

本笔记为北京《初级架构师》v9.0版随堂笔记，笔记版本为v7.0，每个新班内容将不断更新。笔记中所涉及到的案例均针对Red Hat Enterprise Linux 6u4 x86_64环境。笔记内容不包括扩展及提高部分，主要可以为学习《初级架构师》课程的同学提供一个参考的手册。

尚观所有学员均可自由使用和转载该笔记，敬请注明出处。

by 杨生（天云）

mail: yangsheng131420@126.com

职位需求分析

job 1

[系统运维工程师](#)

发布日期：2013-06-07 工作地点：北京 招聘人数：1

工作年限：二年以上 学 历：本科

职位职能：系统工程师 网络工程师

职位描述：

岗位职责：

1. 负责网络及应用服务系统的规划和架构及实施工作；

2. 负责网络及应用服务系统的日常运维工作；

任职要求：

1. 计算机相关专业本科学历，两年以上linux服务器维护经验；

2. 具有大型网站系统维护经验；

3. 熟悉TCP/IP协议；

4. 熟悉mysql数据库服务器安装、配置与维护；

5. 熟悉apache, nginx, lighttpd等服务器的安装、配置与维护；

6. 熟悉DNS、mail (qmail、postfix) 服务器的安装、配置与维护；

7. 熟悉代理服务器squid的安装、配置与维护；

8. 熟悉网络安全及其多种防火墙的配置；

9. 善于分析和解决系统及网络故障；

10. 熟悉存储应用；

11. 有较强的shell脚本编写能力；

12. 工作认真负责，有较强的学习能。

job 2

[系统工程师](#)

发布日期： 2013-06-07 工作地点： 北京 招聘人数： 1

工作年限： 二年以上 学 历： 大专

职位职能：技术支持/维护工程师 系统管理员/网络管理员

职位描述：

职位描述：

- 1、负责公司服务器的维护、管理工作，通过监控系统实时掌握服务运行情况；
- 2、与开发团队密切配合完成项目的部署、监控并维护，保障运维质量；
- 3、解决线上涉及网络、硬件、操作系统和程序的各种问题；
- 4、对进行运维手册编辑整理；
- 5、具备常用运维脚本编写能力，提升运维管理效率；

具备以下行业工作经验：

- 搜索引擎、商务平台、社区产品运维
- 运维安全管理
- 分布式系统运维
- 数据分析与监控
- 大规模数据存储管理

任职资格：

- 1、计算机或相关专业大专以上学历，两年以上互联网行业相关工作经验；
- 2、熟练掌握linux系统管理和维护；至少管理超过50台服务器的工作经验；
- 3、熟练掌握 cacti监控系统，nagios 服务监控系统；
- 4、熟练掌握 SQL语句,熟悉mysql数据库维护工作，熟悉数据库的管理和维护；
- 5、熟悉TCP/IP、HTTP等协议；
- 6、精通shell或python、perl编程；
- 7、熟练配置 Apache、tomcat、LAMP的环境搭建及配置；
- 8、工作踏实，做事仔细，认真负责，有良好的团队合作精神，吃苦耐劳，能承受加班；
- 9、熟悉互联网产品基本架构，有互联网产品研发或运维经验者优先。

job 3

[高级应用运维工程师](#)

发布日期： 2013-06-07 工作地点： 北京 招聘人数： 若干

职位职能： 网站维护工程师

职位描述：

【岗位职责】

- 1、负责国内外服务器的系统、应用运维工作，包括系统监控、系统升级、系统安全、性能优化等；
- 2、参与公共类运维支撑平台的建设；
- 3、参与部门运维自动化实施；
- 4、运维相关的新技术研究；
- 5、负责公司内部mysql数据库管理工作(主要是日常数据库维护工作)。

【岗位要求】

- 1、大专以上学历，两年以上服务器维护经验，了解互联网运用架构，管理过50台Linux服务器以上经验；
- 2、有一定的开发能力（能使用shell、perl、python等脚本语言）；
- 3、有高度的责任心、较强的沟通能力和良好的团队合作精神；
- 4、做事细致，主动性好，善于主动思考问题，并提出自己的解决思路；
- 5、有维护mysql数据库的经验(熟悉mysql主从原理及myisam,innodb存储引擎原理)。

job 4

云计算工程师

公司 北京新网互联科技有限公司

发布日期：2014-05-06 工作地点：北京 招聘人数：1

工作年限：一年以上 学 历：大专 薪水范围：面议

职位标签：云计算 云主机 linux 运维工程师

职位职能：技术支持/维护工程师

职位描述：

岗位要求：

- 1、具有良好的计算机操作系统基础知识、掌握tcp/ip 7层协议设备。
- 2、熟悉redhat/centos linux 操作系统，熟悉linux下各种系统管理命令的使用。
- 3、能书写常用的shell及python，能熟练使用python进行编程优先。
- 4、熟悉mysql数据库日常管理，数据库复制技术，各种常用SQL命令的书写，了解redis等NOSQL。
- 5、熟悉kvm、xenserver等虚拟化技术,掌握相关的灾备方案。
- 6、能熟悉openstack云架构，进行过实际项目开发或部署优先。
- 7、了解NAS/SAN存储系统，熟悉ceph、GlusterFS、hadoop等分布式存储系统优先。

岗位优势：

- 1、能参与到公司公有云平台的建设，有机会接触最新的云计算技术。
- 2、良好的技术氛围，我们对于技术有开放的态度，良好的内部沟通学习机制。
- 3、参与到虚拟主机产品的优化、接触到无线互联网的优化与运维技术。

job 5

高级运维工程师

喜阅网

公司行业：计算机服务(系统、数据服务、维修) 通信/电信运营、增值服务

公司性质：民营公司

公司规模：50-150人

发布日期：2014-05-07 工作地点：北京-东城区 招聘人数：1

工作年限：三年以上 学 历：本科 薪水范围：6000-7999

职位标签：运维经理 运维工程师 运维总监

职位职能：技术支持/维护工程师 技术支持/维护经理

职位描述：

岗位职责：

- 1、负责公司网站相关业务运营系统安全稳定运行；
- 2、负责公司内部办公网络和员工电脑运维，保障公司人员办公环境稳定。

岗位要求：

- 1、计算机相关专业本科以上，具有3年以上大中型网站运维经验；
- 2、精通nginx，apache，tomcat，jboss，memcached等web应用的部署和优化；
- 3、熟练掌握windows及Linux操作系统内核优化策略，有服务器集群及安全加固部署的实际经验，能独立编写服务器脚本；
- 4、熟练掌握Linux监控软件，如nagios，cacti，zabbix等；

- 5、熟练掌握Mysql数据库的优化，主从设置，数据库集群；
- 6、具有防火墙策略及F5设备的配置经验；
- 7、爱岗敬业，对客户的需求能及时处理及反馈，有极佳的创业精神。

需求汇总

=====

职位：

- 1. 系统工程师
- 2. 运维工程师
- 3. 高级运维工程
- 4. 技术支持工程师
- 5. 售后技术支持
- 6. 云计算工程师
- 7. MySQL DBA

要求：

- 1. shell、perl、python
- 2. 对Linux发行版有一定了解： RHEL、CentOS、Fedora、Ubuntu、SUSE等
- 3. 对Unix有一定了解：AIX，HP-UX，Solaris
- 4. 对硬件有一定了解：IBM，HP，EMC，DELL (PC Server、小型机、刀片服务器、路由器、交换机、存储设备)
- 5. 工作经验
- 6. 学历及计算机专业
- 7. 有较强的学习能力
- 8. 有证书者优先: RHCE、RHCA、OCP、OCM

=====

机架式服务器：

机架式服务器的宽度为19英寸，高度以U为单位(1U=1.75英寸=44.45毫米)，通常有1U，2U，3U，4U，5U，7U几种标准的服务器。机柜的尺寸也是采用通用的工业标准，通常从22U到42U不等；机柜内按U的高度有可拆卸的滑动拖架，用户可以根据自己服务器的标高灵活调节高度，以存放服务器、交换机、磁盘阵列柜等网络设备。
U越高：托管成本越高、扩展能力超强

对Linux发行版有一定了解：

==OS 熟悉	通过发行注记 (release note) 对新版本的了解
==提供服务	文件服务：FTP、NFS、CIFS 名称服务：DNS、DHCP WEB服务：Apache LAMP、Nginx LNMP、LNMMMP、Tomcat Mail服务：Postfix 目录服务：LDAP 监控服务：Nagios、Cacti、Zabbix
==教学环境	可以实现这些、那些功能
==工作环境	依赖于需求、通过官方手册解决需求

ULA 课程安排

===== ULA课程时间分配 : 22天 (含考试) =====

TCP/IP协议

2

天

Linux 系统高级安全及入侵恢复

5天

Linux 虚拟化技术

2天

Linux 高可用集群、负载均衡集群

5天

Linux 高级存储、共享存储、云存储

4天

Linux 系统性能分析及调优

3天

Linux 云计算、ULA考试

2天

一、TCP/IP协议

TCP/IP协议概述

=====TCP/IP=====

应用层： 它只负责产生相应格式的数据 ssh ftp nfs cifs dns http smtp pop3

传输层： 定义数据传输的两种模式：

 TCP (传输控制协议：面向连接，可靠的，效率相对不高)

 UDP (用户数据报协议：非面向连接，不可靠的，但效率高)

网络层： 连接不同的网络如以太网、令牌环网

 IP (路由，分片) 、 ICMP、 IGMP

 ARP (地址解析协议，作用是将IP解析成MAC)

数据链路层： 以太网传输

物理层： 主要任务是规定各种传输介质和接口与传输信号相关的一些特性

7 应用层

6 表示层

5 会话层

4 传输层

3 网络层

2 数据链路层

1 物理层

<应用层>

TELNET, SSH, HTTP, SMTP, POP,
SSL/TLS, FTP, MIME, HTML,
SNMP, MIB, SIP, RTP ...

<传输层>

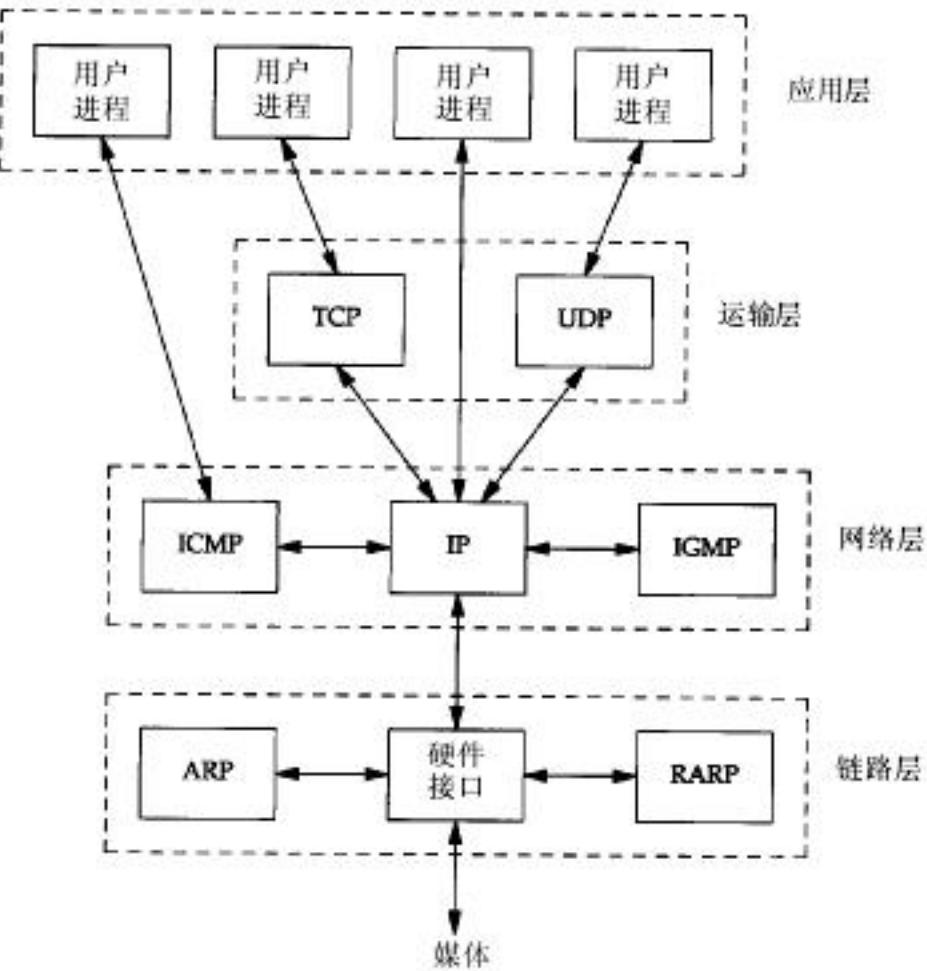
TCP, UDP, UDP-Lite, SCTP, DCCP

<网络层>

ARP, IPv4, IPv6, ICMP, IPsec

以太网、无线LAN、PPP.....

(双绞线电缆、无线、光纤.....)

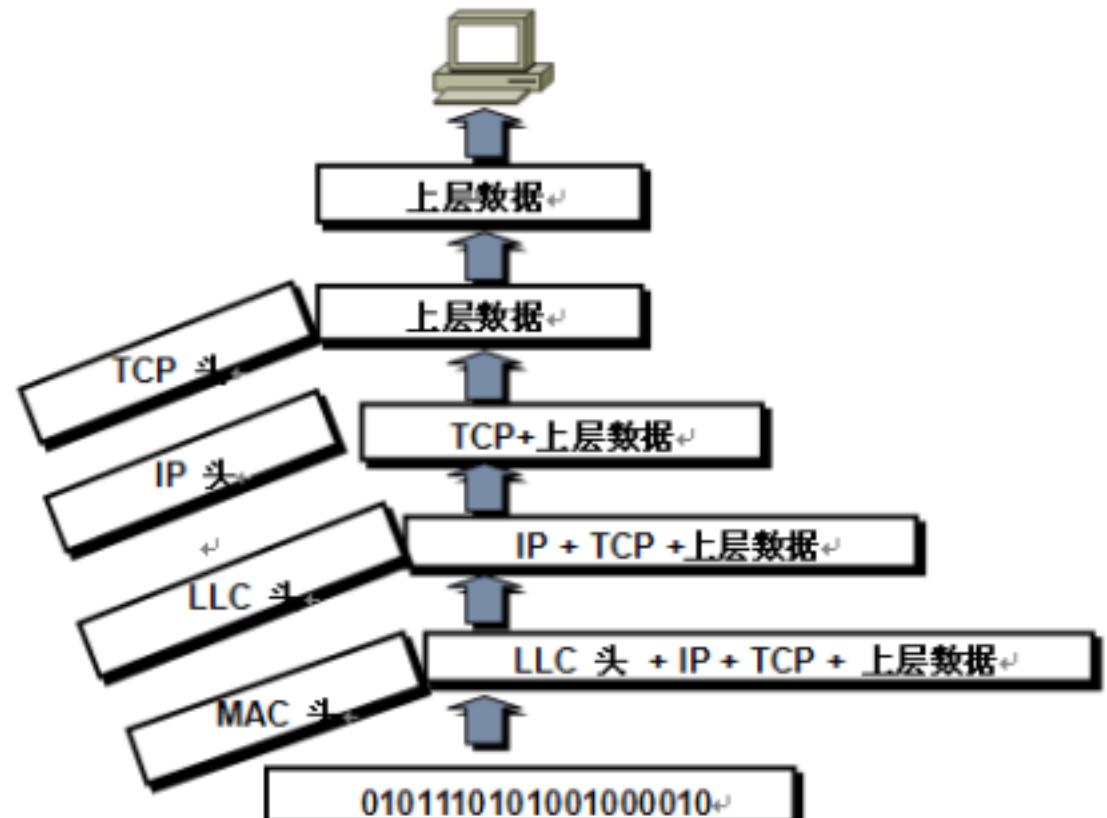
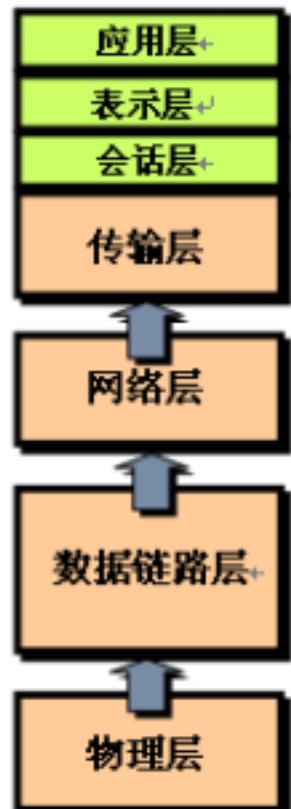
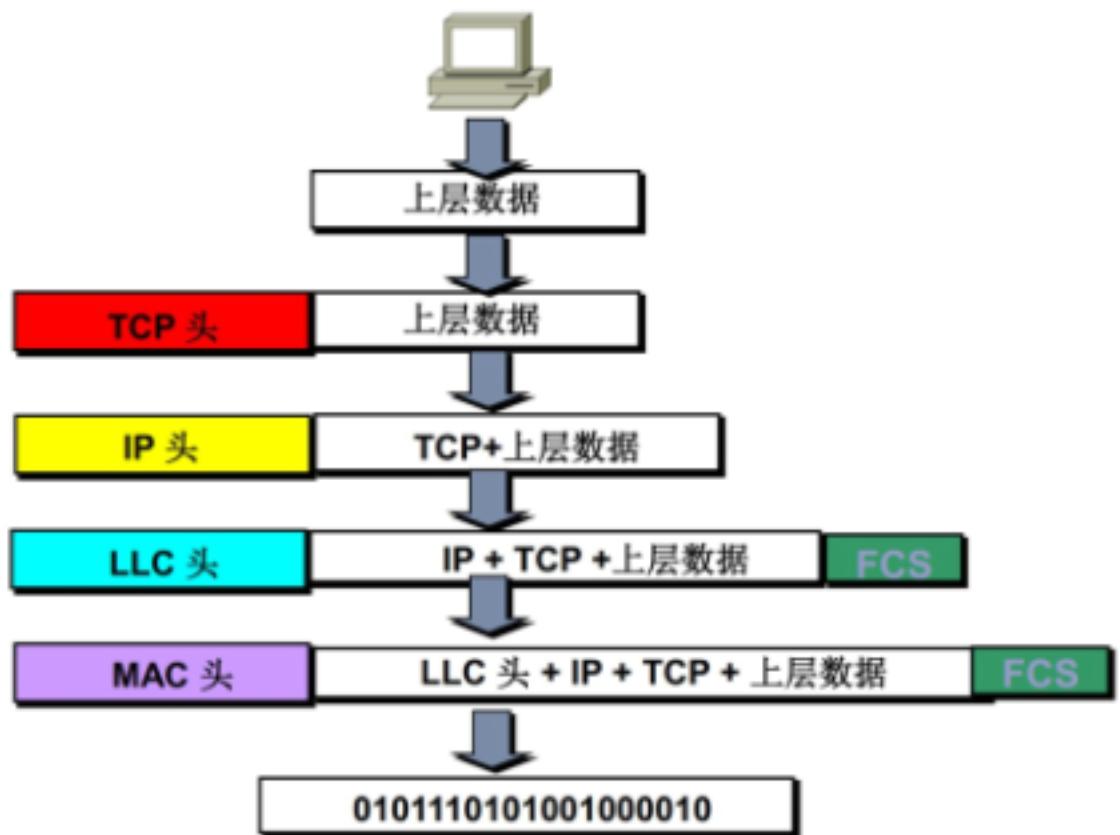
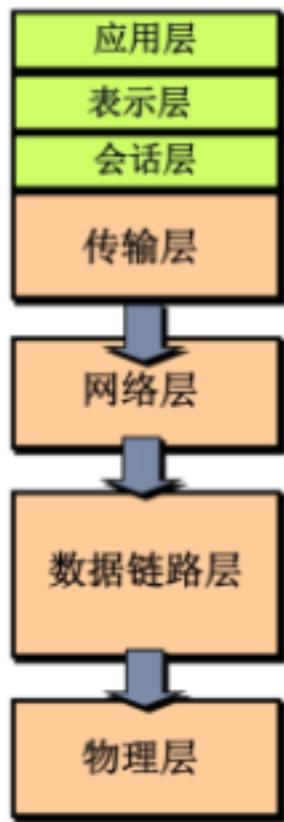


示例：

Client (firefox) -----> HTTP Server

封装

解封装



例如传输100B的数据：

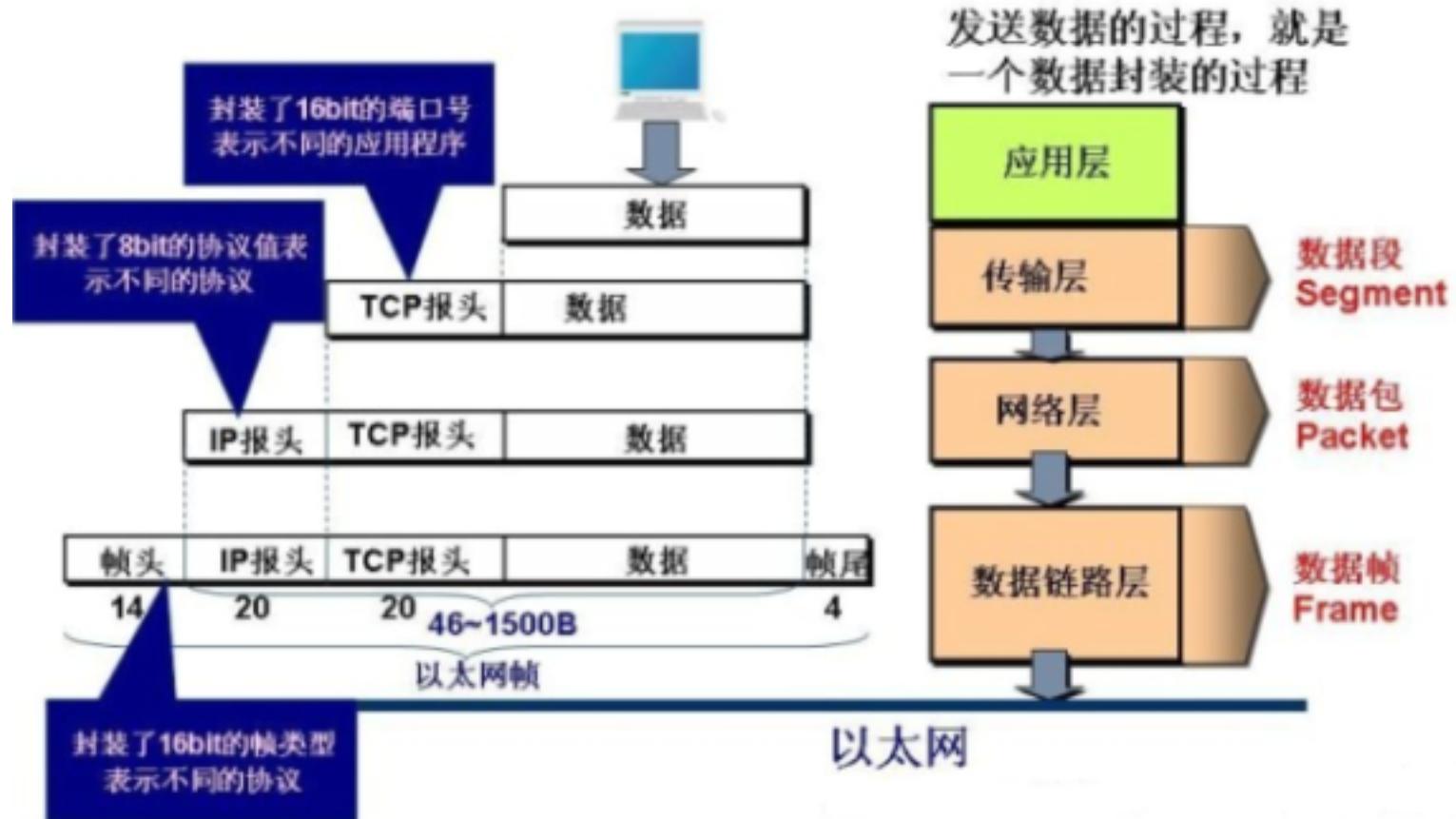
data
TCP首部 (20-40B) 数据段，MSS

应用数据100B

IP首部 (20B)
数据链路层
物理层

数据包，数据报，超过1500字节分片：MTU
数据帧
比特流

数据封装



抓取数据包

抓取数据包

tcpdump,wireshark...

一、tcpdump

```
[root@tianyun ~]# yum -y install tcpdump
```

1.针对特定网口抓包 (-i选项)

```
[root@tianyun ~]# tcpdump
```

```
[root@tianyun ~]# tcpdump -D
```

```
[root@tianyun ~]# tcpdump -i eth0
```

```
[root@tianyun ~]# tcpdump -i eth0 -nn  
[root@tianyun ~]# tcpdump -i eth0 -nn -S
```

2. 抓取指定数目的包 (-c选项)

默认情况下tcpdump将一直抓包，直到按下“ctrl+c”中止

```
[root@tianyun ~]# tcpdump -i eth0 -nn -c 2
```

3. 将抓到包写入文件中 (-w选项)

```
[root@tianyun ~]# tcpdump -i eth0 -nn -w lab1.tcpdump
```

4. 读取tcpdump保存文件 (-r选项)

```
[root@tianyun ~]# tcpdump -nn -r lab1.tcpdump
```

5. 增加抓包时间戳 (-ttt选项)

使用-ttt选项，抓包结果中将包含抓包日期：

```
[root@tianyun ~]# tcpdump -i eth0 -nn -ttt
```

tcpdump过滤器(filter)：

6. 指定抓包的协议类型

```
[root@tianyun ~]# tcpdump -i eth0 -nn 'arp'  
[root@tianyun ~]# tcpdump -i eth0 -nn 'icmp'  
[root@tianyun ~]# tcpdump -i eth0 -nn 'ip'  
[root@tianyun ~]# tcpdump -i eth0 -nn 'tcp'
```

7. 抓和某个主机通讯的包

```
[root@tianyun ~]# tcpdump -i eth0 -nn host 192.168.1.20  
[root@tianyun ~]# tcpdump -i eth0 -nn not host 192.168.1.20
```

8. 指定抓包端口

```
[root@tianyun ~]# tcpdump -i eth0 -nn port 22  
[root@tianyun ~]# tcpdump -i eth0 -nn host 192.168.1.20 and port 22  
[root@tianyun ~]# tcpdump -i eth0 -nn dst 192.168.1.20 and dst port 22  
[root@tianyun ~]# tcpdump -i eth0 -nn src 192.168.1.20 and dst port 22  
[root@tianyun ~]# tcpdump -i eth0 -nn dst 192.168.1.20 and not port 22  
[root@tianyun ~]# tcpdump -i eth0 -nn host 172.16.100.49 and dst port 22
```

```
[root@tianyun ~]# tcpdump -i eth0 -nn host 172.16.30.55 and port 22 -w /tmp/ssh.tcpdump
```

二、wireshark

```
[root@tianyun ~]# yum -y install wireshark wireshark-gnome  
[root@tianyun ~]# wireshark /tmp/ssh.tcpdump
```

应用层协议

HTTP协议

应用层协议：**HTTP**

一、HTTP协议简介

超文本传输协议 (HTTP-Hypertext transfer protocol) 是一种详细规定了浏览器和万维网服务器之间互相通信的规则，通过因特网传送万维网文档的数据传送协议。

二、HTTP协议命令

```
[root@tianyun ~]# tcpdump -i eth0 -nn port 80 -w http.tcpdump  
[root@client ~]# firefox 192.168.1.250
```

wireshark详细查看HTTP：

HTTP GET / HTTP/1.1 -----> Follow TCP Stream (右键)

红色部分：客户端发给服务器

蓝色部分：服务器回给客户端

```
[root@tianyun ~]# curl -I 172.16.70.86          //只看HTTP头部内容  
HTTP/1.1 200 OK  
Server: nginx/1.6.0  
Date: Wed, 24 Dec 2014 02:27:43 GMT  
Content-Type: text/html; charset=utf-8  
Connection: keep-alive  
X-Powered-By: PHP/5.3.27
```

HTTP响应头

HTTP/1.1 200 OK	//协议返回码200
Date: Thu, 25 Oct 2012 06:16:50 GMT	
Server: Apache/2.2.3 (Red Hat)	//服务器软件及版本
Last-Modified: Thu, 25 Oct 2012 05:40:00 GMT	//告诉客户端是否需要从服务器更新，由于客户端有缓存的数据
ETag: "330001-12-a2518c00"	//和服务器比Etag值，作用为Last-Modified相似
Accept-Ranges: bytes	
Content-Length: 18	
Connection: close	
Content-Type: text/html; charset=UTF-8	//mime类型相关

```
[root@server2 ~]# nc www.126.com 80  
HEAD / HTTP/1.1
```

```
HTTP/1.0 403 Forbidden  
Server: Cdn Cache Server V2.0  
Date: Tue, 11 Jun 2013 02:45:15 GMT  
Content-Type: text/html  
Content-Length: 685  
Expires: Tue, 11 Jun 2013 02:45:15 GMT  
X-Via: 1.0 hdwt9:9080 (Cdn Cache Server V2.0)  
Connection: close
```

```
[root@server2 ~]# nc www.sina.com 80  
HEAD / HTTP/1.1
```

```
HTTP/1.1 403 Forbidden  
Server: squid/2.7.STABLE5  
Date: Tue, 11 Jun 2013 02:46:31 GMT  
Content-Type: text/html  
Content-Length: 116  
X-Squid-Error: ERR_ACCESS_DENIED 0  
X-Cache: MISS from xd33-70.sina.com.cn  
Connection: close
```

MIME类型

```
[root@server2 ~]# grep mime /etc/httpd/conf/httpd.conf
LoadModule mime_magic_module modules/mod_mime_magic.so
LoadModule mime_module modules/mod_mime.so
# TypesConfig describes where the mime.types file (or equivalent) is
TypesConfig /etc/mime.types
# The mod_mime_magic module allows the server to use various hints from the
<IfModule mod_mime_magic.c>
# MIMEMagicFile /usr/share/magic.mime
# file mime.types for specific file types.
```

三、HTTP的持久性连接

以Apache为例：

```
[root@server2 ~]# vim /etc/httpd/conf/httpd.conf
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 15
```

关闭KeepAlive off，获得HTTP HEAD信息：

响应头
HTTP/1.1 200 OK
Date: Fri, 01 Mar 2013 20:57:23 GMT
Server: Apache/2.2.3 (Red Hat)
Last-Modified: Fri, 01 Mar 2013 19:18:07 GMT
ETag: "2e8108-8-4d6e1ddfa59c0"
Accept-Ranges: bytes
Content-Length: 8
Connection: close
Content-Type: text/html; charset=UTF-8
tianyun

开启KeepAlive on，获得HTTP HEAD信息：

响应头
HTTP/1.1 200 OK
Date: Fri, 01 Mar 2013 20:59:00 GMT
Server: Apache/2.2.3 (Red Hat)
Last-Modified: Fri, 01 Mar 2013 19:18:07 GMT
ETag: "2e8108-8-4d6e1ddfa59c0"
Accept-Ranges: bytes
Content-Length: 8
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
tianyun

连接 (Connection) : TCP连接

请求 (Request) : HTTP请求，如：GET, HEAD, POST, PUT, DELETE等协议命令

Keepalive off : 一次请求 (HTTP请求) 一次连接 (TCP) , 简图1 即每次请求都要建立一个TCP连接

Keepalive on : 多次请求 (HTTP请求) 一次连接 (TCP) , 简图2

应用场景：

如果一个页面中图片较多（因为获得一个图片需要一个HTTP请求，从而产生TCP连接），更加建议打开持久性连接。

四、TCP状态（补充）

问题一：服务器有大量的 TIME_WAIT 不影响服务器，但会影响客户端再使用刚才的端口

分析及解决方案：主动断开 (FIN) 的一端，会进入该状态。如网站服务器端、SSH客户端。

问题二：服务器有大量的 SYN_RECV <半开的连接>

分析及解决方案：设置超时时间

五、nc命令补充

功能1：连接某主机的某TCP/UDP的端口

功能2：扫描某主机开启了哪些端口

功能3：在本机某TCP/UDP的端口上监听

```
[root@server2 ~]# nc 192.168.0.99 80      //连接192.168.0.99的TCP端口80，等价于telnet 192.168.0.99 80
[root@server2 ~]# nc 192.168.0.99 -u 5555 //连接到主机192.168.0.99的udp端口5555
[root@server2 ~]# nc 192.168.0.99 -z 1-2000 //扫描主机192.168.0.99开放的TCP端口
[root@server2 ~]# nc 192.168.0.99 -z -u 1-2000//扫描主机192.168.0.99开放的UDP端口
[root@server2 ~]# nc 192.168.0.99 -l 2000 //在本地接口192.168.0.99的TCP端口2000上监听
[root@server2 ~]# nc 192.168.0.99 -l -u 2000 //在本地接口192.168.0.99的UDP端口2000上监听
```

例1：

```
[root@server2 ~]# nc 192.168.0.101 -l -u 2000
[root@server2 ~]# netstat -tunlp |grep :2000
  udp      0      0 192.168.0.101:2000        0.0.0.0:*          6339/nc
```

六、HTTP头部解释（扩展）

1. Accept：告诉WEB服务器自己接受什么介质类型，/* 表示任何类型，type/* 表示该类型下的所有子类型，type/sub-type。
2. Accept-Charset：浏览器申明自己接收的字符集
Accept-Encoding：浏览器申明自己接收的编码方法，通常指定压缩方法，是否支持压缩，支持什么压缩方法（gzip，deflate）
Accept-Language：浏览器申明自己接收的语言，语言跟字符集的区别：中文是语言，中文有多种字符集，比如big5，gb2312，gbk等等。
3. Accept-Ranges：WEB服务器表明自己是否接受获取其某个实体的一部分（比如文件的一部分）的请求。bytes：表示接受，none：表示不接受。
4. Age：当代理服务器用自己缓存的实体去响应请求时，用该头部表明该实体从产生到现在经过多长时间了。
5. Authorization：当客户端接收到来自WEB服务器的 WWW-Authenticate 响应时，用该头部来回应自己的身份验证信息给WEB服务器。
6. Cache-Control：请求：no-cache（不要缓存的实体，要求现在从WEB服务器去取）
max-age：（只接受 Age 值小于 max-age 值，并且没有过期的对象）
max-stale：（可以接受过去的对象，但是过期时间必须小于max-stale 值）
min-fresh：（接受其新鲜生命期大于其当前 Age 跟 min-fresh 值之和的缓存对象）
响应：public（可以用 Cached 内容回应任何用户）
private（只能用缓存内容回应先前请求该内容的那个用户）
no-cache（可以缓存，但是只有在跟WEB服务器验证了其有效后，才能返回给客户端）
max-age：（本响应包含的对象的过期时间）
ALL: no-store（不允许缓存）
7. Connection：请求：close（告诉WEB服务器或者代理服务器，在完成本次请求的响应后，断开连接，不要等待本次连接的后续请求）。
keepalive（告诉WEB服务器或者代理服务器，在完成本次请求的响应后，保持连接，等待本次连接的后续请求）。
响应：close（连接已经关闭）。
keepalive（连接保持着，在等待本次连接的后续请求）。
Keep-Alive：如果浏览器请求保持连接，则该头部表明希望 WEB 服务器保持连接多长时间（秒）。
例如：Keep-Alive：300
8. Content-Encoding：WEB服务器表明自己使用了什么压缩方法（gzip，deflate）压缩响应中的对象。例如：Content-Encoding：gzip
Content-Language：WEB 服务器告诉浏览器自己响应的对象的语言。
Content-Length：WEB 服务器告诉浏览器自己响应的对象的长度。例如：Content-Length: 26012
Content-Range：WEB 服务器表明该响应包含的部分对象为整个对象的哪个部分。例如：Content-Range: bytes 21010-47021/47022
Content-Type：WEB 服务器告诉浏览器自己响应的对象的类型。例如：Content-Type : application/xml
9. ETag：就是一个对象（比如URL）的标志值，就一个对象而言，比如一个 html 文件，如果被修改了，其 Etag 也会被修改，所以，ETag 的作用跟 Last-Modified 的作用差不多，主要供 WEB 服务器 判断一个对象是否改变了。比如前一次请求某个 html 文件时，获得了其 ETag，当这次又请求这个文件时，浏览器就会把先前获得的 ETag 值发送给 WEB 服务器，然后 WEB 服务器会把这个 ETag 跟该文件的当前ETag 进行对比，然后就知道这个文件有没有改变了。
10. Expired：WEB服务器表明该实体将在什么时候过期，对于过期了的对象，只有在跟WEB服务器验证了其有效性后，才能用来响应客户请求。
是 HTTP/1.0 的头部。例如：Expires : Sat, 23 May 2009 10:02:12 GMT
11. Host：客户端指定自己想访问的WEB服务器的域名/IP 地址和端口号。例如：Host : rss.sina.com.cn
12. If-Match：如果对象的 ETag 没有改变，其实也就意味著对象没有改变，才执行请求的动作。

If-None-Match : 如果对象的 ETag 改变了，其实也就意味著对象也改变了，才执行请求的动作。

13. If-Modified-Since : 如果请求的对象在该头部指定的时间之后修改了，才执行请求的动作（比如返回对象），否则返回代码304，告诉浏览器该对象没有修改。
例如：If-Modified-Since : Thu, 10 Apr 2008 09:14:42 GMT

If-Unmodified-Since : 如果请求的对象在该头部指定的时间之后没修改过，才执行请求的动作（比如返回对象）。

14. If-Range : 浏览器告诉 WEB 服务器，如果我请求的对象没有改变，就把我缺少的部分给我，如果对象改变了，就把整个对象给我。
浏览器通过发送请求
求对象的ETag 或者自己所知道的最后修改时间给 WEB 服务器，让其判断对象是否改变了。总是跟 Range 头部一起使用。

15. Last-Modified : WEB 服务器认为对象的最后修改时间，比如文件的最后修改时间，动态页面的最后产生时间等等。
例如：Last-Modified : Tue, 06 May 2008 02:42:43 GMT

16. Location : WEB 服务器告诉浏览器，试图访问的对象已经被移到别的位置了，到该头部指定的位置去取。
例如：Location : http://i0.sinainfo.cn/dy/deco/2008/0528/sinahome_0803_ws_005_text_0.gif

17. Pramga : 主要使用 Pramga: no-cache，相当于 Cache-Control : no-cache。
例如：Pragma : no-cache

18. Proxy-Authenticate : 代理服务器响应浏览器，要求其提供代理身份验证信息。
Proxy-Authorization : 浏览器响应代理服务器的身份验证请求，提供自己的身份信息。

19. Range : 浏览器（比如 Flashget 多线程下载时）告诉 WEB 服务器自己想取对象的哪部分。
例如：Range: bytes=1173546-

20. Referer : 浏览器向 WEB 服务器表明自己是从哪个 网页/URL 获得/点击 当前请求中的网址/URL。
例如：Referer : http://www.sina.com/

21. Server: WEB 服务器表明自己是什么软件及版本等信息。例如：Server : Apache/2.0.61 (Unix)

22. User-Agent: 浏览器表明自己的身份（是哪种浏览器）。
例如：User-Agent : Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-CN; rv:1.8.1.14) Gecko/20080404 Firefox/2.0.0.14

23. Transfer-Encoding: WEB 服务器表明自己对本响应消息体（不是消息体里面的对象）作了怎样的编码，比如是否分块（chunked）。
例如：Transfer-Encoding: chunked

24. Vary:
服务器
服务器会
方法，
WEB服务器用该头部的内容告诉 Cache 服务器，在什么条件下才能用本响应所返回的对象响应后续的请求。假如源WEB
在接到第一个请求消息时，其响应消息的头部为：Content-Encoding: gzip; Vary: Content-Encoding，那么 Cache
分析后续请求消息的头部，检查其 Accept-Encoding，是否跟先前响应的 Vary 头部值一致，即是否使用相同的内容编码
这样就可以防止 Cache 服务器用自己Cache 里面压缩后的实体响应给不具备解压能力的浏览器。
例如：Vary : Accept-Encoding

25. Via :
端请求到达第
一个
面， 以此类
列出从客户端到 OCS 或者相反方向的响应经过了哪些代理服务器，他们用什么协议（和版本）发送的请求。当客户
一个代理服务器时，该服务器会在自己发出的请求里面添加 Via 头部，并填上自己的相关信息，当下一个代理服务器 收到
代理服务器的请求时，会在自己发出的请求里面复制前一个代理服务器的请求的Via头部，并把自己的相关信息加到后
推，当 OCS 收到最后一个代理服务器的请求时，检查 Via 头部，就知道该请求所经过的路由。
例如：Via : 1.0 236-81.D07071953.sina.com.cn:80 (squid/2.6.STABLE13)telnet 192.168.1.154 80

其它参考文章：
<http://blog.csdn.net/zhanghefu/article/details/1610030>

SNMP [自修]

FTP协议【自修】

应用层协议：

一、FTP协议简介：

FTP是TCP/IP协议组中的协议之一，是英文File Transfer Protocol的缩写。该协议是Internet文件传送的基础，它由一系列规格说明文档组成，目标是提高文件的共享性，提供非直接使用远程计算机，使存储介质对用户透明和可靠高效地传送数据。简单的说，FTP就是完成两台计算机之间的拷贝，从远程计算机拷贝文件至自己的计算机上，称之为“下载（download）”文件。若将文件从自己计算机中拷贝至远程计算机上，则称之为“上载（upload）”文件。在TCP/IP协议中，FTP标准命令TCP端口号为21，Port方式数据端口为20。

二、FTP两种模式：

两种数据传输模式：主动模式，被动模式

命令端口： 21/tcp

数据端口： 主动模式 Active Mode 20/tcp

被动模式 Passive Mode 随机端口

=====配置被动模式下，服务器所开的数据传输端口的范围=====

```
[root@station95 ~]# man vsftpd.conf
[root@station95 ~]# vim /etc/vsftpd/vsftpd.conf
[root@station95 ~]# tail -n 3 /etc/vsftpd/vsftpd.conf
pasv_enable=YES                                //默认支持被动
pasv_min_port=50000                            //设置端口范围
pasv_max_port=60000
[root@station95 ~]# service vsftpd restart
[root@station95 ~]# iptables -t filter -A INPUT -p tcp --dport 21 -j ACCEPT
[root@station95 ~]# iptables -t filter -A INPUT -p tcp --dport 50000:60000 -j ACCEPT
=====
```

被动模式测试

```
# lftp -d FTP_SERVER_IP      //查看FTP协议命令 debug
```

主动模式测试： 服务器端只开21和20端口

```
# vim /etc/lftp.conf
set ftp:passive-mode off
# lftp -d FTP_SERVER_IP
```

小结：

主动模式： 客户端向FTP服务器提供数据端口，FTP服务器会使用20/tcp和客户端数据端口通信

被动模式： FTP服务器向客户端提供数据端口，客户端和FTP服务器的数据端口通信

三、FTP客户端命令

```
[root@tianyun ~]# lftp -e 'cd pub; get myfile; quit' 192.168.2.51
[root@tianyun ~]# wget ftp://192.168.2.51/pub/myfile -O /tmp/myfile
[root@tianyun ~]# wget ftp://192.168.2.51/pub/myfile -P /tmp/
[root@tianyun ~]# wget http://192.168.2.251/downfile1
[root@tianyun ~]# wget -q ftp://172.16.130.1/11.ctb
[root@tianyun ~]# wget -c http://172.16.130.1/rhel6u5.iso
[root@tianyun ~]# wget -i download.txt
```


FTP命令

命令 描述

ABOR 中断数据连接程序

ACCT <account> 系统特权帐号

ALLO <bytes> 为服务器上的文件存储器分配字节

APPE <filename> 添加文件到服务器同名文件

CDUP <dir path> 改变服务器上的父目录

CWD <dir path> 改变服务器上的工作目录
DELE <filename> 删除服务器上的指定文件
HELP <command> 返回指定命令信息
LIST <name> 如果是文件名列出文件信息，如果是目录则列出文件列表
MODE <mode> 传输模式 (S=流模式，B=块模式，C=压缩模式)
MKD <directory> 在服务器上建立指定目录
NLST <directory> 列出指定目录内容
NOOP 无动作，除了来自服务器上的承认
PASS <password> 系统登录密码
PASV 请求服务器等待数据连接 //被动
PORT <address> IP 地址和两字节的端口 ID //主动
PWD 显示当前工作目录
QUIT 从 FTP 服务器上退出登录
REIN 重新初始化登录状态连接
REST <offset> 由特定偏移量重启文件传递
RETR <filename> 从服务器上找回 (复制) 文件
RMD <directory> 在服务器上删除指定目录
RNFR <old path> 对旧路径重命名
RNTO <new path> 对新路径重命名
SITE <params> 由服务器提供的站点特殊参数
SMNT <pathname> 挂载指定文件结构
STAT <directory> 在当前程序或目录上返回信息
STOR <filename> 储存 (复制) 文件到服务器上
STOU <filename> 储存文件到服务器名称上
STRU <type> 数据结构 (F=文件，R=记录，P=页面)
SYST 返回服务器使用的操作系统
TYPE <data type> 数据类型 (A=ASCII，E=EBCDIC，I=binary)
USER <username>> 系统登录的用户名

=====

FTP响应码
响应代码 解释说明

110 新文件指示器上的重启标记
120 服务器准备就绪的时间 (分钟数)
125 打开数据连接，开始传输
150 打开连接
200 成功
202 命令没有执行
211 系统状态回复
212 目录状态回复
213 文件状态回复
214 帮助信息回复
215 系统类型回复
220 服务就绪
221 退出网络
225 打开数据连接
226 结束数据连接
227 进入被动模式 (IP 地址、ID 端口)
230 登录因特网
250 文件行为完成
257 路径名建立
331 要求密码
332 要求帐号
350 文件行为暂停
421 服务关闭
425 无法打开数据连接
426 结束连接
450 文件不可用
451 遇到本地错误
452 磁盘空间不足
500 无效命令
501 错误参数
502 命令没有执行
503 错误指令序列
504 无效命令参数
530 未登录网络
532 存储文件需要帐号

550 文件不可用
551 不知道的页类型
552 超过存储分配
553 文件名不允许

SMTP协议 [自修]

POP协议【自修】

传输层协议

UDP协议

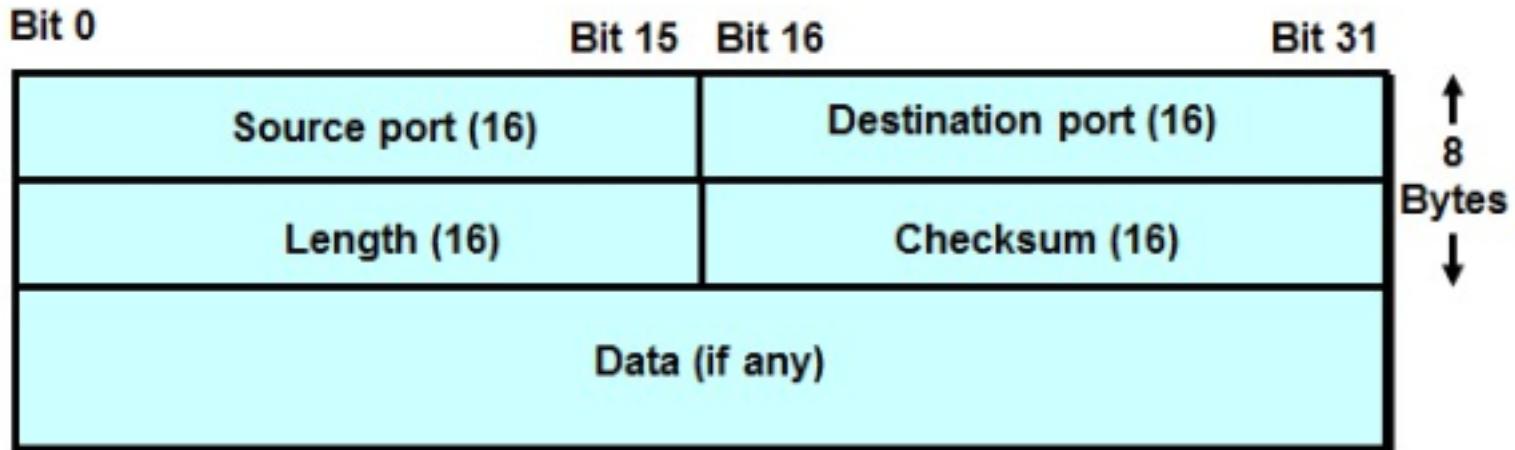
传输层协议：**UDP**

一、 UDP协议简介：

UDP 是User Datagram Protocol的简称，中文名是用户数据报协议。

UDP的特性：它不属于连接型协议，因而具有资源消耗小，处理速度快的优点，所以通常音频、视频和普通数据在传送时使用UDP较多，因为它们即使偶尔丢失一两个数据包，也不会对接收结果产生太大影响。比如我们聊天用的QQ就是使用的UDP协议。

二、 UDP头部 8字节



```
▷ Frame 1 (73 bytes on wire, 73 bytes captured)
▷ Ethernet II, Src: 54:04:a6:cd:9c:14 (54:04:a6:cd:9c:14), Dst: 40:16:9f:f3:48:d6 (40:16:9f:f3:48:d6)
▷ Internet Protocol, Src: 192.168.1.1 (192.168.1.1), Dst: 192.168.1.250 (192.168.1.250)
▽ User Datagram Protocol, Src Port: 43558 (43558), Dst Port: domain (53)
    Source port: 43558 (43558)
    Destination port: domain (53)
    Length: 39
    ▽ Checksum: 0xa151 [validation disabled]
        [Good Checksum: False]
        [Bad Checksum: False]
▷ Domain Name System (query)
```

source port: 源端口 +1024 随机端口
destination port: 目的端口
length: UDP的数据段的长度 (DATA长度 + UDP首部的长度)
checksum : 检查校验和

**非面向连接：不用先与对方建立连接，不握手
不可靠的：没有确认机制**

UDP的可靠是通过上层协议来保障的！！！

[root@tianyun ~]# less /etc/services 查看协议对应的端口号

使用UDP的常见应用层协议：dns(domain),ntp,syslog

```
[root@tianyun ~]# tcpdump -i eth0 -nn host 172.16.8.100 -w ntp.tcpdump  
[root@tianyun ~]# ntpdate 172.16.8.100 对方NTP正常开启
```

```
[root@tianyun ~]# tcpdump -i eth0 -nn host 172.16.8.100 -w dns.tcpdump  
[root@tianyun ~]# dig @172.16.8.100 www.baidu.com
```

小结：当访问目标主机未开启的udp端口时，会收到由ICMP协议发送的 Destination unreachable (Port unreachable)s

TCP协议

传输层协议 : TCP

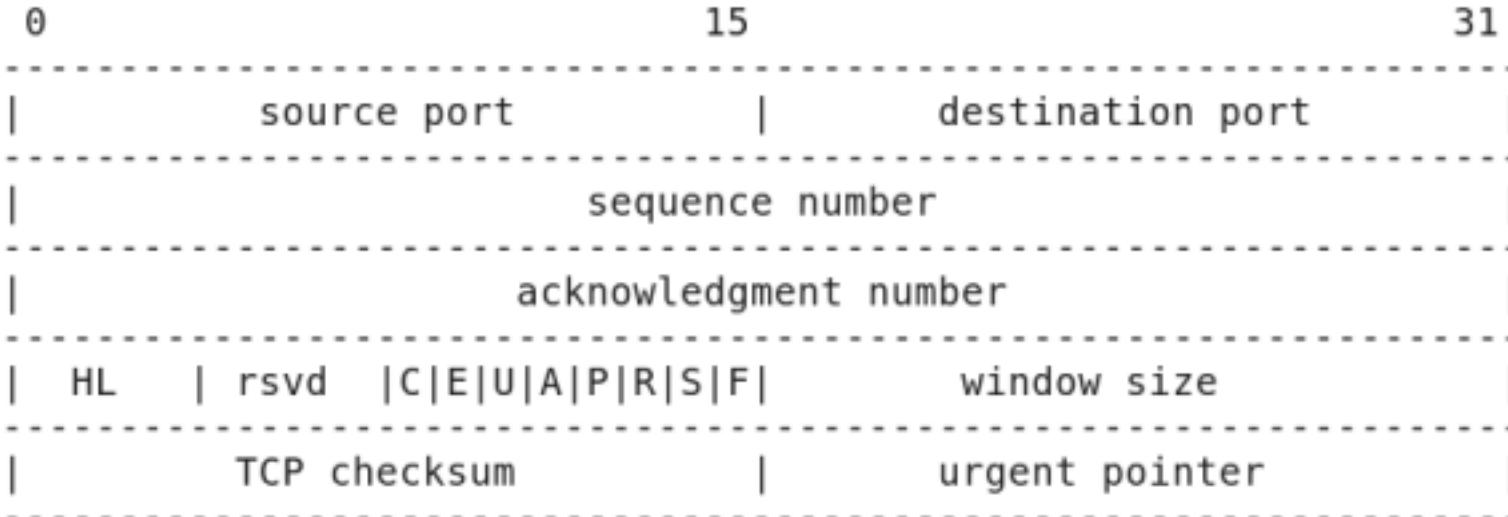
本章重点掌握：

1. TCP建立连接/断开连接（三次握手、四次挥手）的过程 及相应TCP状态
 2. TCP连接标记flags

一、TCP协议简介：

TCP : Transmission Control Protocol 传输控制协议TCP是一种面向连接（连接导向）的、可靠的、基于字节流的传输层（Transport layer）通信协议，由IETF的RFC 793说明（specified）。在简化的计算机网络OSI模型中，它完成第四层传输层所指定的功能。

二、TCP首部 20Byte



使用TCP的常见应用层协议 : http,ftp,ssh

```
[root@tianyun ~]# tcpdump -i eth0 -nn host 172.16.8.100 -w http-80.tcpdump
[root@tianyun ~]# tcpdump -i eth0 -nn host 172.16.8.100 -w ftp.tcpdump
[root@tianyun ~]# tcpdump -i eth0 -nn host 172.16.8.100 -w ssh.tcpdump
[root@tianyun ~]# tcpdump -i eth0 -nn host 172.16.8.100 -w http-8888.tcpdump
[root@tianyun ~]# firefox http://172.16.8.100:8888
```

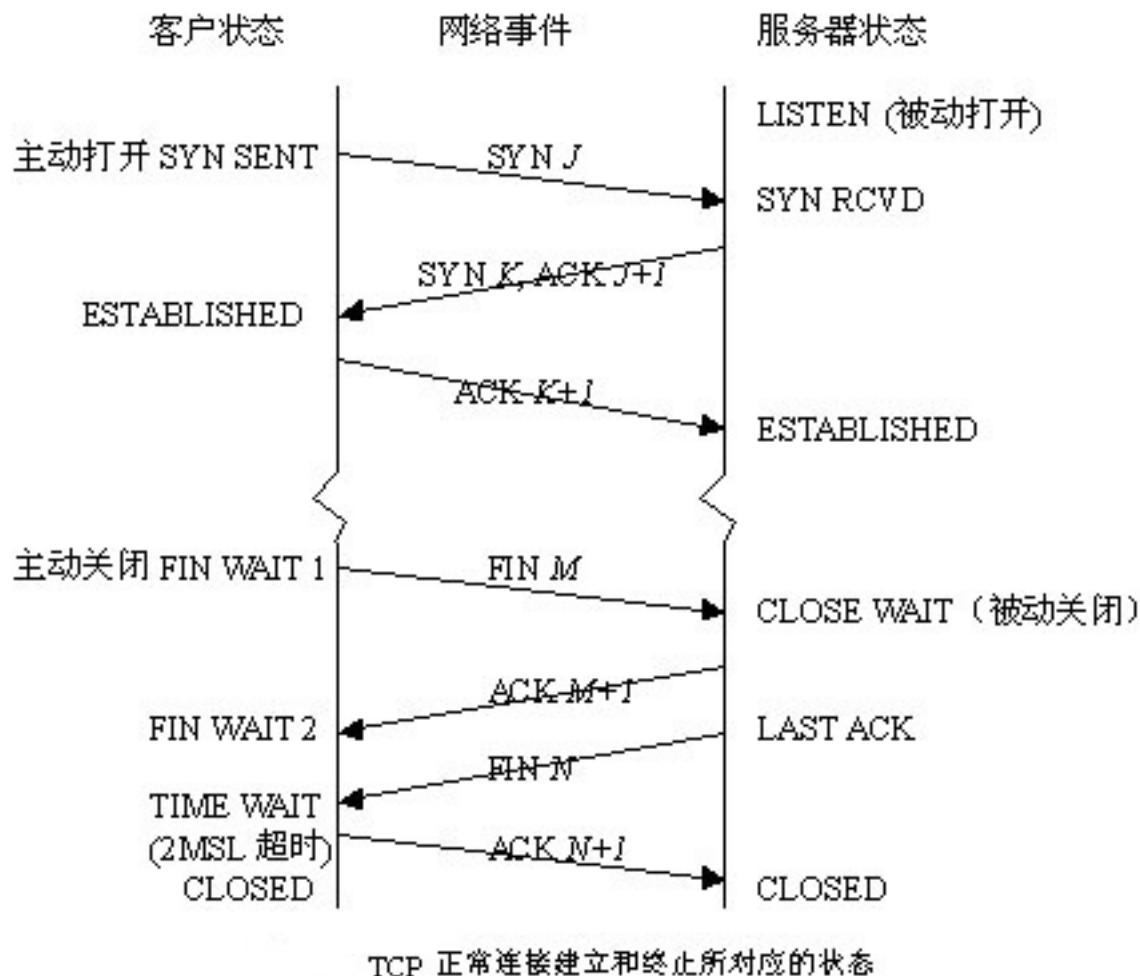
firefox http://172.16.8.100
firefox ftp://172.16.8.100
ssh 172.16.8.100
对方未开启8888/tcp端口

小结：当访问目标主机一个未开启的tcp端口，会收到Reset标记位的响应

```
▷ Frame 1 (74 bytes on wire, 74 bytes captured)
▷ Ethernet II, Src: 54:04:a6:cd:9c:14 (54:04:a6:cd:9c:14), Dst: 40:16:9f:f3:48:d6 (40:16:9f:f3:48:d6)
▷ Internet Protocol, Src: 192.168.1.1 (192.168.1.1), Dst: 192.168.1.250 (192.168.1.250)
⇒ Transmission Control Protocol, Src Port: 43779 (43779), Dst Port: http (80), Seq: 0, Len: 0
    Source port: 43779 (43779)
    Destination port: http (80)
    [Stream index: 0]
    Sequence number: 0      (relative sequence number)
    Header length: 40 bytes
    Flags: 0x02 (SYN)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...0 .... = Acknowledgement: Not set
        .... 0... = Push: Not set
        .... .0.. = Reset: Not set
        .... ..1. = Syn: Set
            .... .0 = Fin: Not set
        Window size: 14680
    Checksum: 0x83fb [validation disabled]
    Options: (20 bytes)
```

source port:	源端
destinaton port:	目标端口
sequence number:	序列号
acknowledgement number:	确认号
header length:	20~40 (首部长度)
flags:	TCP标记位 SYN、ACK、FIN、RST、PSH、URG

三、TCP连接简图：三次握手，数据传输，四次挥手



```
[root@install ~]# netstat -an |grep :80
tcp        0      0 :::80          :::*              LISTEN
tcp        0      0 ::ffff:172.16.8.100:80    ::ffff:172.16.30.35:47553  TIME_WAIT
tcp        0      0 ::ffff:172.16.8.100:80    ::ffff:172.16.30.35:47555  TIME_WAIT
tcp        0      0 ::ffff:172.16.8.100:80    ::ffff:172.16.30.35:47558  TIME_WAIT
tcp        0      0 ::ffff:172.16.8.100:80    ::ffff:172.16.30.35:47547  TIME_WAIT
tcp        0      0 ::ffff:172.16.8.100:80    ::ffff:172.16.30.35:47539  TIME_WAIT
tcp        0      0 ::ffff:172.16.8.100:80    ::ffff:172.16.30.35:47544  TIME_WAIT
tcp        0      0 ::ffff:172.16.8.100:80    ::ffff:172.16.30.199:40099  TIME_WAIT
tcp        0      0 ::ffff:172.16.8.100:80    ::ffff:172.16.30.35:47549  TIME_WAIT
tcp        0      0 ::ffff:172.16.8.100:80    ::ffff:172.16.30.35:47556  TIME_WAIT
tcp        0      0 ::ffff:172.16.8.100:80    ::ffff:172.16.30.35:47551  TIME_WAIT
tcp        0      0 ::ffff:172.16.8.100:80    ::ffff:172.16.30.35:47557  TIME_WAIT
tcp        0      0 ::ffff:172.16.8.100:80    ::ffff:172.16.30.35:47552  TIME_WAIT
tcp        0      0 ::ffff:172.16.8.100:80    ::ffff:172.16.30.35:47554  TIME_WAIT
tcp        0      0 ::ffff:172.16.8.100:80    ::ffff:172.16.30.35:47543  TIME_WAIT
tcp        0      0 ::ffff:172.16.8.100:80    ::ffff:172.16.30.199:40100  TIME_WAIT
tcp        0      0 ::ffff:172.16.8.100:80    ::ffff:172.16.30.35:47540  TIME_WAIT
tcp        0      0 ::ffff:172.16.8.100:80    ::ffff:172.16.30.35:47548  TIME_WAIT
tcp        0      0 ::ffff:172.16.8.100:80    ::ffff:172.16.30.35:47538  TIME_WAIT
tcp        0      0 ::ffff:172.16.8.100:80    ::ffff:172.16.30.35:47537  TIME_WAIT
tcp        0      0 ::ffff:172.16.8.100:80    ::ffff:172.16.30.178:39994  TIME_WAIT
tcp        0      0 ::ffff:172.16.8.100:80    ::ffff:172.16.30.35:47542  TIME_WAIT
tcp        0      0 ::ffff:172.16.8.100:80    ::ffff:172.16.30.178:39993  TIME_WAIT
tcp        0      0 ::ffff:172.16.8.100:80    ::ffff:172.16.30.35:47550  TIME_WAIT
tcp        0      0 ::ffff:172.16.8.100:80    ::ffff:172.16.30.35:47541  TIME_WAIT
tcp        0      0 ::ffff:172.16.8.100:80    ::ffff:172.16.30.199:40101  TIME_WAIT
tcp        0      0 ::ffff:172.16.8.100:80    ::ffff:172.16.30.35:47545  TIME_WAIT
tcp        0      0 ::ffff:172.16.8.100:80    ::ffff:172.16.30.178:39995  TIME_WAIT
```

```
tcp      0      0 ::ffff:172.16.8.100:80    ::ffff:172.16.30.35:47546  TIME_WAIT
```

仅出现在主动断开FIN端，例如用户向网站服务器请求某个文件，服务器已为用户提供，此时服务器会主动FIN

四、TCP flag标记位

SYN : 同步标记位

ACK : 确认标记位（包括三次握手，四次挥手中的确认及数据传输时的确认）

RST : 重置标记位（访问一个未打开的TCP端口时，服务器返回一个带RST标记位的响应）

PSH : 推标记位（**需要低延时**的应用加PSH，比较快的发送出去，不往缓存中放，ssh, telnet, ftp-command）

URG : 紧急标志位（TCP的紧急方式是发送端向另一端发送紧急数据的一种方式。紧急指针指向包内数据段的某个字节不进入接收缓冲就直接交给上层进程，余下的数据要进入接收缓冲。通信的时候网络故障问题引起的数据没传完，会发送一个URG标记位）

FIN : 结束标记位（发送端完成发送任务，双方都需要确认没有数据再发给对方）

五、TCP其它特点

序列号、确认号

通信的双方各自维护着一套 seq 序列号，并对对方发送的数据包进行 ACK 确认

```
[root@tianyun ~]# tcpdump -i eth0 -nn -S // -S详细的方式，适合看序列号
```

```
[root@tianyun ~]# tcpdump -i eth0 -nn -S > tcp.file
```

MSS最大段长

MSS表示TCP传往另一端最大块数据的长度。当一个连接建立时，连接的双方都要通告各自的MSS。一般来说，报文段越大，允许每个报文段传输的数据就越多，相对于IP和TCP首部有更高的网络利用率。

以太网MSS : MTU (1500) - 20 (TCP首部) - 20 (IP首部) = **1460**

网络层协议

IP协议

网络层协议：IP

一、IP协议简介：路由、分片

IP协议是TCP/IP协议族中最为核心的协议。所有TCP、UDP、ICMP及IGMP数据都以IP数据报格式传输。

==特点：

不可靠 (unreliable) : 它不能保证IP数据报能成功地到达目的地，**IP仅提供尽力而为的传输服务**。如果发生某种错误时，如某个路由器暂时用完了缓冲区，IP有一个简单的错误处理算法：丢弃该数据报，然后发送ICMP消息报给信源端。任何要求的可靠性必须由上层来提供（如TCP）。

无连接 (connectionless) : 这个术语的意思是IP并不维护任何关于后续数据报的状态信息。每个数据报的处理是相互独立的。这也说明，IP数据报可以不按发送顺序接收。如果一信源向相同的信宿发送两个连续的数据报（先是A，然后是B），每个数据报都是独立地进行路由选择，可能选择不同的路线，因此B可能在A到达之前先到达。

二、IP首部：



IP协议: 首部 20字节

由于TCP、UDP、ICMP和IGMP都要通过IP协议传送数据，因此IP必须在生成的IP首部中加入某种标识，以表明数据属于哪一层。为此，IP在首部中存入一个长度为8bit的数值，称作**协议域**：

- | | |
|----|----------------|
| 1 | 表示为ICMP协议 0x01 |
| 2 | 表示为IGMP协议 0x02 |
| 6 | 表示为TCP 协议 0X06 |
| 17 | 表示为UDP协议 0x11 |

IP首部详细说明：

[root@tianyun ~]# wireshark icmp.tcpdump

```

▶ Frame 2 (98 bytes on wire, 98 bytes captured)
▶ Ethernet II, Src: 40:16:9f:f3:48:d6 (40:16:9f:f3:48:d6), Dst: 54:04:a6:cd:9c:14 (54:04:a6:cd:9c:14)
└ Internet Protocol, Src: 192.168.1.250 (192.168.1.250), Dst: 192.168.1.1 (192.168.1.1)
    Version: 4
    Header length: 20 bytes
    ▶ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 84
    Identification: 0x12be (4798)
    ▼ Flags: 0x00
        0.. = Reserved bit: Not Set
        .0. = Don't fragment: Not Set
        ..0 = More fragments: Not Set
    Fragment offset: 0
    Time to live: 64
    Protocol: ICMP (0x01)
    ▶ Header checksum: 0xe39f [correct]
    Source: 192.168.1.250 (192.168.1.250)
    Destination: 192.168.1.1 (192.168.1.1)
▶ Internet Control Message Protocol

```

版本 : 4
 首部长度 : 20字节
 服务类型 : tos
 总长度 : 小于等于1500字节
 偏移量 : 判断同一个数据的先后顺序, 0 1480 2960
 分片标识 : 判断片是否属于同一IP包
 上层协议 : 上层协议icmp等
 生存时间ttl : 每经过一个路由器将减1

三、IP分片 [了解]

==超过1500 byte，发送端分片，接收端重组

```
[root@tianyun ~]# ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
  link/ether 00:0c:29:21:4d:a6 brd ff:ff:ff:ff:ff:ff
```

分片DEMO :

```
[root@tianyun ~]# tcpdump -i eth0 -nn icmp and host 192.168.0.100 -w fragment.tcpdump
[root@tianyun ~]# ping -c1 -s 4000 192.168.0.100
PING 192.168.0.100 (192.168.0.100) 4000(4028) bytes of data.
4008 bytes from 192.168.0.100: icmp_seq=1 ttl=128 time=0.463 ms
[root@tianyun ~]# wireshark fragment.tcpdump
```

$$4000 + 8(\text{icmp}) = 4008$$

3个分片：

$$\begin{aligned} 1500 - 20 &= 1480 \\ 1500 - 20 &= 1480 \\ 1068 - 20 &= 1048 \\ 1480 + 1480 + 1048 &= 4008 \end{aligned}$$

只要是同一个IP包，分片的Identification值都一样

对方主机通过 Fragment offset (偏移量) 和 Identification (分片标识) 重组IP包

四、IP路由 (寻址)

IP分类	缺省掩码
A 1 - 127	/8
B 128 - 191	/16

C 192 – 223 /24
D 224 – 239 组播地址
E 240 – 247 保留地址

公有地址：互联网上唯一

私有地址：可以重复，将不会在互联网上被路由

A : 10.0.0.0 - 10.255.255.255 256个
B: 172.16.0.0 - 172.31.255.255 16个
C: 192.168.0.0 - 192.168.255.255 256个

私有地址和公有地址之间是不能直接访问的：

如果互联网上的用户访问的目标主机是私有地址，则会被ISP路由器丢弃
私有地址的主机访问公有地址必须要经过NAT（网络地址转换）

思考：以下两个主机能否直接通信？

主机A 172.16.30.10/24 主机B 172.16.30.20/16

广播地址：

192.168.1.255/24
255.255.255.255

广播：

```
[root@tianyun ~]# tcpdump -i eth0 -nn host 172.16.70.255 -w broadcast.tcpdump
[root@tianyun ~]# ping -b 172.16.70.255
Ethernet II, Src: 3c:97:0e:54:c5:82 (3c:97:0e:54:c5:82), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol, Src: 172.16.70.1 (172.16.70.1), Dst: 172.16.70.255 (172.16.70.255)
```

DHCP获取地址过程：

13	20.9.21814	0.0.0.0	255.255.255.255	DHCP	DHCP Discover	- Transaction ID 0xf958520e
15	21.419304	172.16.90.1	172.16.90.20	DHCP	DHCP Offer	- Transaction ID 0xf958520e
16	21.420076	0.0.0.0	255.255.255.255	DHCP	DHCP Request	- Transaction ID 0xf958520e
17	21.422129	172.16.90.1	172.16.90.20	DHCP	DHCP ACK	- Transaction ID 0xf958520e

组播：

```
[root@tianyun ~]# yum -y install keepalived  
[root@tianyun ~]# service keepalived start  
[root@tianyun ~]# tcpdump -i eth0 -nn vrrp -w mcast.tcpdumpton  
Ethernet II, Src: 3c:97:0e:54:c5:82 (3c:97:0e:54:c5:82), Dst: IPv4mcast_00:00:12 (01:00:5e:00:00:12)  
Internet Protocol, Src: 172.16.70.200 (172.16.70.200), Dst: 224.0.0.18 (224.0.0.18)
```

224.0.0.18 取决于软件设置的组播地址

组播MAC : 01:00:5E:00:00:00 ~ 01:00:5E:7F:FF:FF

具体的组播MAC是根据组播IP计算出来的

	单播	组播	广播
3层(IP)	172.16.70.200	224.0.0.18	255.255.255.255 172.16.70.255
2层(MAC)	3c:97:0e:54:c5:82	01:00:5e:00:00:12	ff:ff:ff:ff:ff:ff

ARP协议

网络层协议 : ARP

ARP : 地址解析协议，作用是将IP解析成MAC。

在TCP/IP网络环境下，每个主机都分配了一个**32位的IP地址**，这种互联网地址是在网际范围标识主机的一种逻辑地址。为了让报文在物理网路上传送，必须知道对方目的主机的物理地址。这样就存在把IP地址转换成物理地址的地址转换问题。以以太网环境为例，为了正确地向目的主机传送报文，必须把目的主机的32位IP地址转换成为**48位以太网的地址**。这就需要在互连层有一组服务将IP地址转换为相应物理地址，这组协议就是ARP协议。

RARP : 反向地址解析协议

tcpdump抓取arp包示例 :

LAB1: 同一网段

```
[root@tianyun ~]# tcpdump -i eth0 -nn host 172.16.70.200 -w arp.tcpdump
```

```
[root@tianyun ~]# ping -c1 172.16.70.200
```

产生2个ARP

产生2个ICMP

```
1 0.000000 3c:97:0e:54:c5:82 Broadcast ARP Who has 172.16.70.200? Tell 172.16.70.1
```

小结: 通过ARP广播寻问目标主机的MAC

LAB2: 不同网段

```
[root@tianyun ~]# tcpdump -i eth0 -nn host 172.16.8.100 or host 172.16.70.254 -w arp-100.tcpdump
```

```
[root@tianyun ~]# ping -c1 172.16.8.100
```

产生2个ARP

产生2个ICMP

```
1 0.000000 3c:97:0e:54:c5:82 Broadcast ARP Who has 172.16.70.254? Tell 172.16.70.1
```

小结: 通过ARP广播寻问网关的MAC

管理ARP缓存表 :

```
[root@tianyun ~]# arp
```

```
[root@tianyun ~]# arp -n
```

```
[root@tianyun ~]# ip neigh
```

```
[root@tianyun ~]# arp -d 192.168.8.109
```

arping获取MAC地址 :

```
[root@tianyun ~]# arping -l eth0 192.168.0.2
```

IP冲突 :

同一个IP有多个MAC地址

防止ARP欺骗 :

```
[root@tianyun ~]# arp -s 192.168.8.1 14:DA:E9:74:95:79
```

ICMP协议

网络层协议 : ICMP

一、ICMP协议简介 :

ICMP经常被认为是IP层的一个组成部分。它**传递差错报文**以及其他需要注意的信息。ICMP报文通常被IP层或更高层协议 (TCP或UDP) 使用。一些ICMP报文把差错报文返回给用户进程

二、ICMP首部 8字节

```
[root@tianyun ~]# ping 192.168.0.100
PING 192.168.0.100 (192.168.0.100) 56(84) bytes of data.
64 bytes from 192.168.0.100: icmp_seq=1 ttl=128 time=0.322 ms
64 bytes from 192.168.0.100: icmp_seq=2 ttl=128 time=0.489 ms
```

```
Frame 1 (98 bytes on wire, 98 bytes captured)
Ethernet II, Src: 54:04:a6:cd:9c:14 (54:04:a6:cd:9c:14), Dst: 40:16:9f:f3:48:d6 (40:16:9f:f3:48:d6)
Internet Protocol, Src: 192.168.1.1 (192.168.1.1), Dst: 192.168.1.250 (192.168.1.250)
Internet Control Message Protocol
  Type: 8 (Echo (ping) request)
  Code: 0 ()
  Checksum: 0x67b7 [correct]
  Identifier: 0x2628
  Sequence number: 1 (0x0001)
  Data (56 bytes)
    Data: 8DFF6A5300000000A7F90B000000000001011121314151617...
      [Length: 56]
```

ICMP消息的类型：

== echo request
== echo reply
== 目标主机不达
== 网络不可达
== udp的port不可达

例1： ping目标主机，会使用ICMP协议发送一个echo request， 目标主机也会使用ICMP协议回应echo reply (正常通的情况下)
例2： ping目标主机，会使用ICMP协议发送一个echo request， 自己会使用ICMP协议（从lo接口）发送Destination Host Unreachable (不通的情况下，且在同一网段)
例3： 访问目标主机未开启的UDP端口，目标主机使用ICMP发送一个destination unreachable (Port unreachable)
例4： ping -t1 -c1 172.16.30.55

说明： ip首部中有8bit TTL值，Linux默认的TTL值为64。TTL每经过一个路由器（一跳hop）会减1，如果减完后为0，该路由器会丢弃IP报文，并使用ICMP协议给源主机

发送一个Time-to-live exceeded的差错报文

...

例1：
[root@tianyun ~]# tcpdump -i eth0 -nn host 172.16.30.55 -w ping-up.tcpdump
[root@tianyun ~]# ping -c1 172.16.30.55
Type: 8 (Echo (ping) request)
Code: 0 ()

例2：
[root@tianyun ~]# tcpdump -i lo -nn icmp -w ping-down.tcpdump
[root@tianyun ~]# ping -c1 172.16.30.250
Type: 3 (Destination unreachable)
Code: 1 (Host unreachable)

例3：
[root@tianyun ~]# tcpdump -i eth0 -nn host 172.16.8.100 -w dns-icmp.tcpdump
[root@tianyun ~]# dig @172.16.8.100 www.tianyun520.com
Type: 3 (Destination unreachable)
Code: 3 (Port unreachable)

例4：
[root@tianyun ~]# tcpdump -i eth0 -nn icmp -w icmp-ttl.tcpdump
[root@tianyun ~]# ping -t1 -c1 172.16.8.100
PING 172.16.8.100 (172.16.8.100) 56(84) bytes of data.
From 172.16.70.254 icmp_seq=1 Time to live exceeded
Type: 11 (Time-to-live exceeded)
Code: 0 (Time to live exceeded in transit)

路由追踪：

```
[root@tianyun ~]# traceroute 172.16.8.100
traceroute to 172.16.8.100 (172.16.8.100), 30 hops max, 60 byte packets
1 172.16.70.254 (172.16.70.254) 1.468 ms 1.787 ms 2.025 ms
2 172.16.8.100 (172.16.8.100) 0.185 ms 0.191 ms 0.183 ms
```

扩展知识

```
[root@tianyun ~]# iptables -m icmp -h
```

```
Valid ICMP Types:
any
echo-reply (pong)
destination-unreachable
network-unreachable
host-unreachable
protocol-unreachable
port-unreachable
fragmentation-needed
source-route-failed
network-unknown
host-unknown
network-prohibited
host-prohibited
TOS-network-unreachable
TOS-host-unreachable
communication-prohibited
host-precedence-violation
precedence-cutoff
source-quench
redirect
    network-redirect
    host-redirect
    TOS-network-redirect
    TOS-host-redirect
echo-request (ping)
router-advertisement
router-solicitation
time-exceeded (ttl-exceeded)
    ttl-zero-during-transit
    ttl-zero-during-reassembly
parameter-problem
    ip-header-bad
    required-option-missing
timestamp-request
timestamp-reply
address-mask-request
address-mask-reply
```

LAB1: 通过防火墙iptables实现阻止其它主机Ping自己

```
[root@tianyun ~]# iptables -A INPUT -p icmp --icmp-type echo-request -j REJECT
[root@tianyun ~]# service iptables save
```

数据链路层协议

基本路由

路由基础

1、什么是路由？

数据从发送端到接受端选路的过程

2、如何查看当前由内核维护的路由表

```
[root@tianyun ~]# ip r  
[root@tianyun ~]# ip route show table main  
[root@tianyun ~]# ip route show table default  
[root@tianyun ~]# ip route show table local
```

3、路由原理

场景1：同一个网段的主机如何路由

场景2：不同网段的主机如何路由

SIP和DIP在整个路由过程中保持不变

SMAC和DMAC一直在变（如快递员送信的过程）

注：网络不可达，指的是没有到达目标主机的路由

注：主机不可达，有路由但可能是其它情况如目标主机DOWN

4、双网卡设置同一网段 IP 对路由的影响

场景一：

```
eth0 down 172.16.30.174/24 例如网线故障  
eth1 up 172.16.30.250/24
```

```
[root@ping ~]# ip route  
172.16.30.0/24 dev eth0 proto kernel scope link src 172.16.30.174  
172.16.30.0/24 dev eth1 proto kernel scope link src 172.16.30.250  
default via 172.16.30.254 dev eth0
```

注：多个网卡不应设置同一网段的IP

5、[root@tianyun ~]# ip addr add dev lo 172.16.130.100/24

```
[root@tianyun ~]# ip addr add dev lo 172.16.130.100/32 不产生路由条目，可以接收数据包
```

6、在lo接口上添加一个IP如：172.16.30.250/32，为什么能够从其它主机ping通？

```
# echo 1 > /proc/sys/net/ipv4/conf/all/arp_ignore
```

思考：需要给51台主机设置同一IP，仅其中1台主机可以响应该IP的arp请求

50主机：(有此IP，但不能声称自己有该IP)

1. 设置在未联网的接口，建议lo
2. 32位掩码
3. 忽略对其他接口的arp请求

高级路由

高级路由

一、维护路由表

```
[root@tianyun ~]# ip route help
[root@tianyun ~]# ip route list table main
[root@tianyun ~]# ip route list table local
[root@tianyun ~]# ip route list table default
[root@tianyun ~]# cat /etc/iproute2/rt_tables
255 local
254 main
253 default
0      unspec
//路由表配置文件

[root@tianyun ~]# ip route flush table local
[root@tianyun ~]# ip route add 172.16.8.0/24 via 172.16.30.252 dev eth0 //添加静态路由
[root@tianyun ~]# ip route add default via 172.16.30.254           //添加默认路由

[root@tianyun ~]# ip route del 192.168.1.0/24 dev eth0
[root@tianyun ~]# ip route show                         //查看路由表信息
[root@tianyun ~]# ip route del default
[root@tianyun ~]# ip route del
[root@tianyun ~]# ip neigh show                         //查看通信过的邻居

[root@tianyun ~]# ip route
172.16.30.0/24 dev eth0 proto kernel scope link src 172.16.30.30      //直连路由，由接口产生
10.10.10.0/24 dev eth0 proto kernel scope link src 10.10.10.20
172.16.8.0/24 via 172.16.30.252 dev eth0             //静态路由，手动添加
default via 172.16.30.254 dev eth0                   //缺省路由，手动添加
```

示例：

到达172.16.130.0/24 网关：172.16.100.4
到达172.16.50.0/24 网关：172.16.100.5
到达172.16.60.0/24 网关：172.16.200.5

二、高级路由扩展 [了解]

策略路由 policy routing
等值多路 equal cost multipath
隧道 tunneling
流量控制 tc

1. 策略路由

a. 查看当前路由规则 (策略)

```
[root@tianyun ~]# ip rule
0:          from all lookup local
32766:     from all lookup main
32767:     from all lookup default
```

第一列：优先级，数值越小，优先级越高

第三列：all

b. 添加自定义路由表

```
[root@tianyun ~]# vim /etc/iproute2/rt_tables
表ID    表名字
255    local
254    main
253    default
0      unspec
10     table1
20     table2
//路由表配置文件
```

c. 添加默认路由

```
[root@tianyun ~]# ip route add default via 192.168.1.254 dev eth1 table table1  
[root@tianyun ~]# ip route add default via 192.168.3.254 dev eth3 table table2  
[root@tianyun ~]# ip route list table table1  
[root@tianyun ~]# ip route list table table2
```

d. 添加路由策略

场景一：把两个网段的网络流量分割

```
[root@tianyun ~]# ip rule add from 192.168.0.0/24 table table1  
[root@tianyun ~]# ip rule add from 192.168.2.0/24 table table2  
[root@tianyun ~]# ip rule show
```

场景二：把一个网段的网络按地址范围流量分割

```
[root@tianyun ~]# iptables -t mangle -A PREROUTING -m iprange --src-range \  
192.168.0.1-192.168.0.100 -j MARK --set-mark 10  
[root@tianyun ~]# iptables -t mangle -A PREROUTING -m iprange --src-range \  
192.168.0.101-192.168.0.253 -j MARK --set-mark 20  
[root@tianyun ~]# service iptables save  
[root@tianyun ~]# ip rule add fwmark 10 table table1  
[root@tianyun ~]# ip rule add fwmark 20 table table2
```

2. 等值多路

a. 添加两个网关

```
[root@tianyun ~]# ip route add default \  
>           nexthop via 192.168.3.1 dev eth0 weight 1 \  
>           nexthop via 192.168.4.1 dev eth1 weight 1      //值越小，权重越低
```

三、ss命令

ss命令条目多的情况下，效命更高

//查看listen TCP socket

```
[root@tianyun ~]# netstat -ntl  
[root@tianyun ~]# netstat -nul  
[root@tianyun ~]# netstat -nutpl  
[root@tianyun ~]# netstat -an  
[root@tianyun ~]# netstat -an |grep :80          //查看all sockets
```

```
[root@tianyun ~]# ss -anp  
[root@tianyun ~]# ss -an  
[root@tianyun ~]# ss -h
```

//查看all sockets

```
root@tianyun:~ [root@tianyun ~]# ip route show table local  
broadcast 127.255.255.255 dev lo proto kernel scope link src 127.0.0.1  
local 172.16.100.93 dev eth0 proto kernel scope host src 172.16.100.93  
broadcast 172.16.100.255 dev eth0 proto kernel scope link src 172.16.100.93  
local 192.168.122.1 dev virbr0 proto kernel scope host src 192.168.122.1  
broadcast 192.168.122.0 dev virbr0 proto kernel scope link src 192.168.122.1  
broadcast 172.16.100.0 dev eth0 proto kernel scope link src 172.16.100.93  
broadcast 127.0.0.0 dev lo proto kernel scope link src 127.0.0.1  
broadcast 192.168.122.255 dev virbr0 proto kernel scope link src 192.168.122.1  
local 127.0.0.1 dev lo proto kernel scope host src 127.0.0.1  
local 127.0.0.0/8 dev lo proto kernel scope host src 127.0.0.1  
[root@tianyun ~]#
```

```
root@tianyun:~ [root@tianyun ~]# ip route show table main  
172.16.100.0/24 dev eth0 proto kernel scope link src 172.16.100.93  
192.168.122.0/24 dev virbr0 proto kernel scope link src 192.168.122.1  
169.254.0.0/16 dev eth0 scope link metric 1002  
default via 172.16.100.254 dev eth0  
[root@tianyun ~]#
```

WHY?

```
[root@rhel6 ~]# ip r s t local
local 192.168.122.52 dev eth0 proto kernel scope host src 192.168.122.52
local 127.0.0.0/8 dev lo proto kernel scope host src 127.0.0.1
[root@rhel6 ~]# ip route del table local local 192.168.122.52 dev eth0
[root@rhel6 ~]# ping -c1 192.168.122.1
connect: Invalid argument
[root@rhel6 ~]#
[root@rhel6 ~]# ip route s t main
192.168.122.0/24 dev eth0 proto kernel scope link src 192.168.122.52
169.254.0.0/16 dev eth0 scope link metric 1002
default via 192.168.122.1 dev eth0
```

本节作业

1. 学习《图解HTTP》
2. URL,URI
3. HTTP状态码
4. TCP 2MSL
5. 简述TCP建立连接、终止连接过程及其状态
6. 掌握使用ip route 命令添加或删除路由条目，静态路由条目
7. RHEL/CentOS静态路由表配置文件

二、高级安全

安全概述

运维面临的安全威胁

运维面临的主要安全威胁

DOS/DDOS

DoS是Denial of Service的简称，即拒绝服务，造成DoS的攻击行为被称为DoS攻击，其目的是使计算机或网络无法提供正常的服务。最常见的DoS攻击有计算机网络带宽攻击和连通性攻击。

SYN Flood

HTTP Flood

缓存、集群、CDN

缓冲区溢出攻击

缓冲区溢出是指当计算机向缓冲区内填充数据位数时超过了缓冲区本身的容量，溢出的数据覆盖在合法数据上。理想的情况是：程序会检查数据长度，而且并不允许输入超过缓冲区长度的字符。但是绝大多数程序都会假设数据长度总是与所分配的储存空间相匹配，这就为缓冲区溢出埋下隐患。操作系统所使用的缓冲区，又被称为“堆栈”，在各个操作进程之间，指令会被临时储存在“堆栈”当中，“堆栈”也会出现缓冲区溢出。

扫描

端口扫描攻击是一种常用的探测技术，攻击者可将它用于寻找他们能够成功攻击的服务。连接在网络中的所有计算机都会运行许多使用TCP或UDP端口的服务，而所提供的已定义端口达6000个以上。通常，端口扫描仅利用对端口所进行的扫描不会造成直接的损失。然而，端口扫描可让攻击者找到可用于发动各种攻击的端口。

病毒

木马

木马（Trojan）这个名字来源于古希腊传说（荷马史诗中木马计的故事，Trojan一词的特洛伊木马本意是特洛伊的，即代指特洛伊木马，也就是木马计的故事）。“木马”程序是目前比较流行的病毒文件，与一般的病毒不同，它不会自我繁殖，也并不“刻意”地去感染其他文件，它通过将自身伪装吸引用户下载执行，向施种木马者提供打开被种主机的门户，使施种者可以任意毁坏、窃取被种者的文件，甚至远程操控被种主机。木马病毒的产生严重危害着现代网络安全运行。

IP欺骗

IP地址欺骗是IP数据包为伪造的源IP地址，以便冒充其他系统或发件人的身份。这是一种黑客的攻击形式，黑客使用一台计算机上网，而借用另外一台机器的IP地址，从而冒充另外一台机器与服务器打交道。

ARP欺骗

由于局域网的网络流通不是根据IP地址进行，而是按照MAC地址进行传输。所以，那个伪造出来的MAC地址在A上被改变成一个不存在的MAC地址，这样就会造成网络不通，导致A不能Ping通C！这就是一个简单的ARP欺骗。

跨站攻击

跨站攻击，即Cross Site Script Execution(通常简写为XSS)是指攻击者利用网站程序对用户输入过滤不足，输入可以显示在页面上对其他用户造成影响的HTML代码，从而盗取用户资料、利用用户身份进行某种动作或者对访问者进行病毒侵害的一种攻击方式

SQL注入攻击

SQL注入攻击是黑客对数据库进行攻击的常用手段之一。随着B/S模式应用开发的发展，使用这种模式编写应用程序的程序员也越来越多。但是由于程序员的水平及经验也参差不齐，相当大一部分程序员在编写代码的时候，没有对用户输入数据的合法性进行判断，使应用程序存在安全隐患。用户可以提交一段数据库查询代码，根据程序返回的结果，获得某些他想得知的数据，这就是所谓的SQL Injection，即SQL注入。

中间人攻击

中间人攻击（Man-in-the-Middle Attack，简称“MITM攻击”）是一种“间接”的入侵攻击，这种攻击模式是通过各种技术手段将受入侵者控制的一台计算机虚拟放置在网络连接中的两台通信计算机之间，这台计算机就称为“中间人”。

密码攻击

0day攻击

在计算机领域中，**0day**通常是指还没有补丁的漏洞，而**0day**攻击则是指利用这种漏洞进行的攻击。提供该漏洞细节或者利用程序的人通常是该漏洞的发现者。0day漏洞的利用程序对网络安全具有巨大威胁，因此0day不但是黑客的最爱，掌握多少0day也成为评价黑客技术水平的一个重要参数。

社会工程学攻击

剖析黑客攻击的手段

一、认识黑客



黑客 (Hacker) 们建设
骇客 (Cracker) 们破坏



黑帽
白帽

二、黑客的攻击手段

1. 确定攻击目标
2. 踩点和信息搜集
3. 获得基本权限
4. 提权
5. 实施攻击
6. 留取后门程序
7. 掩盖入侵痕迹

运维安全技术概览

运维安全技术概览

物理安全

系统安全

- 文件系统
- 系统进程
- 用户安全
- 日志管理

网络安全

- 防火墙
- VPN

应用安全

- 网站服务器
- DNS服务器
- SSH服务器

IDS/IPS

物理安全

系统安全

文件系统安全

文件系统安全

1. 基本权限 rwx
2. 特殊权限 suid sgid sticky
3. 文件ACL getfacl setfacl default mask
4. 文件属性 chattr lsattr +a +i
5. mask umask权限
6. mount权限 -o

```
rw      ro
sync    async
exec   noexec
atime  noatime nodiratime
dev     nodev
suid    nosuid
auto    noauto
```

注意：合理规划权限，禁止777权限出现

示例1：防删除，防修改

```
[root@tianyun ~]# find /bin /sbin /usr/sbin /usr/bin /etc/shadow /etc/passwd /etc/pam.d -type f -exec chattr +i {} \;
```

示例2：日志文件

```
[root@tianyun ~]# chattr +a /var/log/messages /var/log/secure
```

注：日志切割要先去掉a属性，之后增加a属性

```
[root@tianyun ~]# vim /etc/logrotate.d/syslog
prerotate
    chattr -a /var/log/messages
endscript
```

...

```
postrotate
    chattr +a /var/log/messages
endscript
}
```

Sudo

Sudo

一、给普通用户提升（赋予）权限的手段

1. suid, sgid
2. Switching users with su
3. Running commands as root with sudo

二、使用sudo提升普通用户的权限

根据/etc/sudoers文件中设置，普通用户在使用sudo命令时可以以root身份或其他用户的身份运行命令

```
# vim /etc/sudoers      方法一：不推荐
# visudo                  方法二：针对sudo语法检查
```

1. sudo语法：

```
#user      MACHINE=(RUNAS_USER)          COMMANDS
root      ALL=(ALL)                      ALL
允许root用户 在任何主机上=(以任何人的身份) 执行任何命令
```

示例1：

```
jack  ALL=/sbin/ip, /sbin/fdisk
alice ALL=NOPASSWD: /sbin/ip, /sbin/fdisk
```

示例2：

```
[root@tianyun ~]# groupadd it
[root@tianyun ~]# useradd it01 -G it
[root@tianyun ~]# useradd it02 -G it
[root@tianyun ~]# id it01
uid=502(it01) gid=503(it01) 组=503(it01),502(it)
%it  ALL=NOPASSWD: /sbin/
ip
```

```
%it    ALL=NOPASSWD: /usr/sbin/useradd, /usr/sbin/userdel, /usr/bin/passwd  
%it    ALL=NOPASSWD: /usr/sbin/useradd, /usr/sbin/userdel, /usr/bin/passwd, !/usr/bin/passwd root,  
!/usr/bin/passwd root--stdin,!/usr/bin/passwd--stdin  
root
```

=====

sudo测试

```
[it01@tianyun ~]$ sudo /usr/sbin/useradd user10  
[it01@tianyun ~]$ sudo /usr/sbin/useradd user20  
[it01@tianyun ~]$ sudo /usr/sbin/userdel user20  
  
[it01@tianyun ~]$ sudo /usr/bin/passwd user10  
[it01@tianyun ~]$ sudo /usr/bin/passwd root  
[it01@tianyun ~]$ echo 123|sudo /usr/bin/passwd root --stdin  
=====
```

2. sudo Alias别名

```
## Host Aliases  
# Host_Alias FILESERVERS = fs1, fs2  
Host_Alias MAILSERVERS = smtp, smtp2  
  
## User Aliases  
User_Alias ADMINS = jsmith, mikem  
  
## Command Aliases  
## These are groups of related commands...  
## Networking  
Cmnd_Alias NETWORKING = /sbin/route, /sbin/ifconfig, /bin/ping, /sbin/dhclient, /usr/bin/net,  
/sbin/iptables, /usr/bin/rfcomm, /usr/bin/wvdial, /sbin/iwconfig, /sbin/mii-tool  
  
## Installation and management of software  
Cmnd_Alias SOFTWARE = /bin/rpm, /usr/bin/up2date, /usr/bin/yum  
  
## Services  
Cmnd_Alias SERVICES = /sbin/service, /sbin/chkconfig  
  
## Updating the locate database  
Cmnd_Alias LOCATE = /usr/bin/updatedb  
  
## Storage  
Cmnd_Alias STORAGE = /sbin/fdisk, /sbin/sfdisk, /sbin/parted, /sbin/partprobe, /bin/mount, /bin/umount
```

示例1:

alice	ALL=NOPASSWD: NETWORKING	
%it	ALL=NOPASSWD: STORAGE	使用别名
ADMINS	ALL=NOPASSWD: NETWORKING, STORAGE	使用别名

3. sudo日志

```
[root@tianyun ~]# grep '^authpriv' /etc/rsyslog.conf  
authpriv.*          /var/log/secure  
[root@tianyun ~]# service rsyslog status  
rsyslogd (pid 1587) is running...  
[root@tianyun ~]# tail -f /var/log/secure
```

=====

RHEL6 默认不允许wheel中的成员以root使用执行命令

RHEL7 默认允许wheel中的成员以root使用执行命令

=====

PAM认证

PAM

PAM(Pluggable Authentication Modules)即可插拔式认证模块，它是一种高效而且灵活的用户级别的认证方式，它也是当前Linux服务器普遍使用

的认证方式。PAM可以根据用户的网段、时间、用户名、密码等实现认证。并不是所有需要验证的服务都使用PAM来验证，如MySQL-Server就没有安装相应的PAM文件。

例如需要PAM验证的服务：本地(login、gdm、kdm)，sshd，vsftpd，samba...

```
# firefox /usr/share/doc/pam-1.1.1/html/Linux-PAM_SAG.html
```

本节要点：

1. 了解PAM的作用
2. 理解PAM工作原理
3. 掌握常见的PAM模块应用

一、PAM认证的方式

Service(进程)	→	PAM(配置文件)	→	pam_*.so	→	模块的配置文件
/usr/sbin/sshd		/etc/pam.d/sshd		/lib64/security/pam_access.so		/etc/security/access.conf
/bin/su		/etc/pam.d/su		/lib64/security/pam_limits.so		/etc/security/limits.conf
				/lib64/security/pam_rootok.so		

例如sshd service:

```
[root@tianyun ~]# ldd /usr/sbin/sshd |grep -i pam  
libpam.so.0 => /lib64/libpam.so.0 (0x00007f73f1895000)
```

```
[root@tianyun ~]# grep -i pam /etc/ssh/sshd_config  
UsePAM yes
```

```
[root@tianyun ~]# vim /etc/pam.d/sshd
```

二、PAM模块类型

PAM包括四种常见认证类型(module type)：技术面试，体检报告，人力面试，会话管理

auth 认证管理 验证使用者身份，账号和密码

account 用户管理 基于用户表、时间或密码有效期来决定是否允许访问

password 密码(口令) 认证管理 禁止用户反复尝试登录，在变更密码时进行密码复杂性控制

session 会话管理 进行日志记录，或者限制用户登录的次数，资源限制

三、PAM模块的流程控制(控制标记)

技术面试中多道题

Required (必要条件) 验证失败时仍然继续，但返回fail

用户不会知道哪里失败

Requisite (必要条件) 验证失败时则立即结束整个验证过程，返回fail

面试若不成功，马上失败，效率高

Sufficient (充分条件) 验证成功则立即返回，不再继续，否则忽略结果并继续

相当于面试中的拔高题

Optional (可选条件) 无论验证结果如何，均不会影响

通常用于session类型

Include 包含另外一个配置文件中类型相同的行

四、PAM应用示例

模块：pam_rootok.so

功能：用户UID是0，返回成功

示例：限制root切换用户也需要密码

```
[root@tianyun ~]# head -1 /etc/pam.d/su  
#auth sufficient pam_rootok.so
```

示例：sshd不需要密码登录

```
[root@tianyun ~]# head -1 /etc/pam.d/sshd
auth sufficient pam_rootok.so
=====
```

模块：**pam_access.so**

功能：访问控制，默认配置文件/etc/security/access.conf

通常作用于登录程序，如su,login,gdm,**sshd**，例如限制用户从哪些网段登录sshd

示例：不允许root从192.168.1.0/24登录sshd

```
[root@tianyun ~]# grep access.so /etc/pam.d/sshd
auth required pam_access.so
[root@tianyun ~]# vim /etc/security/access.conf
::root:192.168.1.0/24
::root:ALL EXCEPT 192.168.1.0/24
=====
```

示例：使用不同的模块配置文件

```
[root@tianyun ~]# grep access /etc/pam.d/login
auth required pam_access.so accessfile=/accessfile2
[root@tianyun ~]# grep tianyun /accessfile2
::tianyun:tty5 tty6
=====
[root@tianyun ~]# grep access.so /etc/pam.d/sshd
auth requisite pam_access.so accessfile=/accessfile1
[root@tianyun ~]# grep 110 /accessfile1
::root:ALL EXCEPT 192.168.2.110
=====
```

模块：**pam_listfile.so**

功能：基于自定义文件允许或拒绝（黑名单或白名单）

示例：vsftpd黑名单或白名单

```
[root@tianyun ~]# grep listfile /etc/pam.d/vsftpd
auth required pam_listfile.so item=user sense=deny file=/etc/vsftpd/ftpusers onerr=succeed
=====
```

示例：sshd黑名单或白名单

```
[root@tianyunt ~]# grep listfile /etc/pam.d/sshd
auth required pam_listfile.so item=user sense=allow file=/etc/ssh_users onerr=succeed
[root@tianyun ~]# echo root > /etc/ssh_users
=====
```

模块：**pam_time.so**

功能：基于时间的访问控制，默认文件/etc/security/time.conf

示例：基于时间限制sshd的访问

```
[root@tianyun ~]# grep time /etc/pam.d/sshd
account required pam_time.so
[root@tianyun ~]# grep 0800 /etc/security/time.conf
sshd;*:MoTuWeThFr0800-1100
=====
```

模块：**pam_tally2.so**

功能：登录统计

示例：实现防止对sshd暴力破解

```
[root@tianyun ~]# grep tally2 /etc/pam.d/sshd
auth required pam_tally2.so deny=2 even_deny_root root_unlock_time=60 unlock_time=60
[root@tianyun ~]# pam_tally2 --reset -u root
=====
```

模块 : pam_limits.so

功能 : 限制用户会话过程中对各种资源的使用情况。缺省情况下该模块的配置文件是</etc/security/limits.conf>
通常作用于如login,gdm,su,sshd

示例 : 限制用户oracle最大打开的文件数

```
[root@tianyun ~]# grep limits /etc/pam.d/system-auth      //默认支持
session required pam_limits.so
[root@tianyun ~]# grep limits /etc/pam.d/password-auth
session required pam_limits.so
```

```
[root@tianyun ~]# vim /etc/security/limits.conf
oracle    hard  nofile      5
[root@tianyun ~]# su - oracle
-bash: cannot make pipe for command substitution: Too many open files
-bash: redirection error: cannot duplicate fd: Invalid argument
```

示例 : 限制用户oracle最大使用CPU的时间

```
[root@tianyun ~]# vim /etc/security/limits.conf
oracle    hard  cpu        1
```

示例 : 限制用户oracle最大创建的进程数

```
[root@tianyun ~]# vim /etc/security/limits.conf
oracle    hard  nproc      5
```

```
=====
[root@tianyun ~]# vim /etc/security/limits.conf
#<item> can be one of the following:
#   - core - limits the core file size (KB)
#   - data - max data size (KB)
#   - fsize - maximum filesize (KB)
#   - memlock - max locked-in-memory address space (KB)
#   - nofile - max number of open files
#   - rss - max resident set size (KB)
#   - stack - max stack size (KB)
#   - cpu - max CPU time (MIN)
#   - nproc - max number of processes
#   - as - address space limit
#   - maxlogins - max number of logins for this user
#   - maxsyslogins - max number of logins on the system
#   - priority - the priority to run user process with
#   - locks - max number of file locks the user can hold
#   - sigpending - max number of pending signals
#   - msgqueue - max memory used by POSIX message queues (bytes)
#   - nice - max nice priority allowed to raise to
#   - rtprio - max realtime priority
=====
```

```
[root@tianyun ~]# ulimit -a
core file size          (blocks, -c) 0
data seg size            (kbytes, -d) unlimited
scheduling priority      (-e) 0
file size                (blocks, -f) unlimited
pending signals          (-i) 60874
max locked memory        (kbytes, -l) 64
max memory size          (kbytes, -m) unlimited
open files               (-n) 1024
pipe size                (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority       (-r) 0
stack size                (kbytes, -s) 10240
cpu time                 (seconds, -t) unlimited
max user processes        (-u) 60874
virtual memory             (kbytes, -v) unlimited
file locks                (-x) unlimited
[root@tianyun ~]# ulimit -n
1024
[root@tianyun ~]# ulimit -n 10240
[root@tianyun ~]# ulimit -n
10240
```

/etc/security/limits.conf

www hard nofile 10240

~/.bash_profile

ulimit -n 10240

注：PAM资源限制仅针对用户，不针对进程

如果进程以root运行，会受到PAM的限制吗？

参考部分：查看进程文件是否支持PAM认证

```
[root@teacher ~]# which vsftpd  
/usr/sbin/vsftpd  
[root@teacher ~]# which sshd  
/usr/sbin/sshd  
[root@teacher ~]# which login  
/bin/login  
[root@teacher ~]# ldd /usr/sbin/vsftpd |grep pam  
    libpam.so.0 => /lib/libpam.so.0 (0x00b67000)  
[root@teacher ~]# ldd `which sshd` |grep pam  
    libpam.so.0 => /lib/libpam.so.0 (0x00d0c000)  
[root@teacher ~]# ldd `which login` |grep pam  
    libpam.so.0 => /lib/libpam.so.0 (0x00a43000)  
    libpam_misc.so.0 => /lib/libpam_misc.so.0 (0x0050b000)  
[root@teacher ~]# which mysql  
/usr/bin/mysql  
[root@teacher ~]# ldd `which mysql` |grep pam  
=====
```

LDAP认证

LDAP基础

集中式用户认证LDAP

作用：

提供用户集中用户认证的信息

提供用户信息的查询

Lightweight Directory Access Protocol 轻量级目录访问协议

一、相关概念：

1. Directory Services 目录服务

它是一个特殊的数据库，它存储的是一些小的信息块

常见的目录服务：

电话本：电话号码 人名

DNS：主机名 IP

NIS：账户信息：用户名 密码...

LDAP：用户联系信息：国家 省 街道 邮编 邮件 电话...

账户信息：username passwd home uid gid

任何信息：例如鱼香肉丝食材原料

AD : Windows(LDAP + Kerberos)

关系型数据库：MySQL,oracle,db2,sql server 写和读都是频繁

目录服务： LDAP , AD写少，读频繁（一次写入，多次读取），优化读

2. X.500 Directory Service

DAP: 目录访问协议

LDAP: 轻量级目录访问协议

二、Schema架构：

重量级知识点：

entry , entries 条目, LDAP数据库里基本的存储单元，一堆属性的集合

attributes 属性, 每个属性有一个type类型，一个或多个values值

distinguished name DN, 唯一的区分名，用以区分条目，数据库中不能出现两相条目有相同的DN

rdn 相对区分名, dn最左边

Schema 架构, 规定数据库里条目中能存什么，以及怎么存！

object classes: 对象类, name/OID : 规定一个entry可以出现哪些attribute, 它是多个attribute集合

attribute types: 属性类型, name/OID : 规定每个属性如何去描述，例如 身高 怎么表示，是单值还是多值...

常用的属性：

dn 用来唯一的标识entry

cn 全名

sn 用户的姓

uid 登录名

userPassword

shadowLastChange

shadowMin

shadowMax

shadowWarning

loginShell

uidNumber 用户UID

gidNumber

homeDirectory

c 两位国家代码CN

o Name of an organization 组织名，一般公司名

ou Name of an organizational unit 组织单元，部门名

mail Internet e-mail address

st 省

l 区

street 街道

使用标准的Schema:

[root@tianyun ~]# rpm -qa |grep openldap

openldap-2.3.43-25.el5

openldap-devel-2.3.43-25.el5

[root@tianyun ~]# yum -y install openldap-servers

[root@tianyun ~]# cd /etc/openldap/schema/

[root@tianyun schema]# ls

corba.schema cosine.schema java.schema openldap.ldif README

core.ldif dyngroup.schema misc.schema openldap.schema redhat

core.schema inetorgperson.schema nis.schema ppolicy.schema

[root@station11 schema]# vim core.schema

objectClasses: (2.5.6.6 NAME 'person'

SUP top STRUCTURAL

MUST (sn \$ cn)

MAY (userPassword \$ telephoneNumber \$ seeAlso \$ description))

SUP top 继承top对象类，top最顶层的对象类

MUST 强制的属性 mandatory

MAY 可选的属性

ABSTRACT 抽象对象类 top

STRUCTURAL 主对象类

AUXILIARY 辅对象类

注意：在一个entry里有且只有一个主对象类STRUCTURAL(同一级别)，

可以有多个辅助对象AUXILIARY

```
[root@station11 schema]# vim nis.schema
```

```
objectclass ( 1.3.6.1.1.2.0 NAME 'posixAccount'
```

```
    DESC 'Abstraction of an account with POSIX attributes'
```

```
    SUP top AUXILIARY
```

```
    MUST ( cn $ uid $ uidNumber $ gidNumber $ homeDirectory )
```

```
    MAY ( userPassword $ loginShell $ gecos $ description ) )
```

```
objectclass ( 1.3.6.1.1.2.1 NAME 'shadowAccount'
```

```
    DESC 'Additional attributes for shadow passwords'
```

```
    SUP top AUXILIARY
```

```
    MUST uid
```

```
    MAY ( userPassword $ shadowLastChange $ shadowMin $  
          shadowMax $ shadowWarning $ shadowInactive $  
          shadowExpire $ shadowFlag $ description ) )
```

```
objectclass ( 1.3.6.1.1.2.2 NAME 'posixGroup'
```

```
    DESC 'Abstraction of a group of accounts'
```

```
    SUP top STRUCTURAL
```

```
    MUST ( cn $ gidNumber )
```

```
    MAY ( userPassword $ memberUid $ description ) )
```

人 entry

dn: id=5108241982111111

objectclasses: top

objectclasses: 生活

objectclasses: 工作 STRUCTURAL

objectClasses: person STRUCTURAL

身高 : 170

体重 : 90kg

姓名 : 张三

爱好 : 足球

爱好 : 游泳

翅膀 : 2

id : 5108241982111111

homeDirectory: /home/tom

dn: id=5108444444444444

objectclasses: 生活

身高 : 170

体重 : 90kg

姓名 : 张三

公司 : xx

ID : 5108444444444444

dn: cn=tom,dc=tianyun,dc=com

objectclasses: top

objectclass: posixAccount

cn: tom

cn: aaa tom

uid: 1000

home: /home/tom

dn: cn=tom,dc=uplooking,dc=com

objectclasses: top

objectclass: posixAccount

cn: tom

uid: 1001

home: /home/tom

三、LDIF: A standard text-based format for describing directory entries

是一个标准的基于文本格式，用来描述entry

作用：用来将条目批量导入到LDAP数据库中

```
dn: uid=zhuzhu2,ou=People,dc=tianyun,dc=com
uid: zhuzhu2
cn: zhuzhu2
sn: zhuzhu2
mail: zhuzhu2@tianyun.com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
userPassword:: e2NyeXB0fSQ2JFRrS0hRMFM4JHBWbXZvaGF0VGcySUPwdGUvOFU3WFUuUHFIUzQ
uTnkVnlnOExZWTZqUU0uV1ZUMm1sRW5nN2V5ak9JUS5HNWF3aXguRG85REF1eXRSRGdxSnZ2Uzcx
shadowLastChange: 16429
shadowMin: 0
shadowMax: 30
shadowWarning: 7
loginShell: /bin/bash
uidNumber: 532
gidNumber: 537
homeDirectory: /rhome/zhuzhu2
```

LDAP部署服务器

配置LDAPS支持

配置LDAPS支持

1. CA配置

作用：配置CA服务器，并创建CA自己的私钥（ca.key）和证书（ca.crt）

```
[root@ldap ~]# rpm -q openssl  
openssl-1.0.0-27.el6.x86_64  
[root@ldap ~]# cd /etc/pki/tls/certs/  
[root@ldap certs]# openssl genrsa -des3 -out ca.key 4096  
[root@ldap certs]# openssl req -new -x509 -days 3650 -key ca.key -out ca.crt  
Country Name (2 letter code) [XX]:CN  
State or Province Name (full name) []:BJ  
Locality Name (eg, city) [Default City]:BJ  
Organization Name (eg, company) [Default Company Ltd]:tianyun  
Organizational Unit Name (eg, section) []:it  
Common Name (eg, your name or your server's hostname) []:ca.tianyun.com  
Email Address []:
```

可选操作：共享CA的证书

```
[root@ldap ~]# yum -y install httpd  
[root@ldap ~]# cp -rf ca.crt /var/www/html  
[root@ldap ~]# service httpd restart; chkconfig httpd on
```

2. LDAP Server

作用：LDAP Server创建私钥（ldap.key），并创建证书签名请求文件（ldap.csr）

```
[root@ldap certs]# openssl genrsa -des3 -out ldap.key 4096  
[root@ldap certs]# openssl rsa -in ldap.key -out ldap.key  
[root@ldap certs]# openssl req -new -key ldap.key -out ldap.csr  
Country Name (2 letter code) [XX]:CN  
State or Province Name (full name) []:BJ  
Locality Name (eg, city) [Default City]:BJ  
Organization Name (eg, company) [Default Company Ltd]:tianyun  
Organizational Unit Name (eg, section) []:hr  
Common Name (eg, your name or your server's hostname) []:ldap.tianyun.com  
Email Address []:
```

Please enter the following 'extra' attributes
to be sent with your certificate request

A challenge password []: 不设置额外密码

An optional company name []: 不设置额外密码

3. CA

作用：CA为LDAP Server签名，并产生LDAP Server的证书（ldap.crt）

```
[root@ldap certs]# openssl x509 -req -days 365 -in ldap.csr -CA ca.crt -CAkey ca.key -set_serial 01 -out ldap.crt
```

4. LDAP Server

作用：LDAP Server最终拥有自己的私钥和证书，以及CA的证书

```
[root@ldap certs]# cp -rf ca.crt ldap.crt ldap.key /etc/openldap/certs/  
[root@ldap certs]# ls /etc/openldap/certs/  
ca.crt ldap.crt ldap.key
```

5. Client (之后所有需要使用LDAP实现认证的客户端)

ca.crt

配置LDAP Server

配置基本LDAP Server

一、规划DIT，目录信息树

suffix后缀，建议使用公司DNS域名作为整个DIT的后缀
BaseDN: dc=tianyun,dc=com
DN: ou=beijing,dc=tianyun,dc=com
DN: ou=shanghai,dc=tianyun,dc=com
DN: ou=hr,ou=beijing,dc=tianyun,dc=com
DN: ou=it,ou=beijing,dc=tianyun,dc=com

二、安装软件包

```
[root@tianyun ~]# yum -y install openldap openldap-devel openldap-clients openldap-servers migrationtools
```

三、配置openldap

1. 查看相关的文件

```
[root@station11 openldap]# ls /etc/openldap/  
certs ldap.conf schema slapd.d  
slapd.d          //ldap服务器配置文件  
ldap.conf        //ldap客户端配置文件  
schema/*         //schema文件 nis.schema, core.schema  
cacerts          //存放如CA证书，实现LDAP安全ldaps
```

```
[root@ldapserver openldap]# ls /usr/share/openldap-servers/  
DB_CONFIG.example  slapd.conf.obsolete  
DB_CONFIG.example  //数据库模板  
slapd.conf.obsolete //LDAP服务器配置文件模板
```

//复制ldap的主配置文件

```
[root@ldapserver openldap]# cp /usr/share/openldap-servers/slapd.conf.obsolete /etc/openldap/slapd.conf  
[root@ldapserver openldap]# mv /etc/openldap/slapd.d /etc/openldap/slapd.d.bak
```

//复制模板数据库文件

```
[root@ldapserver openldap]# cp /usr/share/openldap-servers/DB_CONFIG.example /var/lib/ldap/DB_CONFIG  
[root@ldapserver openldap]# chown -R ldap.ldap /var/lib/ldap/
```

2. 配置slapd.conf

```
[root@tianyun openldap]# slappasswd -s tianyun 生成管理DN口令  
{SSHA}Prmw+sqlt1oID6EXdW31Q4tXFcDsLWaJ
```

```
[root@tianyun openldap]# vim /etc/openldap/slapd.conf
```

修改或添加：

```
include      /etc/openldap/schema/core.schema  
include      /etc/openldap/schema/cosine.schema  
include      /etc/openldap/schema/inetorgperson.schema  
include      /etc/openldap/schema/nis.schema  
TLSCACertificateFile /etc/openldap/certs/ca.crt  
TLSCertificateFile /etc/openldap/certs/ldap.crt  
TLSCertificateKeyFile /etc/openldap/certs/ldap.key  
database    bdb  
suffix      "dc=tianyun,dc=com"  
rootdn     "cn=admin,dc=tianyun,dc=com"  
rootpw     {SSHA}BwICKPwAyGxBWiPNsEMi+62Mhi+OsFS2  
directory   /var/lib/ldap //后端数据目录
```

```
[root@ldapserver ~]# chown ldap.ldap /etc/openldap/certs/ca.crt  
[root@ldapserver ~]# chown ldap.ldap /etc/openldap/certs/ldap.key  
[root@ldapserver ~]# chown ldap.ldap /etc/openldap/certs/ldap.crt
```

3. 启动LDAP

```
[root@ldapserver ~]# service slapd start  
[root@ldapserver ~]# chkconfig slapd on  
[root@ldapserver ~]# ps aux |grep slapd  
ldap  10617  0.8  6.8 422960 71156 ?  Ssl 20:47 0:00 /usr/sbin/slapd -h ldap:/// -u ldap  
root  10626  0.0  0.0  4264 700 pts/1 R+ 20:48 0:00 grep slapd  
[root@tianyun openldap]# grep ldap /etc/services  
ldap 389/tcp
```

```
ldap      389/udp
ldaps    636/tcp          # LDAP over SSL
ldaps      636/udp          # LDAP over SSL
[root@station11 ~]# netstat -tnlp |grep :389
tcp      0      0 0.0.0.0:389          0.0.0.0:*      LISTEN      10617/slapd
```

4. 导入基础DN (Base DN)

利用 `migrate_base.pl`生成Base DN LDIF文件

注：如果已有Base DN LDIF文件则直接导入

```
[root@tianyun ~]# ls /usr/share/migrationtools/
migrate_aliases.pl      migrate_hosts.pl
migrate_all_netinfo_offline.sh  migrate_netgroup_byhost.pl
migrate_all_netinfo_online.sh   migrate_netgroup_byuser.pl
```

```
# cd /usr/share/migrationtools
# vim migrate_common.ph
$DEFAULT_MAIL_DOMAIN = "tianyun.com";           //邮件域
$DEFAULT_BASE = "dc=tianyun,dc=com";      //Base DN
$EXTENDED_SCHEMA = 1;                          //支持扩展Schema,可选
# ./migrate_base.pl > /tmp/base.ldif
```

导入Base DN

```
# vim /etc/openldap/ldap.conf
BASE dc=tianyun, dc=com
URI ldap://172.16.70.1           //本机IP
```

```
# ldapsearch -x
# ldapadd -x -D "cn=admin,dc=tianyun,dc=com" -w tianyun -f /tmp/base.ldif
# ldapsearch -x
```

LDAP 账号管理

LDAP账号管理

批量导入账号

删除账号信息

一、批量增加（导入）账号

1. 生成用户和组的ldif文件

```
# mkdir /rhome
# for i in `seq 10` 
> do
> useradd ldapuser$i -d /rhome/ldapuser$i
> echo "123" |passwd --stdin ldapuser$i
> done
# grep '^ldapuser' /etc/passwd > user.txt
# grep '^ldapuser' /etc/group > group.txt
# /usr/share/migrationtools/migrate_passwd.pl user.txt > /tmp/user.ldif
# /usr/share/migrationtools/migrate_group.pl group.txt > /tmp/group.ldif
```

2. 导入ldif到LDAP数据库

```
[root@tianyun migration]# vim /tmp/user.ldif
[root@tianyun migration]# vim /tmp/group.ldif

# ldapsearch -x
# ldapadd -x -D "cn=admin,dc=tianyun,dc=com" -w tianyun -f /tmp/group.ldif
# ldapadd -x -D "cn=admin,dc=tianyun,dc=com" -w tianyun -f /tmp/user.ldif
```

二、查询，默认使用的是匿名用户

```
# ldapsearch -x
# ldapsearch -x 'uid=ldapuser8' -LLL
```

三、删除用户

```
[root@tianyun ~]# ldapsearch -x 'uid=ldapuser8' -LLL
dn: uid=ldapuser8,ou=People,dc=tianyun,dc=com
uid: ldapuser8
cn: ldapuser8
sn: ldapuser8
mail: ldapuser8@tianyun.com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
userPassword:: e2NyeXB0fSQ2JDhPSDBMNfOJGZxV3pLTHJOMUhVaFl2QXNPNTAvTVoxejV3dTl
BcIZHYzdWaEpBc2F5dnjWNFpvbDhyNIRCakk2bWjwbmZYcmIJUmNrTi8wMm92cXM2dIRqVGZrazEw
shadowLastChange: 16429
shadowMin: 0
shadowMax: 30
shadowWarning: 7
loginShell: /sbin/nologin
uidNumber: 514
gidNumber: 519
homeDirectory: /rhome/ldapuser8

[root@tianyun ~]# ldapdelete -x -D "cn=admin,dc=tianyun,dc=com" -w tianyun
"uid=ldapuser8,ou=People,dc=tianyun,dc=com"
[root@tianyun ~]# ldapsearch -x 'uid=ldapuser8' -LLL
```

四、安装phpldapadmin

```
# yum -y install httpd php php-ldap
# tar xf phpldapadmin-1.2.0.4.tgz
# cp -rf phpldapadmin-1.2.0.4 /var/www/html/ldap
# service httpd start
# chkconfig httpd on
```

Linux使用LDAP验证

Linux客户端（系统）使用LDAP实现身份验证

基本PAM认证的service

ssh client --(ssh)--> sshd(ldap client) --> PAM /etc/pam.d/sshd (pam_unix.so,pam_sss.so) --> /etc/nsswitch.conf (files sss) -----
(ldap+ldaps)-----> ldap server
login (ldap client) --> PAM /etc/pam.d/login (pam_unix.so,pam_sss.so) --> /etc/nsswitch.conf (files sss) -----**(ldap+ldaps)-----> ldap server**
=====

ldap server端 :

1. 通过nfs实现对用户家目录共享 (可选)

```
[root@tianyun ~]# vim /etc/exports  
/rhome *(rw,sync)  
[root@tianyun ~]# service rpcbind restart  
[root@tianyun ~]# service nfs restart  
[root@tianyun ~]# chkconfig rpcbind on  
[root@tianyun ~]# chkconfig nfs on
```

2. autofs.ldif (可选)

```
[root@ldapserver ~]# vim autofs.ldif  
dn: nisMapName=auto.master,dc=tianyun,dc=com  
objectClass: top  
objectClass: nisMap  
nisMapName: auto.master  
  
dn: cn=/rhome,nisMapName=auto.master,dc=tianyun,dc=com  
objectClass: nisObject  
cn: /rhome  
nisMapEntry: ldap:nisMapName=auto.home,dc=tianyun,dc=com  
nisMapName: auto.master  
  
dn: nisMapName=auto.home,dc=tianyun,dc=com  
objectClass: top  
objectClass: nisMap  
nisMapName: auto.home  
  
dn: cn=/,nisMapName=auto.home,dc=tianyun,dc=com  
objectClass: nisObject  
cn: /  
nisMapEntry: -rw 192.168.8.110:/rhome/&  
nisMapName: auto.home
```

```
[root@ldapserver ~]# ldapadd -x -D "cn=admin,dc=tianyun,dc=com" -w redhat -f autofs.ldif  
adding new entry "nisMapName=auto.master,dc=tianyun,dc=com"
```

```
adding new entry "cn=/rhome,nisMapName=auto.master,dc=tianyun,dc=com"
```

```
adding new entry "nisMapName=auto.home,dc=tianyun,dc=com"
```

```
adding new entry "cn=/,nisMapName=auto.home,dc=tianyun,dc=com"
```

ldap客户端配置

1. 安装软件包

```
# yum -y install openldap openldap-clients
```

2. 查询 , 可选

```
# ldapsearch -x -h ldap.tianyun.com -b "dc=tianyun,dc=com"
```

2. 加入ldap域

a. authconfig-gtk 图形

b. authconfig-tui 文本

```
# wget http://ldap.tianyun.com/ca.crt -P /etc/openldap/cacerts  
# authconfig-tui  
# service autofs restart //第一次使用  
# ssh ldapuser7@localhost  
# su - ldapuser7
```

LDAP Settings

[*] Use TLS

Server: ldap.tianyun.com

Base DN: dc=tianyun,dc=com

Back

Ok

c. kickstart在安装时自动加入

```
auth --useshadow --passalgo=sha512 --enableldap --enableldapauth --ldapserver=ldap.tianyun.com --  
ldapbasedn=dc=tianyun,dc=com \  
--ldapcacert=http://ldap.tianyun.com/ca.crt
```

=====

无图形界面：

1. 下载ca证书到 /etc/openldap/cacerts
2. setup

=====

Apache 使用 LDAP 认证

Apache 使用 LDAP 认证

=====

client(firefox,curl) ----(http|https)--> http server auth_basic_module (ldap client) ----(ldap | ldaps)----> ldap server

基于用户的的身份认证

```
[root@tianyun ~]# mkdir /var/www/html/download  
[root@tianyun ~]# echo "ldap test" > /var/www/html/download/index.html
```

方法一：使用传统的无格式文本文件认证

```
[root@tianyun ~]# htpasswd -cm /etc/httpd/htpasswd user1  
[root@tianyun ~]# vim /etc/httpd/conf/httpd.conf  
<Directory "/var/www/html/download">
```

```
AuthType basic
AuthName "local passwd auth test"
AuthUserFile /etc/httpd/htpasswd
Require valid-user
</Directory>
```

方法二：使用LDAP实现身份认证

```
<Directory "/var/www/html/download">
    AuthName "ldap auth test"
    AuthType basic
    AuthBasicProvider ldap
    AuthLDAPUrl "ldap://192.168.0.2/dc=tianyun,dc=com"
    Require valid-user
</Directory>
```

方法三：使用LDAP实现身份认

```
[root@tianyun ~]# wget http://ldap.tianyun.com/ca.crt -P /etc/httpd
```

```
LDAPTrustedGlobalCert CA_BASE64 /etc/httpd/ca.crt
```

```
<Directory "/var/www/html/download">
    AuthName "welcome..."
    AuthType Basic
    AuthBasicProvider ldap
    AuthLDAPUrl "ldap://ldap.tianyun.com/dc=tianyun,dc=com" TLS
    Require valid-user
</Directory>
```

Nginx 使用 LDAP 认证

LDAP 客户端管理命令

LDAP客户端管理命令

```
=====
```

查询：

```
# ldapsearch -x -h 192.168.0.243 -b "dc=tianyun,dc=com"
//h 指定主机
//b 指定Base DN
//x 使用简单认证
```

```
# ldapsearch -x -D "cn=admin,dc=tianyun,dc=com" -W -h 192.168.0.243 -b "dc=tianyun,dc=com"
//D 指定DN
//W 输入密码
//w 密码
```

```
# vim /etc/openldap/ldap.conf
BASE dc=tianyun, dc=com
URI ldap://192.168.0.243
# ldapsearch -x
# ldapadd -x -D "cn=admin,dc=tianyun,dc=com" -W -f user.ldif
```

删除用户

```
# ldapsearch -x 'uid=root' -LLL
# ldapdelete -x -D "cn=admin,dc=chao,dc=com" -w tianyun 'uid=root,ou=People,dc=chao,dc=com' //dn
# ldapsearch -x 'uid=root' -LLL
```

导入新用户：

```
[root@tianyun ~]# ldapsearch -x 'uid=ldapuser10' -LLL >> new.ldif
[root@tianyun ~]# vim new.ldif
[root@tianyun ~]# ldapadd -x -D "cn=admin,dc=tianyun,dc=com" -w tianyun -f new.ldif
```

DIT 定义

DIT 目录信息树

```
dn: dc=tianyun,dc=com
dc: tianyun
objectClass: top
objectClass: domain
objectClass: domainRelatedObject
associatedDomain: tianyun.com
```

```
dn: ou=beijing,dc=tianyun,dc=com
ou: beijing
objectClass: top
objectClass: organizationalUnit
objectClass: domainRelatedObject
associatedDomain: tianyun.com
```

```
dn: ou=shanghai,dc=tianyun,dc=com
ou: shanghai
objectClass: top
objectClass: organizationalUnit
objectClass: domainRelatedObject
associatedDomain: tianyun.com
```

网络安全

端口安全 **Iptables**

iptables 基础部分

iptables 基础部分

Netfilter/Iptables

=====

一、iptables基础

iptables语法

iptables [-t 要操作的表] <操作命令> [要操作的链] [规则号码] [匹配条件] [-j 匹配后的动作]
小写 大写 大写 小写 大写

表及应用顺序

raw ---> mangle ---> nat ---> filter

常见的操作命令

-L	查看,v详细,n不反解	--line-number
-A	追加,放置最后一条	
-I	插入,默认插入成第一条	
-D	删除	
-F	清空flush	
-X	删除空的自定义链	
-P	设置默认策略	
-Z	计数器归零	

要操作的链

INPUT
OUTPUT
FORWARD
PREROUTING
POSTROUTING

基本匹配

-s 192.168.2.0/24	源地址
-d 192.168.2.1	目标地址
-p tcp upd icmp	协议
-i eth0	input 从eth0接口进入的数据包
-o eth0	output 从eth0出去的数据包
-p tcp --sport 80	源端口是80的数据包
-p tcp --dport 80	目标端口是80,必须和-p tcp udp 连用

基本动作 Target

-j ACCEPT	接受	filter
-j REJECT	拒绝	filter
-j DROP	丢弃	filter
-j LOG	记录日志	filter
-j SNAT	源地址转换	nat
-j MASQUERADE	伪装	nat
-j DNAT	目标地址转换	nat
-j MARK	标记	mangle

二、iptables基本应用

netfilter:

```
[root@tianyun ~]# iptables -F
[root@tianyun ~]# iptables -X
[root@tianyun ~]# iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
[root@tianyun ~]# iptables -A INPUT -i lo -j ACCEPT
[root@tianyun ~]# iptables -A INPUT -s 192.168.2.0/24 -j ACCEPT
[root@tianyun ~]# iptables -A INPUT -s 192.168.3.0/24 -j ACCEPT
[root@tianyun ~]# iptables -A INPUT -p tcp --dport 80 -j ACCEPT
[root@tianyun ~]# iptables -A INPUT -p tcp --dport 25 -j ACCEPT
[root@tianyun ~]# iptables -A INPUT -p tcp --dport 110 -j ACCEPT
[root@tianyun ~]# iptables -A INPUT -p tcp --dport 22 -j ACCEPT
[root@tianyun ~]# iptables -A INPUT -p tcp --dport 21 -j ACCEPT //对外开放FTP控制端口
[root@tianyun ~]# iptables -A INPUT -p tcp --dport 50000:60000 -j ACCEPT //对外开放FTP数据端口
[root@tianyun ~]# iptables -A INPUT -j REJECT
[root@tianyun ~]# service iptables save
[root@tianyun ~]# chkconfig iptables on
[root@tianyun ~]# service iptables restart
[root@tianyun ~]# iptables -vnL INPUT
```

```
#tail -2 /etc/vsftpd/vsftpd.conf  
pasv_min_port=50000  
pasv_max_port=60000
```

员工出差：

上海（上网）---拨号到---> 公司VPN服务器 ---访问---> 公司内网服务器（samba,nfs）

注：

从INPUT链进来的包包括

其它主机访问本机的数据包

本机访问其它主机时，其它主机给本机回的数据包

从OUTPUT链出去的包包括

本机访问其它主机的数据包

其它主机访问本机时，本机给其它主机回的数据包

1. 如何实现？

```
Client -----rsync----- 22/tcp Server      ACCEPT  
Client -----ssh----- 22/tcp Server      REJECT
```

2. 允许主机192.168.1.2 - 192.168.1.50访问网站？

3. 如果客户端发给服务器的第一个包只带有ACK或FIN怎么办？

4. 如果只允许在某些时间段如23:00-23:05访问sshd？

iptables高级应用

iptables高级部分

```
扩展匹配      man iptables      /MATCH EXTENSIONS  
扩展动作      man iptables      /TARGET EXTENSIONS
```

一、扩展匹配 MATCH EXTENSIONS

获得帮助：

```
[root@tianyun ~]# iptables -m icmp -h          //从后往前查看  
[root@tianyun ~]# iptables -m iprange -h         //从后往前查看
```

```
[root@tianyun ~]# iptables -F  
[root@tianyun ~]# iptables -A INPUT -j REJECT
```

-m icmp

```
[root@tianyun ~]# iptables -t filter -I INPUT -p icmp --icmp-type echo-reply -j ACCEPT      //回应
```

-m iprange

```
[root@tianyun ~]# iptables -t filter -I INPUT -m iprange --src-range 192.168.2.20-192.168.2.100 -j REJECT
```

-m multiport

```
[root@tianyun ~]# iptables -m multiport -h  
[root@tianyun ~]# iptables -t filter -I INPUT -p tcp -m multiport --dports 21,22,25,80,110 -j ACCEPT
```

-m state

跟TCP中的状态没有关系

NEW 新生态

ESTABLISHED 连接态

RELATED 衍生态

INVALID 无效态

```
[root@tianyun ~]# iptables -t filter -I INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

=====

lab1：使用状态防火墙，放行本机FTP服务

```
[root@tianyun ~]# iptables -t filter -I INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

```
[root@tianyun ~]# iptables -t filter -I INPUT -p tcp --dport 21 -j ACCEPT
```

```
[root@tianyun ~]# iptables -t filter -A INPUT -j REJECT
```

```
[root@tianyun ~]# modprobe nf_conntrack_ftp
```

```
[root@tianyun ~]# vim /etc/sysconfig/iptables-config
```

IPTABLES_MODULES="nf_conntrack_ftp"

小结：该内核模块的作用是在<连接数据端口时>，将第一次握手的数据包状态由原来的 NEW 识别成 RELATED

=====

-m tos //type of service

```
[root@tianyun ~]# iptables -F
```

```
[root@tianyun ~]# tcpdump -i eth0 -nn port 22 -vvv //抓取远程主机访问本机ssh数据包，分别于输入密码前和后观察TOS值
```

```
[root@tianyun ~]# tcpdump -i eth0 -nn port 22 -vvv //抓取远程从本机rsync或scp复制文件，分别于输入密码前和后观察TOS值
```

小结：都是使用22/tcp，但可以通过IP报文中的TOS值来区分应用

ssh : tos 0x0 0x10

scp : tos 0x0 0x8

rsync: tos 0x0 0x8

```
[root@tianyun ~]# iptables -m tos -F
```

```
[root@tianyun ~]# iptables -t filter -A INPUT -p tcp --dport 22 -m tos --tos 0x10 -j REJECT //仅拒绝客户端ssh到本机
```

```
[root@tianyun ~]# iptables -t filter -A INPUT -p tcp --dport 22 -j ACCEPT
```

=====

-m ttl //TTL是IP首部的一部分

```
[root@tianyun ~]# cat /proc/sys/net/ipv4/ip_default_ttl
```

```
[root@tianyun ~]# iptables -t filter -A FORWARD -m ttl --ttl-eq 128 -j DROP //不帮windows转发，本机作为网关
```

```
[root@tianyun ~]# iptables -t filter -A FORWARD -m ttl --ttl-gt 64 -j DROP //大于64
```

=====

-m tcp 按TCP标记匹配

Flags are: SYN ACK FIN RST URG PSH ALL NONE

```
[root@tianyun ~]# iptables -t filter -A INPUT -p tcp -m tcp --tcp-flags SYN,ACK,FIN,RST SYN --dport 80 -j ACCEPT
```

```
[root@tianyun ~]# iptables -t filter -A INPUT -p tcp --syn --dport 80 -j ACCEPT
```

--tcp-flags SYN,ACK,FIN,RST SYN

检查四个标记位SYN,ACK,FIN,RST 但只有SYN标记位才匹配

则允许三次握手中的第一次握手，等价于 --syn

=====

=====

-m limit

```
[root@tianyun ~]# iptables -F
```

实验：从客户端ping本机，观察序列号

```
[root@tianyun ~]# iptables -t filter -A INPUT -p icmp -m limit --limit 20/minute -j ACCEPT
```

```
[root@tianyun ~]# iptables -t filter -A INPUT -j REJECT
```

进入本机INPUT链的ICMP，如果匹配第一条则放行，不匹配的将被第二条拒绝，默认前5个不限

16/second

16/minute

16/hour

16/day

```
[root@tianyun ~]# iptables -t filter -A INPUT -p tcp --syn --dport 80 -m limit --limit 50/second -j ACCEPT
```

```
[root@tianyun ~]# iptables -t filter -A INPUT -j REJECT
```

=====

-m connlimit 限同一IP最大连接数

```
[root@tianyun ~]# iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

```
[root@tianyun ~]# iptables -A INPUT -p tcp --syn --dport 22 -m connlimit ! --connlimit-above 2 -j ACCEPT
```

//仅允许每个客户端有两个ssh连接

=====

等价于：

```
[root@tianyun ~]# iptables -A INPUT -p tcp --syn --dport 22 -m connlimit --connlimit-above 2 -j REJECT
```

//超过两个连接拒绝

=====

```
[root@tianyun ~]# iptables -A INPUT -p tcp --syn --dport 80 -m connlimit ! --connlimit-above 100 -j ACCEPT
```

//仅允许每个客户端有100个requests

```
[root@tianyun ~]# iptables -A INPUT -j REJECT
```

-m time

```
[root@tianyun ~]# iptables -A INPUT -m time --timestart 12:00 --timestep 13:00 -j ACCEPT  
[root@tianyun ~]# iptables -A INPUT -p tcp --syn --dport 22 -m time --timestart 12:00 --timestep 13:00 -j ACCEPT  
[root@tianyun ~]# iptables -A INPUT -j REJECT
```

-m comment

```
[root@tianyun ~]# iptables -A INPUT -s 172.16.130.7 -m comment --comment "cloud class" -j REJECT
```

-m mark

```
[root@tianyun ~]# iptables -t filter -A INPUT -m mark --mark 2 -j REJECT
```

二、动作扩展 TARGET EXTENSIONS

filter:

```
-j ACCEPT  
-j DROP  
-j REJECT  
-j LOG
```

nat:

```
-j SNAT  
-j MASQUERADE  
-j DNAT  
-j REDIRECT //端口重定向
```

mangle:

```
-j MARK
```

-j LOG

```
[root@tianyun ~]# grep 'kern.*' /etc/rsyslog.conf  
kern.*          /var/log/kernel.log  
[root@tianyun ~]# service rsyslog restart
```

```
[root@tianyun ~]# iptables -j LOG -h  
[root@tianyun ~]# iptables -t filter -A INPUT -p tcp --syn --dport 22 -j LOG --log-prefix " tianyun_ssh "  
[root@tianyun ~]# iptables -t filter -A INPUT -p tcp --syn --dport 22 -j ACCEPT  
[root@tianyun ~]# iptables -t filter -A INPUT -j REJECT
```

-j REJECT

当访问一个未开启的TCP端口时，应该返回一个带有RST标记的数据包

当访问一个开启的TCP端口，但被防火墙REJECT，结果返回port xxx unreachable

```
[root@tianyun ~]# iptables -j REJECT -h  
[root@tianyun ~]# iptables -t filter -A INPUT -p tcp --dport 22 -j REJECT --reject-with tcp-reset //返回一个自定义消息类型
```

-j MARK

```
[root@tianyun ~]# iptables -t mangle -L  
[root@tianyun ~]# iptables -j MARK -h  
[root@tianyun ~]# iptables -t mangle -A PREROUTING -s 192.168.2.110 -j MARK --set-mark 1  
[root@tianyun ~]# iptables -t mangle -A PREROUTING -s 192.168.2.25 -j MARK --set-mark 2  
[root@tianyun ~]# iptables -t filter -A INPUT -m mark --mark 1 -j ACCEPT //按照标记匹配  
[root@tianyun ~]# iptables -t filter -A INPUT -m mark --mark 2 -j REJECT
```

NAT表：

POSTROUTING : SNAT, MASQUERADE

PRETROUTING : DNAT, REDIRECT

OUTPUT : DNAT, 针对本机

源地址转换

```
-j SNAT  
-j MASQUERADE  
Client : (192.168.2.25) -> (eth0: 192.168.2.110) NAT Server (eth1: 1.1.1.1) -> 1.1.1.10 (Internet web)  
[root@nat_server ~]# echo 1 > /proc/sys/net/ipv4/ip_forward  
[root@nat_server ~]# iptables -t nat -A POSTROUTING -s 192.168.2.0/24 -o eth1 -j SNAT --to-source 1.1.1.1  
[root@nat_server ~]# iptables -t nat -A POSTROUTING -s 192.168.2.0/24 -j MASQUERADE
```

目的地址转换

```
-j DNAT  
web server (192.168.2.25) < - (eth0: 192.168.2.110) NAT Server (eth1: 1.1.1.1) < - 1.1.1.10 (Internet client)  
[root@nat_server ~]# iptables -t nat -A PREROUTING -i eth1 -d 1.1.1.1 -p tcp --dport 80 -j DNAT --to-destination  
192.168.2.25:80  
[root@nat_server ~]# iptables -t nat -A OUTPUT -d 1.1.1.1 -p tcp --dport 80 -j DNAT --to 192.168.2.25:80 //从本机访问  
1.1.1.1:80
```

-j REDIRECT //端口转发

```
[root@nat_server ~]# iptables -t nat -A PREROUTING -s 172.16.130.0/24 -p tcp --dport 8888 -j REDIRECT --to-ports 22  
=====
```

iptables扩展应用

Netfilter/Iptables

扩展匹配 [自修] :

1. iptables库文件

//安装较新版本的iptables

```
[root@tianyun ~]# iptables -m time -h  
[root@tianyun ~]# iptables -m layer7 -h  
iptables v1.4.7: Couldn't load match `layer7':/lib64/xtables/libipt_layer7.so: cannot open shared object file: No such file or  
directory  
Try `iptables -h' or 'iptables --help' for more information.
```

2. 内核模块

//重新编译并增加kernel模块

//使用第三方的模块

```
[root@tianyun ~]# ls /lib/modules/`uname -r`/kernel/net/netfilter/ |grep time
```

xt_time.ko

```
[root@tianyun ~]# ls /lib/modules/`uname -r`/kernel/net/ipv4/netfilter/
```

作业 :

Lab: 支持L7

Lab: 透明网桥

实验 : 验证各种包的状态 : 虚拟机

```
[root@tianyun ~]# iptables -t filter -A INPUT -m state --state NEW -j LOG --log-prefix "INPUT_NEW"  
[root@tianyun ~]# iptables -t filter -A INPUT -m state --state ESTABLISHED -j LOG --log-prefix "INPUT_ESTAB"  
[root@tianyun ~]# iptables -t filter -A INPUT -m state --state RELATED -j LOG --log-prefix "INPUT RELATED"  
[root@tianyun ~]# iptables -t filter -A INPUT -m state --state INVALID -j LOG --log-prefix "INPUT_INVALID"  
=====  
[root@tianyun ~]# iptables -t filter -A OUTPUT -m state --state NEW -j LOG --log-prefix "OUTPUT_NEW"  
[root@tianyun ~]# iptables -t filter -A OUTPUT -m state --state ESTABLISHED -j LOG --log-prefix "OUTPUT_ESTAB"  
[root@tianyun ~]# iptables -t filter -A OUTPUT -m state --state RELATED -j LOG --log-prefix "OUTPUT RELATED"  
[root@tianyun ~]# iptables -t filter -A OUTPUT -m state --state INVALID -j LOG --log-prefix "OUTPUT_INVALID"  
  
[root@tianyun ~]# tail -f /var/log/kernel.log
```

lab1: 从远程主机ping

lab2: 从本机ping远程主机，能ping通

lab3: 从本机ping远程主机，未ping通

lab4: 从远程主机访问开启的sshd

lab5: 从远程主机访问未开启的sshd

lab6: 从远程主机 # nmap -sA 192.168.1.250 -p 80 //A, ACK

lab7: 从远程主机 # nmap -sF 192.168.1.250 -p 80 //F, FIN

```
[root@tianyun ~]# iptables -nL
```

```

Chain INPUT (policy ACCEPT)
target  prot opt source          destination
LOG    all  --  0.0.0.0/0      0.0.0.0/0      state NEW LOG flags 0 level 4 prefix `eth0_INPUT_NEW '
LOG    all  --  0.0.0.0/0      0.0.0.0/0      state ESTABLISHED LOG flags 0 level 4 prefix `eth0_INPUT_ESTAB '
LOG    all  --  0.0.0.0/0      0.0.0.0/0      state RELATED LOG flags 0 level 4 prefix `eth0_INPUT RELATED '
LOG    all  --  0.0.0.0/0      0.0.0.0/0      state INVALID LOG flags 0 level 4 prefix `eth0_INPUT_INVALID '
LOG    all  --  0.0.0.0/0      0.0.0.0/0      state NEW LOG flags 0 level 4 prefix `lo_INPUT_NEW '
LOG    all  --  0.0.0.0/0      0.0.0.0/0      state ESTABLISHED LOG flags 0 level 4 prefix `lo_INPUT_ESTAB '
LOG    all  --  0.0.0.0/0      0.0.0.0/0      state RELATED LOG flags 0 level 4 prefix `lo_INPUT RELATED '
LOG    all  --  0.0.0.0/0      0.0.0.0/0      state INVALID LOG flags 0 level 4 prefix `lo_INPUT_INVALID '
Chain FORWARD (policy ACCEPT)
target  prot opt source          destination
Chain OUTPUT (policy ACCEPT)
target  prot opt source          destination
LOG    all  --  0.0.0.0/0      0.0.0.0/0      state NEW LOG flags 0 level 4 prefix `eth0_OUTPUT_NEW '
LOG    all  --  0.0.0.0/0      0.0.0.0/0      state ESTABLISHED LOG flags 0 level 4 prefix `eth0_OUTPUT_ESTAB '
LOG    all  --  0.0.0.0/0      0.0.0.0/0      state RELATED LOG flags 0 level 4 prefix `eth0_OUTPUT RELATED '
LOG    all  --  0.0.0.0/0      0.0.0.0/0      state INVALID LOG flags 0 level 4 prefix `eth0_OUTPUT_INVALID '
LOG    all  --  0.0.0.0/0      0.0.0.0/0      state NEW LOG flags 0 level 4 prefix `lo_OUTPUT_NEW '
LOG    all  --  0.0.0.0/0      0.0.0.0/0      state ESTABLISHED LOG flags 0 level 4 prefix `lo_OUTPUT_ESTAB '
LOG    all  --  0.0.0.0/0      0.0.0.0/0      state RELATED LOG flags 0 level 4 prefix `lo_OUTPUT RELATED '
LOG    all  --  0.0.0.0/0      0.0.0.0/0      state INVALID LOG flags 0 level 4 prefix `lo_OUTPUT_INVALID '
=====
```

虚拟专用网络VPN

VPN基本概念1

VPN 隧道协议PPTP、L2TP、IPSec 和 SSLVPN 的区别

虚拟私有网络（VPN）隧道是通过Internet隧道技术将两个不同地理位置的网络安全的连接起来的技术。当两个网络是使用私有IP地址的私有局域网络时，它们之间是不能相互访问的，这时使用隧道技术就可以使得两个子网内的主机进行通讯。例如，VPN隧道技术经常被用于大型机构中不同办公区域子网的连接。

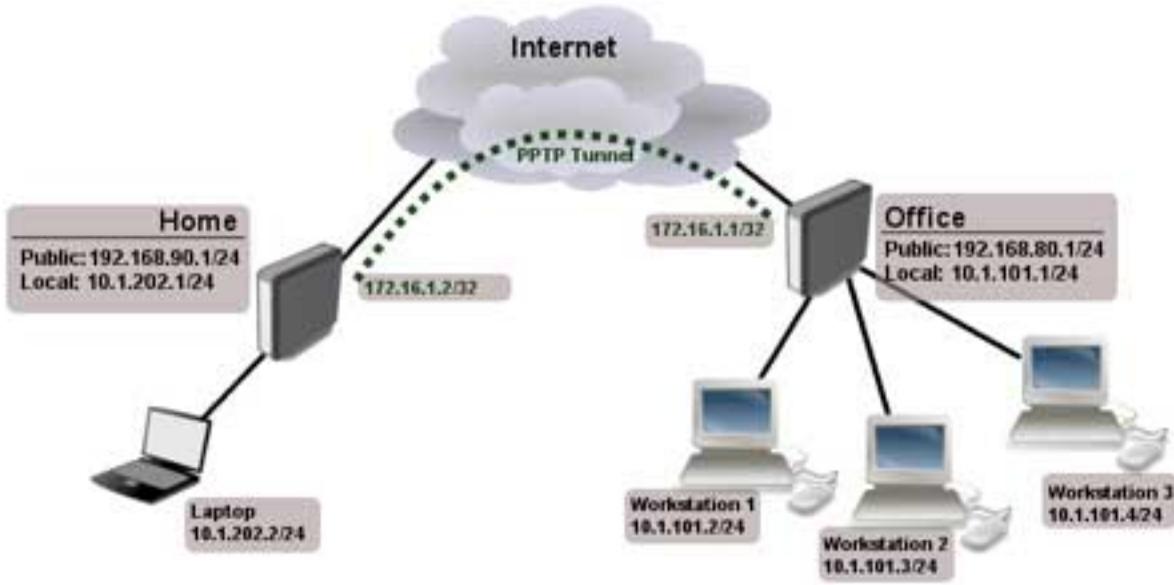
有时，使用VPN隧道仅仅是因为它很安全。服务提供商与公司会使用这样一种方式架设网络，他们将重要的服务器（如，数据库，VoIP，银行服务器）放置到一个子网内，仅仅让有权限的用户通过VPN隧道进行访问。如果需要搭建一个安全的VPN隧道，通常会选用IPsec，因为IPsec VPN隧道被多重安全层所保护。

VPN（虚拟专用网）发展至今已经不在是一个单纯的经过加密的访问隧道了，它已经融合了访问控制、传输管理、加密、路由选择、可用性管理等多种功能，并在全球的信息安全体系中发挥着重要的作用。也在网络上，有关各种VPN协议优缺点的比较是仁者见仁，智者见智，很多技术人员由于出于使用目的的考虑，包括访问控制、安全和用户简单易用，灵活扩展等各方面，权衡利弊，难以取舍；尤其在VOIP语音环境中，网络安全显得尤为重要，因此现在越来越多的网络电话和语音网关支持VPN协议。



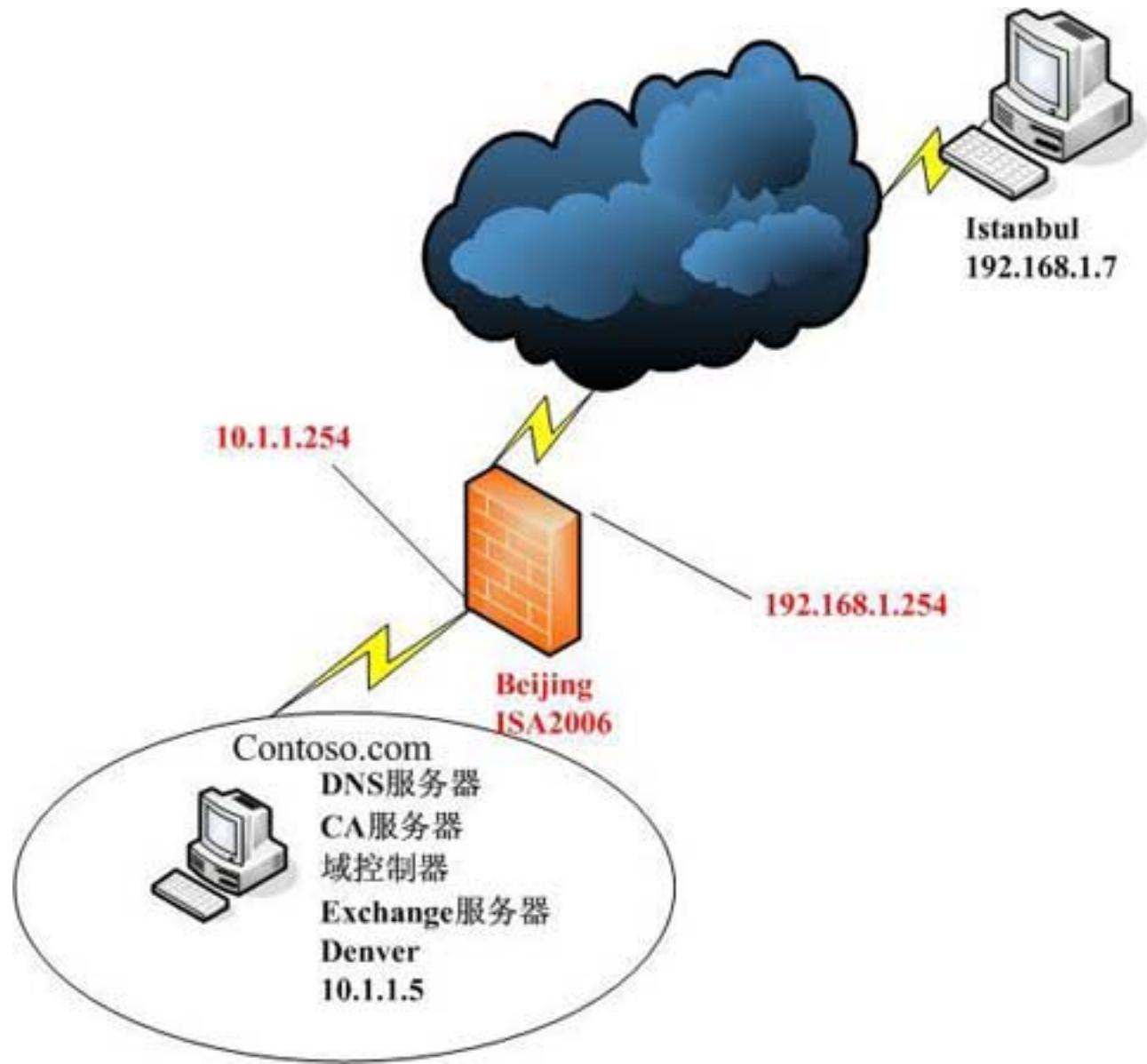
PPTP

点对点隧道协议 (PPTP) 是由包括微软和3Com等公司组成的PPTP论坛开发的一种点对点隧道协议，基于拨号使用的PPP协议使用PAP或CHAP之类的加密算法，或者使用Microsoft的点对点加密算法MPPE。其通过跨越基于 TCP/IP 的数据网络创建 VPN 实现了从远程客户端到专用企业服务器之间数据的安全传输。PPTP 支持通过公共网络(例如 Internet)建立所需的、多协议的、虚拟专用网络。PPTP 允许加密 IP 通讯，然后在要跨越公司 IP 网络或公共 IP 网络(如 Internet)发送的 IP 头中对其进行封装。



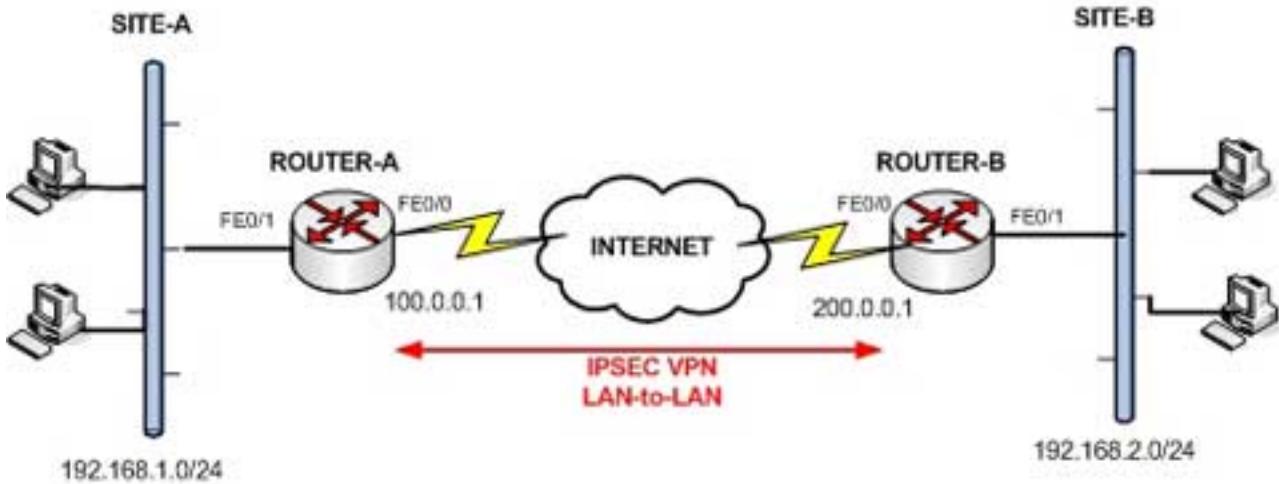
L2TP

第 2 层隧道协议 (L2TP) 是 IETF 基于 L2F (Cisco 的第二层转发协议) 开发的 PPTP 的后续版本。是一种工业标准 Internet 隧道协议，其可以为跨越面向数据包的媒体发送点到点协议 (PPP) 框架提供封装。PPTP 和 L2TP 都使用 PPP 协议对数据进行封装，然后添加附加包头用于数据在互联网络上的传输。PPTP 只能在两端点间建立单一隧道。L2TP 支持在两端点间使用多隧道，用户可以针对不同的服务质量创建不同的隧道。L2TP 可以提供隧道验证，而 PPTP 则不支持隧道验证。但是当 L2TP 或 PPTP 与 IPSEC 共同使用时，可以由 IPSEC 提供隧道验证，不需要在第 2 层协议上验证隧道使用 L2TP。PPTP 要求互联网络为 IP 网络。L2TP 只要求隧道媒介提供面向数据包的点对点的连接，L2TP 可以在 IP (使用 UDP)，帧中继永久虚拟电路 (PVCs), X.25 虚拟电路 (VCs) 或 ATM VCs 网络上使用。



IPSec

IPSec 的隧道是封装、路由与解封装的整个过程。隧道将原始数据包隐藏(或封装)在新的数据包内部。该新的数据包可能会有新的寻址与路由信息，从而使其能够通过网络传输。隧道与数据保密性结合使用时，在网络上窃听通讯的人将无法获取原始数据包数据(以及原始的源和目标)。封装的数据包到达目的地后，会删除封装，原始数据包用于将数据包路由到最终目的地。



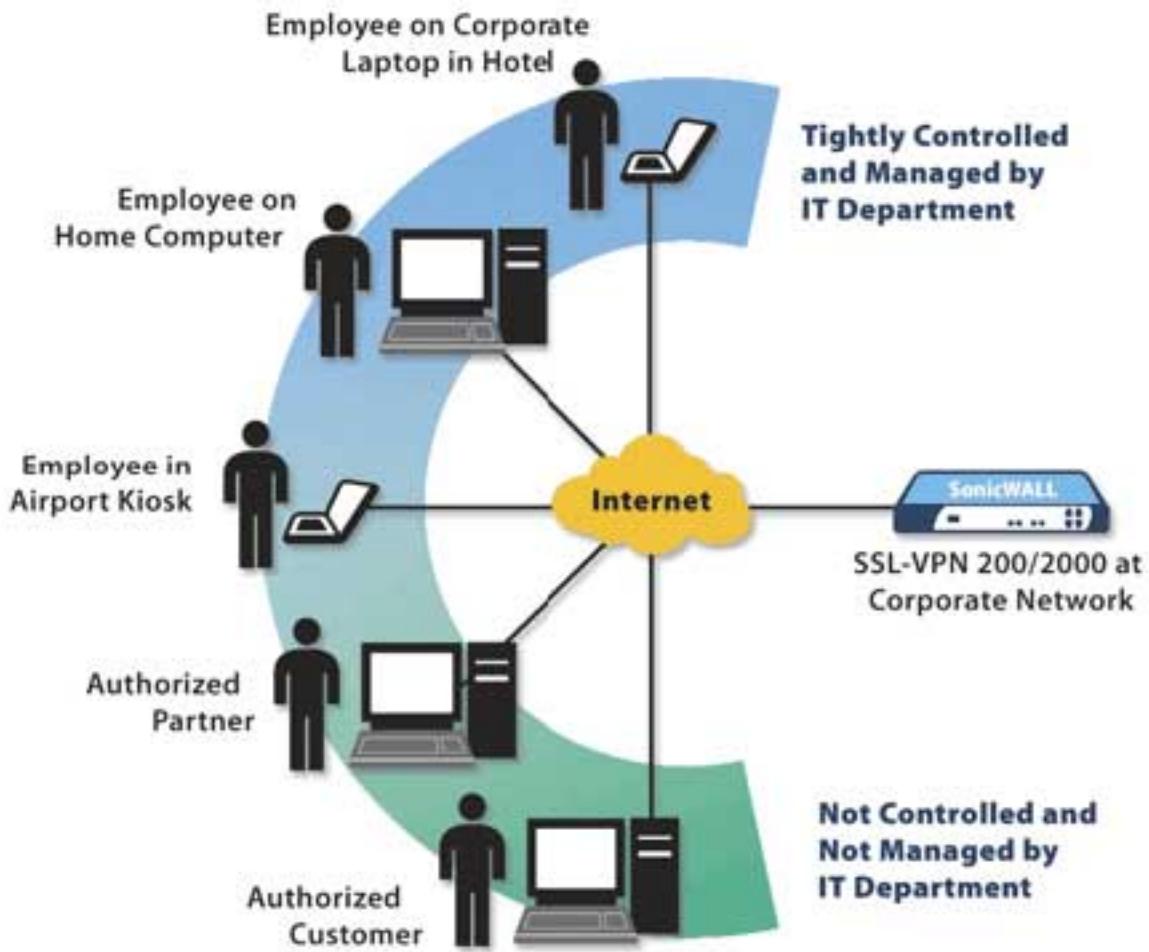
隧道本身是封装数据经过的逻辑数据路径，对原始的源和目的端，隧道是不可见的，而只能看到网络路径中的点对点连接。连接双方并不关心隧道起点和终点之间的任何路由器、交换机、代理服务器或其他安全网关。将隧道和数据保密性结合使用时，可用于提供VPN。

封装的数据包在网络中的隧道内部传输。在此示例中，该网络是 Internet。网关可以是外部 Internet 与专用网络间的周界网关。周界网关可以是路由器、防火墙、代理服务器或其他安全网关。另外，在专用网络内部可使用两个网关来保护网络中不信任的通讯。

当以隧道模式使用 IPSec 时，其只为 IP 通讯提供封装。使用 IPSec 隧道模式主要是为了与其他不支持 IPSec 上的 L2TP 或 PPTP VPN 隧道技术的路由器、网关或终端系统之间的相互操作。

SSLVPN

SSL协议提供了数据私密性、端点验证、信息完整性等特性。SSL协议由许多子协议组成，其中两个主要的子协议是握手协议和记录协议。握手协议允许服务器 和客户端在应用协议传输第一个数据字节以前，彼此确认，协商一种加密算法和密码钥匙。在数据传输期间，记录协议利用握手协议生成的密钥加密和解密后来交换 的数据。



SSL独立于应用，因此任何一个应用程序都可以享受它的安全性而不必理会执行细节。SSL置身于网络结构体系的传输层和应用层之间。此外，SSL本身就被几乎所有的Web浏览器支持。这意味着客户端不需要为了支持SSL连接安装额外的软件。这两个特征就是SSL能应用于VPN的关键点。

VPN基本概念2

什么是VPN使用的隧道技术与隧道协议？

隧道技术是一种通过公共网络的基础设施，在专用网络或专用设备之间实现加密数据通信的技术。通信的内容可以是任何通信协议的数据包。隧道协议将这些协议的数据包重新封装在新的包中发送。新的包头提供了路由信息，从而使封装的数据能够通过公共网络传递，传递时所经过的逻辑路径称为隧道。当数据包到达通信终点后，将被拆封并转发到最终目的地。隧道技术是指包括数据封装、传输和数据拆封在内的全过程。

VPN隧道所使用的公共网络可以是任何类型的通信网络。可以是Internet，也可以是企业内部网。为创建隧道，VPN的客户机和服务器必须使用相同的隧道协议，常用的隧道协议包括点对点隧道协议PPTP、第2层隧道协议L2TP和安全IP隧道模式IPSec。

按照开放系统互联OSI参考模型划分，隧道技术可以分为以第2层隧道协议为基础的技术和以第3层隧道协议为基础的技术。第2层隧道协议对应

OSI模型中的数据链路层，使用帧作为数据传输单位。PPTP和L2TP协议属于第2层隧道协议，都是将数据封装在点对点协议(PPP)的帧中通过Internet发送。第3层隧道协议对应OSI模型中的网络层，使用包作为数据传输单位。安全IP隧道模式IPSec属于第3层隧道协议，是将数据包封装在附加了IP包头的新数据包中通过IP网络传送。

点对点隧道协议(PPTP，Point-to-Point Tunneling Protocol)将点对点协议(PPP，Point-to-Point Protocol)的数据帧封装进IP数据包中，通过TCP／IP网络进行传输。PPTP可以对IP、IPX或NetBEUI数据进行加密传递。PPTP通过PPTP控制连接来创建、维护和终止一条隧道，并使用通用路由封装(GRE，Generic Routing Encapsulation)对PPP数据帧进行封装。封装前，PPP数据帧的有效载荷(有效传输数据)首先必须经过加密、压缩或是两者的混合处理。

第2层隧道协议(L2TP, Layer Two Tunneling Protocol)是PPTP和第2层转发技术(L2F，Layer Two Forward)的结合。第2层转发是Cisco公司提出的隧道技术。为了避免PPTP和L2F两种互不兼容的隧道技术在市场上彼此竞争给用户造成困惑和带来不便，Internet工程任务委员会IETF要求将两种技术结合在单一隧道协议中，并在该协议中综合PPTP和L2F两者的特点，由此产生了L2TP。L2TP协议将PPP数据帧封装后，可通过TCP／IP、X.25、帧中继或ATM等网络进行传送。L2TP可以对IP、IPX或NetBEUI数据进行加密传递。目前，仅定义了基于TCP／IP网络的L2TP。L2TP隧道协议既可用于Internet，也可用于企业内部网。

为了实现在专用或公共IP网络上的安全传输，安全IP隧道模式IPSec使用安全方式封装和加密整个IP包。它首先对IP数据包进行加密，然后将密文数据包再次封装在明文IP包内，通过网络发送到接收端的VPN服务器。VPN服务器对收到的数据包进行处理，在去除明文IP包头，对内容进行解密之后，获得原始的IP数据包，再将其路由到目标网络的接收计算机。

在这三种隧道协议中，点对点隧道协议PPTP和第2层隧道协议L2TP的优点是对用微软公司操作系统的用户来说很方便，因为微软公司已把它们作为路由软件的一部分；缺点是PPTP和L2TP将不安全的IP数据包封装在安全的IP数据包内。PPTP和L2TP适用于远程访问虚拟专用网。安全IP隧道模式IPSec的优点是它定义了一套用于认证、保护私密和数据完整性的标准协议，缺点是微软公司对IPSec的支持不够。IPSec适用于可信的局域网之间的虚拟专用网，即企业内部网VPN应用。

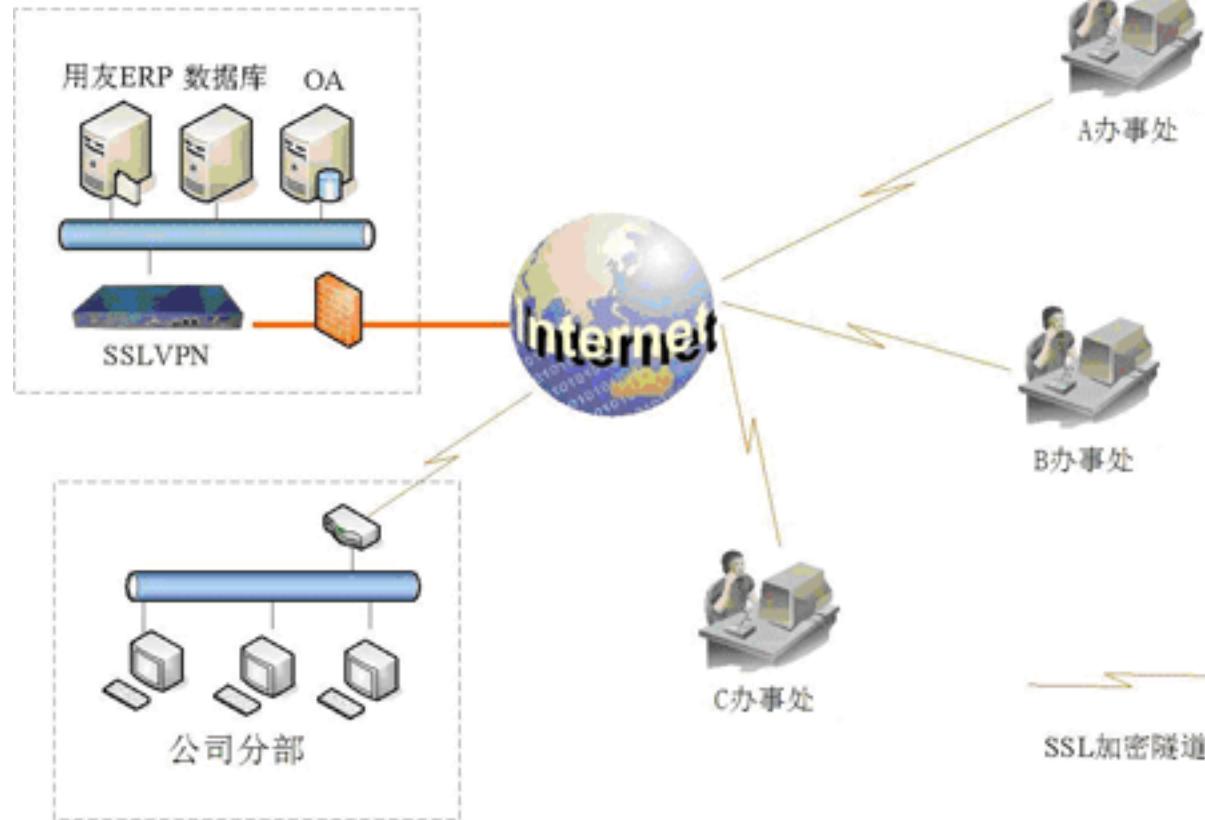
VPN基本概念3

虚拟专用网VPN

VPN解决的问题：

功能：在不安全的公共网络上建立安全的专用网络，进行数据加密传输

公司总部



VPN与隧道技术

1. 隧道协议包括：

乘客协议：被封装的协议，如PPP,SLIP

封装协议：隧道的建立、维持及断开，如L2TP、IPSec

承载协议：承载经过封装后的数据包的协议，如IP



OpenVPN

OpenVPN

典型的SSL VPN应用如OpenVPN，是一个比较好的开源软件。PPTP主要为那些经常外出移动或家庭办公的用户考虑；而OpenVPN主要是针对企业异地两地总分公司之间的VPN不间断按需连接，例如ERP在企业中的应用。

OpenVPN 允许参与建立VPN的单点使用预设的私钥，第三方证书，或者用户名/密码来进行身份验证。它大量使用了OpenSSL加密库，以及SSLv3/TLSv1 协议。OpenVPN能在Linux、xBSD、Mac OS X与Windows 2000/XP上运行。它并不是一个基于Web的VPN软件，也不与IPsec及其他VPN软件包兼容。

隧道加密

OpenVPN使用OpenSSL库加密数据与控制信息：它使用了OpenSSL的加密以及验证功能，意味着，它能够使用任何OpenSSL支持的算法。它提供了可选的数据包HMAC功能以提高连接的安全性。此外，OpenSSL的硬件加速也能提高它的性能。

验证

OpenVPN提供了多种身份验证方式，用以确认参与连接双方的身份，包括：预享私钥，第三方证书以及用户名/密码组合。预享密钥最为简单，但同时它只能用于建立点对点的VPN；基于PKI的第三方证书提供了最完善的功能，但是需要额外的精力去维护一个PKI证书体系。OpenVPN2.0后引入了用户名/口令组合的身份验证方式，它可以省略客户端证书，但是仍有一份服务器证书需要被用作加密。

网络

OpenVPN所有的通信都基于一个单一的IP端口，默认且推荐使用UDP协议通讯，同时TCP也被支持。OpenVPN连接能通过大多数的代理服务器，并且能够在NAT的环境中很好地工作。服务端具有向客户端“推送”某些网络配置信息的功能，这些信息包括：IP地址、路由设置等。

OpenVPN提供了两种虚拟网络接口：通用Tun/Tap驱动，通过它们，可以建立三层IP隧道，或者虚拟二层以太网，后者可以传送任何类型的二层以太网络数据。传送的数据可通过LZO算法压缩。IANA(Internet Assigned Numbers Authority)指定给OpenVPN的官方端口为1194。

OpenVPN 2.0以后版本每个进程可以同时管理数个并发的隧道。

OpenVPN使用通用网络协议(TCP与UDP)的特点使它成为IPsec等协议的理想替代，尤其是在ISP(Internet service provider)过滤某些特定VPN协议的情况下。在选择协议时候，需要注意2个加密隧道之间的网络状况，如有高延迟或者丢包较多的情况下，请选择 TCP协议作为底层协议，UDP协议由于存在无连接和重传机制，导致要隧道上层的协议进行重传，效率非常低下。

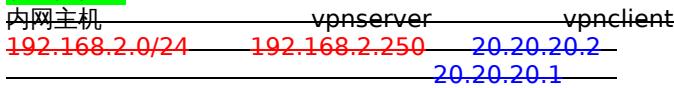
安全

OpenVPN与生俱来便具备了许多安全特性：它在用户空间运行，无须对内核及网络协议栈作修改；初始完毕后以chroot方式运行，放弃root权限；使用mlockall以防止敏感数据交换到磁盘。

OpenVPN通过PKCS#11支持硬件加密标识，如智能卡。

VPN项目环境

项目拓扑：



在内网主机上指定网关：VPN Server内网地址

```
[root@intra host ~]# ip route del
RTNETLINK answers: No such process
[root@intra host ~]# ip route add dev eth0 default via 192.168.2.250
[root@intra host ~]# ip route
192.168.2.0/24 dev eth0 proto kernel scope link src 192.168.2.25
169.254.0.0/16 dev eth0 scope link
default via 192.168.2.250 dev eth0
```

VPN Server添加内、外网接口地址

```
[root@vpnserver ~]# ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:16:3e:2e:3d:b1 brd ff:ff:ff:ff:ff:ff
    inet 192.168.2.250/24 brd 192.168.2.255 scope global eth0
[root@vpnserver ~]# ip addr show eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:16:3e:2e:3d:11 brd ff:ff:ff:ff:ff:ff
    inet 20.20.20.1/24 brd 20.20.20.255 scope global eth1
[root@vpnserver ~]# ip route
192.168.2.0/24 dev eth0 proto kernel scope link src 192.168.2.250
20.20.20.0/24 dev eth1 proto kernel scope link src 20.20.20.1
169.254.0.0/16 dev eth1 scope link
[root@vpnserver ~]#
[root@vpnserver ~]# echo 1 > /proc/sys/net/ipv4/ip_forward
```

bj-vpnserver配置

bj-vpnserver配置

安装openvpn软件

CA配置

自签名证书

为 bj-vpnserver 签发证书

为 sh-vpnclient 签发证书

bj-vpnserver配置

一、安装软件

```
[root@bj-vpnserver ~]# yum install openvpn
```

二、CA配置

```
[root@bj-vpnserver ~]# cd /usr/share/doc/openvpn-2.0.9/easy-rsa/
```

```
[root@bj-vpnserver ~]# chmod +x *
```

```
[root@bj-vpnserver ~]# vim vars
```

```
export KEY_COUNTRY=CN
```

```
export KEY_PROVINCE=BJ
```

```
export KEY_CITY=BJ
```

```
export KEY_ORG="bj-vpnserver"
```

```
export KEY_EMAIL="tianyun@126.com"
```

```
[root@bj-vpnserver easy-rsa]# source vars
```

```
[root@bj-vpnserver easy-rsa]# ./clean-all
```

```
[root@bj-vpnserver easy-rsa]# ./build-ca //生成ca私钥和证书
```

```
Common Name (eg, your name or your server's hostname) []:ca
```

三、为 bj-vpnserver 签发证书

```
[root@bj-vpnserver easy-rsa]# ./build-key-server server
```

```
Common Name (eg, your name or your server's hostname) []:bj-server
```

四、为 sh-vpnclient 签发证书

```
[root@bj-vpnserver easy-rsa]# ./build-key sh-vpnclient
```

```
Common Name (eg, your name or your server's hostname) []:sh-vpnclient
```

五、查看证书

创建密钥协商文件 迪菲·赫尔曼密钥

```
[root@bj-vpnserver easy-rsa]# ./build-dh
```

六、查看相关的证书和私钥

```
[root@bj-vpnserver easy-rsa]# ls keys/
```

```
01.pem ca.key index.txt.attr serial server.csr sh-vpnclient.csr  
02.pem dh1024.pem index.txt.attr.old serial.old server.key sh-vpnclient.key  
ca.crt index.txt index.txt.old server.crt sh-vpnclient.crt
```

七、配置bj-vpnserver

1. 检查相应的密钥文件

```
[root@bj-vpnserver keys]# pwd
```

```
/usr/share/doc/openvpn-2.0.9/easy-rsa/keys
```

```
[root@bj-vpnserver keys]# cp ca.crt server.key server.crt dh1024.pem /etc/openvpn/
```

```
[root@bj-vpnserver keys]# ls /etc/openvpn/
```

```
ca.crt dh1024.pem server.crt server.key
```

2. server.conf

```
[root@bj-vpnserver keys]# cp /usr/share/doc/openvpn-2.0.9/sample-config-files/server.conf /etc/openvpn/
```

```
[root@bj-vpnserver keys]# vim /etc/openvpn/server.conf
```

```
local 1.1.1.1
```

```
port 1194
```

```
proto udp
```

```
dev tun
```

```
ca ca.crt
```

```
cert server.crt
```

```
key server.key
```

```
dh dh1024.pem
```

```
server 10.8.0.0 255.255.255.0
```

```
ifconfig-pool-persist ip.txt
```

```
;server-bridge 10.8.0.4 255.255.255.0 10.8.0.50 10.8.0.100
```

```
push "route 192.168.10.0 255.255.255.0" //为所有客户添加到北京内网的路由
```

```
push "route 192.168.20.0 255.255.255.0" //为出差用户添加到上海内网的路由
```

```
route 192.168.20.0 255.255.255.0 //为本地设置到上海内网路由
```

```
client-config-dir ccd //客户自定义配置
```

```
;route 10.9.0.0 255.255.255.252
```

```
;learn-address ./script
```

```
;push "redirect-gateway"
```

```
;push "dhcp-option DNS 10.8.0.1"
```

```

;push "dhcp-option WINS 10.8.0.1"
client-to-client
duplicate-cn
keepalive 10 120
;tls-auth ta.key 0 # This file is secret
;cipher BF-CBC      # Blowfish (default)
;cipher AES-128-CBC # AES
;cipher DES-EDE3-CBC # Triple DES
comp-lzo
max-clients 100
user nobody
group nobody
persist-key
persist-tun
status openvpn-status.log
log   openvpn.log
verb 3
mute 20

```

3. 为上海sh-vpnclient建立配置文件

```

[root@bj-vpnserver ~]# mkdir /etc/openvpn/ccd
[root@bj-vpnserver ~]# vim /etc/openvpn/ccd/sh-vpnclient          //与common name一致
iroute 192.168.20.0 255.255.255.0                                     //允许访问client的私网
ifconfig-push 10.8.0.6 10.8.0.5

```

4. 路由转发

```

[root@bj-vpnserver ~]# vim /etc/sysctl.conf
net.ipv4.ip_forward = 1
[root@bj-vpnserver ~]# sysctl -p

```

5. iptables SNAT

```

[root@bj-vpnserver ~]# iptables -F
[root@bj-vpnserver ~]# iptables -t nat -A POSTROUTING -s 10.8.0.0/24 -o eth1 -j SNAT -to 1.1.1.1
[root@bj-vpnserver ~]# iptables -t nat -A POSTROUTING -s 10.9.0.0/24 -o eth1 -j SNAT -to 1.1.1.1
[root@bj-vpnserver ~]# service iptables save

```

6. 启动bj-vpnserver

```

[root@bj-vpnserver ~]# service openvpn start
[root@bj-vpnserver ~]# chkconfig openvpn on

```

```

[root@bj-vpnserver ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 52:54:00:ea:e7:d2 brd ff:ff:ff:ff:ff:ff
    inet 192.168.10.254/24 brd 192.168.10.255 scope global eth0
        inet6 fe80::5054:ff:fea:e7d2/64 scope link
            valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 52:54:00:37:4a:87 brd ff:ff:ff:ff:ff:ff
    inet 1.1.1.1/24 brd 1.1.1.255 scope global eth1
        inet6 fe80::5054:ff:fe37:4a87/64 scope link
            valid_lft forever preferred_lft forever
4: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN qlen 100
    link/[65534]
    inet 10.8.0.5 peer 10.8.0.2/32 scope global tun0

```

```

[root@bj-vpnserver ~]# ip r
10.8.0.2 dev tun0 proto kernel scope link src 10.8.0.1
192.168.20.0/24 via 10.8.0.2 dev tun0
10.8.0.0/24 via 10.8.0.2 dev tun0
1.1.1.0/24 dev eth1 proto kernel scope link src 1.1.1.1
192.168.10.0/24 dev eth0 proto kernel scope link src 192.168.10.254

```

sh-vpnclient配置

sh-vpnclient 配置

二、安装软件

```
[root@sh-vpnclient ~]# yum install openvpn
```

一、基本环境准备

1. 密钥

```
[root@sh-vpnclient ~]# ls /etc/openvpn/  
ca.crt sh-vpnclient.crt sh-vpnclient.key
```

三、配置sh-vpnclient

```
[root@sh-vpnclient ~]# cp /usr/share/doc/openvpn-2.0.9/sample-config-files/client.conf /etc/openvpn/  
[root@sh-vpnclient ~]# vim /etc/openvpn/client.conf  
client  
dev tun  
;dev-node MyTap  
proto udp  
remote 1.1.1.1 1194 //拨号地址  
;remote-random  
nobind  
user nobody  
group nobody  
persist-key  
persist-tun  
;http-proxy-retry # retry on connection failures  
;http-proxy [proxy server] [proxy port #]  
;mute-replay-warnings  
ca ca.crt  
cert sh-vpnclient.crt  
key sh-vpnclient.key  
;ns-cert-type server  
;tls-auth ta.key 1  
;cipher x  
comp-lzo  
verb 3  
mute 20
```

三、启动并测试

4. 路由转发

```
[root@sh-vpnclient ~]# vim /etc/sysctl.conf  
net.ipv4.ip_forward = 1  
[root@sh-vpnclient ~]# sysctl -p
```

5. iptables SNAT

```
[root@sh-vpnclient ~]# iptables -F  
[root@sh-vpnclient ~]# iptables -t nat -A POSTROUTING -o eth1 -j SNAT --to 1.1.1.2  
[root@sh-vpnclient ~]# service iptables save
```

```
[root@sh-vpnclient ~]# service openvpn start  
[root@sh-vpnclient ~]# chkconfig openvpn on
```

```
[root@sh-vpnclient ~]# ip a
```

```

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 52:54:00:ea:e7:d3 brd ff:ff:ff:ff:ff:ff
    inet 192.168.20.254/24 brd 192.168.20.255 scope global eth0
        inet6 fe80::5054:ff:fea:e7d3/64 scope link
            valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 52:54:00:8f:0c:63 brd ff:ff:ff:ff:ff:ff
    inet 1.1.1.2/24 brd 1.1.1.255 scope global eth1
        inet6 fe80::5054:ff:fe8f:c63/64 scope link
            valid_lft forever preferred_lft forever
4: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN qlen 100
    link/[65534]
    inet 10.8.0.6 peer 10.8.0.5/32 scope global tun0

```

```

[root@sh-vpnclient ~]# ip r
10.8.0.5 dev tun0 proto kernel scope link src 10.8.0.6
192.168.20.0/24 dev eth0 proto kernel scope link src 192.168.20.254
10.8.0.0/24 via 10.8.0.5 dev tun0
1.1.1.0/24 dev eth1 proto kernel scope link src 1.1.1.2
192.168.10.0/24 via 10.8.0.5 dev tun0

```

cd-client1配置—同一隧道网段

cd-client1配置

Linux:

一、<bj-vpnserver>

1. 为 cd-client1 签发证书

```

[root@bj-vpnserver ~]# cd /usr/share/doc/openvpn-2.0.9/easy-rsa/
[root@bj-vpnserver easy-rsa]# source vars
[root@bj-vpnserver easy-rsa]# ./build-key cd-client1
Common Name (eg, your name or your server's hostname) []:cd-client1

```

2. 建立配置文件

```

[root@bj-vpnserver ~]# vim /etc/openvpn/ccd/cd-client1
ifconfig-push 10.9.0.2 10.9.0.1

```

3. 重启vpn

二、配置sh-vpnclient

```

[root@bj-vpnserver ~]# yum install openvpn
[root@sh-vpnclient ~]# cp /usr/share/doc/openvpn-2.0.9/sample-config-files/client.conf /etc/openvpn/
[root@sh-vpnclient ~]# ls /etc/openvpn/
ca.crt cd-client1.crt cd-client1.key

```

```

[root@sh-vpnclient ~]# vim /etc/openvpn/client.conf
client

```

```
dev tun
;dev-node MyTap
proto udp
remote 1.1.1.1 1194 //拨号地址
;remote-random
nobind
user nobody
group nobody
persist-key
persist-tun
;http-proxy-retry # retry on connection failures
;http-proxy [proxy server] [proxy port #]
;mute-replay-warnings
ca ca.crt
cert sh-vpnclient.crt
key sh-vpnclient.key
;ns-cert-type server
;tls-auth ta.key 1
;cipher x
comp-lzo
verb 3
mute 20
```

Windows:

1. 安装openvpn
openvpn-install-2.3.3-I002-x86_64.exe

2. 证书文件 ca.crt cd-client1.crt cd-client1.key
C:\Program Files\OpenVPN\config

3. 建立配置文件
C:\Program Files\OpenVPN\config\client1.ovpn

```
client
dev tun
proto udp
remote 1.1.1.1 1194
resolv-retry infinite
nobind
persist-key
persist-tun
ca ca.crt
cert cd-client1.crt
key cd-client1.key
ns-cert-type server
cipher BF-CBC
comp-lzo
verb 3
mute 20
```

参考文件：
C:\Program Files\OpenVPN\sample-config*.ovpn

cd-client1配置—不同隧道网段

cd-client1配置

Linux:

一、<bj-ovpnserver>

1. 为 cd-client1 签发证书

```
[root@bj-vpnserver ~]# cd /usr/share/doc/openvpn-2.0.9/easy-rsa/  
[root@bj-vpnserver easy-rsa]# source vars  
[root@bj-vpnserver easy-rsa]# ./build-key cd-client1  
Common Name (eg, your name or your server's hostname) []:cd-client1
```

2. 建立配置文件

```
[root@bj-vpnserver ~]# cp /etc/openvpn/server.conf /etc/openvpn/client.conf  
[root@bj-vpnserver ~]# vim /etc/openvpn/client.conf  
local 1.1.1.1  
port 1195  
proto udp  
dev tun  
ca ca.crt  
cert server.crt  
key server.key  
dh dh1024.pem  
server 10.9.0.0 255.255.255.0  
ifconfig-persist ipp.txt  
;server-bridge 10.8.0.4 255.255.255.0 10.8.0.50 10.8.0.100  
push "route 192.168.10.0 255.255.255.0" //为所有客户添加到北京内网的路由  
push "route 192.168.20.0 255.255.255.0" //为出差用户添加到上海内网的路由  
route 192.168.20.0 255.255.255.0 //为本地设置到上海内网路由  
client-config-dir ccd //客户自定义配置  
;route 10.9.0.0 255.255.255.252  
;learn-address ./script  
;push "redirect-gateway"  
;push "dhcp-option DNS 10.8.0.1"  
;push "dhcp-option WINS 10.8.0.1"  
client-to-client  
duplicate-cn  
keepalive 10 120  
;tls-auth ta.key 0 # This file is secret  
;cipher BF-CBC # Blowfish (default)  
;cipher AES-128-CBC # AES  
;cipher DES-EDE3-CBC # Triple DES  
comp-lzo  
max-clients 100  
user nobody  
group nobody  
persist-key  
persist-tun  
status openvpn-status.log  
log openvpn.log  
verb 3  
mute 20
```

3. 重启vpn

```
[root@vpnserver ~]# ip a  
4: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN qlen 100  
    link/[65534]  
    inet 10.8.0.6 peer 10.8.0.5/32 scope global tun0  
5: tun1: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN qlen 100  
    link/[65534]  
    inet 10.9.0.1 peer 10.8.0.5/32 scope global tun0
```

二、配置sh-vpnclient

```
[root@bj-vpnserver ~]# yum install openvpn  
[root@sh-vpnclient ~]# cp /usr/share/doc/openvpn-2.0.9/sample-config-files/client.conf /etc/openvpn/  
[root@sh-vpnclient ~]# ls /etc/openvpn/  
ca.crt cd-client1.crt cd-client1.key
```

```
[root@sh-vpnclient ~]# vim /etc/openvpn/client.conf
client
dev tun
;dev-node MyTap
proto udp
remote 1.1.1.1 1195 //拨号地址
;remote-random
nobind
user nobody
group nobody
persist-key
persist-tun
;http-proxy-retry # retry on connection failures
;http-proxy [proxy server] [proxy port #]
;mute-replay-warnings
ca ca.crt
cert sh-vpnclient.crt
key sh-vpnclient.key
;ns-cert-type server
;tls-auth ta.key 1
;cipher x
comp-lzo
verb 3
mute 20
```

Windows:

1. 安装openvpn
openvpn-install-2.3.3-i002-x86_64.exe
2. 证书文件 ca.crt cd-client1.crt cd-client1.key
C:\Program Files\OpenVPN\config

3. 建立配置文件
C:\Program Files\OpenVPN\config\client1.ovpn

```
client
dev tun
proto udp
remote 1.1.1.1 1195
resolv-retry infinite
nobind
persist-key
persist-tun
ca ca.crt
cert cd-client1.crt
key cd-client1.key
;ns-cert-type server
cipher BF-CBC
comp-lzo
verb 3
mute 20
```

参考文件：
C:\Program Files\OpenVPN\sample-config*.opvn

[基于帐号方式验证](#)

[通过MySQL帐号方式验证](#)

[通过LDAP帐号方式验证](#)

[VPN Bridge模式](#)

服务安全

SELinux

SELinux基本概念

[SELinux](#)

FILE CONTEXTS PORT TYPES BOOLEANS SHARING FILES

SELinux 由美国国家安全局 (NSA) 开发

SELinux (安全增强Linux) 实现系统安全性的额外机制

SELinux的目标之一是保护用户的数据免受已泄露的系统服务的威胁！

一、基本SELinux安全性概念

DAC (Discretionary Access Control) : 自主访问控制，依据进程的所有者与文件资源的rwx权限来决定有无访问能力，有以下隐患：
root具有最高权限，如果某个进程以root运行，且被有心人士取得...

用户可以取得进程来获得文件的访问权限，比如系统上的某个目录为777...

小结：DAC针对控制的“主体”是[用户](#)

MAC (Mandatory Access Control) : 强制访问控制，依据策略规则决定进程可以访问哪些文件，优点：

即使是root，在使用不同的进程时，所能取得的权限并不一定是root，而得看当时该进程的设置而定

小结：MAC针对控制的“主体”是[进程](#)

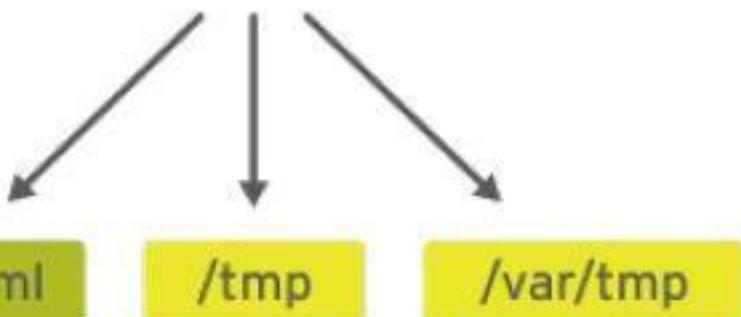
SELinux提供一些默认的策略 (Policy)，并在该策略内提供多个规则 (rule)，让用户可以选择是否启用该控制规则

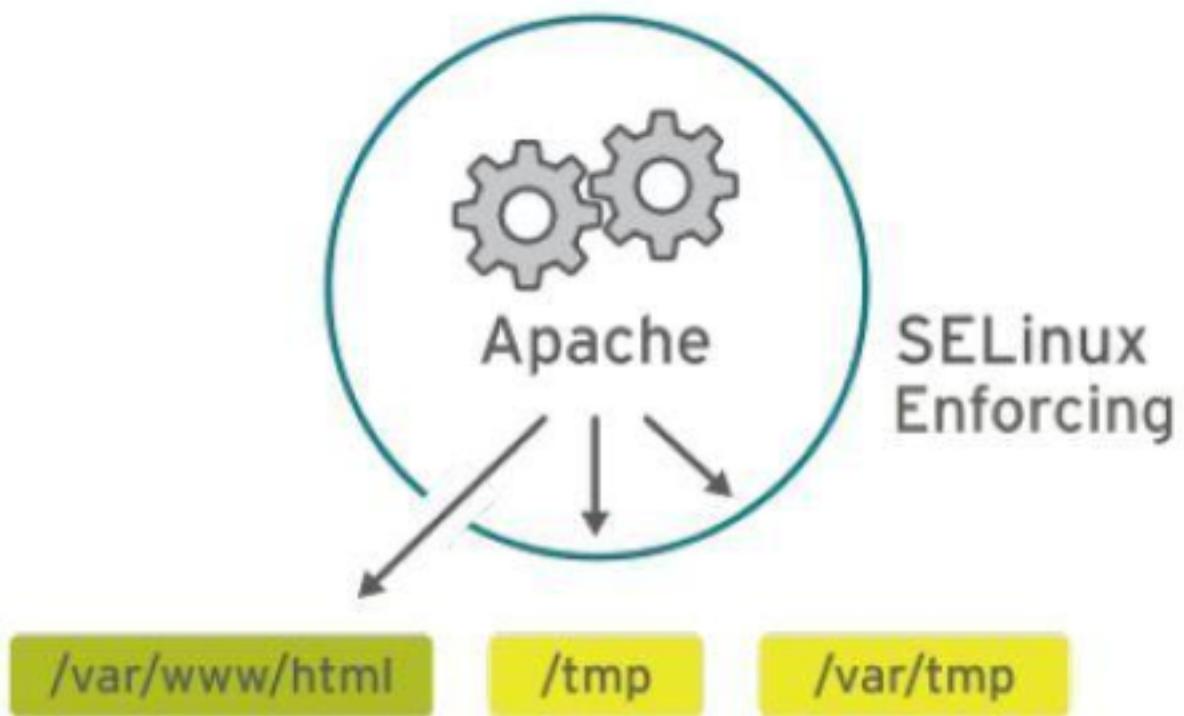
在强制访问控制的设置下，进程能够活动的空间变小了！例如规则规定httpd进程默认只能访问/var/www目录中的文件，所以，即使httpd被黑客取得了控制权，

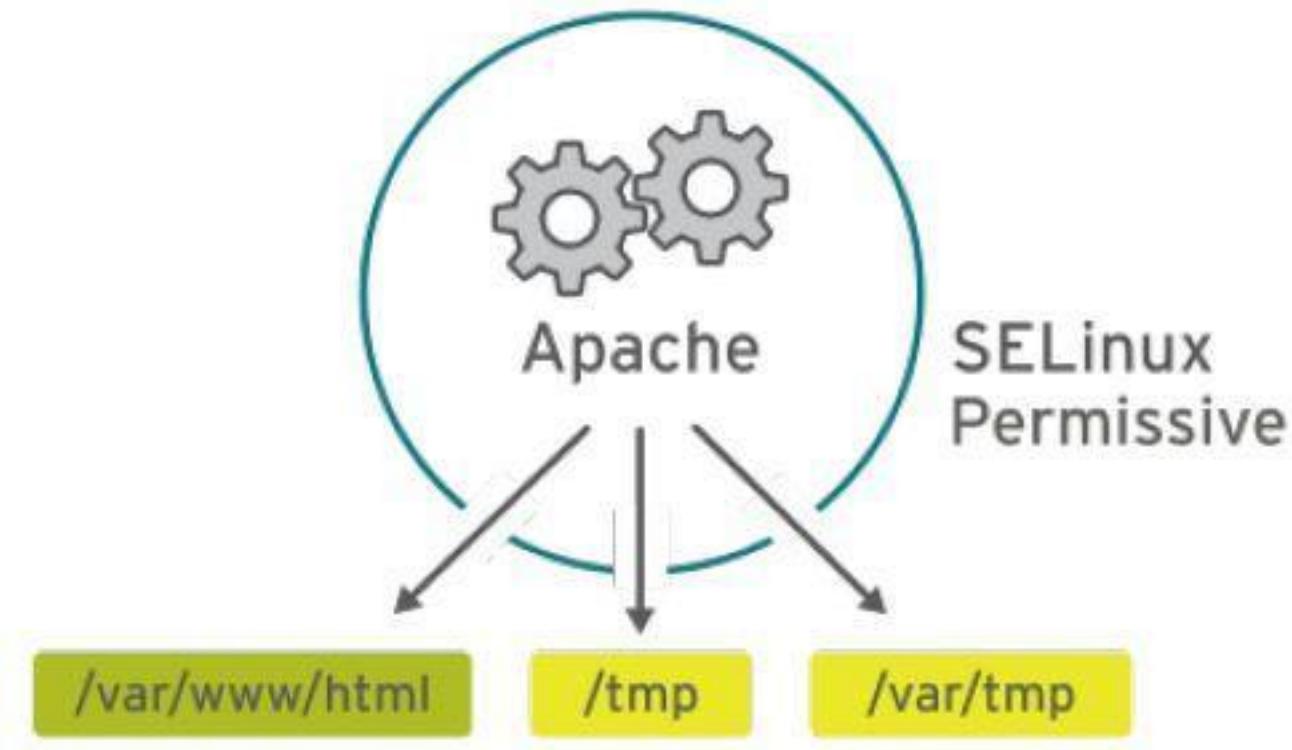
他也无权浏览/etc/shadow等重要的配置文件。



Apache







SELinux保护网络服务

使用SELinux保护网络服务

一、显示和设置SELinux模式

```
[root@tianyun ~]# vim /etc/sysconfig/selinux      //强模式制 许可模式 禁用模式
[root@tianyun ~]# getenforce                      //查看当前SELinux的状态
[root@tianyun ~]# setenforce                      //可以在强模式制 许可模式之间切换
[root@tianyun ~]# sestatus                         //列出目录SELinux使用的策略 (Policy)
```

二、安全上下文CONTEXT

1. 查看文件、进程的上下文

```
[root@tianyun ~]# ps auxZ                  //查看进程的SELinux安全上下文
```

```
[root@tianyun ~]# ps -ZC httpd  
[root@tianyun ~]# ps -ZC vsftpd  
[root@tianyun ~]# ps -ZC sshd  
[root@tianyun ~]# ls -Zd /home //查看文件的SELinux安全上下文  
[root@tianyun ~]# ls -Zd /var/www
```

2. 修改文件SELinux上下文

lab1: 父目录确定文件的初始上下文

```
[root@tianyun ~]# ls -dZ /var/www/html  
[root@tianyun ~]# touch /var/www/html/index.html  
[root@tianyun ~]# ls -Z /var/www/html/index.html
```

Lab2: cp、mv对文件上下文的影响

```
[root@tianyun ~]# touch /tmp/file1 /tmp/file2  
[root@tianyun ~]# cp /tmp/file1 /var/www/html  
[root@tianyun ~]# mv /tmp/file2 /var/www/html  
[root@tianyun ~]# ls -Z /var/www/html/file*
```

方法一 : chcon 修改文件上下文

```
[root@tianyun ~]# mkdir /virtual  
[root@tianyun ~]# chcon -R -t httpd_sys_content_t /virtual  
[root@tianyun ~]# chcon -R --reference /var/www/html /virtual
```

方法二 : semanage fcontext 修改SELinux规则

```
[root@tianyun ~]# yum provides */semanage  
[root@tianyun ~]# yum -y install policycoreutils policycoreutils-python  
[root@tianyun ~]# semanage fcontext -a -t httpd_sys_content_t "/web1(.*)?"  
[root@tianyun ~]# semanage fcontext -l |grep ^/web1  
[root@tianyun ~]# restorecon -RFvv /web1
```

```
[root@tianyun ~]# semanage fcontext -a -t httpd_sys_content_t "/web2" //针对目录  
[root@tianyun ~]# semanage fcontext -a -t httpd_sys_content_t "/web2/(.*)?" //针对目录及目录下的所有文件  
[root@tianyun ~]# semanage fcontext -a -t httpd_sys_content_t "/web2/file4" //针对文件  
[root@tianyun ~]# semanage fcontext -a -t httpd_sys_content_t "/web2/index\.txt" //针对文件
```



```
[root@tianyun ~]# semanage fcontext -a -t httpd_sys_content_t "/web2"  
[root@tianyun ~]# semanage fcontext -a -t httpd_sys_content_t "/web2/*"  
[root@tianyun ~]# semanage fcontext -a -t httpd_sys_content_t "/web2/(.*)?"
```

Lab3:

```
[root@tianyun ~]# touch /tmp/file1 /tmp/file2  
[root@tianyun ~]# cp /tmp/file1 /var/www/html  
[root@tianyun ~]# mv /tmp/file2 /var/www/html  
[root@tianyun ~]# ls -Z /var/www/html/file*  
[root@tianyun ~]# semanage fcontext -l |grep ^/var/www  
[root@tianyun ~]# restorecon -Rv /var/www
```

Lab4: 针对新目录添加上下文

```
[root@tianyun ~]# mkdir /virtual  
[root@tianyun ~]# touch /virtual/index.html  
[root@tianyun ~]# ls -Zd /virtual  
[root@tianyun ~]# semanage fcontext -a -t httpd_sys_content_t '/virtual/(.*)?'  
[root@tianyun ~]# restorecon -RFvv /virtual
```

```
[root@tianyun ~]# yum -y install selinux-policy  
[root@tianyun ~]# man -k '_selinux'  
[root@tianyun ~]# man ftpd_selinux  
[root@tianyun ~]# man samba_selinux
```

lab5: SHARING FILES

If you want to share files with multiple domains (Apache, FTP, rsync, Samba)
public_content_t
public_content_rw_t

三、SELinux布尔值

通过SELinux BOOLEANS调整策略行为

```
man -k '_selinux'  
cd /selinuxBOOLEANS
```

相当于变量

setsebool	用于修改布尔值
setsebool -P	修改SELinux策略，以永久保留修改
getsebool -a grep httpd	用于显示布尔值（当前生效的）
semanage boolean -l grep httpd_enable_homedirs	显示布尔值是否永久生效（当前生效的，永久的）

Lab6: 实现FTP匿名用户上传

1. 文件系统的权限

```
[root@tianyun ~]# mkdir /var/ftp/music
[root@tianyun ~]# setfacl -m u:ftp:rwx /var/ftp/music/
```

2. FTP服务器配置

```
anonymous_enable=YES
anon_upload_enable=YES
anon_mkdir_write_enable=YES
```

3. SELinux

文件上下文

```
[root@tianyun ~]# semanage fcontext -a -t public_content_rw_t "/var/ftp/music(/.*)?""
[root@tianyun ~]# restorecon -RFvv /var/ftp/music
```

SELinux BOOLEANS

```
[root@tianyun ~]# getsebool -a |grep ftp
allow_ftpd_anon_write --> off
ftp_home_dir --> off
[root@tianyun ~]# setsebool -P allow_ftpd_anon_write on
[root@tianyun ~]# semanage boolean -l |grep ftp
ftp_home_dir      (开 , 关) Allow ftp to read and write files in the user home directories
allow_ftpd_anon_write (开 , 开) Allow ftp servers to upload files,used for public file transfer services.
```

四、SELinux端口

Lab7: SELinux Web安全性（端口问题）

```
[root@tianyun ~]# grep '^Listen' /etc/httpd/conf/httpd.conf
Listen 8888
[root@tianyun ~]# service httpd restart
(13)Permission denied: make_sock: could not bind to address [::]:8888
(13)Permission denied: make_sock: could not bind to address 0.0.0.0:8888
no listening sockets available, shutting down
Unable to open logs
```

[FAILED]

```
[root@tianyun ~]# lsof -i TCP:8888 //检查端未被占用
[root@tianyun ~]# semanage port -l |grep http //列出当前端SELinux上下文
[root@tianyun ~]# semanage port -a -t http_port_t -p tcp 8888 //为SELinux上下文指定端口
[root@tianyun ~]# semanage port -l |grep http_port_t
http_port_t          tcp      8888, 80, 443, 488, 8008, 8009, 8443
pegasus_http_port_t    tcp      5988
[root@tianyun ~]# service httpd start
```

Lab8 : SELinux Samba安全性

布尔值

```
[root@tianyun ~]# getsebool -a |grep samba
samba_enable_home_dirs //允许共享用户HOME目录
user_samba_home_dirs //允许挂载远程CIFS文件共享并将其用作用户HOME目录
```

```
mkdir -p /shared/dir
semanage fcontext -a -t public_content_t '/shared(/.*)?'
semanage fcontext -a -t samba_share_t '/shared/dir(/.*)?'
semanage fcontext -l
restorecon -FRvv /shared
=====
```

五、监控SELinux策略冲突情况

部署SELinux日志分析工具

必须安装setroubleshoot-server软件包，才能将SELinux消息发送至/var/log/messages

setroubleshoot-server侦听/var/log/audit/audit.log中的审核信息并将简短摘要发送至/var/log/messages

该摘要包括SELinux冲突的唯一标识符（UUIDs），可用于收集更多信息

```
[root@tianyun ~]# yum -y install setroubleshoot setroubleshoot-server
[root@tianyun ~]# service rsyslog restart; chkconfig rsyslog on
[root@tianyun ~]# service auditd restart; chkconfig auditd on
```

Lab9:

```
[root@tianyun ~]# touch /root/file3  
[root@tianyun ~]# mv /root/file3 /var/www/html  
[root@tianyun ~]# service httpd restart  
[root@tianyun ~]# elinks --dump http://192.168.1.231/file3
```

//访问失败

```
[root@tianyun ~]# tail /var/log/messages
```

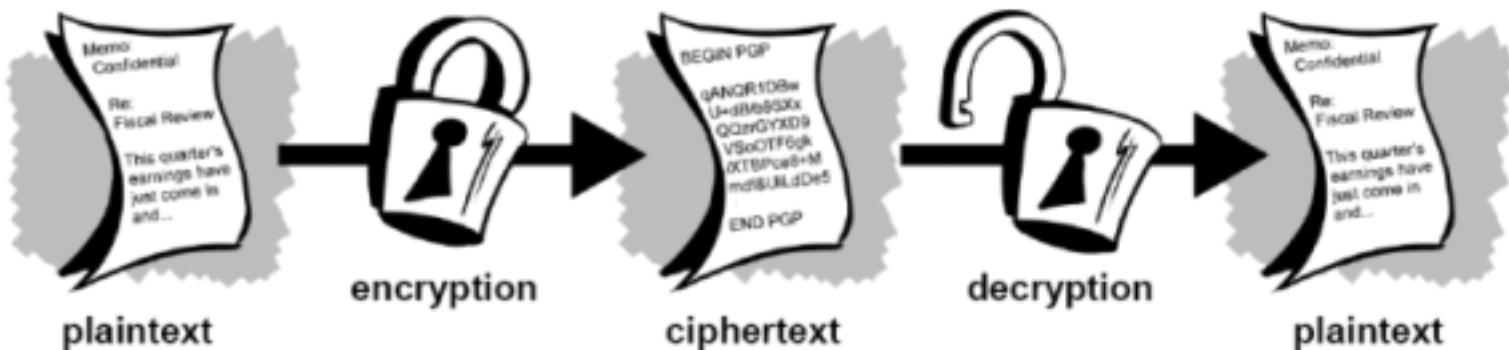
```
Nov  6 06:18:02 tianyun setroubleshoot: SELinux is preventing httpd from getattr access on the file /var/www/html/index.html.  
For complete SELinux messages. run sealert -l dca1f406-90ce-4818-b887-80de204d8194  
[root@tianyun ~]# sealert -l dca1f406-90ce-4818-b887-80de204d8194
```

数据加密技术

数据加密概述

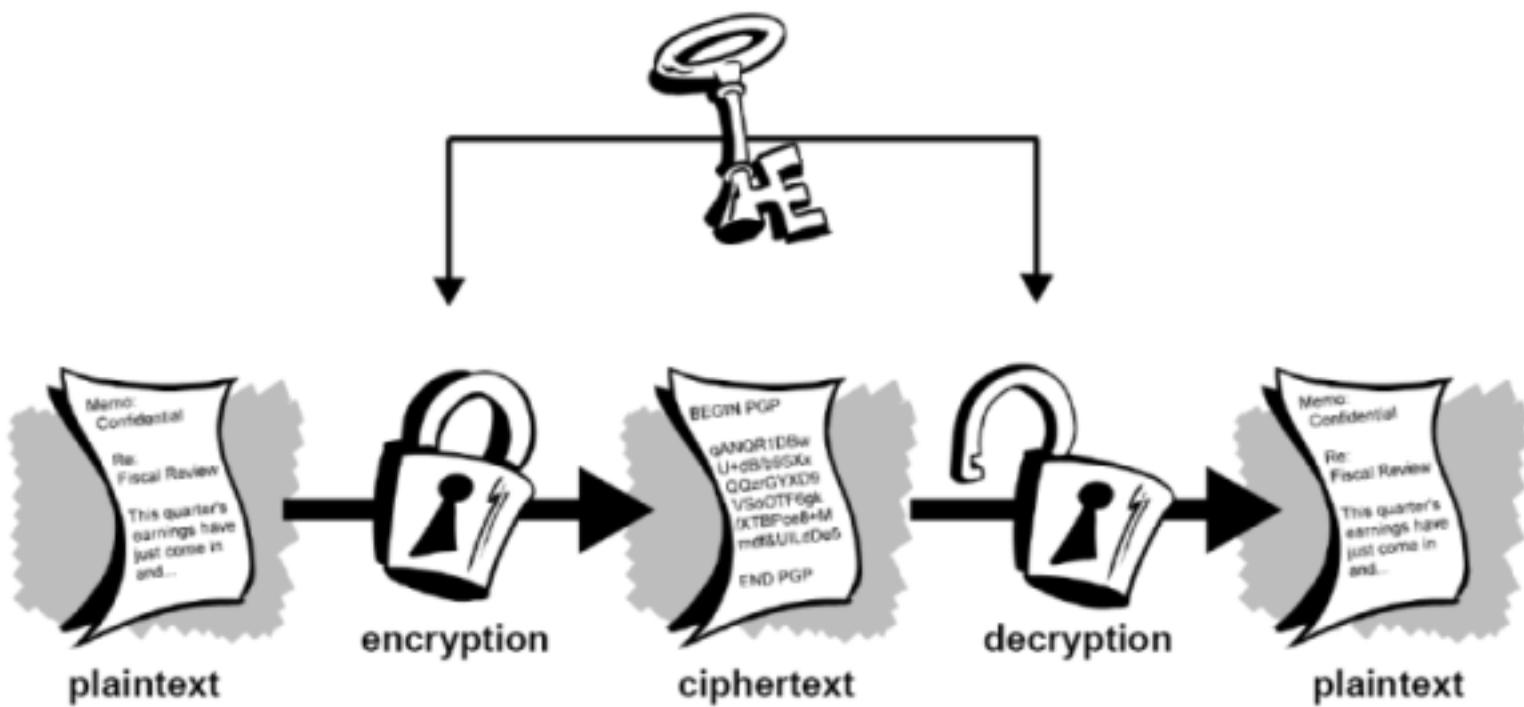
数据加密

加/解密其实就是函数变换的过程



根据加密钥匙的不同，分类两种加密体系：

第一类：传统加密/对称加密



加密和解密使用同一把钥匙

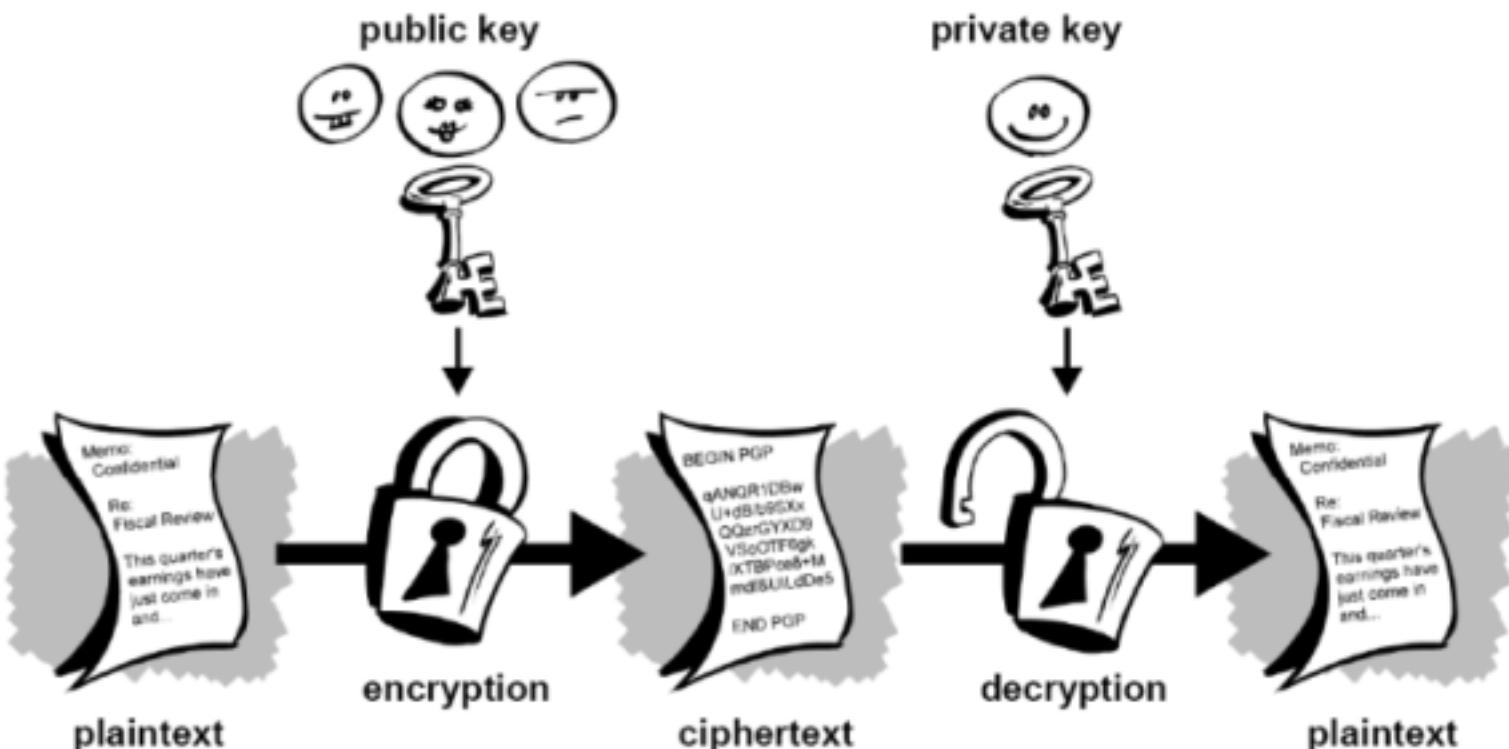
优点：效率高，加密速度快，可以加密大量的数据，几G到几十G

缺点：密钥的传递问题，特别是多个人同时通信，军队通信

对称加密算法：

DES, 3DES, RC4, RC5, RC6, BASE64, AES256

第二类：公钥加密/非对称加密

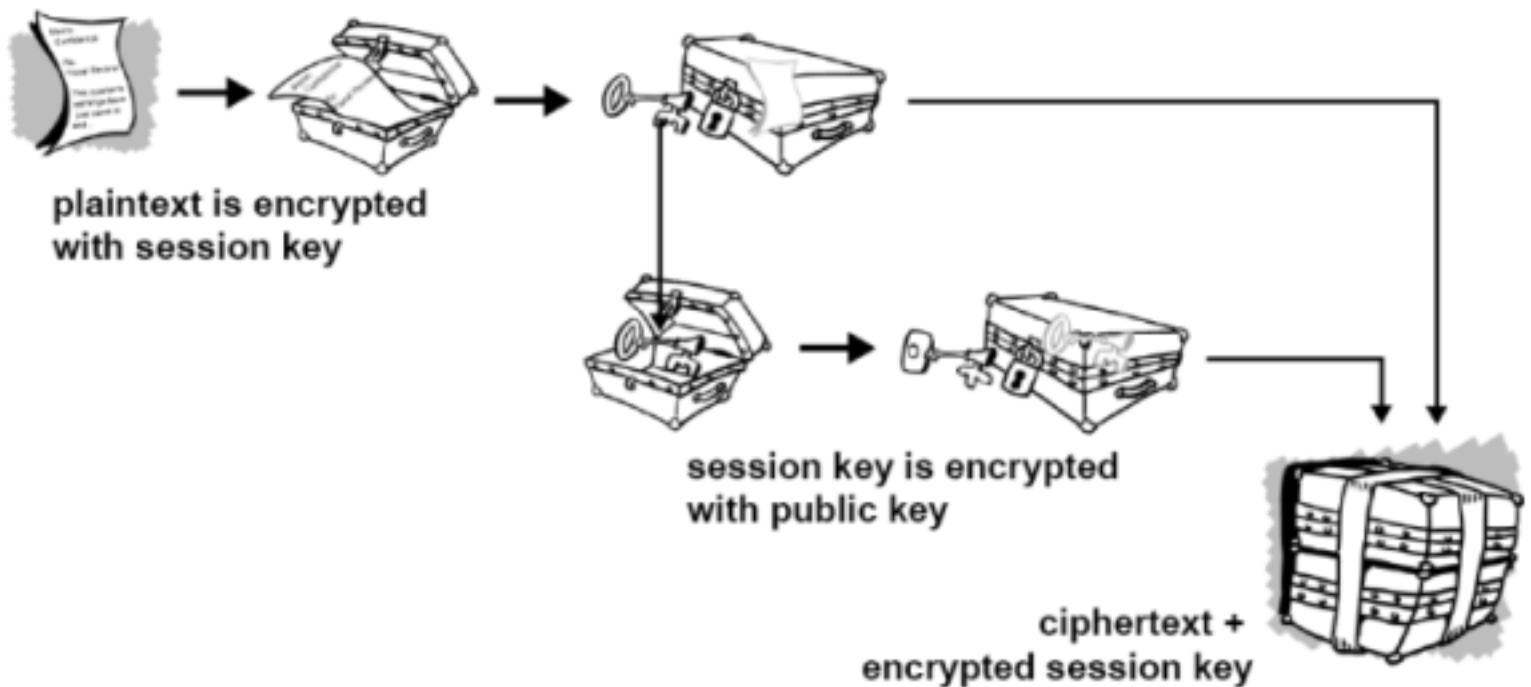


加密和解密使用不同的同钥匙，一般是公钥加密，私钥解密

优点：解决了密钥传递的问题

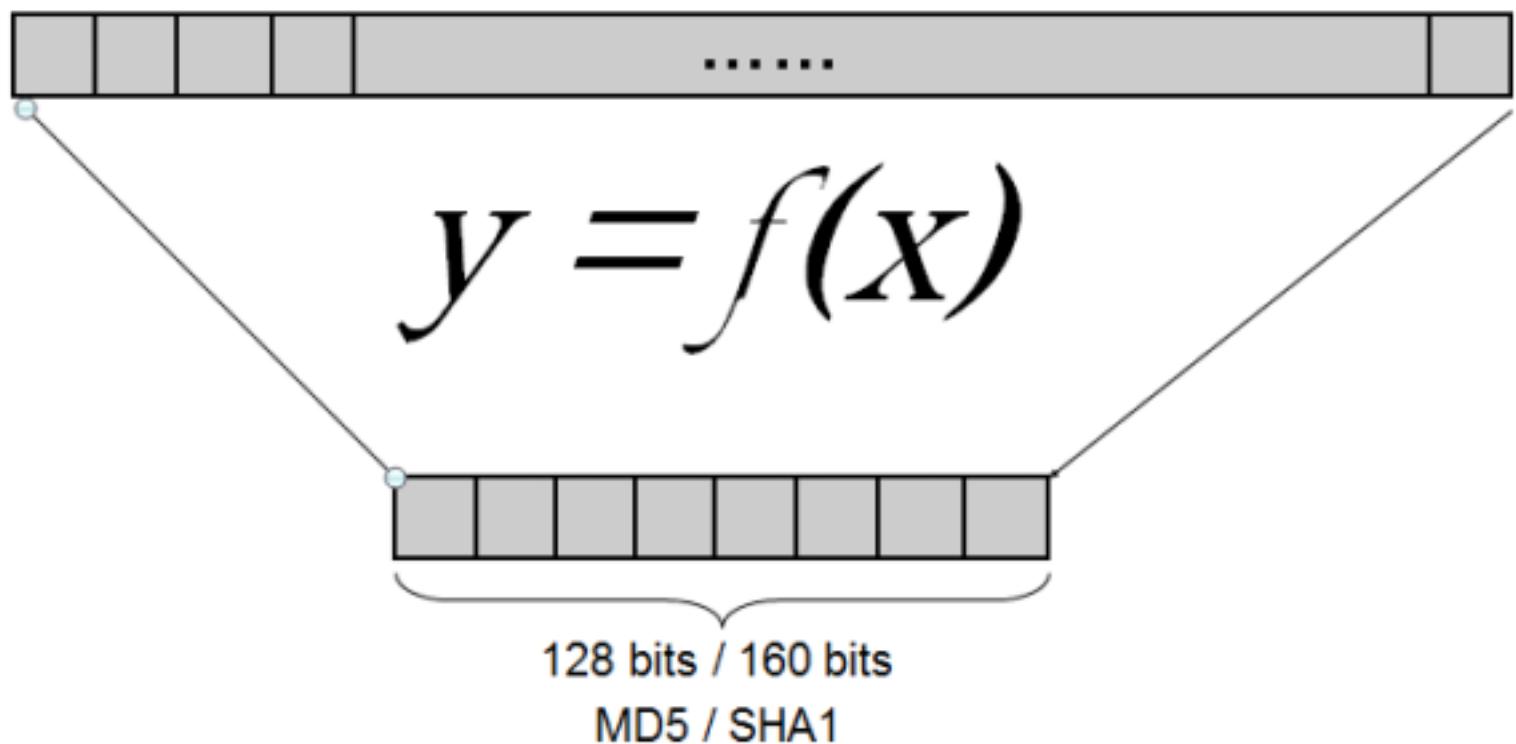
缺点：效率低，加密速度慢，比对称加密速度慢1000倍，只能加密少量数据

第三种：两者结合



非对称加密算法：
RSA, DSA

Hash算法（散列）：
MD5, SHA1



对称加密**OpenSSL**

对称加密

openssl实现对称加密

```
=====  
[root@tianyun ~]# openssl enc -des3 -in /etc/passwd -out /home/passwd.enc  
[root@tianyun ~]# openssl enc -des3 -a -in /etc/passwd -out /home/passwd1.enc
```

====解密====

```
[root@tianyun ~]# openssl enc -des3 -d -in /home/passwd.enc  
[root@tianyun ~]# openssl enc -des3 -a -d -in /home/passwd1.enc -out /password_new.txt
```

非对称加密**GnuPG**

非对称加密 PGP

GnuPG (开源) 非对称加密：

主机 A : 192.168.1.250

1. 生成密钥对

```
[jack@tianyun ~]# gpg --gen-key
```

2. 列出公钥

```
[root@tianyun ~]# gpg --list-keys  
/root/.gnupg/pubring.gpg
```

```
-----  
pub 2048R/B1B799CB 2014-05-15  
uid          jackcc  
sub 2048R/FBC65F10 2014-05-15
```

3. 导出公钥

```
[root@tianyun ~]# gpg -a -o yang --export jackcc  
[root@tianyun ~]# rsync -va yang 192.168.1.4:/
```

4. 导入对方的公钥

```
[root@tianyun ~]# gpg --import /wangyan  
gpg: 密钥 D0F15E1A : 公钥“alicecc”已导入  
gpg: 合计被处理的数量 : 1  
gpg: 已导入 : 1 (RSA: 1)
```

```
[root@tianyun ~]# gpg --list-key  
/root/.gnupg/pubring.gpg
```

```
-----  
pub 2048R/B1B799CB 2014-05-15  
uid          jackcc  
sub 2048R/FBC65F10 2014-05-15
```

```
pub 2048R/D0F15E1A 2014-05-15
uid          alicecc
sub 2048R/F08F1A39 2014-05-15
```

5. 使用对方的公钥加密数据

```
[root@tianyun ~]# gpg -e -a -r alicecc c.txt
[root@tianyun ~]# ls c.txt
c.txt      c.txt.asc
```

主机B : 192.168.1.4

1. 生成密钥对
2. 列出公钥
3. 导出公钥
4. 导入对方的公钥
5. 使用对方的公钥加密数据
6. 使用自己的私钥解密数据

```
[root@hostb ~]# gpg -d -a -o new_cc.txt /c.txt.asc
```

```
[root@tianyun ~]# export DISPLAY=:0.0
```

```
[root@tianyun ~]# xhost +
```

时钟

非对称加密**SSL/TLS**

HTTPS/LDAPS/POP3S/SMTP+TLS

加密传输的数据

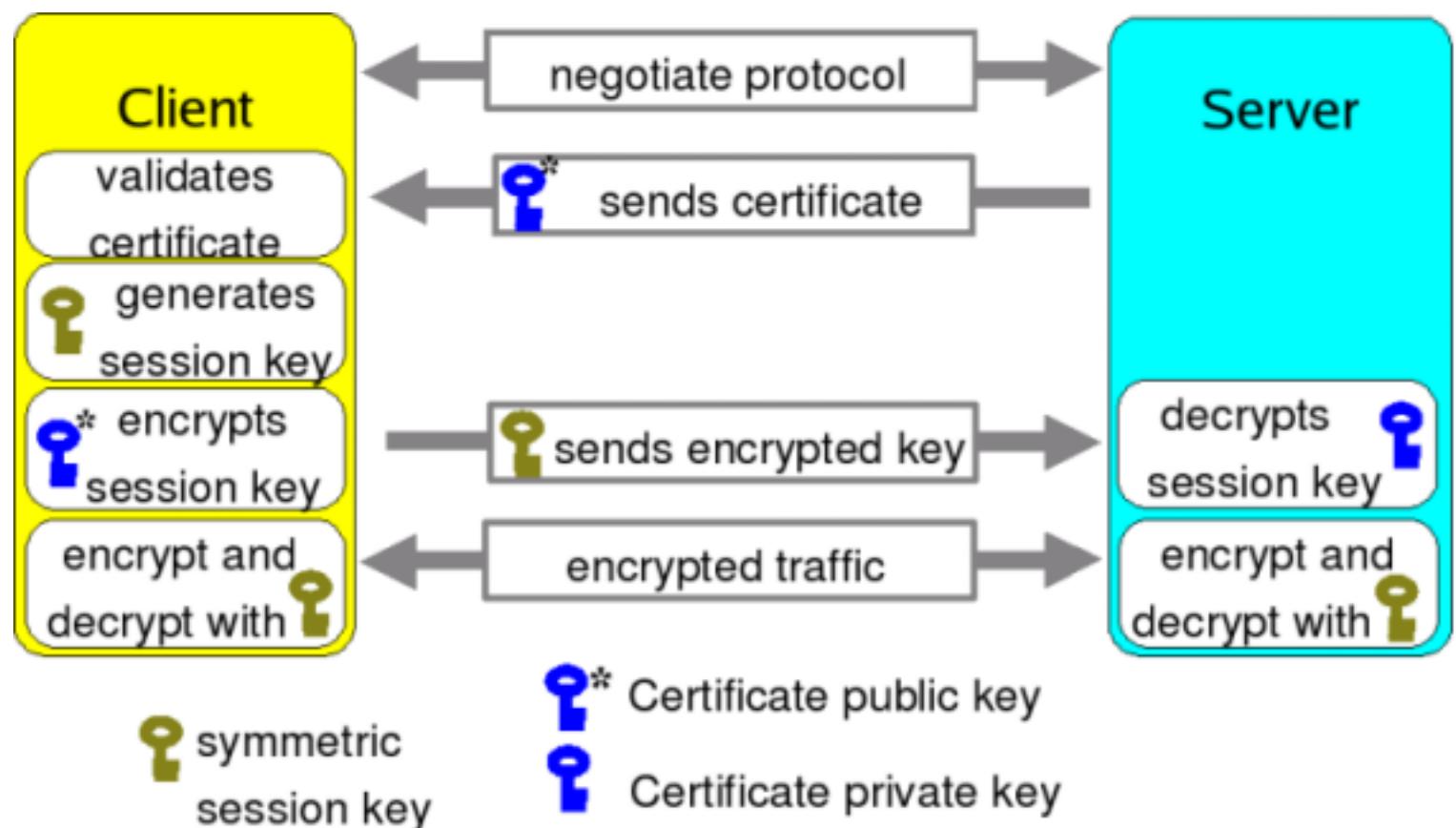
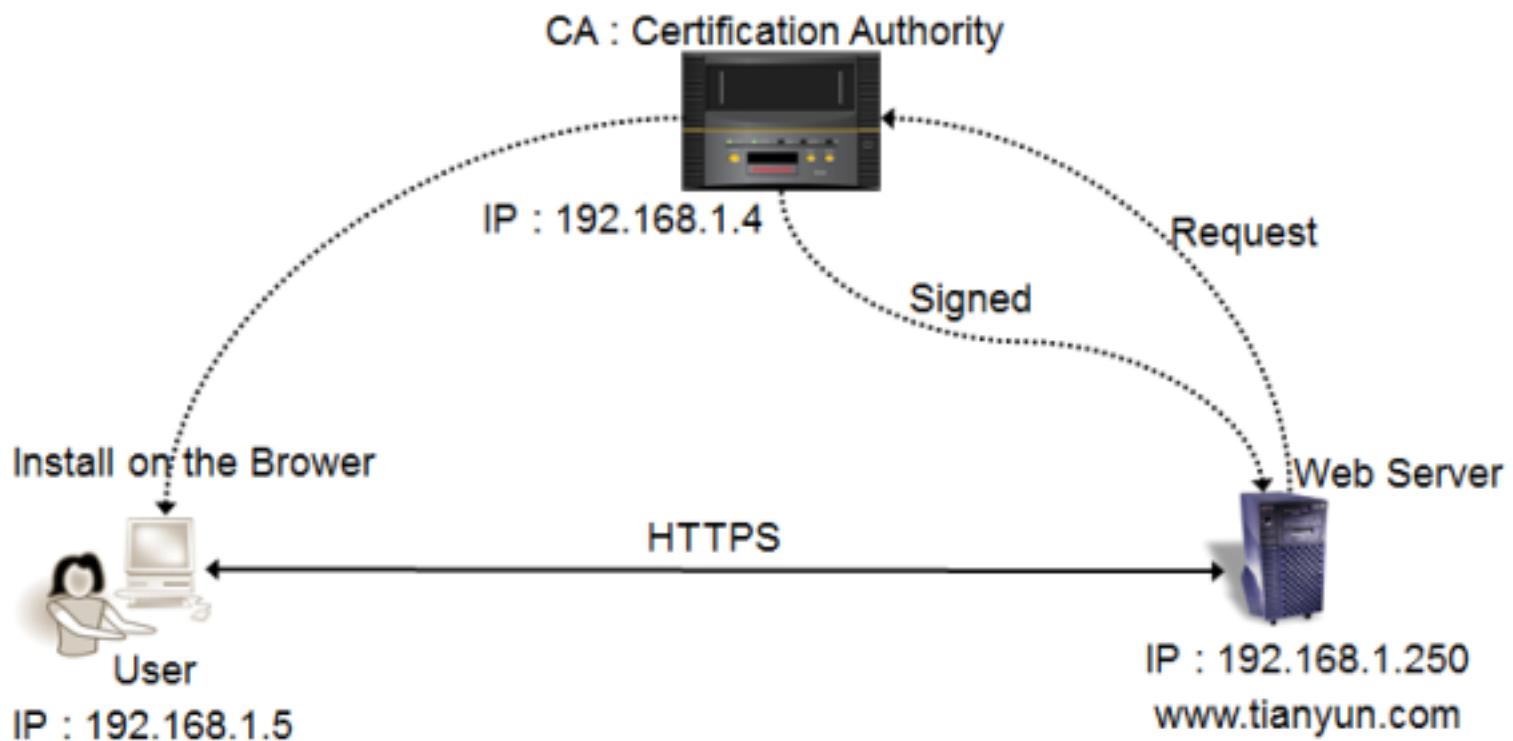
用于验证通信的双方是彼此声称的那个人！

SSL 3.0 - - - TLS 1.0 (SSL 3.1)

ldap/tcp	389	
ldap + SSL = ldaps/tcp	636	LDAP over SSL
ldap + TLS = ldaps/tcp	389	
smtp	25	
smt�ps	25	

<https://www.tianyun.com>

Certificate Signed by a CA



一、搭建CA机构

1. 安装软件包

```
[root@CA ~]# rpm -qa |grep openssl  
openssl-0.9.8e-22.el5  
openssl-devel-0.9.8e-22.el5
```

2. CA配置文件

```
[root@CA ~]# vim /etc/pki/tls/openssl.cnf  
[ ca ]  
default_ca = CA_default # The default ca section  
  
[ CA_default ]  
dir = /etc/pki/CA # Where everything is kept  
certs = $dir/certs # Where the issued certs are kept  
crl_dir = $dir/crl # Where the issued crl are kept  
database = $dir/index.txt # database index file.  
#unique_subject = no # Set to 'no' to allow creation of  
# several certificates with same subject.  
new_certs_dir = $dir/newcerts # default place for new certs.  
  
certificate = $dir/ca.crt # The CA certificate  
serial = $dir/serial # The current serial number  
crlnumber = $dir/crlnumber # the current crl number  
# must be commented out to leave a V1 CRL  
crl = $dir/ca.crl # The current CRL  
private_key = $dir/private/ca.key # The private key  
RANDFILE = $dir/private/.rand # private random number file  
  
[ req_distinguished_name ]  
countryName = Country Name (2 letter code)  
countryName_default = CN  
countryName_min = 2  
countryName_max = 2  
  
stateOrProvinceName = State or Province Name (full name)  
stateOrProvinceName_default = BJ  
  
localityName = Locality Name (eg, city)  
localityName_default = BJ  
  
0.organizationName = Organization Name (eg, company)  
0.organizationName_default = tianyun
```

3. 生成私钥和自签名的证书

```
[root@CA ~]# cd /etc/pki/CA  
[root@CA CA]# mkdir certs crl newcerts  
[root@CA CA]# touch index.txt  
[root@CA CA]# echo 00 > serial  
[root@CA CA]# ls  
certs crl index.txt newcerts private serial
```

CA生成私钥

```
[root@CA CA]# (umask 077 ; openssl genrsa -out /etc/pki/CA/private/ca.key -des3 2048)  
Generating RSA private key, 2048 bit long modulus  
.....+++  
.....+++  
e is 65537 (0x10001)  
Enter pass phrase for /etc/pki/CA/private/ca.key: //设置密码，之后为客户签名使用该私钥  
Verifying - Enter pass phrase for /etc/pki/CA/private/ca.key:  
  
[root@CA CA]# ll /etc/pki/CA/private/ca.key //查看私钥  
-rw----- 1 root root 1743 Mar 3 17:31 /etc/pki/CA/private/ca.key
```

CA自签名生成CA证书

```
[root@CA CA]# openssl req -new -x509 -days 7300 -key private/ca.key > ca.crt //CA自签名  
Enter pass phrase for private/ca.key: //提供私钥的密码  
You are about to be asked to enter information that will be incorporated  
into your certificate request.  
What you are about to enter is what is called a Distinguished Name or a DN.  
There are quite a few fields but you can leave some blank
```

For some fields there will be a default value,
If you enter '.', the field will be left blank.

Country Name (2 letter code) [CN]:
State or Province Name (full name) [BJ]:
Locality Name (eg, city) [BJ]:
Organization Name (eg, company) [tianyun]:
Organizational Unit Name (eg, section) []:it
Common Name (eg, your name or your server's hostname) []:ca.tianyun.com
Email Address []:

二、HTTP Server实现HTTPS

```
[root@tianyun ~]# yum -y install httpd mod_ssl
[root@tianyun ~]# openssl genrsa -out /etc/httpd/httd.key
[root@tianyun ~]# openssl req -new -key /etc/httpd/httd.key -out /tmp/httd.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
```

Country Name (2 letter code) [CN]:**CN**
State or Province Name (full name) [BJ]:**BJ**
Locality Name (eg, city) [BJ]:**BJ**
Organization Name (eg, company) [tianyun]:**tianyun**
Organizational Unit Name (eg, section) []:it
Common Name (eg, your name or your server's hostname) []:www.tianyun.com //客户最终访问名
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []: //不设置额外的密码
An optional company name []:
[root@tianyun ~]#

Generating RSA private key, 512 bit long modulus
....+++++
.....+++++
e is 65537 (0x10001)

HTTP Server等待CA为其签名，并生成数字证书...

三、CA服务器签名

```
[root@CA CA]# openssl ca -in /tmp/httd.csr -out /tmp/httd.crt      //签名
Using configuration from /etc/pki/tls/openssl.cnf
Enter pass phrase for /etc/pki/CA/private/ca.key: //提供CA私钥密码
Check that the request matches the signature
Signature ok
Certificate Details:
    Serial Number: 1 (0x1)
    Validity
        Not Before: Mar 3 10:02:20 2013 GMT
        Not After : Mar 3 10:02:20 2014 GMT
    Subject:
        countryName      = CN
        stateOrProvinceName = BJ
        organizationName    = tianyun
        organizationalUnitName = it
        commonName         = www.tianyun.com
X509v3 extensions:
    X509v3 Basic Constraints:
        CA:FALSE
    Netscape Comment:
        OpenSSL Generated Certificate
X509v3 Subject Key Identifier:
    11:12:DB:3B:92:8C:CD:4E:4B:FE:80:7D:BC:0B:AE:2E:84:1D:A9:16
```

X509v3 Authority Key Identifier:
keyid:83:91:3F:1C:71:C2:0C:A2:08:EF:3B:D5:CD:47:B7:32:2F:C1:73:1A

Certificate is to be certified until Mar 3 10:02:20 2014 GMT (365 days)
Sign the certificate? [y/n]:**y**

1 out of 1 certificate requests certified, commit? [y/n]**y**

Write out database with 1 new entries

Data Base Updated

签名完成后，将数字证书发给HTTP Server

四、HTTP Server (Apache)

1. Apache支持SSL

```
[root@tianyun ~]# yum -y install mod_ssl  
[root@tianyun ~]# cp /tmp/httpd.crt /etc/httpd/
```

2. 配置指定证书和私钥

```
[root@tianyun ~]# grep '^SSLCert' /etc/httpd/conf.d/ssl.conf  
SSLCertificateFile /etc/httpd/httpd.crt  
SSLCertificateKeyFile /etc/httpd/httpd.key
```

```
[root@tianyun ~]# service httpd restart
```

```
Stopping httpd: [ OK ]  
Starting httpd: [ OK ]
```

3. 虚拟主机支持（扩展知识）

https://www.zhuzhu.com
https://www.yang.com

证书申请及签名

```
[root@tianyun CA]# mkdir /webroot/zhuzhu -p  
[root@tianyun CA]# echo "zhuzhu test..." > /webroot/zhuzhu/index.html  
[root@tianyun CA]# openssl genrsa -out /etc/httpd/zhuzhu.key  
[root@tianyun CA]# openssl req -new -key /etc/httpd/zhuzhu.key -out /tmp/zhuzhu.csr  
[root@tianyun CA]# openssl ca -in /tmp/zhuzhu.csr -out /etc/httpd/zhuzhu.crt
```

虚拟主机配置

```
[root@tianyun ~]# vim /etc/httpd/conf.d/www.zhuzhu.com.conf  
<VirtualHost 1.1.1.10:443>  
    DocumentRoot /webroot/zhuzhu  
    ServerName www.zhuzhu.com  
  
    SSLEngine on  
    SSLProtocol all -SSLv2 -SSLv3  
    SSLCipherSuite ALL:!ADH:!EXPORT:!SSLv2:RC4+RSA:+HIGH:+MEDIUM:+LOW  
    SSLCertificateFile /etc/httpd/zhuzhu.crt  
    SSLCertificateKeyFile /etc/httpd/zhuzhu.key  
    SSLCACertificateFile /etc/httpd/ca.crt  
</VirtualHost>
```

```
[root@tianyun ~]# cat /etc/httpd/conf.d/www.yang.com.conf  
<VirtualHost 1.1.1.20:443>  
    DocumentRoot /var/www/html  
    ServerName www.yang.com  
  
    SSLEngine on  
    SSLProtocol all -SSLv2 -SSLv3  
    SSLCipherSuite ALL:!ADH:!EXPORT:!SSLv2:RC4+RSA:+HIGH:+MEDIUM:+LOW  
    SSLCertificateFile /etc/httpd/httpd.crt  
    SSLCertificateKeyFile /etc/httpd/httpd.key  
    SSLCACertificateFile /etc/httpd/ca.crt  
</VirtualHost>
```

五、HTTP Server (Nginx)

1. Nginx支持SSL

```
[root@tianyun nginx-1.2.0]# ./configure \  
> --user=www \
```

```
> --group=www \
> --prefix=/usr/local/nginx \
> --with-http_stub_status_module \
> --with-http_sub_module \
> --with-http_ssl_module
```

2. 配置指定证书和私钥

```
[root@tianyun ~]# vim /usr/local/nginx/conf/nginx.conf
listen 443;
ssl on;
ssl_certificate /usr/local/nginx/nginx.crt;
ssl_certificate_key /usr/local/nginx/nginx.key;

ssl_session_timeout 3m;
ssl_protocols SSLv2 SSLv3 TLSv1;
```

六、LDAP

```
[root@tianyun ~]# vim /etc/openldap/slapd.conf
TLSCACertificateFile /etc/openldap/certs/ca.crt
TLSCertificateFile /etc/openldap/certs/ldap.crt
TLSCertificateKeyFile /etc/openldap/certs/ldap.key
```

七、从客户端测试

1. 导入ca的公钥
2. <https://www.tianyun.com>

八、其它服务器

Postfix: SMTP + TLS
Dovecot: POP3S

常用服务安全

SSHD

SSHD

=====一般情况不对外开放sshd服务，只允许特定的主机（跳板机）连接

一、基本安全

```
[root@teacher ~]# vim /etc/ssh/sshd_config
Protocol 2
SyslogFacility AUTHPRIV
LogLevel INFO
#PubkeyAuthentication yes
#AuthorizedKeysFile    .ssh/authorized_keys
PermitRootLogin no
PermitEmptyPasswords no
PasswordAuthentication no
AllowUsers tom
AllowGroups hr
```

二、高级安全

公钥认证

```
[root@localhost ~]# ssh-keygen -t rsa //客户端生成公钥和私钥
[root@localhost ~]# ssh-copy-id -i 192.168.2.34 //将公钥复制到远程主机
[root@localhost ~]# ssh-copy-id -i alice@192.168.2.34 //将公钥复制到远程主机
[root@localhost ~]# ssh 192.168.2.34 'useradd alice'
[root@localhost ~]# ssh root@192.168.2.34
[root@localhost ~]# ssh 172.16.8.100 -p 3333
```

SSH端口转发

例1：

192.168.2.100

```
# ssh -g -L 7000:127.0.0.1:22 192.168.2.1
# ssh -g -L 8000:127.0.0.1:22 192.168.2.2
# ssh -g -L 9000:127.0.0.1:22 192.168.2.3
```

client test:

```
# ssh 192.168.2.100 -p 7000      =====> 192.168.2.1:22
# ssh 192.168.2.100 -p 8000      =====> 192.168.2.3:22
# ssh 192.168.2.100 -p 9000      =====> 192.168.2.3:22
```

JumpServer

三、SSH防暴力破解

1. 仅使用公钥认证

2. PAM pam_tally2.so

```
# vim /etc/pam.d/sshd
auth    required    pam_securetty.so
auth    required    pam_tally2.so deny=2 even_deny_root unlock_time=100 root_unlock_time=500
auth    required    pam_env.so
auth    required    pam_unix.so
auth    required    pam_nologin.so
account required    pam_unix.so
password required    pam_unix.so
session required    pam_limits.so
session required    pam_unix.so
session required    pam_lastlog.so nowtmp
session optional    pam_mail.so standard
```

四、iptables -j LOG

```
[root@tianyun ~]# grep 'kern.*' /etc/rsyslog.conf
kern.*          /var/log/kernel.log
[root@tianyun ~]# service rsyslog restart
```

```
[root@tianyun ~]# iptables -t filter -A INPUT -p tcp --syn --dport 22 -j LOG --log-prefix "ssh-conn "
```

五、/var/log/secure

Apache

隐藏服务器的版本信息

关注软件bug,升级版本

基于主机的访问控制

基于用户的访问控制

基于文件访问控制

例如：网站的上传目录不允许执行php程序

BIND

BIND

隐藏BIND版本

使用VIEW建立内部和外部视图

1. 外部view只提供必要的RR

2. 外部view不支持递归

主/辅区域传传输使用TSIG

以chroot的方式运行

```
# yum -y install bind-chroot
```

入侵检测

查看系统日志

查看安全相关日志

```
[root@instructor ~]# grep -i Failed /var/log/secure
```

```
May 20 12:15:35 instructor sshd[12070]: Failed password for root from 192.168.1.12 port 50720 ssh2
```

```
May 20 12:15:48 instructor sshd[12086]: Failed password for root from 192.168.1.144 port 52765 ssh2
```

```
[root@instructor ~]# grep -i Accepted /var/log/secure
```

```
Oct 24 12:18:06 chao sshd[7086]: Accepted password for root from 172.16.130.91 port 41415 ssh2
```

```
Oct 24 12:18:06 chao sshd[7084]: Accepted password for root from 172.16.130.81 port 42986 ssh2
```

```
[root@tianyun ~]# egrep -i Failed /var/log/secure |awk '{print $(NF-3)}' |egrep '^-[0-9]' |sort |uniq -c
```

```
[root@instructor ~]# last
```

```
[root@instructor ~]# lastlog
```

```
[root@instructor ~]# w
```

```
13:37:48 up 4:11, 4 users, load average: 1.88, 0.92, 0.53
```

USER	TTY	FROM	LOGIN@	IDLE	JCPU	PCPU	WHAT
root	tty1	:0	09:30	?	7:36	7:36	/usr/bin/Xorg :0 -
root	pts/0	:0.0	09:42	0.00s	0.94s	0.29s	w

```
root pts/2 :0.0      09:57 1:26m 0.94s 0.52s ssh -g -L 7000:loc  
root pts/3 :0.0      11:47 1:49m 0.06s 0.06s bash
```

```
[root@instructor ~]# less /var/log/messages  
[root@instructor ~]# less /var/log/cron
```

存储日志到远程服务器：

```
[root@tianyun ~]# vim /etc/rsyslog.conf  
authpriv.*          /var/log/secure  
authpriv.*          @172.16.8.100
```

用户级计划任务

```
[root@tianyun ~]# ls /var/spool/cron/  
alice  root  
[root@tianyun ~]# cat /var/spool/cron/*  
* * * * * date  
* * * * * ls
```

系统级计划任务

```
[root@tianyun ~]# cat /etc/crontab  
[root@tianyun ~]# ls /etc/cron.*
```

查看异常流量

常用工具

iftop, Ntopng, Cacti, Zabbix, tcpdump, wireshark, Sniffer

```
[root@tianyun ~]# iftop -i eth0  
[root@tianyun ~]# iftop
```

tcpdump

Tcpdump

man tcpdump

tcpdump基本用法

```
# tcpdump -i eth0 -nnv  
# tcpdump -i eth0 -nnv -c 100  
# tcpdump -i eth0 -nnv -w /file1.tcpdump  
# tcpdump -i eth0 -nnv -r /file1.tcpdump
```

条件 : port , host , net

```
# tcpdump -i eth0 -nnv not port 80
# tcpdump -i eth0 -nnv port 22
# tcpdump -i eth0 -nnv port 80
# tcpdump -i eth0 -nnv net 192.168.0.0/24
# tcpdump -i eth0 -nnv host 192.168.0.15
# tcpdump -i eth0 -nnv dst port 22
# tcpdump -i eth0 -nnv src port 22
```

协议作为条件 :

```
# tcpdump -i eth0 -nnv arp
# tcpdump -i eth0 -nnv icmp
# tcpdump -i eth0 -nnv udp
# tcpdump -i eth0 -nnv tcp
# tcpdump -i eth0 -nnv ip
```

多个条件 : 与关系 或关系 非关系

and or not

```
# tcpdump -i eth0 -nnv not net 192.168.0.0/24
# tcpdump -i eth0 -nnv not port 80
# tcpdump -i eth0 -nnv host 192.168.0.15 and port 22
# tcpdump -i eth0 -nnv host 192.168.0.15 and host 192.168.0.33
# tcpdump -i eth0 -nnv host 192.168.0.15 or host 192.168.0.33
# tcpdump -i eth0 -nnv 'host 192.168.0.15 and port 22' or 'host 192.168.0.33 and port 80'

# tcpdump -i eth0 -nnv host 192.168.0.110 and port 22 or port 80
# tcpdump -i eth0 -nnv host 192.168.0.110 and !( port 22 or port 80)

# tcpdump -i eth0 -nnv host 192.168.0.110 and port 80
# tcpdump -i eth0 -nnv host 192.168.0.110 and ! port 80
```

条件是 : TCP仅有SYN标记的

```
# man tcpdump
# tcpdump -i eth0 -nnv tcp[13]==2
```

C E U A P R S F

0 0 0 0 0 1 0

7 6 5 4 3 2 1 0

条件是 : TCP仅有SYN/ACK标记的

```
# tcpdump -i eth0 -nnv tcp[13]==18
```

C E U A P R S F

0 0 0 1 0 0 1 0

7 6 5 4 3 2 1 0

NIDS: snort

Snort + base 搭建IDS入侵检测系统

Snort是美国Sourcefire公司开发的发布在GPLv2下的IDS (Intrusion Detection System) 软件

Snort有三种工作模式：**嗅探器、数据包记录器、网络入侵检测系统模式。**

嗅探器模式仅仅是从网络上读取数据包并作为连续不断的流显示在终端上。

数据包记录器模式把数据包记录到硬盘上。

网络入侵检测模式分析网络数据流以匹配用户定义的一些**规则**，并根据检测结果采取一定的动作。网络入侵检测系统模式是最复杂的，而且是可配置的。

Snort可以用来监测各种数据包如端口扫描等之外，还提供了以XML形式或数据库形式记录日志的各种插件。

Snort作为常见的支持分布式的网络入侵检测系统(NIDS)，能够进行实时网络流量分析并记录各类攻击行为和相关网络数据包。

BASE(Basic Analysis and Security Engine)是基于PHP的广泛使用的一种高效Snort分析查询系统。虽然这两者配合安装配置有些复杂，但是因为其比较灵活，扩展性好，只要配置使用得当，也适合用来构建校园网入侵检测平台。

Snort 支持多种操作系统(Windows/Linux/Solaris等)，源代码和安装包可以从<http://www.snort.org>获取，由于Snort版本一直都在持续更新。综合性能和功能考虑，不建议在Windows下安装Snort。可以选择的Linux发行版本，推荐CentOS、Fedora、Redhat。虽然Snort都有rpm 包提供，安装比较方便，不过从源代码编译会更加灵活和便于进行优化。

Snort的一些功能：

- 实时通讯分析和信息包记录
- 包装有效载荷检查
- 协议分析和内容查询匹配
- 探测缓冲溢出、秘密端口扫描、CGI攻击、SMB探测、操作系统侵入尝试
- 对系统日志、指定文件、Unix socket或通过Samba的WinPopus 进行实时警

Snort有三种主要模式：信息包嗅探器、信息包记录器或成熟的侵入探测系统。遵循开发/自由软件最重要的惯例，Snort支持各种形式的插件、扩充和定制，包括数据库或是XML记录、小帧探测和统计的异常探测等。

信息包有效载荷探测是Snort最有用的一个特点，这就意味着很多额外种类的敌对行为可以被探测到。

检查可疑进程

基本工具

ps **pstree** **top** **netstat**

lsof fuser

lsof fuser

查看某文件正在被哪些进程使用

```
[root@test ~]# lsof /usr/sbin/vsftpd
COMMAND PID USER FD TYPE DEVICE SIZE NODE NAME
vsftpd 7220 root txt REG 253,0 131000 4942709 /usr/sbin/vsftpd
```

查看某个端口被哪些进程使用

```
[root@localhost ~]# lsof -i TCP:21
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
```

```
vsftpd 1826 root 3u IPv4 13869 0t0 TCP *:ftp (LISTEN)
```

Audit

Kernel Audit 内核审计

作用：将系统中所有的行为记录下来（或者有选择性的记录）：

如：什么时间，哪个用户，执行了哪个程序，操作了哪个文件，成功与否，例如执行过su命令，/tmp目录是否被写入过

```
[root@tianyun ~]# rpm -q audit
audit-2.2-2.el6.x86_64
[root@tianyun ~]# /etc/init.d/auditd status
auditd (pid 2577) is running...
```

audit Linux系统中用于记录用户底层调用情况的系统，如记录用户执行的open，exit等系统调用，并将记录写到日志文件中，可以通过使用auditctl

命令来添加或者删除audit规则。设置针对某个用户进行记录，或针对某个进程的进行记录。

auditd audit服务进程

auditctl audit规则设置工具

auditd是**audit**系统的用户空间程序。主要作用是将audit记录信息写到磁盘上，auditd在启动时会读取两个配置文件：

```
/etc/audit/auditd.conf      audit配置文件
/etc/audit/audit.rules      audit规则文件
```

查看audit日志

1. ausearch
2. aureport -l

auditctl 添加审计规则

临时生效

LAB1: 审计对/tmp目录的写入

```
[root@tianyun ~]# auditctl -w /tmp -p w
[root@tianyun ~]# auditctl -l
LIST_RULES: exit,always dir=/tmp (0x4) perm=w
[root@tianyun ~]# auditctl -D                                         //全部删除

[root@tianyun ~]# auditctl -w /tmp -p w -k "tmp_w"
[root@tianyun ~]# auditctl -l
LIST_RULES: exit,always dir=/tmp (0x4) perm=w key=tmp_w      //方便之后使用关键字索引
[jack@tianyun ~]$ mkdir /tmp/dir2
[jack@tianyun ~]$ touch /tmp/file30

[root@tianyun ~]# ausearch -k "tmp_w"                                         //检索audit
time->Wed Dec 31 10:46:56 2014
type=PATH msg=audit(1419994016.093:19): item=1 name="/tmp/ccc" inode=656806 dev=fd:01 mode=0100664 ouid=501
ogid=501 rdev=00:00
type=PATH msg=audit(1419994016.093:19): item=0 name="/tmp/" inode=655361 dev=fd:01 mode=041777 ouid=0 ogid=0
rdev=00:00
type=CWD msg=audit(1419994016.093:19): cwd="/home/alice"
type=SYSCALL msg=audit(1419994016.093:19): arch=c000003e syscall=2 success=yes exit=3 a0=7fff8f0108c1 a1=941
a2=1b6 a3=3bd838e14c items=2 ppid=9873 pid=9897 auid=0 uid=501 gid=501 euid=501 suid=501 fsuid=501 egid=501
sgid=501 fsgid=501 tty=pts1 ses=2 comm="touch" exe="/bin/touch" key="tmp_w"
---
time->Wed Dec 31 10:52:33 2014
type=PATH msg=audit(1419994353.833:26): item=1 name="/tmp/ddd" inode=656807 dev=fd:01 mode=0100664 ouid=0
ogid=501 rdev=00:00
type=PATH msg=audit(1419994353.833:26): item=0 name="/tmp/" inode=655361 dev=fd:01 mode=041777 ouid=0 ogid=0
rdev=00:00
type=CWD msg=audit(1419994353.833:26): cwd="/home/alice"
type=SYSCALL msg=audit(1419994353.833:26): arch=c000003e syscall=2 success=yes exit=3 a0=7ffe82bd8c1 a1=941
a2=1b6 a3=3bd838e14c items=2 ppid=9873 pid=9949 auid=0 uid=501 gid=501 euid=0 suid=0 fsuid=0 egid=501
sgid=501 fsgid=501 tty=pts1 ses=2 comm="touch" exe="/bin/touch" key="tmp_w"
```

注意：

auid=0 auid是登录时用户的UID，即初始的ID

uid=503 uid是当前用户的UID

euid=0 有效UID

suid=0 当前文件set uid，并属主为root

两种写法一样：

[root@tianyun ~]# auditctl -w /etc -p wa -k "etc_w" //方法一

[root@tianyun ~]# auditctl -a exit,always -F dir=/etc/ -F perm=wa //方法二

-p [r|w|x|a] Set permissions filter on watch
r=read, w=write, x=execute, a=attribute

LAB2: 审计身份切换

[root@tianyun ~]# auditctl -a exit,always -S all -F auid!=0 -F uid=0 -k "SU-ROOT" //初始UID不为0，后来成为0，即从普通用户切换到root

[root@tianyun ~]# auditctl -a exit,always -S all -F auid=0 -F uid!=0 -k "SU-USER" //root切普通用户

测试：重新打开终端 Ctrl + Alt + [F2-F6]

1. 先登录成普通用户，su成root

2. 先登录成root，su成普通用户

[root@tianyun ~]# ausearch -k "SU-USER"

[root@tianyun ~]# ausearch -k "USER-SU"

LAB3: 审计/bin/su被执行

[root@tianyun ~]# auditctl -w /bin/su -p x -k "EXEC-SU"

永久生效：

[root@tianyun ~]# vim /etc/audit/audit.rules

First rule - delete all

-D

-a exit,always -F path=/etc/passwd -F perm=w

-a exit,always -S all -F auid!=0 -F uid=0 -k tianyun

-a exit,always -F uid>=500 -F dir=/tmp -F perm=rwx -k TMP500

-w /bin/su -p x

[root@tianyun ~]# service auditd restart

[root@tianyun ~]# auditctl -l

LIST_RULES: exit,always auid!=0 uid=0 key="tianyun" syscall=all

LIST_RULES: exit,always uid>=500 (0x1f4) dir=/tmp (0x4) perm=rwx key="TMP500"

LIST_RULES: exit,always watch=/bin/su perm=x

LIST_RULES: exit,always watch=/etc/passwd perm=w

审计规则参考文件

[root@tianyun audit]# pwd

/usr/share/doc/audit-2.2

[root@tianyun audit]# ll

-rw-r--r--. 1 root root 271 Mar 2 2012 auditd.cron

-rw-r--r--. 1 root root 9770 Mar 2 2012 capp.rules

-rw-r--r--. 1 root root 31551 Mar 2 2012 ChangeLog

-rw-r--r--. 1 root root 17992 Mar 2 2012 COPYING

-rw-r--r--. 1 root root 10918 Mar 2 2012 lspp.rules

-rw-r--r--. 1 root root 5914 Mar 2 2012 nispom.rules

-rw-r--r--. 1 root root 5348 Mar 2 2012 README

-rw-r--r--. 1 root root 7892 Mar 2 2012 stig.rules

文件完整性检查

RPM -V

rpm -V

文件完整性检查，只针对RPM包安装的文件

```
=====
[root@test ~]# rpm -V bash
[root@test ~]# rpm -V kernel

[root@test ~]# rpm -Vf /etc/passwd
.....T c /etc/bashrc
S.5....T c /etc/printcap

[root@tianyun Desktop]# rpm -V coreutils
[root@tianyun Desktop]# chmod u+s /bin/rm
[root@tianyun Desktop]# rpm -V coreutils
.M..... /bin/rm

[root@test ~]# man rpm
denotes failure of the corresponding --verify test:

S file Size differs
M Mode differs (includes permissions and file type)
5 MD5 sum differs
D Device major/minor number mismatch
L readLink(2) path mismatch
U User ownership differs
G Group ownership differs
T mTime differs
```

md5sum

md5sum/sha1sum

```
=====
# find /etc -type f -exec md5sum {} \; > file1

# md5sum -c file1 |grep -i FAILED
/etc/hosts: FAILED
md5sum: WARNING: 1 of 1932 computed checksums did NOT match
```

HIDS: AIDE

HIDS: AIDE

高级入侵检测环境

一、安装并配置aide

```
[root@tianyun ~]# yum -y install aide
```

```
[root@tianyun ~]# vim /etc/aide.conf
@@define DBDIR /var/lib/aide
@@define LOGDIR /var/log/aide

# The location of the database to be read.
database=file:@@{DBDIR}/aide.db.gz

# The location of the database to be written.
#database_out=sql:host:port:database:login_name:passwd:table
#database_out=file:aide.db.new
database_out=file:@@{DBDIR}/aide.db.new.gz

/etc          NORMAL
/var/www/html    NORMAL
```

二、初始化数据库

```
[root@tianyun ~]# aide --init
AIDE, version 0.13.1
### AIDE database at /var/lib/aide/aide.db.new.gz initialized.
```

警告：把生成的数据库放到其它位置...

三、定期检查

```
[root@tianyun ~]# aide --check
Couldn't open file /var/lib/aide/aide.db.gz for reading
[root@tianyun ~]# cd /var/lib/aide/
[root@tianyun aide]# ls
aide.db.new.gz
[root@tianyun aide]# cp aide.db.new.gz aide.db.gz
[root@tianyun aide]# ls
aide.db.gz  aide.db.new.gz
```

四、更新数据库

```
[root@tianyun ~]# aide --update
=====
```

渗透测试

渗透工具

BT (Back Track) ===> kali

ARP欺骗

一、静态绑定如网关的MAC

```
[root@tianyun ~]# arp -s 172.16.70.254 00:0d:65:c8:a9:ff
```

二、ARP防火墙

```
arpTables
```

中间人攻击

DNS欺骗

Shellshock

本节作业

LDAP:

1. 编写ldap部署脚本
2. 编写批量导入ldap用户脚本

Iptables:

1. -j SNAT 和 -j MASQUERADE 作用场景
2. -j DNAT 和 -j REDIRECT 作用场景

三、虚拟化

虚拟化技术概述

虚拟化技术概述

一、虚拟化技术概述

虚拟化 (Virtualization) 技术最早出现在 20 世纪 60 年代的 IBM 大型机系统，在 70 年代的 System 370 系列中逐渐流行起来，这些机器通过一种叫虚拟机监控器 (Virtual Machine Monitor, VMM) 的程序在物理硬件之上生成许多可以运行独立操作系统软件的虚拟机 (Virtual Machine) 实例。随着近年多核系统、集群、网格甚至云计算的广泛部署，虚拟化技术在商业应用上的优势日益体现，不仅降低了 IT 成本，而且还增强了系统安全性和可靠性，虚拟化的概念也逐渐深入到人们日常的工作与生活中。

虚拟化是一个广义的术语，对于不同的人来说可能意味着不同的东西，这要取决于他们所处的环境。在计算机科学领域中，**虚拟化**代表对计算资源的抽象，而不仅仅局限于虚拟机的概念。例如对物理内存的抽象，产生了**虚拟内存技术**，使得应用程序认为其自身拥有连续可用的地址空间（Address Space），而实际上，应用程序的代码和数据可能是被分隔成多个碎片页或段），甚至被交换到磁盘、闪存等外部存储器上，即使物理内存不足，应用程序也能顺利执行。

二、虚拟化技术的分类

平台虚拟化（Platform Virtualization），针对计算机和操作系统的虚拟化

资源虚拟化（Resource Virtualization），针对特定的系统资源的虚拟化，比如内存、存储、网络资源等。

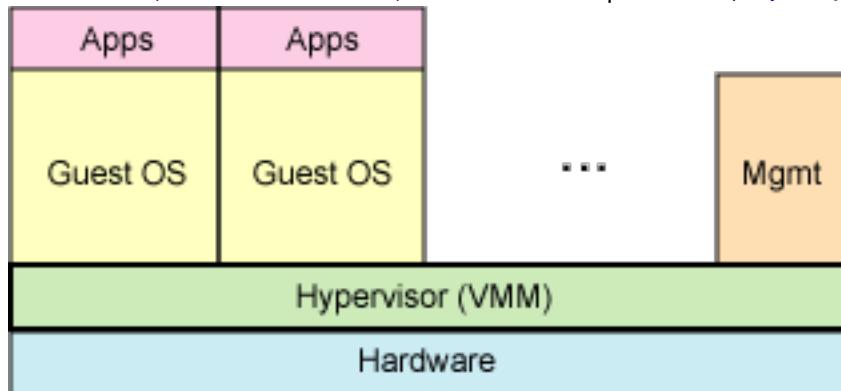
应用程序虚拟化（Application Virtualization），包括仿真、模拟、解释技术等。

我们通常所说的虚拟化主要是指平台虚拟化技术，通过使用控制程序（Control Program，也被称为 **Virtual Machine Monitor** 或 **Hypervisor**），隐藏特定计算平台的实际物理特性，为用户提供抽象的、统一的、模拟的计算环境（称为虚拟机）。虚拟机中运行的操作系统被称为客户机操作系统（Guest OS），运行虚拟机监控器的操作系统被称为主机操作系统（Host OS），当然某些虚拟机监控器可以脱离操作系统直接运行在硬件之上（如 VMWARE 的 ESX/ESXi 产品）。运行虚拟机的真实系统我们称之为宿主系统。

三、平台虚拟化技术子类

全虚拟化（Full Virtualization）

全虚拟化是指虚拟机模拟了完整的底层硬件，包括处理器、物理内存、时钟、外设等，使得为原始硬件设计的操作系统或其它系统软件完全不做任何修改就可以在虚拟机中运行。操作系统与真实硬件之间的交互可以看成是通过一个预先规定的硬件接口进行的。全虚拟化 VMM 以完整模拟硬件的方式提供全部接口（同时还必须模拟特权指令的执行过程）。比较著名的全虚拟化 VMM 有 Microsoft Virtual PC、VMware Workstation、Oracle Virtual Box、Parallels Desktop for Mac 和 QEMU。



全虚拟化(Full virtualization)，也称为原始虚拟化技术。该模型使用虚拟机协调客户操作系统和原始硬件。这里“协调”个关键词，因为VMM在客户操作系统和裸硬件之间用于工作协调。一些受保护的指令必须由Hypervisor(虚拟机管理程序)来捕获和处理。因为操作系统是通过Hypervisor来分享底层硬件。

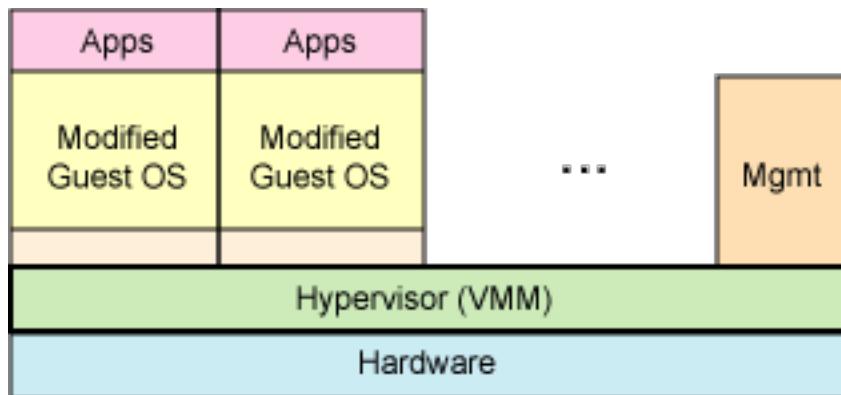
全虚拟化最大的优点是操作系统没有经过任何修改。它的唯一限制是操作系统必须能够支持底层硬件(比如，PowerPC)。

硬件辅助虚拟化（Hardware-Assisted Virtualization）

硬件辅助虚拟化是指借助硬件（主要是主机处理器）的支持来实现高效的全虚拟化。例如有了 **Intel-VT** 技术的支持，Guest OS 和 VMM 的执行环境自动地完全隔离开来，Guest OS 有自己的“全套寄存器”，可以直接运行在最高级别。因此在上面的例子中，Guest OS 能够执行修改页表的汇编指令。**Intel-VT** 和 **AMD-V** 是目前 x86 体系结构上可用的两种硬件辅助虚拟化技术。

半虚拟化（Para-virtualization）

这是一种修改 Guest OS 部分访问特权状态的代码以便直接与 VMM 交互的技术。在半虚拟化虚拟机中，部分硬件接口以软件的形式提供给客户机操作系统，这可以通过 Hypercall（VMM 提供给 Guest OS 的直接调用，与系统调用类似）的方式来提供。由于不需要产生额外的异常和模拟部分硬件执行流程，半虚拟化可以大幅度提高性能，比较著名的 VMM 有 Denali、**Xen**。



半虚拟化(Para-virtualization)是另一种类似于全虚拟化的热门技术。它使用Hypervisor(虚拟机管理程序)分享存取底层的硬件，但是它的客户操作系统集成了虚拟化方面的代码。该方法无需重新编译或引起陷阱，因为操作系统自身能够与虚拟进程进行很好的协作。半虚拟化需要客户操作系统做一些修改(配合Hypervisor)，这是一个不足之处。但是半虚拟化提供了与原始系统相近的性能。与全虚拟化一样，半虚拟化可以同时支持多个不同的操作系统。

常见虚拟化方案

常见虚拟机 VMM Hypervisor

XEN

[QEMU/KVM](#)

VMware Workstation
VMware Server
[VMware ESX/ESXi](#)

VirtualBox

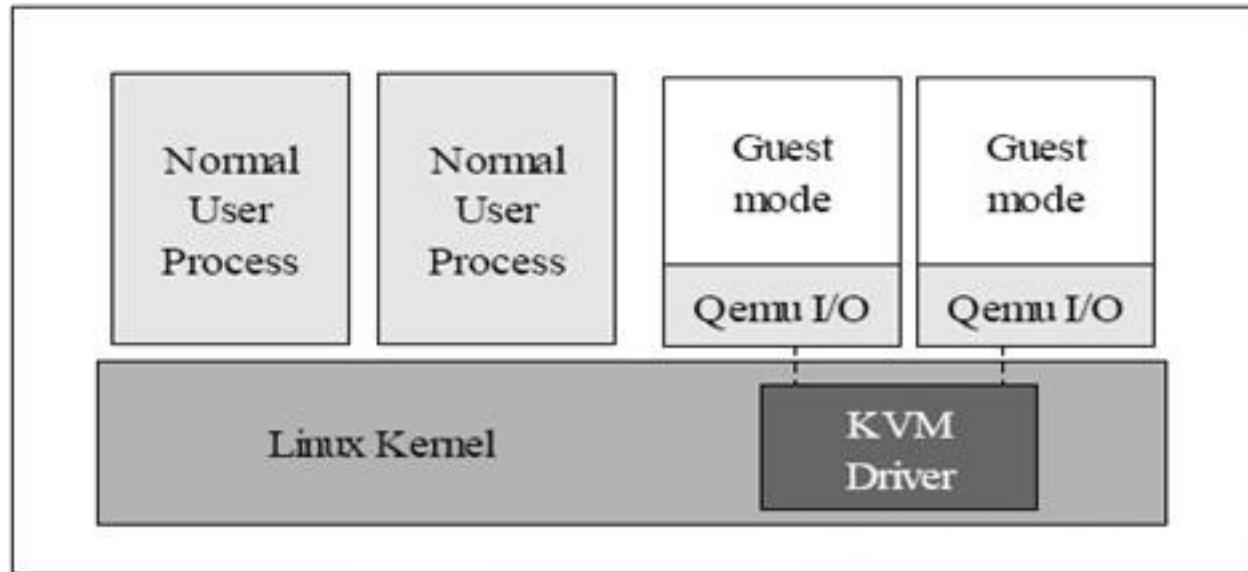
Hyper-V

Ctrix

内核虚拟机 **KVM**

KVM虚拟化简介

KVM简介



虚拟化是一项允许单个物理计算机分为多个虚拟机的功能，每个虚拟机可以分别运行独立的操作系统。
X86-64的RHEL6支持KVM，KVM允许内核作为支持guest虚拟机的管理程序(hypervisor)

安装虚拟化软件包

Installing the virtualization packages

一、安装前的准备

1. CPU支持虚拟化 (Inter VT-X、AMD-V)

```
[root@tianyun ~]# lscpu
Virtualization: VT-x
[root@tianyun ~]# egrep --color 'svm|vmx|lm' /proc/cpuinfo
```

相关CPU功能标志包括：

svm=安全虚拟机 (AMD-V)

vmx=虚拟机x86 (Inter VT-X)

lm=长模式 (64位支持)

2. BIOS开启CPU虚拟化支持

二、安装虚拟化软件

1. 虚拟相关组件

Package Group	Description	Mandatory Packages	Optional Packages
Virtualization	Provides an environment for hosting virtual machines	qemu-kvm	qemu-guest-agent, qemu-kvm-tools
Virtualization Client	Clients for installing and managing virtualization instances	python-virtinst, virt-manager, virt-viewer	virt-top
Virtualization Platform	Provides an interface for accessing and controlling virtual machines and containers	libvirt, libvirt-client, virt-who, virt-what	fence-virtd-libvirt, fence-virtd-multicast, fence-virtd-serial, libvirt-cim, libvirt-java, libvirt-qmf, libvirt-snmp, perl-Sys-Virt
Virtualization Tools	Tools for offline virtual image management	libguestfs	libguestfs-java, libguestfs-tools, virt-v2v

2. 安装虚拟化软件KVM

Configuring a Virtualization Host installation 在安装系统的过程中安装KVM

The default installation of Red Hat Enterprise Linux is a basic server install. You can optionally select a different set of software now.

- Basic Server
- Database Server
- Web Server
- Identity Management Server
- Virtualization Host
- Desktop
- Software Development Workstation
- Minimal

Please select any additional repositories that you want to use for software installation.

- High Availability
- Load Balancer
- Red Hat Enterprise Linux

 [Add additional software repositories](#)

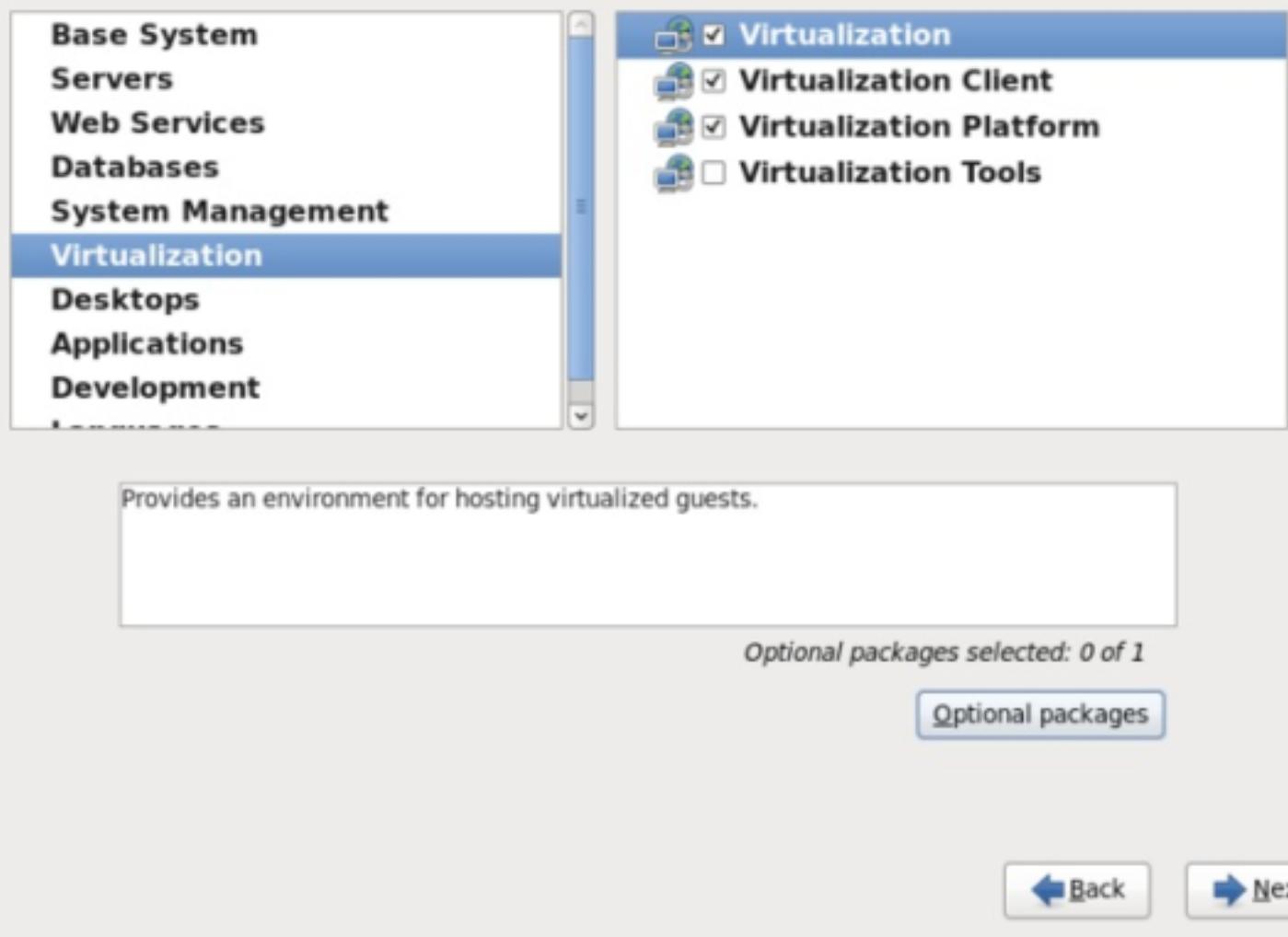
 [Modify repository](#)

You can further customize the software selection now, or after install via the software management application.

- Customize later
- Customize now

 [Back](#)

 [Next](#)



Installing KVM packages with Kickstart files 使用Kickstart文件安装KVM

In the %packages section of your Kickstart file, append the following package groups:

```
@ virtualization  
@ virtualization-client  
@ virtualization-platform  
@ virtualization-tools
```

Installing virtualization packages on an existing Red Hat Enterprise Linux system 在已经存在的系统上安装

```
[root@tianyun ~]# wget ftp://172.16.8.100/rhel6.repo -P /etc/yum.repos.d/  
[root@tianyun ~]# LANG=C yum grouplist |grep -i 'virtual'  
[root@tianyun ~]# LANG=C yum -y groupinstall "Virtualization" "Virtualization Client" "Virtualization Platform" "Virtualization Tools"  
[root@tianyun ~]# lsmod |grep kvm  
  
[root@tianyun ~]# service libvirtd start  
[root@tianyun ~]# chkconfig libvirtd on  
[root@tianyun ~]# virsh list --all  
[root@tianyun ~]# virt-manager
```

Guest虚拟机安装

KVM管理工具

KVM管理工具

=====

qemu-kvm (原生)

libvirt

virsh

virt-manager

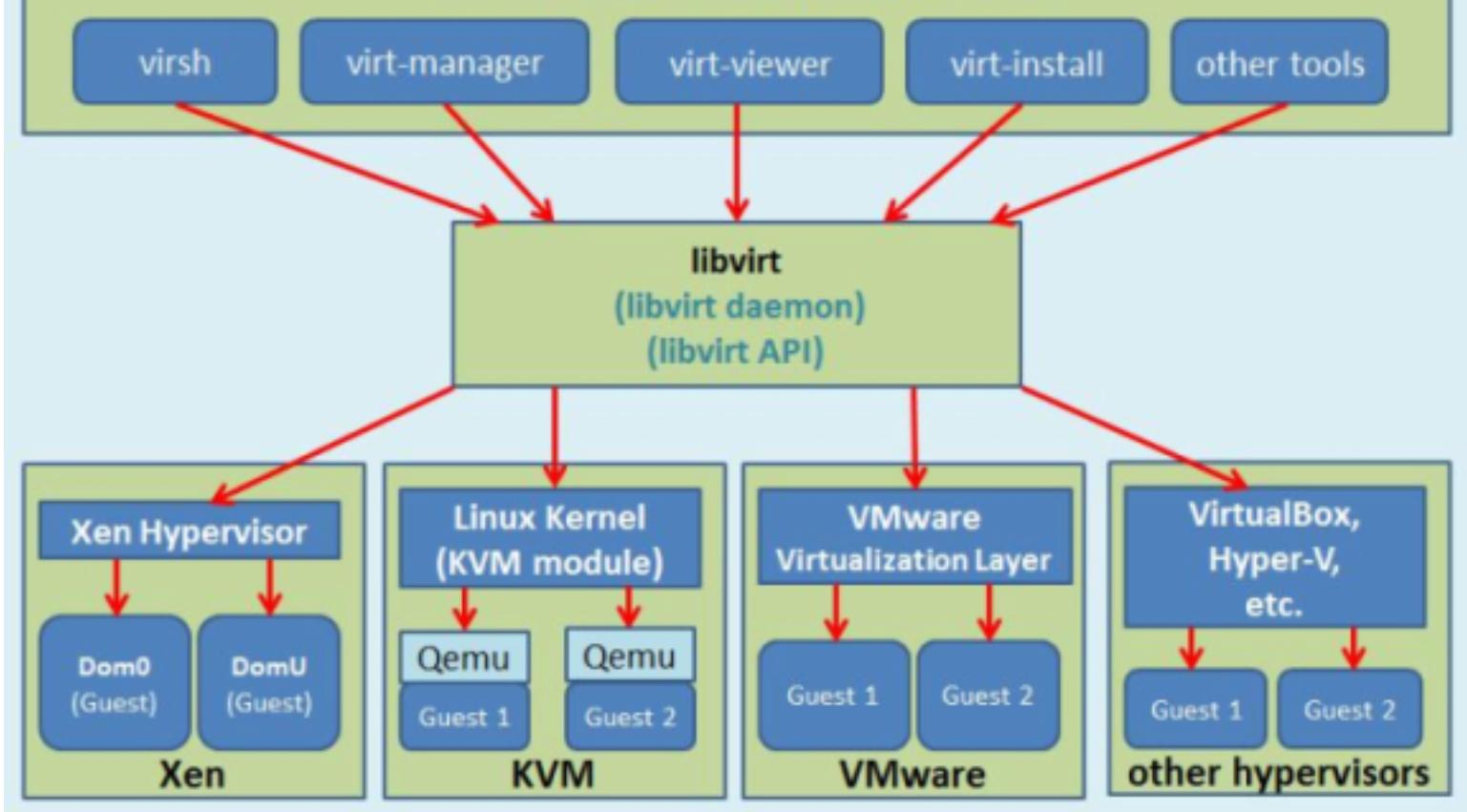
virt-viewer

virt-install

virt-top

OpenStack

User-space management tools

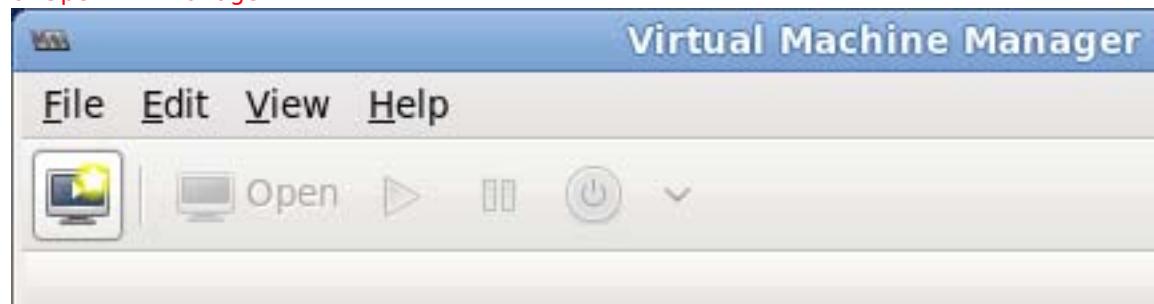


virt-manager

Guest virtual machine installation

Creating guests with **virt-manager** 使用virt-manager创建虚拟机

0. Open **virt-manager**



1. Specify name and installation type

New VM



Create a new virtual machine

Step 1 of 5

Enter your virtual machine details

Name:

Connection: localhost (QEMU/KVM)

Choose how you would like to install the operating system

- Local install media (ISO image or CDROM)
- Network Install (HTTP, FTP, or NFS)
- Network Boot (PXE)
- Import existing disk image

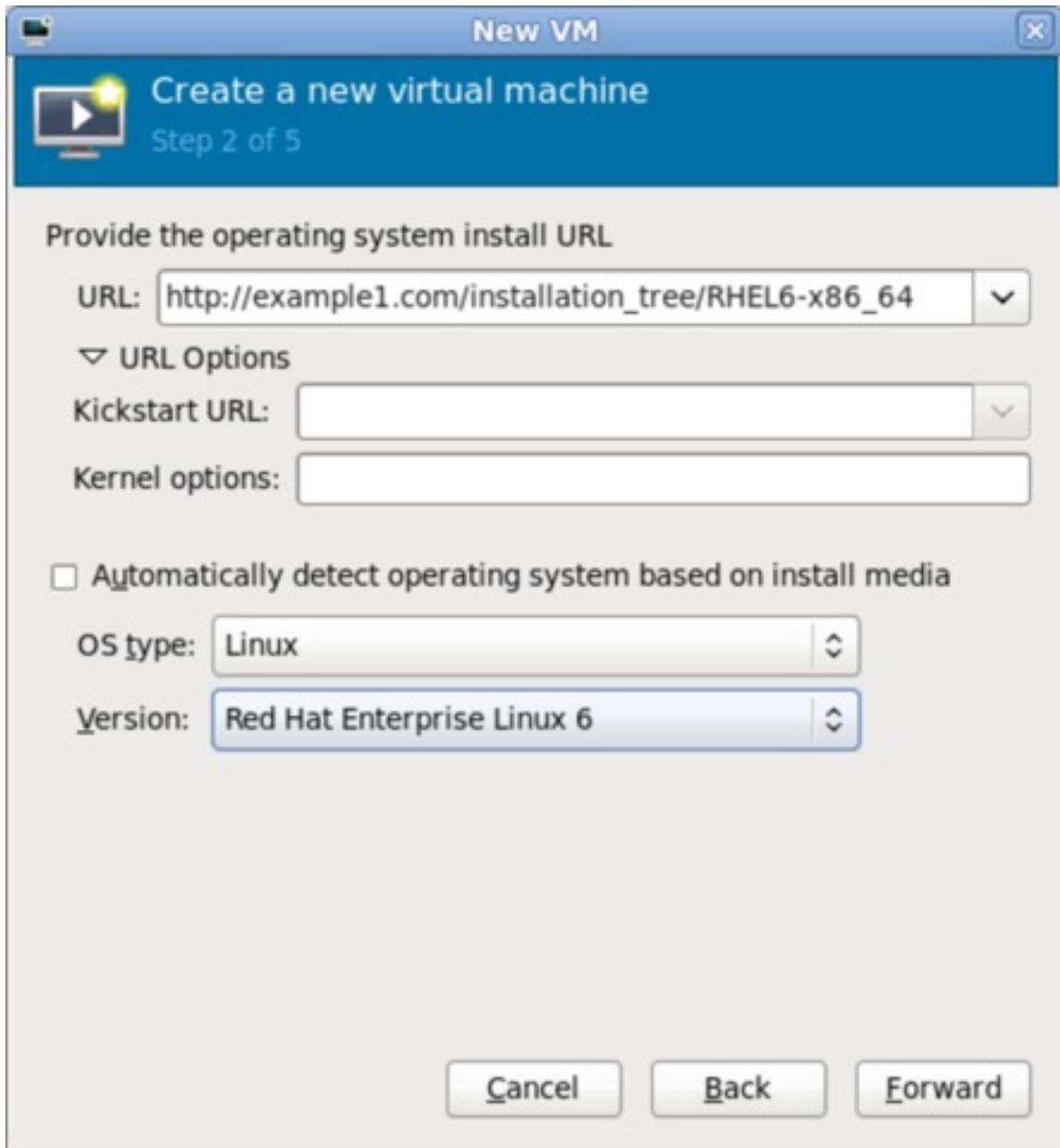
[Cancel](#)

[Back](#)

[Forward](#)

2. Configure installation

a. Remote installation URL



b. Local ISO image installation

New VM



Create a new virtual machine

Step 2 of 5

Locate your install media

Use CDROM or DVD

Use ISO image:

/var/lib/libvirt/images/RHEL6.2-Server-x86

Choose an operating system type and version

OS type: Linux

Version: Red Hat Enterprise Linux 6

[Cancel](#)

[Back](#)

[Forward](#)

3. Configure CPU and memory

New VM



Create a new virtual machine

Step 3 of 5

Choose Memory and CPU settings

Memory (RAM): MB

Up to 6033 MB available on the host

CPUs:

Up to 6 available

Cancel

Back

Forward

4. Configure storage

New VM



Create a new virtual machine

Step 4 of 5

Enable storage for this virtual machine

Create a disk image on the computer's hard drive

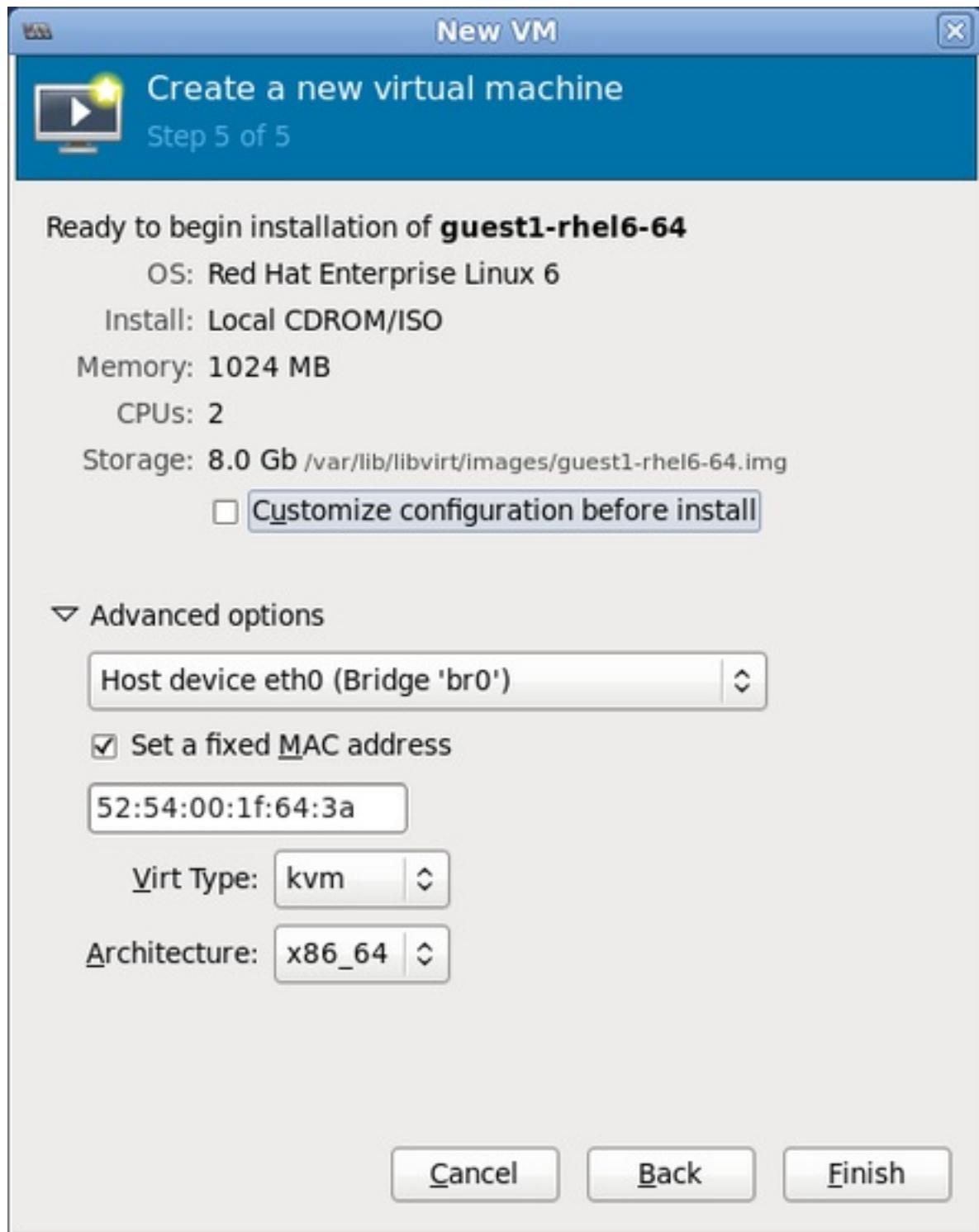
8.0 GB

14.8 Gb available in the default location

Allocate entire disk now

Select managed or other existing storage

5. Final configuration



URL: <ftp://172.16.8.100/rhel6.4>
Kickstart URL: <ftp://172.16.8.100/rhel6.4.ks>

磁盘：

1. 设备文件 /dev/sdb5 或 /dev/vg1/lv5
2. 文件 /vms/web1.img (raw) /vms/web2.qcow2(推荐qcow2)

virt-install

Guest virtual machine installation

Creating guests with **virt-install** 使用**virt-install**创建虚拟机

```
[root@tianyun ~]# qemu-img create -f qcow2 /var/lib/libvirt/images/tianyun.img 8G
```

```
[root@tianyun ~]# virt-install --help
```

```
[root@tianyun ~]# /usr/sbin/virt-install \
--vnc \
--name=tianyun \
--ram=512 \
--arch=x86_64 \
--vcpus=1 \
--os-type=linux \
--os-variant=rhel6 \
--hvm \
--accelerate \
--file=/var/lib/libvirt/images/tianyun.img \
--bridge=virbr0 \
--location=ftp://172.16.8.100/rhel6.4 \
--extra-args="ks=ftp://172.16.8.100/rhel6.4.ks"
```

虚拟机相关文件

```
[root@tianyun ~]# virsh list --all
```

Id	Name	State
-	node1	shut off
-	tianyun	shut off

1. 配置文件

```
[root@tianyun ~]# ls /etc/libvirt/qemu
networks node1.xml tianyun.xml
```

```
[root@tianyun ~]# virsh edit tianyun
[root@tianyun ~]# virsh dumpxml tianyun > newhost
```

设置自动运行vm

```
[root@tianyun ~]# virsh autostart tianyun
Domain tianyun marked as autostarted
```

```
[root@tianyun ~]# ls /etc/libvirt/qemu/autostart/
tianyun.xml
```

```
[root@tianyun ~]# ll /etc/libvirt/qemu/autostart/tianyun.xml
lwxrwxrwx. 1 root root 29 Jul 30 13:46 /etc/libvirt/qemu/autostart/tianyun.xml -> /etc/libvirt/qemu/tianyun.xml
```

2. 镜像文件[磁盘文件]

```
[root@tianyun ~]# ls /var/lib/libvirt/images/
node1.img tianyun.img
```

KVM 网络配置

网络配置

Network configuration

Red Hat Enterprise Linux 6 supports the following networking setups for virtualization:

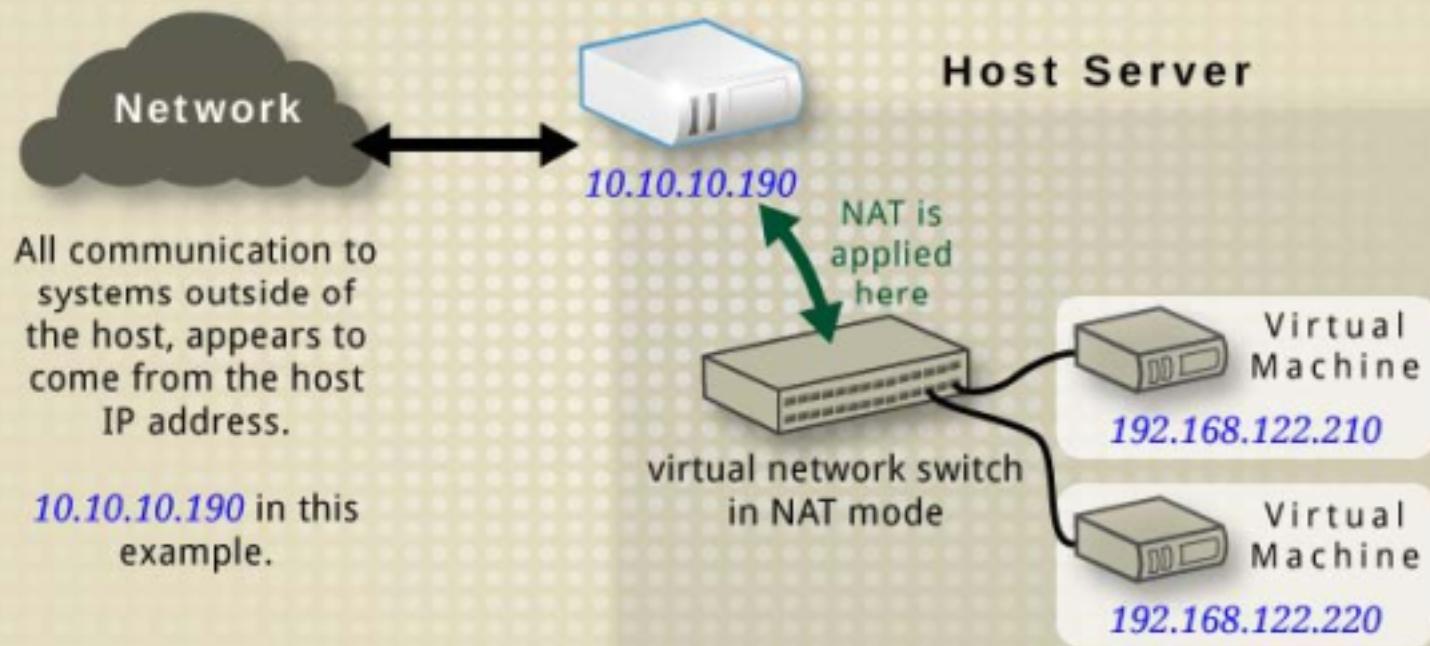
- virtual networks using Network Address Translation (**NAT**)
- directly allocated physical devices using PCI device assignment
- directly allocated virtual functions using PCIe SR-IOV
- **bridged networks**

```
[root@tianyun networks]# rpm -q bridge-utils  
bridge-utils-1.2-10.el6.x86_64
```

NAT

Network Address Translation (NAT) with libvirt (default)

Virtual switch: NAT mode



Virtualization Host Server



Virtual network switch



DNS and DHCP server (dnsmasq)

Using DHCP range:
192.168.122.2 - 192.168.122.254



Host configuration

1. 查看网络情况

```
[root@tianyun ~]# virsh net-list --all  
Name   State  Autostart
```

```
-----  
default active yes
```

```
[root@tianyun ~]# virsh net-edit default
```

```
[root@tianyun ~]# ls /etc/libvirt/qemu/networks/autostart/
```

```
default.xml //默认启动的网络
```

```
[root@tianyun ~]# ls /etc/libvirt/qemu/autostart/
```

```
tianyun.xml //开机启动的虚拟机
```

2. 启动网络 default

```
[root@tianyun ~]# virsh net-start default
```

```
Network default started
```

```
[root@tianyun ~]# virsh net-destroy default
```

```
Network default destroyed
```

```
[root@tianyun ~]# virsh net-list --all
```

Name	State	Autostart	Persistent
default	inactive	yes	yes

3. 重启加载 default 配置

```
[root@tianyun ~]# virsh net-define /usr/share/libvirt/networks/default.xml
```

4. 查看是否生效

```
[root@tianyun ~]# sysctl -a |grep ip_forward //当前状态
```

```
net.ipv4.ip_forward = 1
```

```
[root@tianyun ~]# vim /etc/sysctl.conf
```

```
net.ipv4.ip_forward = 1
```

```
[root@tianyun ~]# iptables -t nat -nL POSTROUTING
```

```
Chain POSTROUTING (policy ACCEPT)
```

```
target prot opt source destination
```

```
MASQUERADE tcp -- 192.168.122.0/24 !192.168.122.0/24 masq ports: 1024-65535
```

```
MASQUERADE udp -- 192.168.122.0/24 !192.168.122.0/24 masq ports: 1024-65535
```

```
MASQUERADE all -- 192.168.122.0/24 !192.168.122.0/24
```

```
[root@tianyun ~]# brctl show
```

Guest virtual machine configuration

1. 客户机配置文件

NAT模式 :

```
<interface type='bridge'>
  <mac address='52:54:00:60:2a:a8'/>
  <source network='default' />
  <model type='virtio' />
  <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0' />
</interface>
```

桥接(Route)模式 :

```
<interface type='bridge'>
  <mac address='52:54:00:60:2a:a8'/>
  <source bridge='br0' />
  <model type='virtio' />
  <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0' />
</interface>
```

为虚拟机映射外网IP :

SNAT

```
[root@tianyun ~]# iptables -t nat -F
```

```
[root@tianyun ~]# iptables -F
```

```
[root@tianyun ~]# iptables -t nat -A POSTROUTING -s 192.168.122.0/24 ! -d 192.168.122.0/24 -j MASQUERADE
```

DNAT

```
[root@tianyun ~]# iptables -t nat -A PREROUTING -d 172.16.30.30 -p tcp --dport 80 -j DNAT --to 192.168.122.66:80
```

```
[root@tianyun ~]# iptables -t nat -A PREROUTING -d 172.16.30.30 -p tcp --dport 8080 -j DNAT --to 192.168.122.77:80
```

```
[root@tianyun ~]# iptables -t nat -A PREROUTING -p tcp --dport 2222 -j DNAT --to 192.168.122.66:22
```

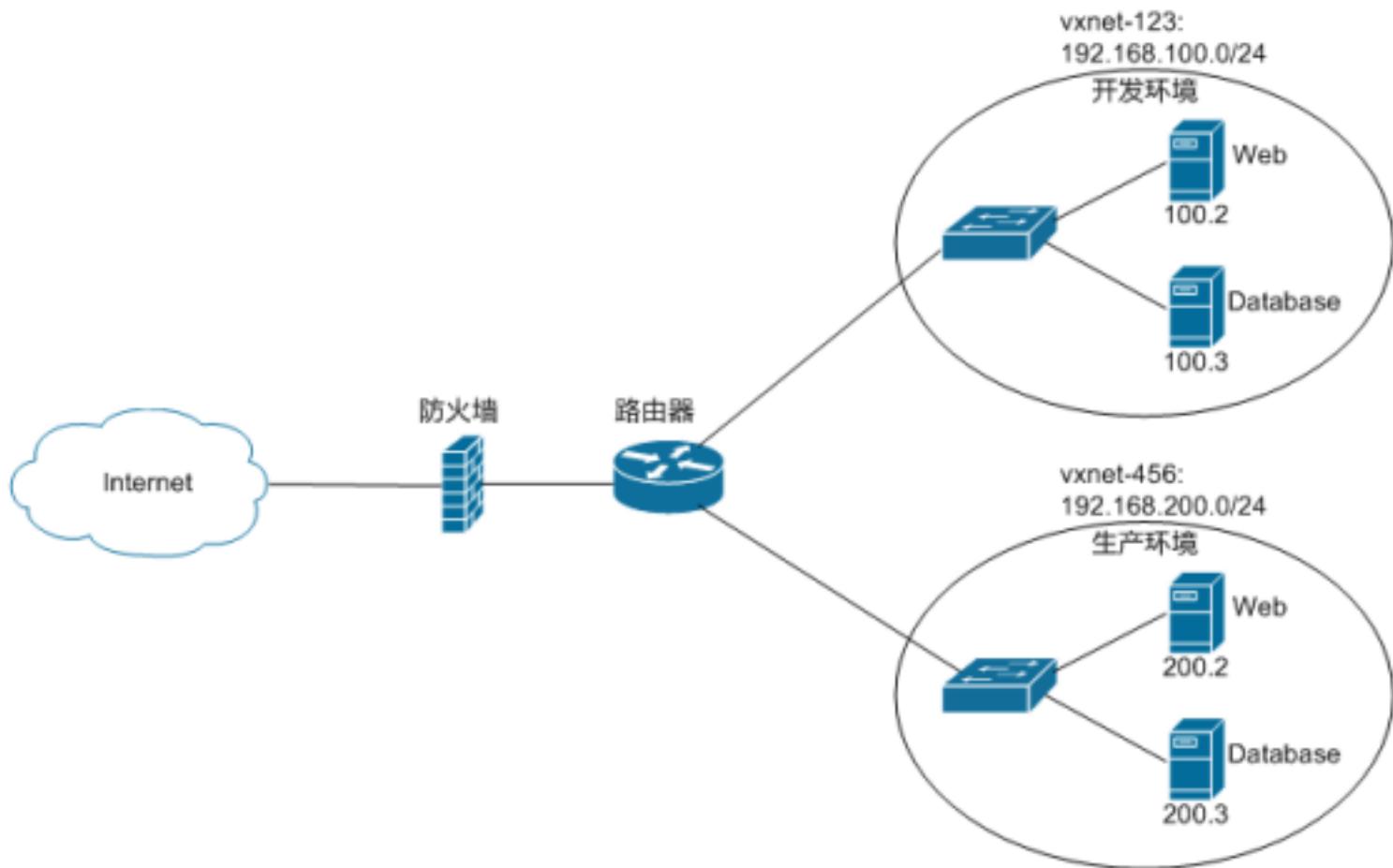
```
[root@tianyun ~]# iptables -t nat -A PREROUTING -d 172.16.30.240 -j DNAT --to 192.168.122.66
```

```
[root@tianyun ~]# iptables -t nat -A PREROUTING -d 172.16.30.241 -j DNAT --to 192.168.122.67
```

为接口绑定地址

```
[root@tianyun ~]# ip addr add dev eth0 172.16.30.240/24
[root@tianyun ~]# ip addr add dev eth0 172.16.30.241/24
[root@tianyun ~]# ip a s eth0
3: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 3c:97:0e:54:c5:82 brd ff:ff:ff:ff:ff:ff
    inet 172.16.30.30/24 brd 172.16.30.255 scope global eth0
        inet 172.16.30.240/24 scope global secondary eth0
        inet 172.16.30.241/24 scope global secondary eth0
    inet6 fe80::3e97:eff:fe54:c582/64 scope link
        valid_lft forever preferred_lft forever
```

```
[root@tianyun ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
IPADDR=172.16.30.30
PREFIX=24
IPADDR1=172.16.30.240
PREFIX1=24
IPADDR2=172.16.30.241
PREFIX2=24
GATEWAY=172.16.30.254
```



添加其它的私有网络 (NAT)

```
tianyun155 192.168.155.0/24
tianyun166 192.168.166.0/24
tianyun177 192.168.177.0/24
```

1. 创建私有网络

```
[root@tianyun ~]# virsh net-list --all
Name      State  Autostart  Persistent
-----
default   active yes       yes
```

```
[root@tianyun ~]# cp -rf /etc/libvirt/qemu/networks/{default.xml,net155.xml}
[root@tianyun ~]# uuidgen
7144aa9f-e71b-40dc-8fae-26d653fc07a4
[root@tianyun ~]# openssl rand -hex 6 |sed -r 's/(..)\1:/g; s/$//'
58:1c:32:8a:e8:74
[root@tianyun ~]# vim /etc/libvirt/qemu/networks/net155.xml
<network>
  <name>net155</name>
  <uuid>7144aa9f-e71b-40dc-8fae-26d653fc07a4</uuid>
  <bridge name="net155" />
  <mac address='52:54:00:5E:50:E8' />
  <forward/>
  <ip address="192.168.155.1" netmask="255.255.255.0">
    <dhcp>
      <range start="192.168.155.2" end="192.168.155.254" />
    </dhcp>
  </ip>
</network>

[root@tianyun ~]# virsh net-define /etc/libvirt/qemu/networks/net155.xml
Network net155 defined from /etc/libvirt/qemu/networks/net155.xml

[root@tianyun ~]# virsh net-start net155
Network net155 started

[root@tianyun ~]# virsh net-autostart net155
Network net155 marked as autostarted

[root@tianyun ~]# virsh net-list --all
Name          State   Autostart  Persistent
-----
133           active   yes       yes
144           active   yes       yes
default        active   yes       yes
net155         active   yes       yes

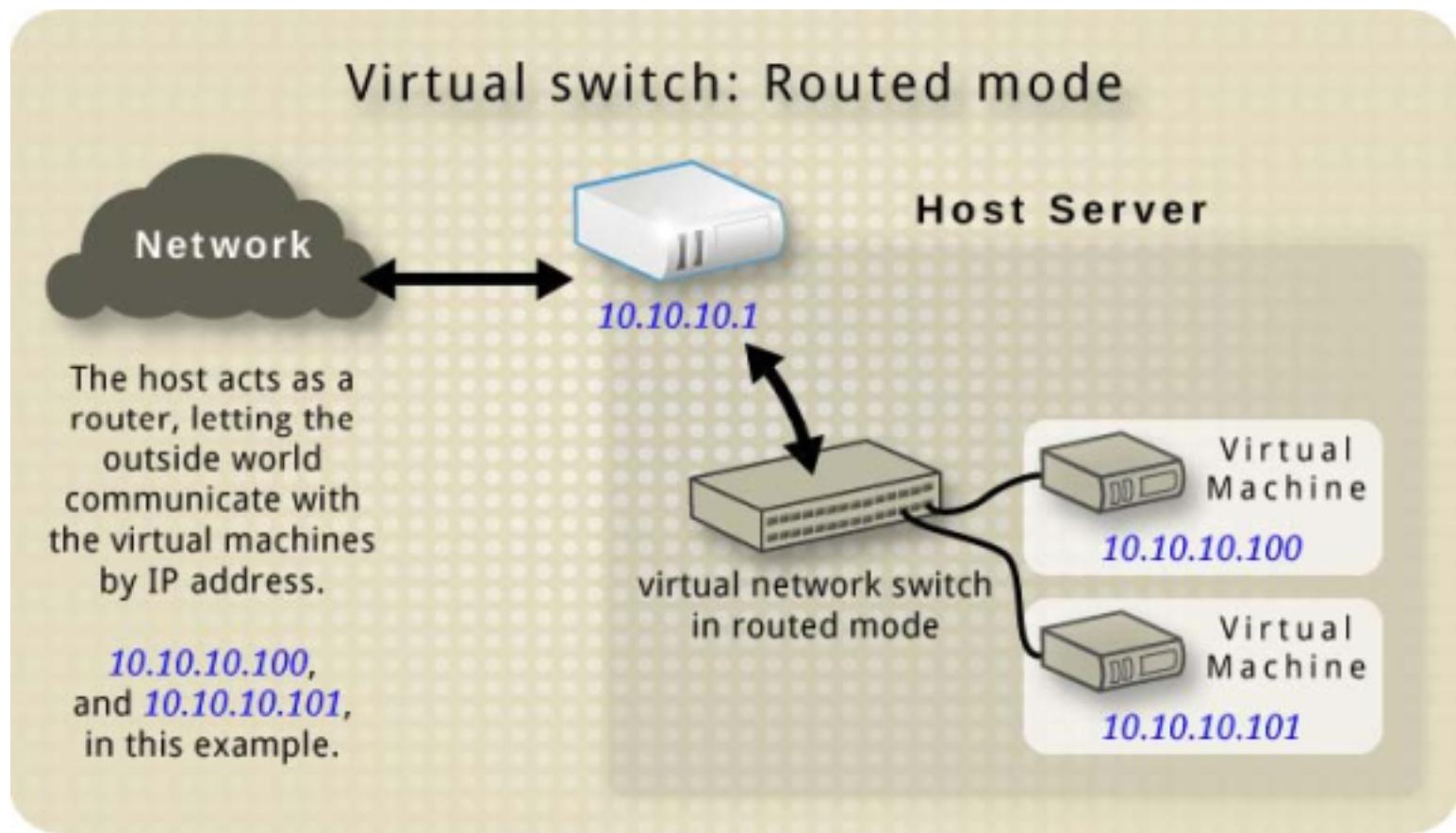
[root@tianyun ~]# ip addr show net155
17: net155: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN
  link/ether 52:54:00:5e:50:e8 brd ff:ff:ff:ff:ff:ff
    inet 192.168.155.1/24 brd 192.168.155.255 scope global net155

[root@tianyun ~]# brctl show
bridge name  bridge id          STP enabled  interfaces
net155       8000.5254005e50e8  yes         net155-nic
pan0         8000.000000000000  no          virbr0-nic
virbr0       8000.5254005e50e7  yes         virbr0-nic
                                         vnet1
virbr1       8000.525400be5027  yes         virbr1-nic
                                         vnet0
virbr144    8000.525400be5028  yes         virbr144-nic
                                         vnet2
```

2. 为虚拟机设置网络

Bridged

Bridged networking with libvirt



Host configuration : 创建桥

1. 停用NetworkManager

```
[root@tianyun ~]# chkconfig NetworkManager off  
[root@tianyun ~]# chkconfig network on  
[root@tianyun ~]# service NetworkManager stop  
[root@tianyun ~]# service network start
```

2. 创建桥

方法一 : virt-manager

方法二 : 配置文件

方法三 : virsh

To create a bridge (br0) based on the eth0 interface, execute the following command on the host:

```
[root@tianyun ~]# virsh iface-bridge eth0 br0
```

方法四：brctl (临时)

```
[root@tianyun ~]# brctl addbr br1
[root@tianyun ~]# brctl addif br1 eth1
[root@tianyun ~]# brctl stp br1 on
[root@tianyun ~]# ifconfig eth1 0
[root@tianyun ~]# ifconfig br1 1.1.1.1/24 up
```

Guest virtual machine configuration

1. 客户机配置文件 <offline>

```
[root@tianyun ~]# virsh edit tianyun
<interface type='bridge'>
<source bridge='br0' />
</interface>
```

2. 查看是否生效

```
[root@tianyun ~]# brctl show
```

参考：配置bridge

创建桥

```
[root@tianyun ~]# cat /etc/sysconfig/network-scripts/ifcfg-br0
DEVICE=br0
TYPE=Bridge
BOOTPROTO=none
IPADDR=192.168.0.230
PREFIX=24
GATEWAY=192.168.1.254
DNS1=8.8.8.8
ONBOOT=yes
DELAY=0
STP=on
```

DHCP:

```
DEVICE=br0
ONBOOT=yes
TYPE=Bridge
BOOTPROTO=dhcp
STP=on
DELAY=0
```

将物理接口桥接到br0

```
[root@tianyun ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
ONBOOT=yes
BRIDGE=br0
```

重启加载网络服务

```
[root@tianyun ~]# service network restart
```

查看当前桥接情况

```
[root@tianyun ~]# brctl show
```

```
root@tianyun:~#
File Edit View Search Terminal Help
[root@tianyun ~]# brctl show
bridge name      bridge id          STP enabled    interfaces
br0              8000.3c970e54c582    yes           eth0
                                         vnet4
pan0              8000.000000000000    no
virbr0            8000.525400623810    yes           virbr0-nic
                                         vnet0
                                         vnet1
                                         vnet2
                                         vnet3
[root@tianyun ~]# 
```

brctl

brctl 基本用法

```
[root@tianyun ~]# brctl --help
addbr  <bridge>          add bridge
delbr  <bridge>          delete bridge
addif <bridge> <device>  add interface to bridge
delif  <bridge> <device>  delete interface from bridge
```

```
[root@tianyun ~]# brctl show
bridge name      bridge id          STP enabled    interfaces
br0              8000.3c970e54c582    yes           eth0
                                         vnet0
br1              8000.000000000000    yes
virbr0            8000.525400d99d77    yes           virbr0-nic
                                         vnet1
```

```
[root@tianyun ~]# brctl addif br0 vnet1
device vnet1 is already a member of a bridge; can't enslave it to bridge br0.
[root@tianyun ~]# brctl delif virbr0 vnet1
[root@tianyun ~]# brctl addif br0 vnet1
[root@tianyun ~]# brctl show
```

手动删除桥：

1. 从桥中删除所有接口 brctl delif br0 vnet0
2. 停用桥 ifconfig br0 down
3. 删除桥 brctl delbr br0

4. 删除桥相关的配置文件 rm -rf /etc/sysconfig/network-scripts/ifcfg-br0
5. 将物理机IP配置到如eth0接口 vim /etc/sysconfig/network-scripts/ifcfg-eth0

KVM存储配置

KVM存储池管理

KVM存储池管理 (Storage Pool)

一、本地存储

Directory-based

Disk-based

Partition-based

LVM-based

```
[root@tianyun ~]# virsh pool-list --all
```

Name	State	Autostart
default	active	yes

```
-----
```

```
[root@tianyun ~]# ls /etc/libvirt/storage/
```

```
autostart default.xml
```

```
[root@tianyun ~]#
```

```
[root@tianyun ~]# ls /etc/libvirt/storage/autostart/
```

```
default.xml
```

```
[root@tianyun ~]# virsh pool-edit default
```

```
<target>
```

```
  <path>/var/lib/libvirt/images</path>
```

```
  <permissions>
```

```
    <mode>0755</mode>
```

```
    <owner>-1</owner>
```

```
    <group>-1</group>
```

```
  </permissions>
```

```
</target>
```

```
[root@tianyun ~]# ls /var/lib/libvirt/images/
new.img node1-1.img node1.img tianyun.img web2.img web2.qcow2
```

二、网络存储

NFS-based

GlusterFS-based

iSCSI-based

GFS-based

SAN-based

LAB : 使用NFS-based实现存储池

存储端：

```
[root@install ~]# mkdir /var/nfs-storage  
[root@install ~]# vim /etc/exports  
/var/nfs-storage 172.16.30.0/24(rw,sync,no_root_squash)
```

KVM端：

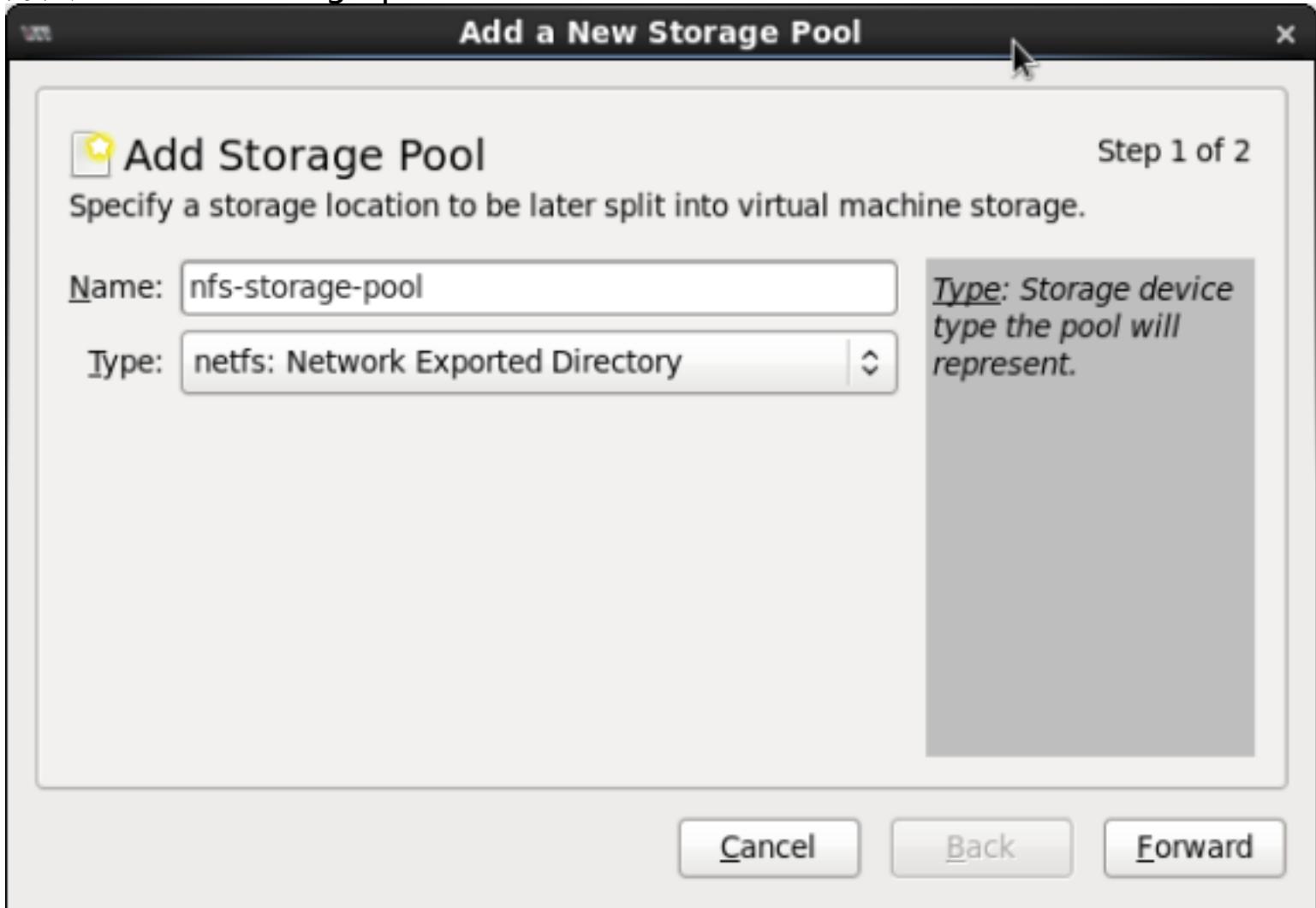
```
[root@tianyun ~]# showmount -e 172.16.8.100  
Export list for 172.16.8.100:  
/var/nfs-storage 172.16.30.0/24
```

方法一：/etc/fstab

```
[root@tianyun ~]# mkdir /var/lib/libvirt/images/nfs-storage-pool1/  
[root@tianyun ~]# vim /etc/fstab  
172.16.8.100:/var/nfs-storage /var/lib/libvirt/images/nfs-storage-pool1/ nfs defaults 0 0
```

```
[root@tianyun ~]# mount -a  
[root@tianyun ~]# df -P  
172.16.8.100:/var/nfs-storage 60476416 7085056 50319360 13% /var/lib/libvirt/images/nfs-storage-pool1
```

方法二：kvm storage pool



Add a New Storage Pool

X

Add Storage Pool

Step 2 of 2

Specify a storage location to be later split into virtual machine storage.

Target Path:

Format:

Host Name:

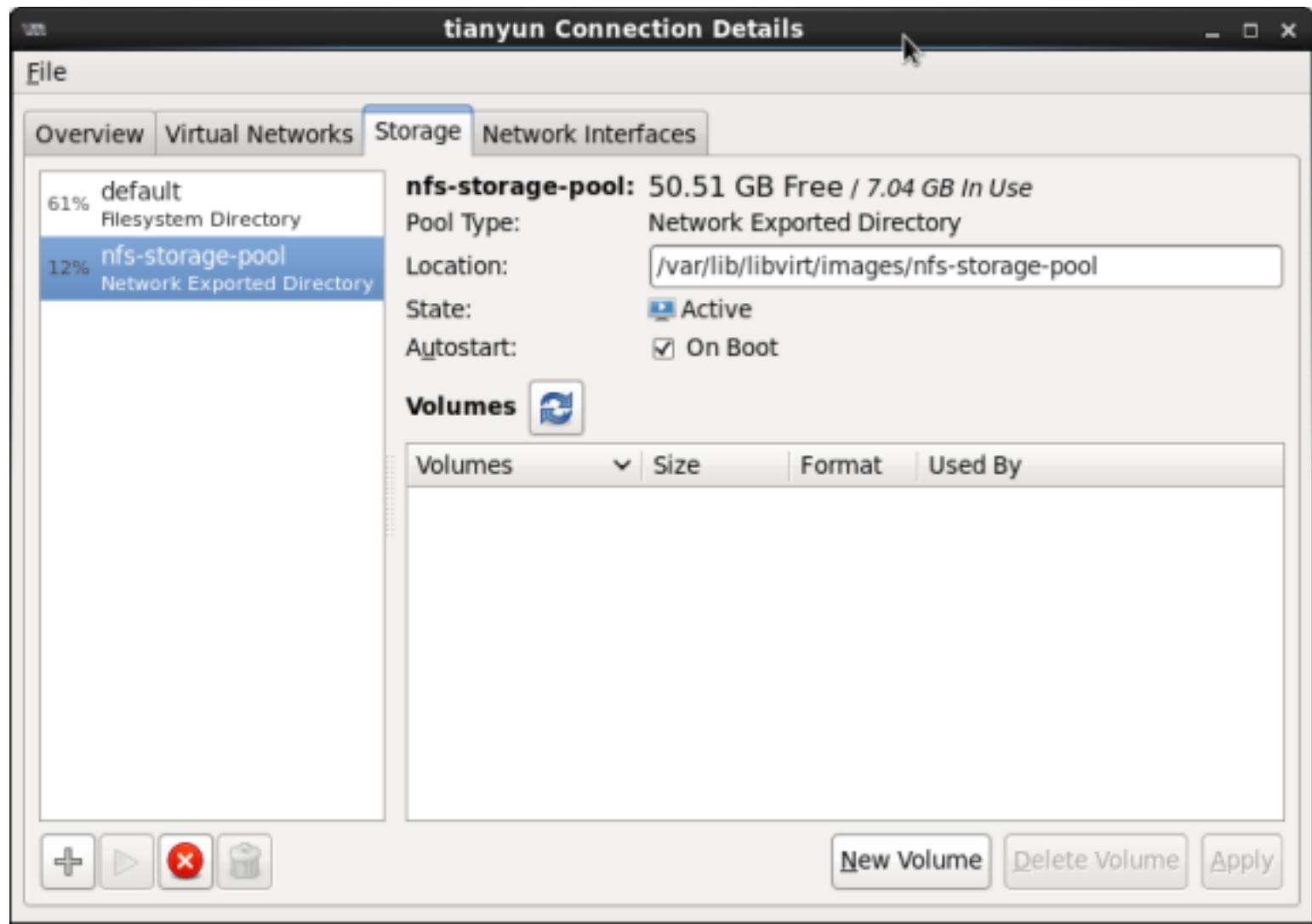
Source Path:

Source path: Path on
the host that is being
shared.

Cancel

Back

Finish



```
[root@tianyun ~]# virsh pool-list --all
```

Name	State	Autostart
default	active	yes
nfs-storage-pool	active	yes

```
[root@tianyun ~]# virsh pool-edit nfs-storage-pool
```

```
<pool type='netfs'>
  <name>nfs-storage-pool</name>
  <uuid>83d872db-154a-f721-a6ae-9d883583a739</uuid>
  <capacity unit='bytes'>61793632256</capacity>
  <allocation unit='bytes'>7562330112</allocation>
  <available unit='bytes'>54231302144</available>
  <source>
    <host name='172.16.8.100'/>
    <dir path='/var/nfs-storage'/'>
    <format type='nfs'/'>
  </source>
  <target>
    <path>/var/lib/libvirt/images/nfs-storage-pool</path>
    <permissions>
      <mode>0755</mode>
      <owner>0</owner>
      <group>0</group>
    </permissions>
  </target>
</pool>
```

方法三 : virsh pool-define-as

```
pool-create-as name --print-xml type [source-host] [source-path] [source-dev] [source-name]
```

```
[<target>] [--source-format format]
```

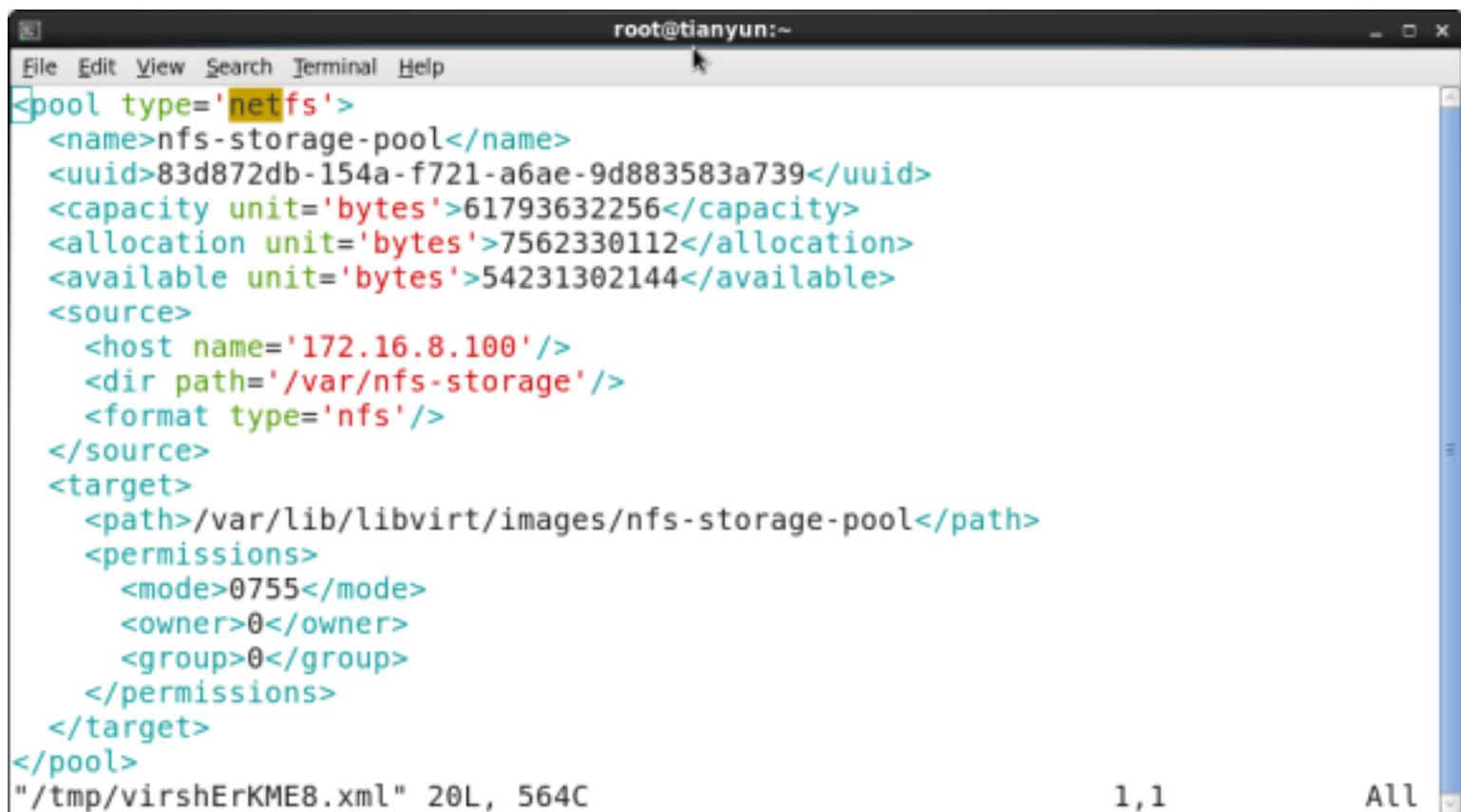
```
[root@tianyun ~]# mkdir /var/kvm-pool2
[root@tianyun ~]# virsh pool-define-as --source-format nfs my-pool2 --type netfs --source-host 172.16.8.100 --source-path /var/kvm-nfs --target /var/kvm-pool2
[root@tianyun ~]# virsh pool-autostart my-pool2
Pool my-pool2 marked as autostarted
[root@tianyun ~]# virsh pool-start my-pool2
Pool my-pool2 started
[root@tianyun ~]# virsh pool-autostart my-pool2
Pool my-pool2 marked as autostarted
```

安装虚拟机到存储池nfs-storage-pool:

```
[root@tianyun ~]# /usr/sbin/virt-install \
    --vnc \
    --name=tianyun \
    --ram=512 \
    --arch=x86_64 \
    --vcpus=1 \
    --os-type=linux \
    --os-variant=rhel6 \
    --hvm \
    --accelerate \
    --disk path=/var/lib/libvirt/images/nfs-storage-pool/tianyun.img,size=8 \
    --bridge=virbr0 \
    --location=ftp://172.16.8.100/rhel6.4 \
    --extra-args="ks=ftp://172.16.8.100/rhel6.4.ks"
```

```
[root@tianyun ~]# ls /etc/libvirt/storage/
autostart default.xml nfs.xml
```

```
[root@tianyun ~]# ls /etc/libvirt/storage/autostart/
default.xml nfs.xml
```



The screenshot shows a terminal window titled "root@tianyun:~". The window displays an XML configuration file for an NFS storage pool. The XML code is color-coded for syntax highlighting, with tags in blue and values in various colors like red, green, and cyan. The configuration includes details such as the pool type (netfs), name (nfs-storage-pool), UUID, capacity, allocation, available bytes, source information (host, directory, format), target path, and permissions (mode, owner, group). The file is located at "/tmp/virshErKME8.xml".

```
<pool type='netfs'>
  <name>nfs-storage-pool</name>
  <uuid>83d872db-154a-f721-a6ae-9d883583a739</uuid>
  <capacity unit='bytes'>61793632256</capacity>
  <allocation unit='bytes'>7562330112</allocation>
  <available unit='bytes'>54231302144</available>
  <source>
    <host name='172.16.8.100' />
    <dir path='/var/nfs-storage' />
    <format type='nfs' />
  </source>
  <target>
    <path>/var/lib/libvirt/images/nfs-storage-pool</path>
    <permissions>
      <mode>0755</mode>
      <owner>0</owner>
      <group>0</group>
    </permissions>
  </target>
</pool>
```

1,1

All

```
root@tianyun:~
```

```
[root@tianyun ~]# df
Filesystem      1K-blocks    Used Available Use% Mounted on
/dev/sdb5        35501808  21717964  11980428  65% /
tmpfs            3905984     980   3905004   1% /dev/shm
172.16.8.100:/var/nfs-storage
                           60345344  8171520  49085440  15% /var/lib/libvirt/images
/nfs-storage-pool
[root@tianyun ~]# 
```

方法四：基于xml添加存储池

```
virsh pool-define *.xml
```

qemu-img 镜像管理

qemu-img 磁盘镜像管理

一、QEMU支持的镜像文件格式

```
[root@tianyun ~]# qemu-img -h
Supported formats: raw cow qcow vdi vmdk cloop dmg bochs vpc vvfat qcow2 qed parallels nbd blkdebug host_cdrom
host_floppy host_device file
```

raw 原始磁盘镜像格式，容易移植到其它模拟器。

qcow2 QEMU推荐的镜像格式，支持稀疏文件，支持AES加密，支持基于 zlib 的压缩，[支持snapshot](#)，[支持后端镜像backing_file](#)

二、磁盘镜像格式转换

1. 查看镜像文件格式

```
[root@tianyun ~]# cd /var/lib/libvirt/images
[root@tianyun images]# qemu-img info rhel6u4_01.img
image: rhel6u4_01.img
file format: raw
```

virtual size: 8.0G (8589934592 bytes)

disk size: 1.5G

2. 创建镜像文件

```
[root@tianyun images]# qemu-img create -f qcow2 -o ? win7.qcow2
```

Supported options:

size	Virtual disk size
backing_file	File name of a base image
backing_fmt	Image format of the base image
encryption	Encrypt the image
cluster_size	qcow2 cluster size
preallocation	Preallocation mode (allowed values: off, metadata, full)

```
[root@tianyun images]# qemu-img create -f qcow2 -o size=20G win8.qcow2
```

Formatting 'win8.qcow2', fmt=qcow2 size=21474836480 encryption=off cluster_size=65536

```
[root@tianyun images]# qemu-img create -f qcow2 win7.qcow2 20G
```

Formatting 'win7.qcow2', fmt=qcow2 size=21474836480 encryption=off cluster_size=65536

后端镜像qcow2

```
[root@tianyun images]# qemu-img create -f qcow2 -b tianyun.img new.img 8G
```

Formatting 'new.img', fmt=qcow2 size=8589934592 backing_file='tianyun.img' encryption=off cluster_size=65536

```
[root@tianyun images]# qemu-img info tianyun.img
```

image: tianyun.img

file format: qcow2

virtual size: 8.0G (8589934592 bytes)

disk size: 1.6G

cluster_size: 65536

```
[root@tianyun images]# qemu-img info new.img
```

image: new.img

file format: qcow2

virtual size: 8.0G (8589934592 bytes)

disk size: 136K

cluster_size: 65536

backing file: tianyun.img

3. 转换镜像文件

```
[root@tianyun images]# qemu-img convert -f raw -O qcow2 rhel6u4_01.img rhel6u4_01.qcow2
```

```
[root@tianyun images]# qemu-img info rhel6u4_01.qcow2
```

4. 改变镜像文件大小

```
[root@tianyun images]# qemu-img resize node1.img +1G
```

5. qcow2镜像snapshot

注意：对KVM快照进行管理需要在关机状态下操作

```
[root@tianyun images]# qemu-img snapshot -l rhel6u4_01.qcow2
```

```
[root@tianyun images]# qemu-img snapshot -c sp1 rhel6u4_01.qcow2
```

```
[root@tianyun images]# qemu-img snapshot -a sp1 rhel6u4_01.qcow2
```

```
[root@tianyun images]# qemu-img snapshot -d sp1 rhel6u4_01.qcow2
```

LAB1：基于后端镜像创建多台虚拟机

LAB2：利用snapshot备份虚拟机不同阶段的状态

qemu-img commit !!!

qemu-img rebase

```
[root@tianyun images]# qemu-img convert -f qcow2 -O qcow2 rhel6-am-5.qcow2 rhel6-am-5.qcow2
```

KVM磁盘高级参数

KVM磁盘性能参数

Disk bus:

Virtio 使用半虚拟化驱动

Cache mode:

none 关闭缓存，读写数据都绕过缓存直接从块设备中读写；

writeback 回写模式

writethrough 直写模式

IO mode:

threads 让一个线程池去处理异步IO

native 只适用于cache=none，使用Linux原生的AIO

node1 Virtual Machine

File Virtual Machine View Send Key



- Overview
- Performance
- Processor
- Memory
- Boot Options
- VirtIO Disk 1**
- NIC :fe:7e:d6
- Tablet
- Mouse
- Display VNC
- Sound: ich6
- Serial 1
- Video
- Controller usb

Virtual Disk

Target device: VirtIO Disk 1

Source path: /var/lib/libvirt/images/node1.img

Storage size: 8.00 GB

Readonly:

Shareable:

Advanced options

Disk bus: **Virtio**

Serial number:

Storage format: **raw**

Performance options

Cache mode: **writeback**

IO mode: **threads**

Tip: 'source' refers to information seen from the host OS, while 'target' refers to information seen from the guest OS

Add Hardware

Remove

Cancel

Apply

```
root@tianyun:~  
File Edit View Search Terminal Help  
<on_crash>restart</on_crash>  
<devices>  
  <emulator>/usr/libexec/qemu-kvm</emulator>  
  <disk type='file' device='disk'>  
    <driver name='qemu' type='raw' cache='writeback' io='threads' />  
    <source file='/var/lib/libvirt/images/node1.img' />  
    <target dev='vda' bus='virtio' />  
    <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0' />  
  </disk>  
  <controller type='usb' index='0'>  
    <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x2' />  
  </controller>  
  <interface type='network'>  
    <mac address='52:54:00:fe:7e:d6' />  
    <source network='default' />  
    <model type='virtio' />  
    <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0' />  
  </interface>  
  <serial type='pty'>  
    22,5  
    47%  
</serial>
```

KVM虚拟机管理

virsh

Managing guest virtual machines with virsh

=====

非交互模式

交互模式

```
[root@tianyun ~]# virsh help  
[root@tianyun ~]# virsh version
```

连接其它主机的KVM

```
[root@tianyun ~]# virsh -c qemu:///system list  
[root@tianyun ~]# virsh -c qemu+ssh://root@172.16.8.100/system
```

Displaying the guest virtual machines 查看虚拟机

```
[root@tianyun ~]# virsh list --all
```

Getting the domain ID 获得虚拟机的domain ID

```
[root@tianyun ~]# virsh domid {domain-name or domain-uuid}
```

Getting the domain name 获得虚拟机的domain name

```
[root@tianyun ~]# virsh domname {domain-id or domain-uuid}
```

Getting the UUID

```
[root@tianyun ~]# virsh domuuid {domain-id or domain-name}
```

Displaying guest virtual machine information 显示虚拟机的详细信息

```
[root@tianyun ~]# virsh dominfo {domain-id, domain-name or domain-uuid}
```

```
[root@tianyun ~]# virsh dominfo tianyun
```

```
Id: -  
Name: tianyun  
UUID: 5759b246-d449-a1c5-fa44-238f95cd96f2  
OS Type: hvm  
State: shut off  
CPU(s): 1  
Max memory: 524288 KiB  
Used memory: 524288 KiB  
Persistent: yes  
Autostart: disable  
Managed save: no  
Security model: none  
Security DOI: 0
```

设置虚拟机开机自动启动/不自动启动

```
[root@tianyun ~]# virsh autostart node0
```

```
Domain node0 marked as autostarted
```

```
[root@tianyun ~]# ls /etc/libvirt/qemu/autostart/
```

```
node0.xml
```

暂停/恢复

```
[root@tianyun ~]# virsh suspend node1
```

```
[root@tianyun ~]# virsh resume node1
```

保存/加载

```
[root@tianyun ~]# virsh save node1 filename
```

```
[root@tianyun ~]# virsh restore filename
```

启动/停止/强制停止/重启

```
[root@tianyun ~]# virsh start node1
```

```
[root@tianyun ~]# virsh shutdown node1
```

```
[root@tianyun ~]# virsh destroy node1
```

```
[root@tianyun ~]# virsh reboot node1
```

```
[root@tianyun ~]# virsh domstate node0
```

Deleting guest virtual machine

```
[root@tianyun ~]# virsh destroy node1
```

```
[root@tianyun ~]# virsh undefine node1
```

```
[root@tianyun ~]# rm -rf /var/lib/libvirt/images/node1.img
```

virt-edit

```
[root@tianyun ~]# virt-edit -d rhel6-pm-4 /etc/grub.conf
```

```
[root@tianyun ~]# virt-edit -d rhel6-pm-3 /etc/shadow
```

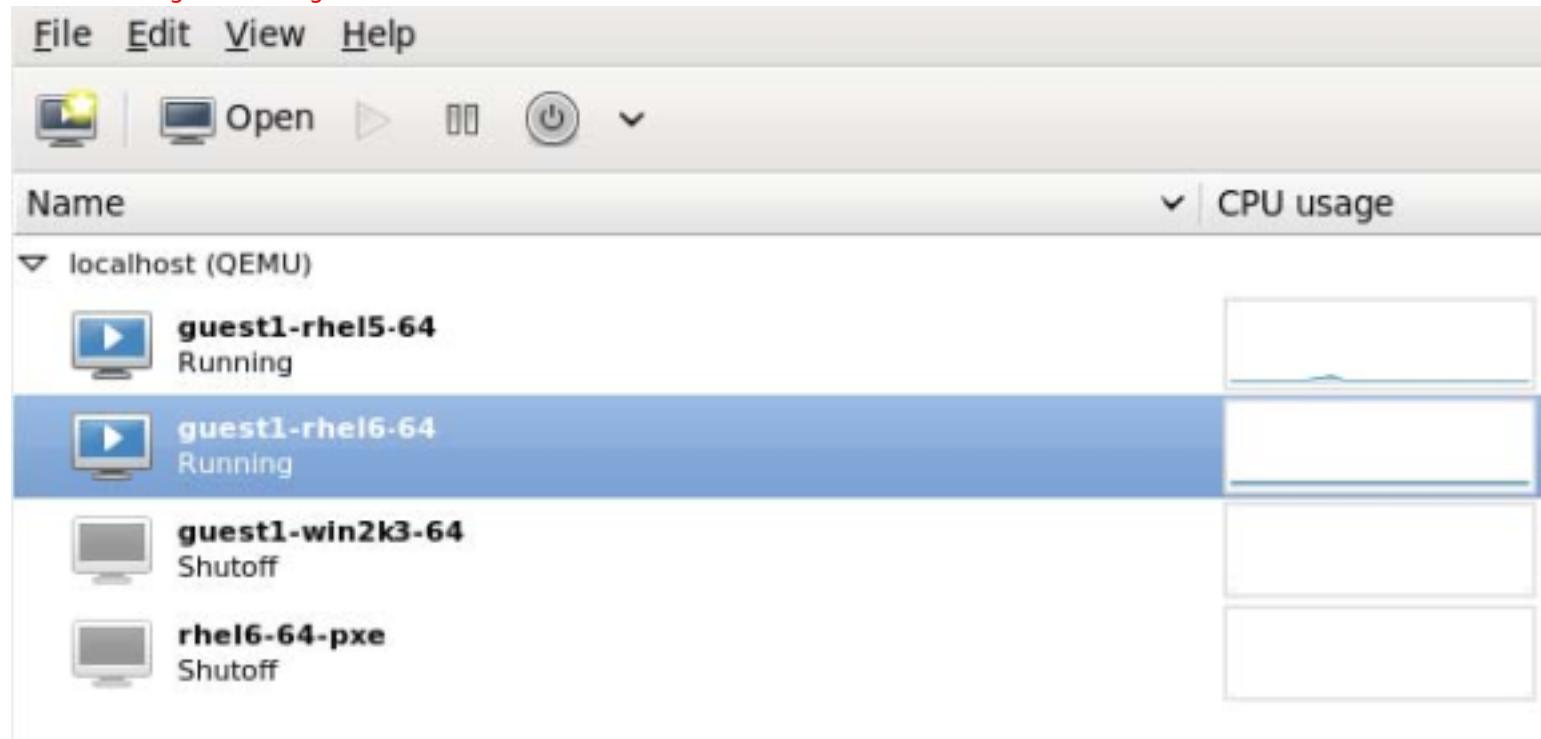
virt-manager

Managing guests with the Virtual Machine Manager (virt-manager)

virt-manager provides a graphical view of hypervisors and guests on your host system and on remote host systems. virt-manager can perform virtualization management tasks, including:

- defining and creating guests,
- assigning memory,
- assigning virtual CPUs,
- monitoring operational performance,
- saving and restoring, pausing and resuming, and shutting down and starting guests,
- links to the textual and graphical consoles, and
- live and offline migrations.

LAB1: Starting virt-manager



LAB2: The virtual hardware details window



virtual hardware details

Basic Details

Name: guest1-rhel6-64
UUID: b8d7388a-bbf2-db3a-e962-b97ca6e514bd
Status: Running
Description:

- Performance
- Processor
- Memory
- Boot Options
- VirtIO Disk 1
- NIC :79:35:e9
- Tablet
- Mouse
- Display VNC
- Sound: ich6
- Serial 1
- Video

Hypervisor Details

Hypervisor: kvm
Architecture: x86_64
Emulator: /usr/libexec/qemu-kvm

▷ **Machine Settings**

▷ **Security**

Add Hardware

Apply

LAB3: Adding a remote connection

Hypervisor: QEMU/KVM

Connect to remote host

Method: SSH

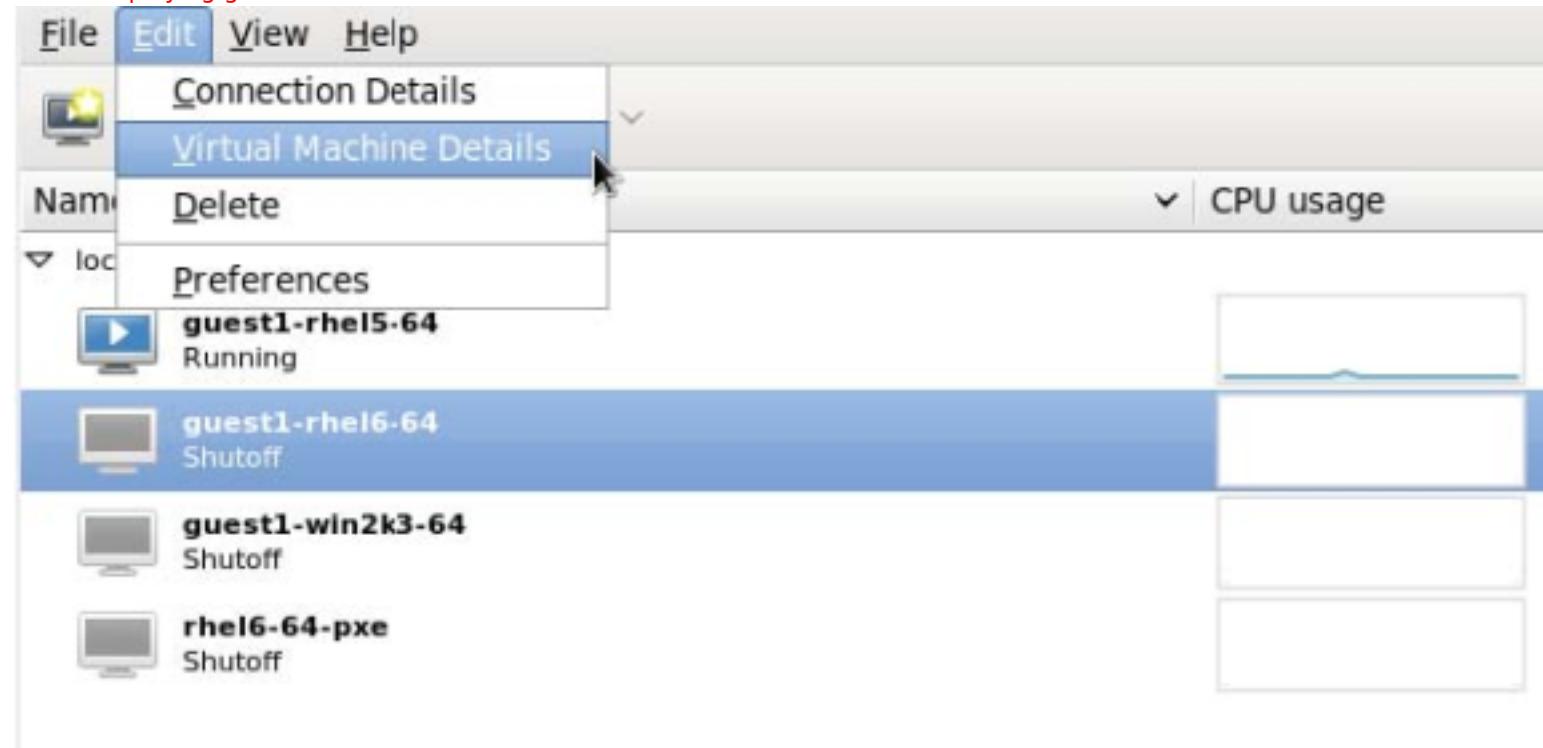
Username: root

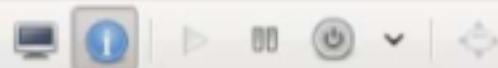
Hostname: dhcp-100-19-175

Autoconnect:

Generated URI: qemu+ssh://root@dhcp-100-19-175/system

LAB4: Displaying guest details





- Overview**
- Performance
- Processor
- Memory
- Boot Options
- VirtIO Disk 1
- IDE CDROM 1
- NIC :74:47:c2
- Tablet
- Mouse
- Display VNC
- Serial 1
- Video
- Controller IDE
- Controller usb

Add Hardware

Basic Details

Name:	VM-RHEL
UUID:	d2ce805d-caab-757a-e8e9-9b8f570259e2
Status:	Running
Description:	

Hypervisor Details

Hypervisor: kvm
Architecture: x86_64
Emulator: /usr/libexec/qemu-kvm

Operating System

Hostname: unknown
Product name: unknown

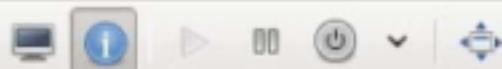
▷ Applications

▷ Machine Settings

▷ Security

Cancel

Apply



- Overview
- Performance**
- Processor
- Memory
- Boot Options
- VirtIO Disk 1
- NIC :79:35:e9
- Tablet
- Mouse
- Display VNC
- Sound: ich6
- Serial 1
- Video

Performance

CPU usage:

0 %

Memory usage:

2048 MB of 6033 MB

Disk I/O:

0 KB/s read
0 KB/s write

Network I/O:

0 KB/s in
0 KB/s out

Add Hardware

Apply



Overview
Performance
Processor
Memory
Boot Options
VirtIO Disk 1
NIC :79:35:e9
Tablet
Mouse
Display VNC
Sound: ich6
Serial 1
Video

CPUs

Logical host CPUs: 6

Current allocation: Maximum allocation:

▼ Configuration

Model:

Copy host CPU configuration

▼ Topology

 Manually set CPU topologySockets: Cores: Threads:

▼ Pinning

Default pinning: (ex: 0,1,3-5,7)

Generate from host NUMA configuration

Runtime pinning:

VCPU	On CPU	Pinning
0	3	0,1,2,3,4,5
1	0	0,1,2,3,4,5



Overview

Performance

Processor

Memory

Boot Options

VirtIO Disk 1

NIC :79:35:e9

Tablet

Mouse

Display VNC

Sound: ich6

Serial 1

Video

Memory

Total host memory: 6033 MB

Current allocation: 2048 MB

Maximum allocation: 2048 MB



- Overview
- Performance
- Processor
- Memory
- Boot Options
- IDE Disk 1
- IDE CDROM 1
- NIC :e8:05:34
- Tablet
- Mouse
- Display VNC
- Sound: ich6
- Serial 1
- USB 0d8c:000c
- Video
- Controller IDE
- Controller usb

Virtual Disk

Target device: IDE Disk 1

Source path: /home/guest-images/guest1-rhel6-64.img

Storage size: 12.00 GB

Readonly:

Shareable:

Advanced options

Cache mode:

Storage format:

Disk Bus:

Tip: 'source' refers to information seen from the host OS, while 'target' refers to information seen from the guest OS



Virtual Network Interface

Source device: Bridge 'br0'

Device model: virtio

MAC address: 52:54:00:79:35:e9

- Overview
- Performance
- Processor
- Memory
- Boot Options
- VirtIO Disk 1
- NIC :79:35:e9**
- Tablet
- Mouse
- Display VNC
- Sound: ich6
- Serial 1
- Video

LAB5: Performance monitoring

File Edit View Help

Name: guest1-rhel5-64 CPU usage

loc: Preferences

- guest1-rhel5-64** Running
- guest1-rhel6-64** Running
- guest1-win2k3-64** Shutoff
- rhel6-64-pxe** Shutoff

General Stats VM Details Feedback

Stats Options

Update status every seconds

Maintain history of samples

Enable Stats Polling

Disk I/O

Network I/O

[Close](#)

File Edit View Help

Guest CPU Usage

Host CPU Usage

Disk I/O (disabled)

Network I/O (disabled)

CPU usage

Name
192.168.122.1 (QEMU)

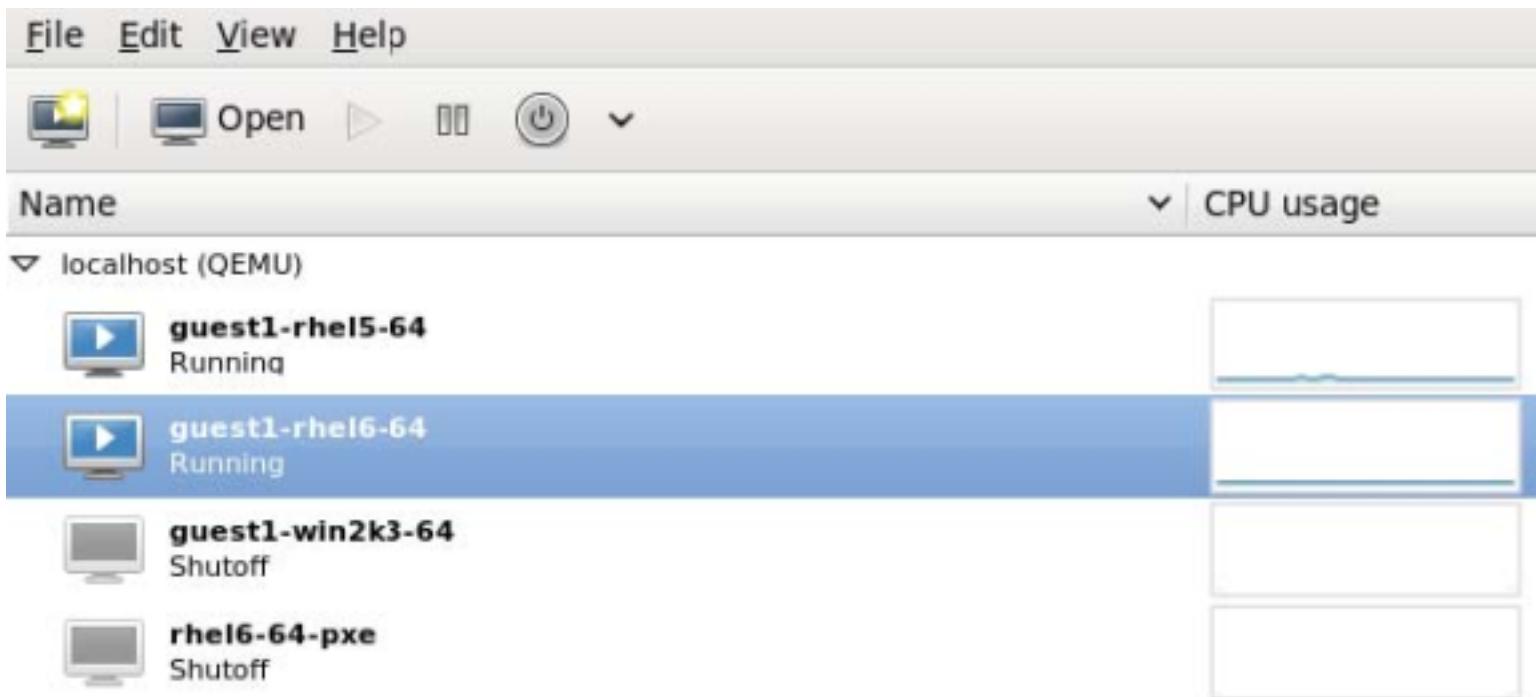
 VM-RHEL
Running

localhost (QEMU)

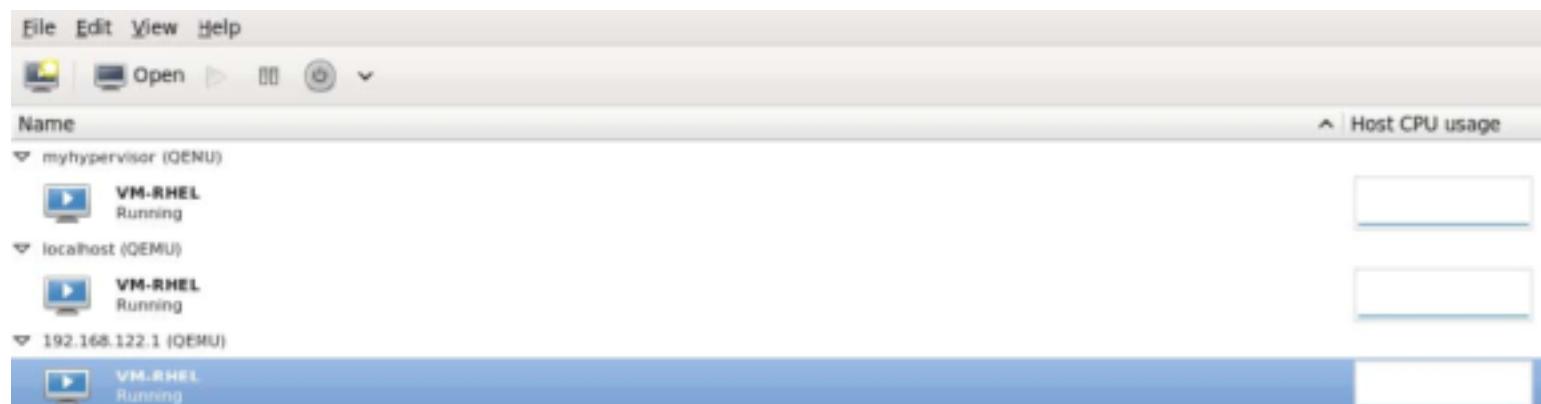
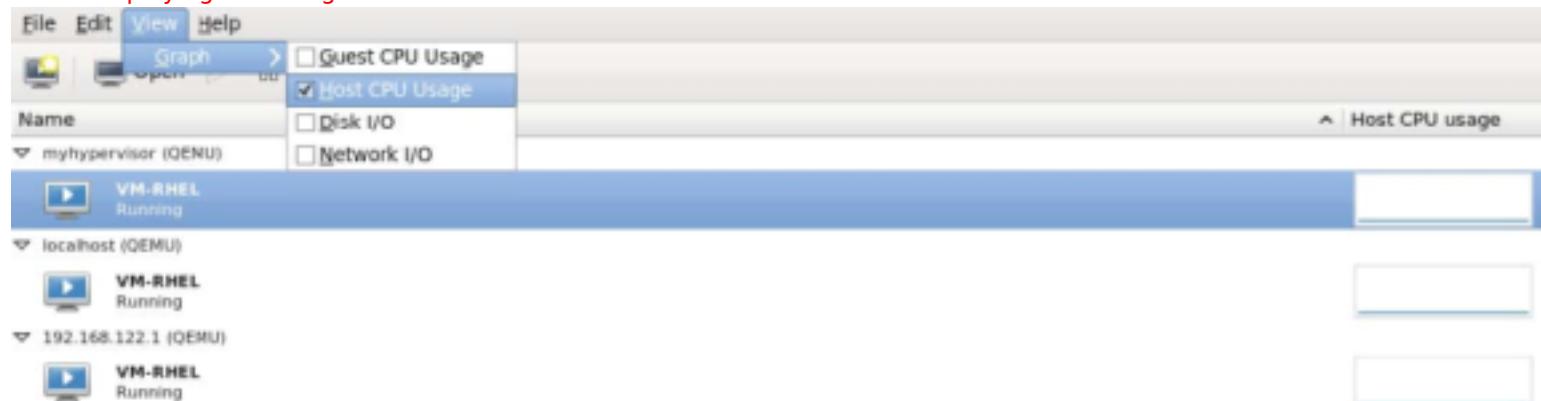
 VM-RHEL
Running

myhypervisor (QEMU)

 VM-RHEL
Running



LAB6: Displaying CPU usage for hosts



LAB7: Displaying Disk I/O

General Stats VM Details Feedback

Stats Options

Update status every seconds

Maintain history of samples

Enable Stats Polling

Disk I/O

Network I/O

[Close](#)

File Edit View Help

Graph >

Guest CPU Usage

Host CPU Usage

Disk I/O

Network I/O

Name	Age	Host CPU usage	Disk I/O
myhypervisor (QEMU)			
VM-RHEL			
localhost (QEMU)			
VM-RHEL			
192.168.122.1 (QEMU)			
VM-RHEL			

General Stats VM Details Feedback

Stats Options

Update status every seconds

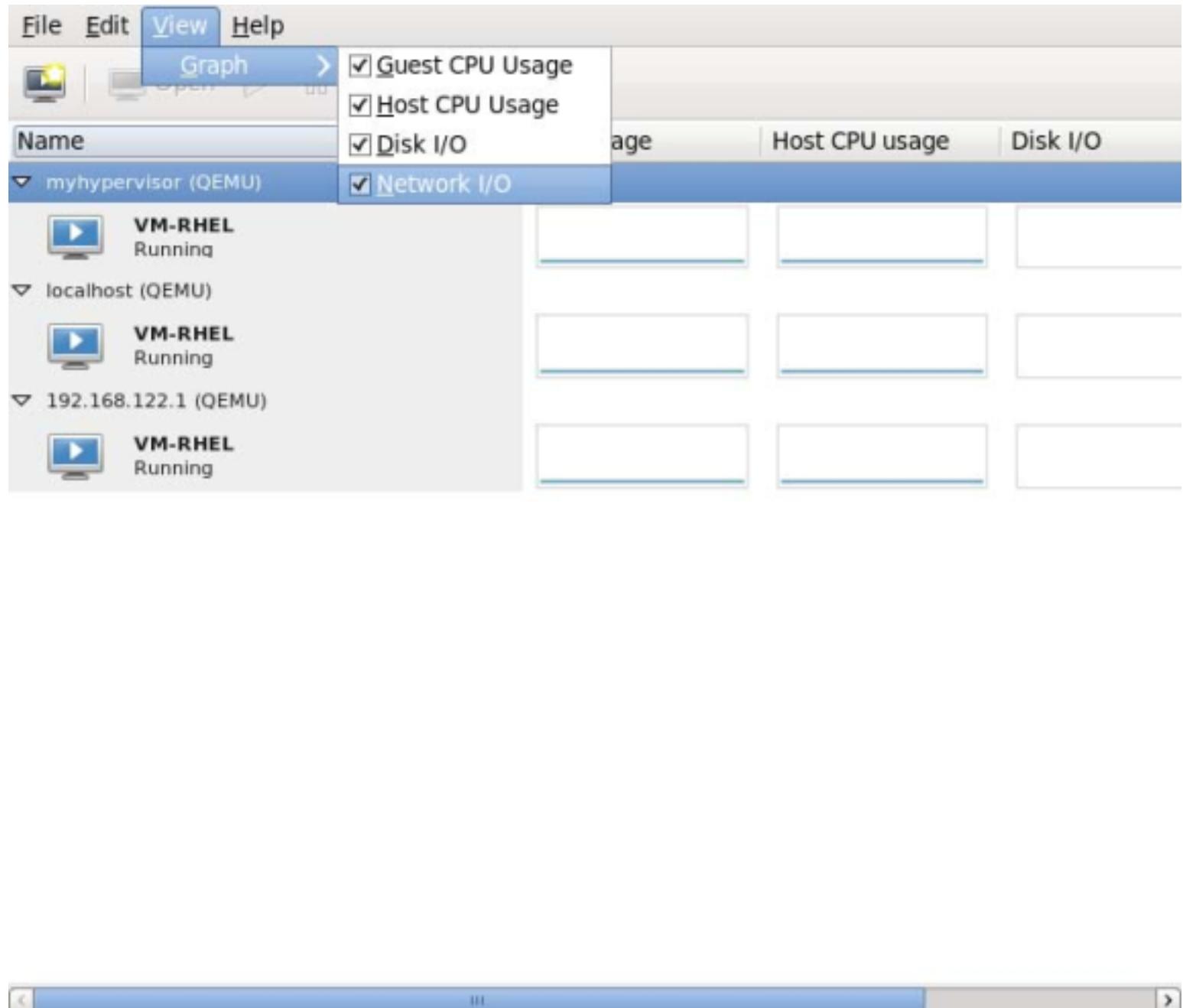
Maintain history of samples

Enable Stats Polling

Disk I/O

Network I/O

[Close](#)



virt-clone

Clone:

```
[root@tianyun ~]# virt-clone -o test3 -n test4 -f /var/lib/libvirt/images/test4.img
```

-o ORIGINAL_GUEST, --original=ORIGINAL_GUEST 原始客体的名称；必须为关闭或者暂停状态。
-n NEW_NAME, --name=NEW_NAME 新客户端的名称
-f NEW_DISKEFILE --file=NEW_DISKEFILE 作为新客户端磁盘映像的新文件

热添加硬盘

热添加硬盘

方法一：

```
[root@tianyun ~]# virsh list
Id  Name          State
-----
1   rhel6-am-2    running
2   rhel6-am-3    running

[root@tianyun ~]# virsh domblklist rhel6-am-3
Target  Source
-----
vda    /var/lib/libvirt/images/rhel6-am-3.qcow2

[root@tianyun ~]# qemu-img create -f qcow2 /disk10.img 1G
Formatting '/disk10.img', fmt=qcow2 size=1073741824 encryption=off cluster_size=65536

[root@tianyun ~]# virsh attach-disk rhel6-am-3 --source /disk10.img --target vdb --cache writeback --subdriver qcow2
Disk attached successfully

[root@tianyun ~]# virsh domblklist rhel6-am-3
Target  Source
-----
vda    /var/lib/libvirt/images/rhel6-am-3.qcow2
vdb    /disk10.img

[root@tianyun ~]# virsh attach-disk rhel6-pm-1 --source /disk30.img --target vdc --cache writeback --subdriver qcow2 --
persistent
```

方法二：基于xml文件

```
[root@tianyun ~]# qemu-img create -f qcow2 /disk20.img 1G
Formatting '/disk20.img', fmt=qcow2 size=1073741824 encryption=off cluster_size=65536

[root@tianyun ~]# vim temp.xml
<disk type='file' device='disk'>
  <driver name='qemu' type='qcow2' cache='writeback' />
  <source file='/disk20.img' />
  <target dev='vdc' bus='virtio' />
</disk>

[root@tianyun ~]# virsh attach-device rhel6-am-3 temp.xml
Device attached successfully

[root@tianyun ~]# virsh domblklist rhel6-am-3
Target  Source
-----
vda    /var/lib/libvirt/images/rhel6-am-3.qcow2
vdb    /disk10.img
vdc    /disk20.img
```

强制移除：

```
[root@tianyun ~]# virsh detach-disk rhel6-am-3 vdb
Disk detached successfully

[root@tianyun ~]# virsh domblklist rhel6-am-3
Target  Source
-----
vda    /var/lib/libvirt/images/rhel6-am-3.qcow2
vdc    /disk20.img
```

热添加网卡

添加内存

添加**CPU**

P2V

V2V

KVM半虚拟化驱动

Linux使用半虚拟机驱动

disk, network默认使用virtio

添加的新设备同样选择半虚拟化驱动程序

Windows使用半虚拟化驱动

Installing the drivers on an installed Windows guest virtual machine

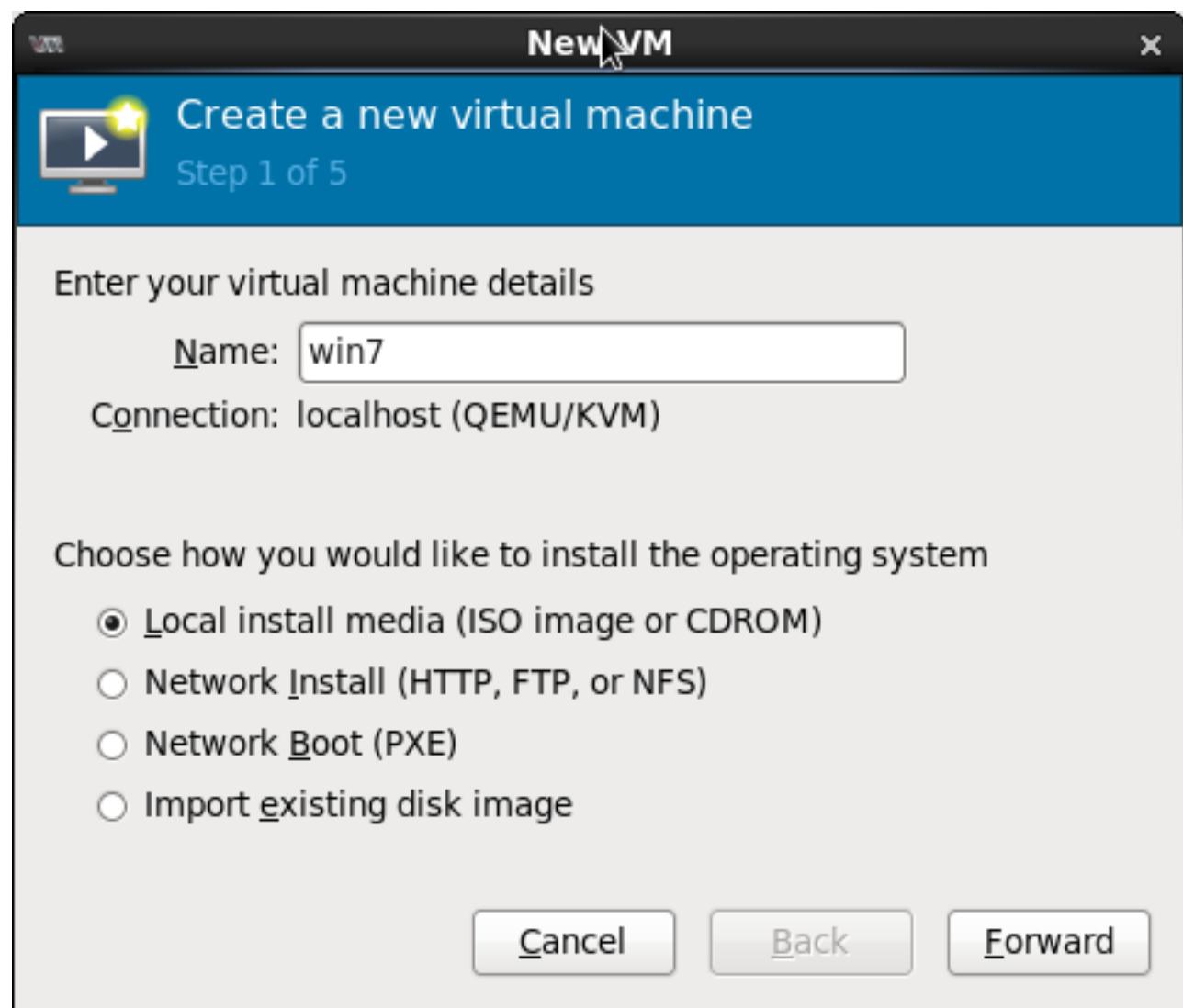
Windows installation on a Windows 7 virtual machine

disk, network默认使用的是全虚拟化驱动程序，效率低

一、下载KVM Para-virtualized Drivers

virtio-win-1.1.1.iso (disk, network)

二、安装Windows时提供半虚拟化驱动



New VM



Create a new virtual machine

Step 2 of 5

Locate your install media

Use CDROM or DVD

Use ISO image:

/media/ssd_data/iso_images/WIN7x32x64.iso

[Browse...](#)

Choose an operating system type and version

OS type:

Version:

[Cancel](#)

[Back](#)

[Forward](#)

New VM



Create a new virtual machine
Step 3 of 5

Choose Memory and CPU settings

Memory (RAM): MB

Up to 7628 MB available on the host

CPUs:

Up to 4 available

Cancel

Back

Forward

New VM



Create a new virtual machine

Step 4 of 5

Enable storage for this virtual machine

Create a disk image on the computer's hard drive

20.0 GB

18.3 Gb available in the default location

Allocate entire disk now 

Select managed or other existing storage

Browse...

Cancel

Back

Forward



Create a new virtual machine

Step 5 of 5

Ready to begin installation of **win7**

OS: Generic

Install: Local CDROM/ISO

Memory: 4096 MB

CPUs: 1

Storage: 20.0 Gb /var/lib/libvirt/images/win7.img

Customize configuration before install

▽ Advanced options

Virtual network 'default' : NAT



Set a fixed MAC address

52:54:00:d9:7a:b2

Virt Type: kvm



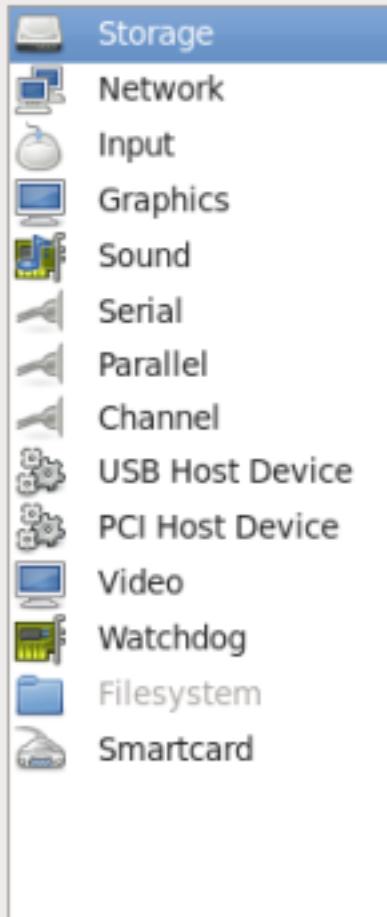
Architecture: x86_64



Cancel

Back

Finish



Storage



Please indicate how you would like to assign space on the host system for your virtual storage device.

- Create a disk image on the computer's hard drive

8.0 GB

18.3 Gb available in the default location

Allocate entire disk now

- Select managed or other existing storage

[Browse...](#)

/root/Desktop/ftp/Virtualization/virtio-win-1.1.16.iso

Device type:

IDE cdrom

Cache mode:

default

Storage format:

[Cancel](#)

[Finish](#)

 Begin Installation  Cancel

-  Overview
-  Processor
-  Memory
-  Boot Options
-  **Disk 1**
-  NIC :92:0c:1d
-  Input
-  Display VNC
-  Sound: default
-  Console
-  Video

Virtual Disk

Target device: Disk 1

Source path: /var/lib/libvirt/images/win7.img

Storage size: Unknown

Readonly:

Shareable:

Advanced options

Disk bus: Virtio 

Serial number:

Storage format: raw 

Performance options

 **Tip:** 'source' refers to information seen from the host OS,
while 'target' refers to information seen from the guest OS

Add Hardware

Remove

Cancel

Apply

 Begin Installation  Cancel

-  Overview
-  Processor
-  Memory
-  Boot Options
-  VirtIO Disk 1
-  NIC :92:0c:1d
-  Input
-  Display VNC
-  Sound: default
-  Console
-  Video

Virtual Network Interface

Source device: Virtual network 'default' : NAT

Device model: virtio

MAC address: 52:54:00:92:0c:1d

[Add Hardware](#)[Remove](#)[Cancel](#)[Apply](#)

File Virtual Machine View Send Key



安装 Windows

您想将 Windows 安装在何处？

	名称	总计大小	可用空间	类型

刷新 (R)

驱动器选项 (高级) (A)

加载驱动程序 (L)

未找到任何驱动器。单击“加载驱动程序”提供用于安装的大容量存储驱动程序。

下一步 (N)

1 收集信息

2 安装 Windows

File Virtual Machine View Send Key



安装 Windows

选择要安装的驱动程序。

Red Hat VirtIO SCSI controller (E:\viosster\Win7\x86\virtio.inf)

 检查与此计算机上的硬件不兼容的驱动程序 (O)。

浏览 (B)

重新扫描 (R)

下一步 (N)

1 收集信息

2 安装 Windows

File Virtual Machine View Send Key



1 收集信息

2 安装 Windows

三、基于已安装的Windwos安装半虚拟化驱动

加载virtio.iso



Storage

Please indicate how you would like to assign space on the host system for your virtual storage device.

- Create a disk image on the computer's hard drive

 GB

17.9 Gb available in the default location

Allocate entire disk now

- Select managed or other existing storage

Browse...

/usr/share/virtio-win/virtio-win.iso

Device type:

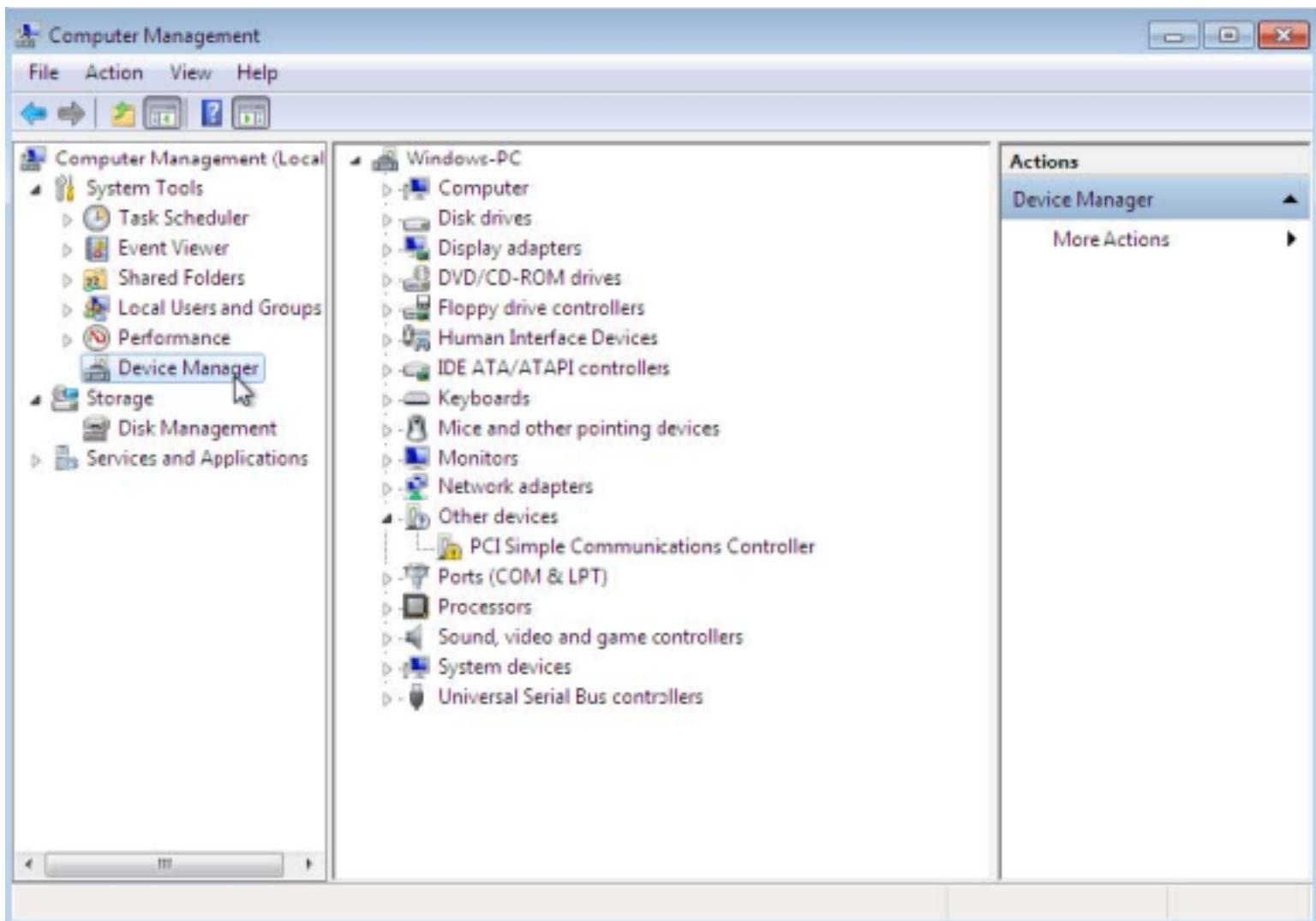
Cache mode:

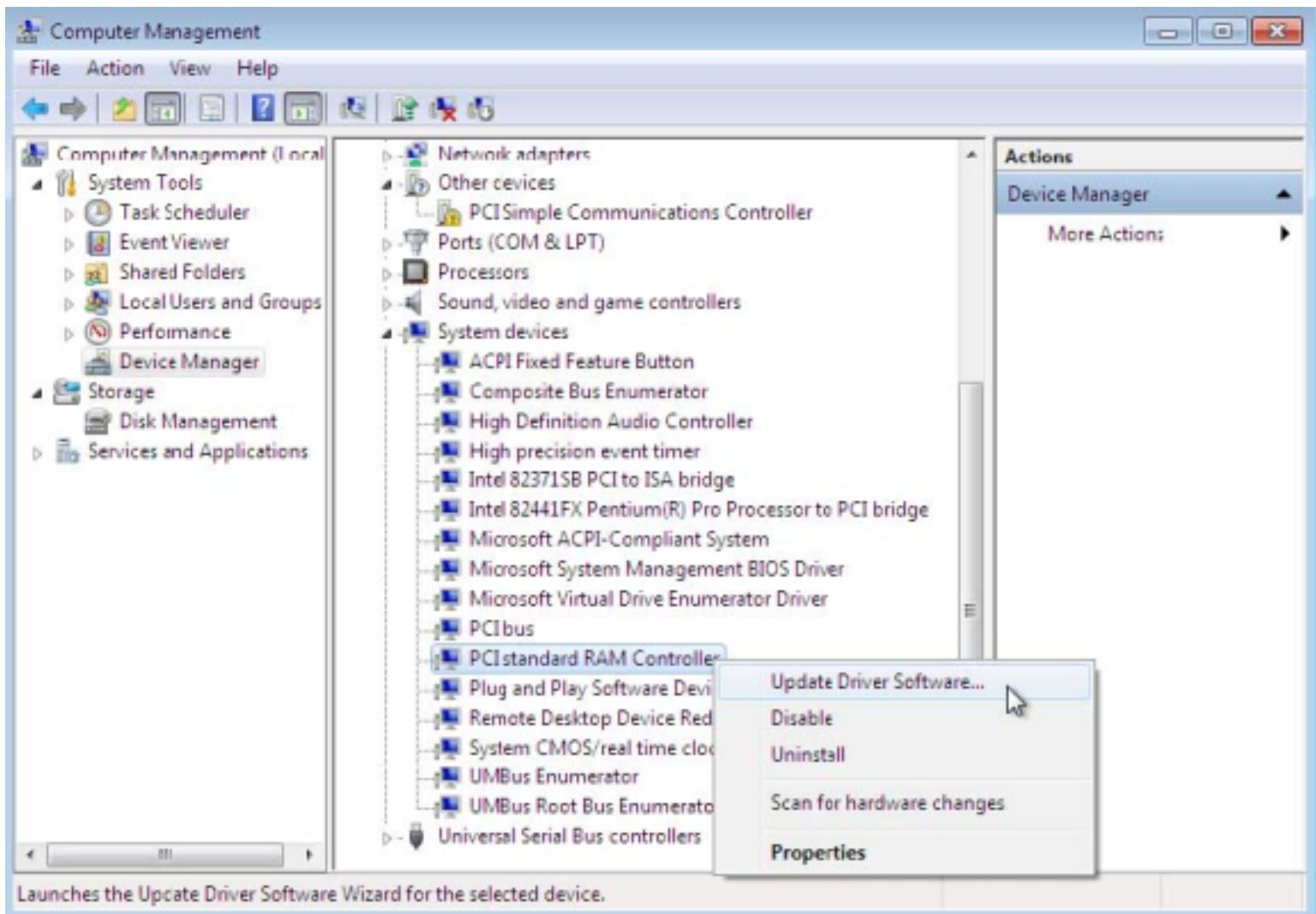
Storage format:

Cancel

Finish

安装驱动







How do you want to search for driver software?

→ Search automatically for updated driver software

Windows will search your computer and the Internet for the latest driver software for your device, unless you've disabled this feature in your device installation settings.

→ Browse my computer for driver software

Locate and install driver software manually.



Cancel



Browse for driver software on your computer

Search for driver software in this location:

[Browse...](#) [Include subfolders](#)

→ Let me pick from a list of device drivers on my computer

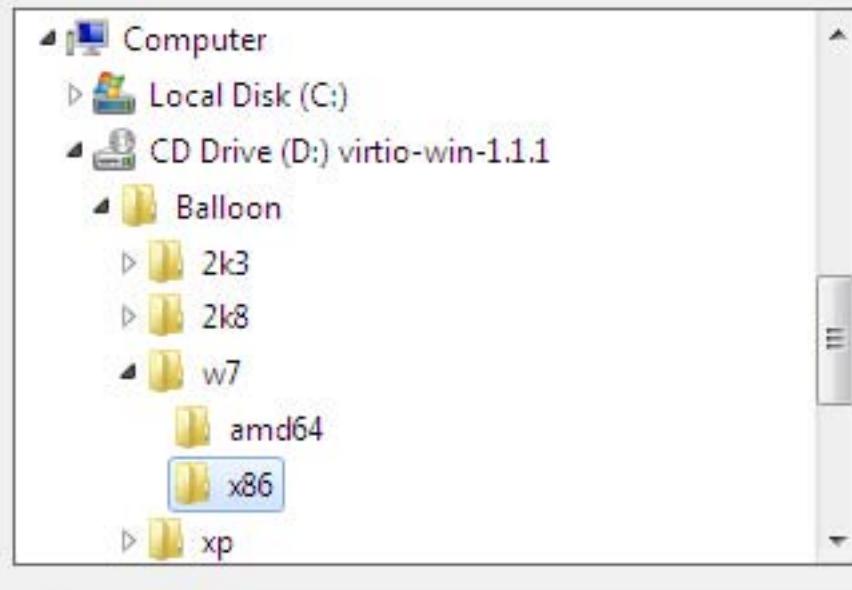
This list will show installed driver software compatible with the device, and all driver software in the same category as the device.

[Next](#)[Cancel](#)

Browse For Folder



Select the folder that contains drivers for your hardware.

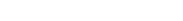
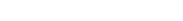


Folder:

x86

OK

Cancel





Update Driver Software - PCI standard RAM Controller

Browse for driver software on your computer

Search for driver software in this location:

D:\Balloon\w7\x86

[Browse...](#)

[Include subfolders](#)

Let me pick from a list of device drivers on my computer

This list will show installed driver software compatible with the device, and all driver software in the same category as the device.

[Next](#)

[Cancel](#)



Update Driver Software - VirtIO Balloon Driver

Windows has successfully updated your driver software

Windows has finished installing the driver software for this device:

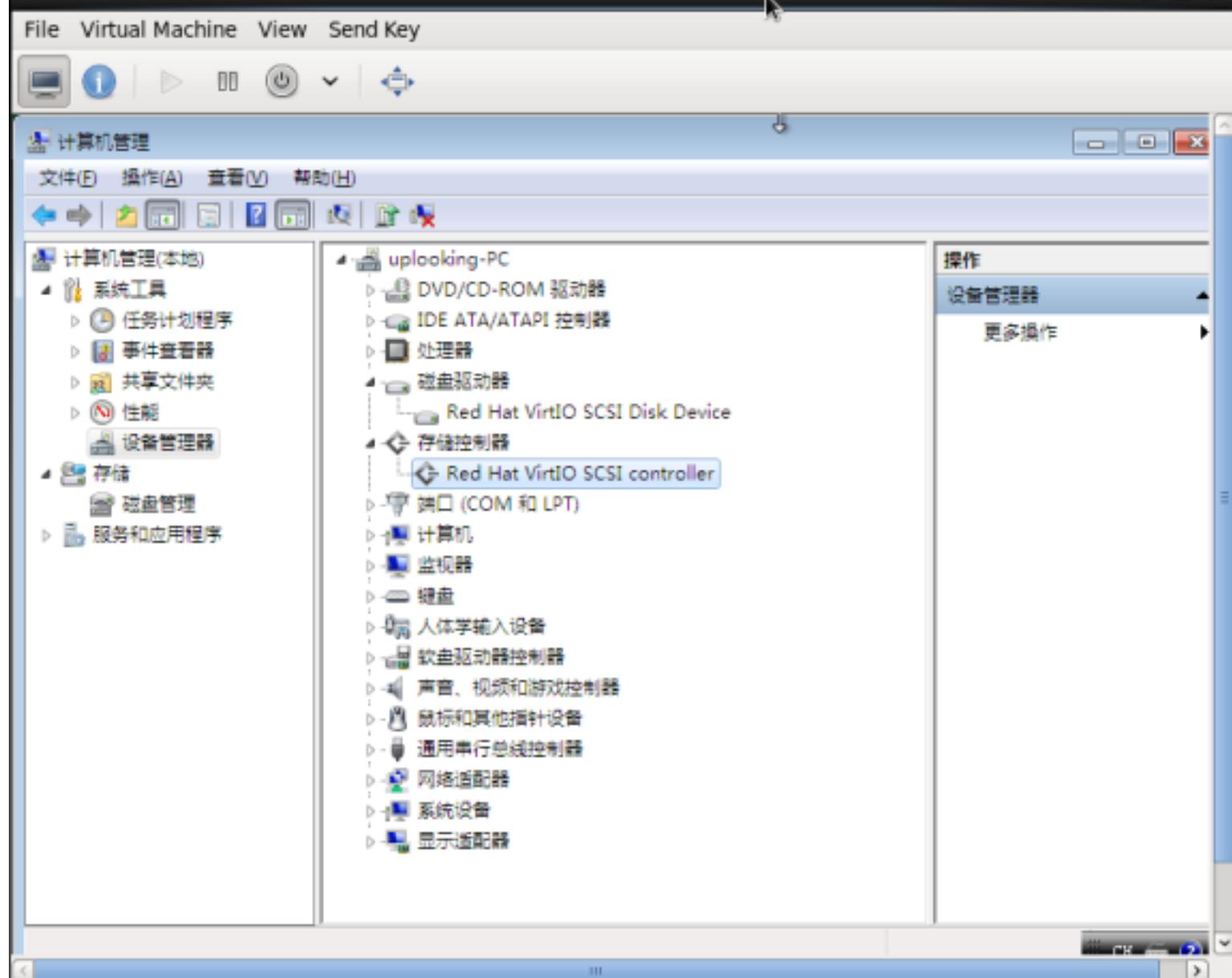


VirtIO Balloon Driver

[Close](#)

Reboot

windows7 Virtual Machine



KVM实现在线迁移

在线迁移准备工作

在线迁移准备工作

一、Live migration requirements 在线迁移的需求

Migrating guest virtual machines requires the following:

1. Migration requirements

A guest virtual machine installed on [shared storage](#) using one of the following protocols:

Fibre Channel-based LUNs

iSCSI

FCoE

NFS

GFS2

SCSI RDMA protocols (SCSI RCP): the block export protocol used in Infiniband and 10GbE iWARP

2. Make sure that the libvirtd service is enabled

二、Live migration and Red Hat Enterprise Linux version compatibility 版本兼容性

Migration Method	Release Type	Example	Live Migration Support	Notes
Forward	Major release	5.x → 6.y	Not supported	
Forward	Minor release	5.x → 5.y ($y > x, x \geq 4$)	Fully supported	Any issues should be reported
Forward	Minor release	6.x → 6.y ($y > x, x \geq 0$)	Fully supported	Any issues should be reported
Backward	Major release	6.x → 5.y	Not supported	
Backward	Minor release	5.x → 5.y ($x > y, y \geq 4$)	Supported	Refer to Troubleshooting problems with migration for known issues
Backward	Minor release	6.x → 6.y ($x > y, y \geq 0$)	Supported	Refer to Troubleshooting problems with migration for known issues

三、Shared storage example: NFS for a simple migration

KVM: 172.16.70.1 172.16.70.88

Storage: 172.16.70.17

1. 准备NFS存储 172.16.70.17
/nas1 *(rw,no_root_squash,sync)

2. KVM主机挂载存储
存储池配置NFS挂载

3. 在任意一台KVM上创建虚拟机（克隆一样）
虚拟机磁盘文件选择NFS存储

4. 主机名解析

virsh实现在线迁移

virsh实现在线迁移

Example: live migration with virsh

host1.example.com host2.example.com
guest1-rhel6-64 -----to----->

1. Verify the guest virtual machine is running

[root@kvm1 ~]# virsh list

Id	Name	State
1	rhel6-am-2	running
2	rhel6-am-3	running
3	tianyun	running

2. Migrate the guest virtual machine

[root@ host1 ~]# virsh migrate --live guest1-rhel6-64 qemu+ssh://host2.example.com/system

3. Wait

4. Verify the guest virtual machine has arrived at the destination host

[root@ host2 ~]# virsh list

Id	Name	State
10	guest1-rhel6-64	running

注：Nor-running的虚拟机迁移方法不能使用virsh migrate

virsh dumpxml Guest1 > Guest1.xml

virsh -c qemu+ssh://<target-system-FQDN>/system define Guest1.xml

virsh undefine Guest1

[root@kvm1 ~]# virsh dumpxml tianyun > tianyun.xml

[root@kvm1 ~]# virsh -c qemu+ssh://kvm2.tianyun.com/system define tianyun.xml

root@kvm2.tianyun.com's password:

Domain tianyun defined from tianyun.xml

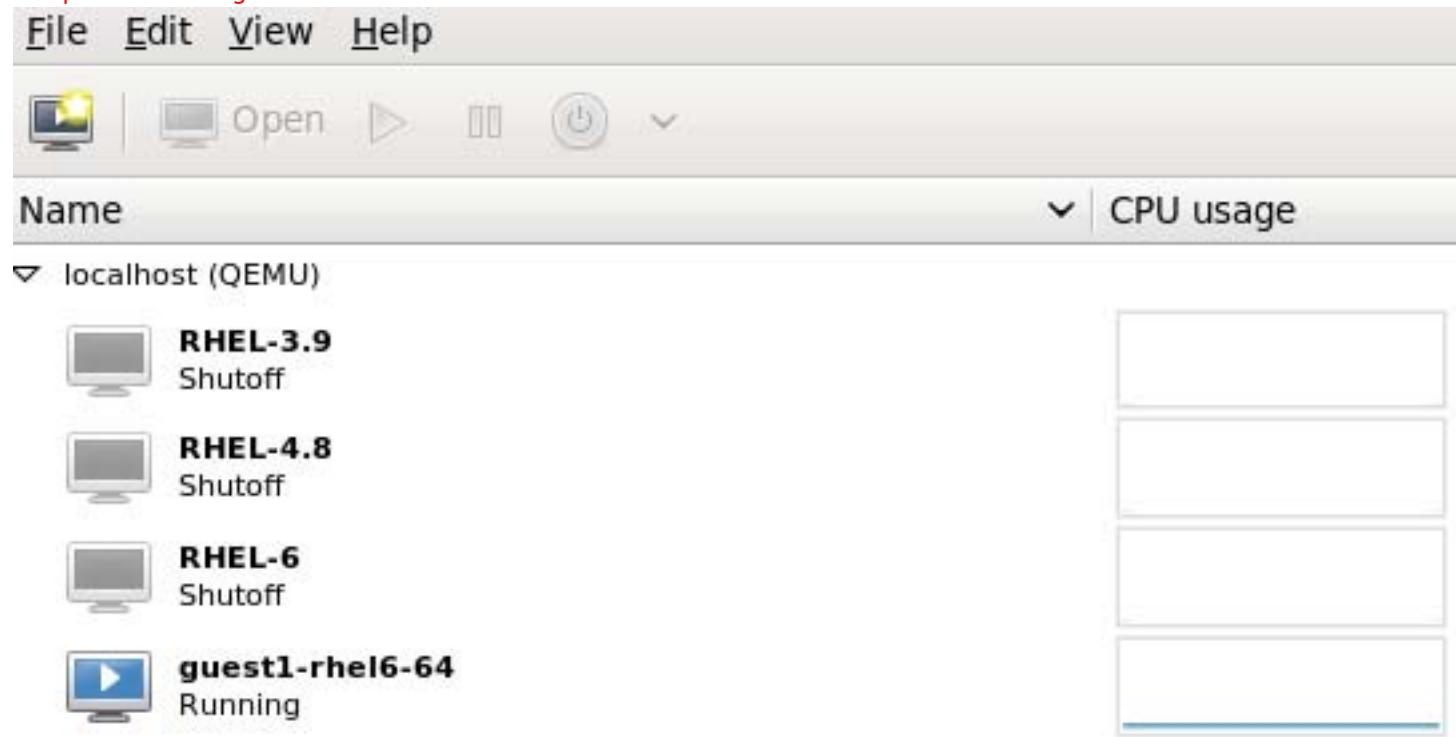
[root@kvm1 ~]# virsh undefine tianyun

Domain tianyun has been undefined

***virt-manager*实现迁移**

***virt-manager*实现在线迁移**

1. Open *virt-manager*



2. Connect to the target host physical machine

File Edit View Help

Add Connection...

Close

Ctrl+W

Quit

Ctrl+Q

▼ CPU usage

localhost (QEMU)



RHEL-3.9

Shutoff



RHEL-4.8

Shutoff



RHEL-6

Shutoff



guest1-rhel6-64

Running



Hypervisor:

QEMU/KVM



Connect to remote host

Method:

SSH



Username:

root

Hostname:

virtlab22



Autoconnect:

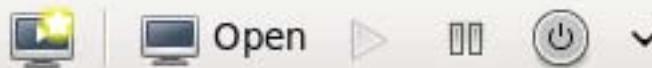


Generated URI: qemu+ssh://root@virtlab22/system

Cancel

Connect

3. Migrate guest virtual machines



Name

CPU usage

localhost (QEMU)

RHEL-3.9
Shutoff

RHEL-4.8
Shutoff

RHEL-6
Shutoff

guest1-rhel6-64
Running

virtlab22 (QEMU)

RHEL6
Shutoff

Run

Pause

Shut Down >

Clone...

Migrate...

Delete

Open



Migrate 'guest1-rhel6-64'

Name: guest1-rhel6-64

Original host: virtlab18

New host: virtlab22 (QEMU)

Migrate offline:

Advanced options

Tunnel migration through libvirt's daemon:

Max downtime: 30 ms

Connectivity

Address:

Port: 49152

Bandwidth: 0 Mbps

[Cancel](#)

[Migrate](#)



Migrating VM 'guest1-rhel6-64' from virtlab18
to virtlab22. This may take awhile.

Migrating domain

87% 912 MB

[Cancel](#)

File Edit View Help



Name

CPU usage

localhost (QEMU)



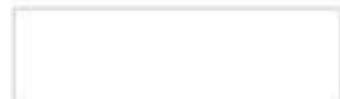
RHEL-3.9

Shutoff



RHEL-4.8

Shutoff



RHEL-6

Shutoff



guest1-rhel6-64

Shutoff



virtlab22 (QEMU)



RHEL6

Shutoff



guest1-rhel6-64

Running



注：主机名解析的问题 /etc/hosts

virsh console 连接

virsh console 本地连接虚拟机

注：没有安装图形，不知道虚拟机IP

File Edit View Search Terminal Help

```
[root@tianyun ~]# virsh console node1
Connected to domain node1
Escape character is ^]
```

```
Red Hat Enterprise Linux Server release 6.4 (Santiago)
Kernel 2.6.32-358.el6.x86_64 on an x86_64
```

```
localhost.localdomain login: root
Password:
Last login: Mon Aug  3 01:12:08 on ttyS0
[root@localhost ~]# 
```

Red Hat Enterprise Linux 6 Virtualization Host Configuration and Guest Installation Guide

File Edit View Go Help

Previous Next 138 of 166 100% ▲

machine.

Solution

Set up a serial console in the guest's XML file.

Procedure B.1. Setting up a serial console in the guest's XML

1. Add the following XML to the guest virtual machine's XML using `virsh edit`:

```
<serial type='pty'>
  <target port='0' />
</serial>
<console type='pty'>
  <target type='serial' port='0' />
</console>
```

2. Set up the console in the guest kernel command line.

To do this, either log in to the guest virtual machine to edit the `/boot/grub/grub.conf` file directly, or use the `virt-edit` command line tool. Add the following to the guest kernel command line:

console=ttyS0,115200

3. Run the following command:

virsh start vm && virsh console vm

vnc 远程连接虚拟机

vnc 远程连接连接虚拟机

```
[root@tianyun ~]# # vi /etc/libvirt/qemu.conf
vnc_listen = "0.0.0.0"
```

```
[root@tianyun ~]# service libvirtd restart
```

```
[root@tianyun ~]# virsh edit web2
<graphics type='vnc' port='5910' autoport='no'/>
```

```
[root@tianyun ~]# netstat -tnlp |grep :59
tcp      0      0 127.0.0.1:5900          0.0.0.0:*          LISTEN    7969/qemu-kvm
tcp      0      0 127.0.0.1:5901          0.0.0.0:*          LISTEN    2861/qemu-kvm
tcp      0      0 127.0.0.1:5902          0.0.0.0:*          LISTEN    2902/qemu-kvm
tcp      0      0 127.0.0.1:5903          0.0.0.0:*          LISTEN    2944/qemu-kvm
tcp      0      0 127.0.0.1:5904          0.0.0.0:*          LISTEN    3000/qemu-kvm
tcp      0      0 127.0.0.1:5906          0.0.0.0:*          LISTEN    7123/qemu-kvm
tcp      0      0 0.0.0.0:5910           0.0.0.0.*          LISTEN    12296/qemu-kvm
```

```
[root@tianyun ~]# ps aux |grep web2
qemu    13226  7.7  2.5 924336 198440 ? S  15:21 0:10 /usr/libexec/qemu-kvm -name web2 -S -M rhel6.4.0 -enable-
```

```
kvm -m 512 -smp 1,sockets=1,cores=1,threads=1 -uuid f3231efa-08c5-09b9-79d6-5f3e0cc309ce -nodefconfig -nodefaults -chardev socket,id=charmonitor,path=/var/lib/libvirt/qemu/web2.monitor,server,nowait -mon chardev=charmonitor,id=monitor,mode=control -rtc base=utc -no-shutdown -device piix3-usb-uhci,id=usb,bus=pci.0,addr=0x1.0x2 -drive file=/var/lib/libvirt/images/web2.qcow2,if=none,id=drive-virtio-disk0,format=qcow2,cache=none -device virtio-blk-pci,scsi=off,bus=pci.0,addr=0x4,drive=drive-virtio-disk0,id=virtio-disk0,bootindex=1 -drive file=/var/lib/libvirt/images/web2.img,if=none,id=drive-virtio-disk1,format=raw,cache=none -device virtio-blk-pci,scsi=off,bus=pci.0,addr=0x6,drive=drive-virtio-disk1,id=virtio-disk1 -netdev tap,fd=35,id=hostnet0,vhost=on,vhostfd=36 -device virtio-net-pci,netdev=hostnet0,id=net0,mac=52:54:00:d9:b3:3a,bus=pci.0,addr=0x3 -netdev tap,fd=37,id=hostnet1,vhost=on,vhostfd=38 -device virtio-net-pci,netdev=hostnet1,id=net1,mac=52:54:00:71:a4:73,bus=pci.0,addr=0x7 -chardev pty,id=charserial0 -device isa-serial,chardev=charserial0,id=serial0 -device usb-tablet,id=input0 -vnc 0.0.0.0:20 -vga cirrus -device virtio-balloon-pci,id=balloon0,bus=pci.0,addr=0x5
```

批量部署虚拟机

Linux镜像模板

创建镜像模板

安装虚拟机

qcow2
磁盘size

调整配置

1. network

```
/etc/init.d/NetworkManager stop  
chkconfig NetworkManager off
```

```
>/etc/udev/rules.d/70-persistent-net.rules
```

```
vim /etc/sysconfig/network-scripts/ifcfg-eth0 (删除mac,uuid)  
vim /etc/sysconfig/network-scripts/ifcfg-eth1 (删除mac,uuid)
```

2. yum repolist

3. raw —> qcow2

4. iptables/selinux

5. ssh公钥认证

6. 配置本地console连接

7. 配置从远程vnc连接

创建XML模板

1. memory 256M (**XML适用于clone**)
2. 磁盘/网卡使用半虚拟化驱动程序 (**XML适用于clone**)

Windows模板

安装虚拟机 qcow2

移除Windows个性化信息

批量部署部署

一、Clone

准备模板主机

自动产生虚拟机的配置文件和磁盘文件

速度慢

```
[root@tianyun images]# virt-clone -o rhel6-am-1 -n tianyun -f /var/lib/libvirt/images/tianyun.img
```

二、通过后端镜像文件

准备后端镜像qcow2

只产生虚拟机的磁盘文件

手动创建虚拟机配置文件 (name,uuid,disk,mac)

```
[root@tianyun ~]# qemu-img create -f qcow2 -b node1.img node2.img
```

```
[root@tianyun ~]# cp /etc/libvirt/qemu/node1.xml /etc/libvirt/qemu/node2.xml
<name>node2</name>
<uuid>69d608b9-4bab-4f44-a7d8-684a88072163</uuid>
<source file='/var/lib/libvirt/images/node2.img' />
<mac address='52:54:00:fe:7e:d7' />
```

```
[root@tianyun ~]# virsh define /etc/libvirt/qemu/node2.xml
```

```
[root@tianyun ~]# virsh start node2
```

```
[root@tianyun ~]# vm_uuid=$(uuidgen); echo $vm_uuid
```

```
[root@tianyun ~]# MACADDR="52:54:$($(dd if=/dev/urandom count=1 2>/dev/null | md5sum | sed -r 's/^(..)(..)(..)(..).*$\A1:\2:\3:\4/')"; echo $MACADDR
```

```
[root@yang ~]# vim create_kvm.sh
```

```
[root@tianyun images]# openssl rand -hex 6
e67ed8281e16
```

```
[root@tianyun images]# openssl rand -hex 6 |sed -r 's/(..)\1:/g; s/.$//'
```

```
[root@tianyun images]# openssl rand -hex 6 |sed -r -e 's/(..)\1:/g' -e 's/.$//'
```

```
[root@tianyun images]# openssl rand -hex 6 |sed -r 's/(..)(..)(..)(..)(..)\1:\2:\3:\4:\5:\6/'
```

```
24:02:d2:57:72:68
```

企业级虚拟化解决方案

企业级虚拟化概述

Vmware vSphere

Red Hat RHEV

Oracle OVM

Ovirt

Openstack

本节作业

1. 什么是虚拟化？
2. 半虚拟化和全虚拟化的区别？
3. 有哪些虚拟化产品Hypervisor？
4. KVM有哪些特性？
5. QEMU和KVM是什么关系？
6. 虚拟机使用网段：192.168.133.0/24, 192.168.144.0/24, 192.168.155.0/24

7. libvirt的作用是什么？
8. SR-IOV
9. 如果查看虚拟机的某个网卡连接到哪个桥设备？
10. virsh dumpxml tianyun
11. vish net-dumpxml mynet
12. virsh pool-dumpxml mypool
13. 在没有图形的情况下，如何修改kvm虚拟机的密码？
14. 描述P2V,V2V的详细流程
15. 描述热添加硬盘的详细流程及注意事项
16. 描述热添加网卡的详细流程及注意事项
17. 描述添加CPU的详细流程及注意事项
18. 描述添加MEM的详细流程及注意事项
19. 如何制作Linux模板
20. 如何制作Windows 2008模板

```
virsh start|net-start|pool-start tianyun|mynet|mypool  
virsh autostart|net-autostart|pool-autostart tianyun|mynet|mypool  
virsh edit|net-edit|pool-edit tianyun|mynet|mypool  
virsh define|net-define|pool-define tianyun|mynet|mypool  
virsh undefine|net-undefine|pool-undefine tianyun|mynet|mypool
```

```
[root@tianyun ~]# virsh --help |grep destroy  
destroy          destroy (stop) a domain  
iface-destroy    destroy a physical host interface (disable it / "if-down")  
net-destroy      destroy (stop) a network  
nodedev-destroy  destroy (stop) a device on the node  
pool-destroy     destroy (stop) a pool
```

四、集群技术

集群技术概述

LB

HA

HPC

LB LVS (L4)

LVS概述

LVS

Linux Virtual Server , Linux虚拟服务器
基于四层的LB

一、LVS 概述

LVS是Linux内核的一部分，因此性能较高

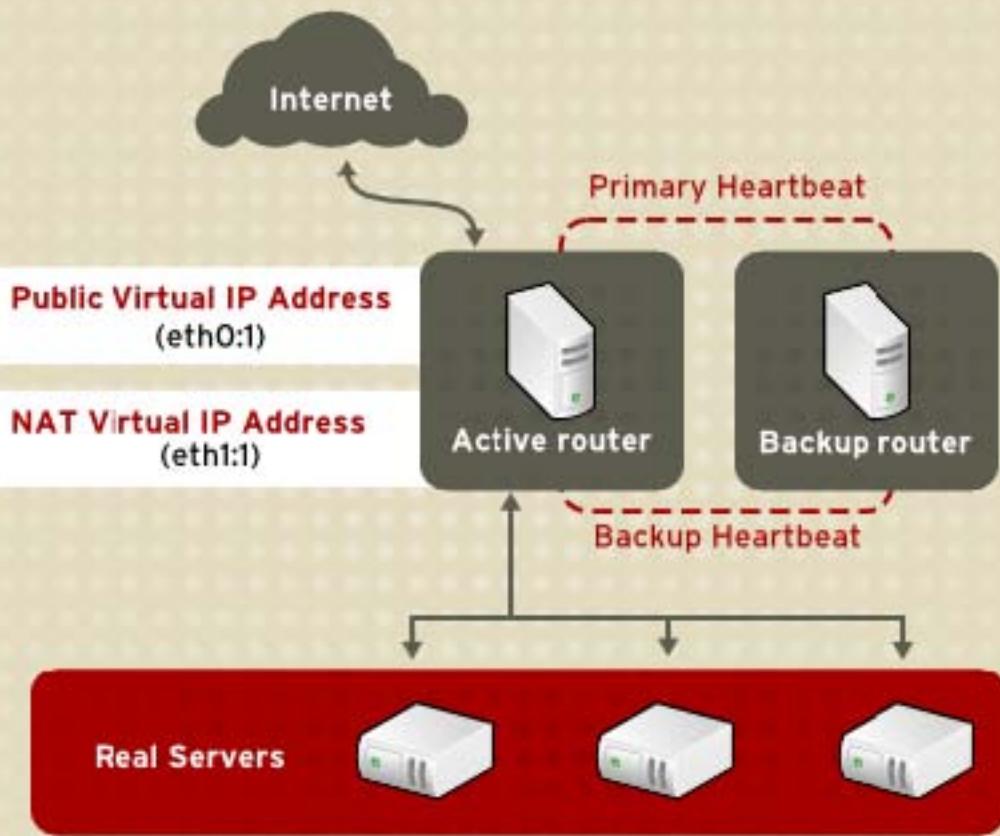
Linux虚拟服务器（即分发器或调度器）：不真正提供服务，但接受客户的访问，为整个集群提供一个唯一的入口

虚拟服务器和真实服务器（Real Server）通信

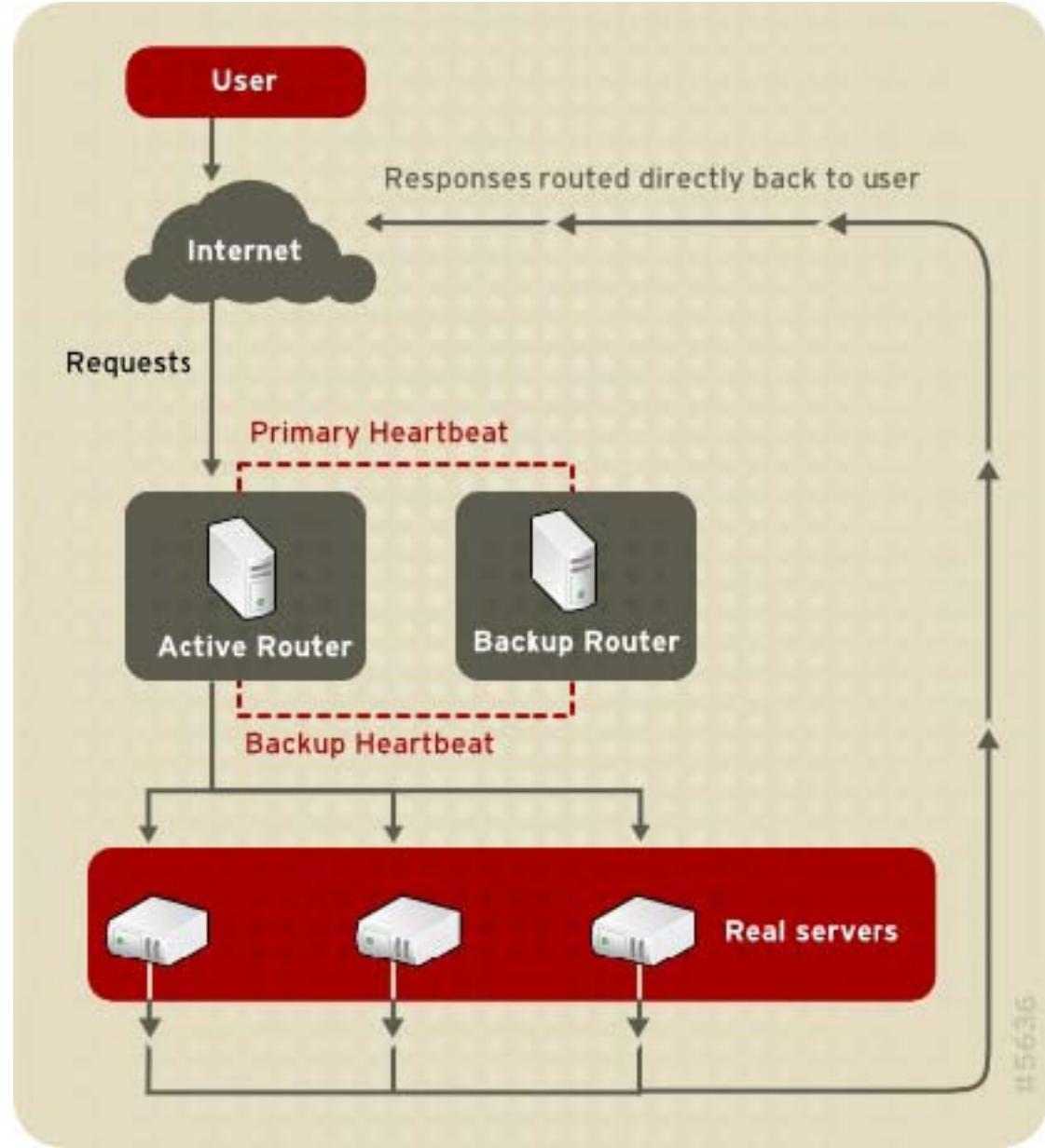
真实服务器（Real Server）：真正提供服务，集群中每个Real Server可以是物理机，也可以是虚拟机

二、LVS三种基本模式

VS/NAT： 网络地址转换模式，进站/出站的数据流量经过分发器



VS/DR : 直接路由模式，只有进站的数据流量经过分发器
调度器和真实服务器必须在同一网段



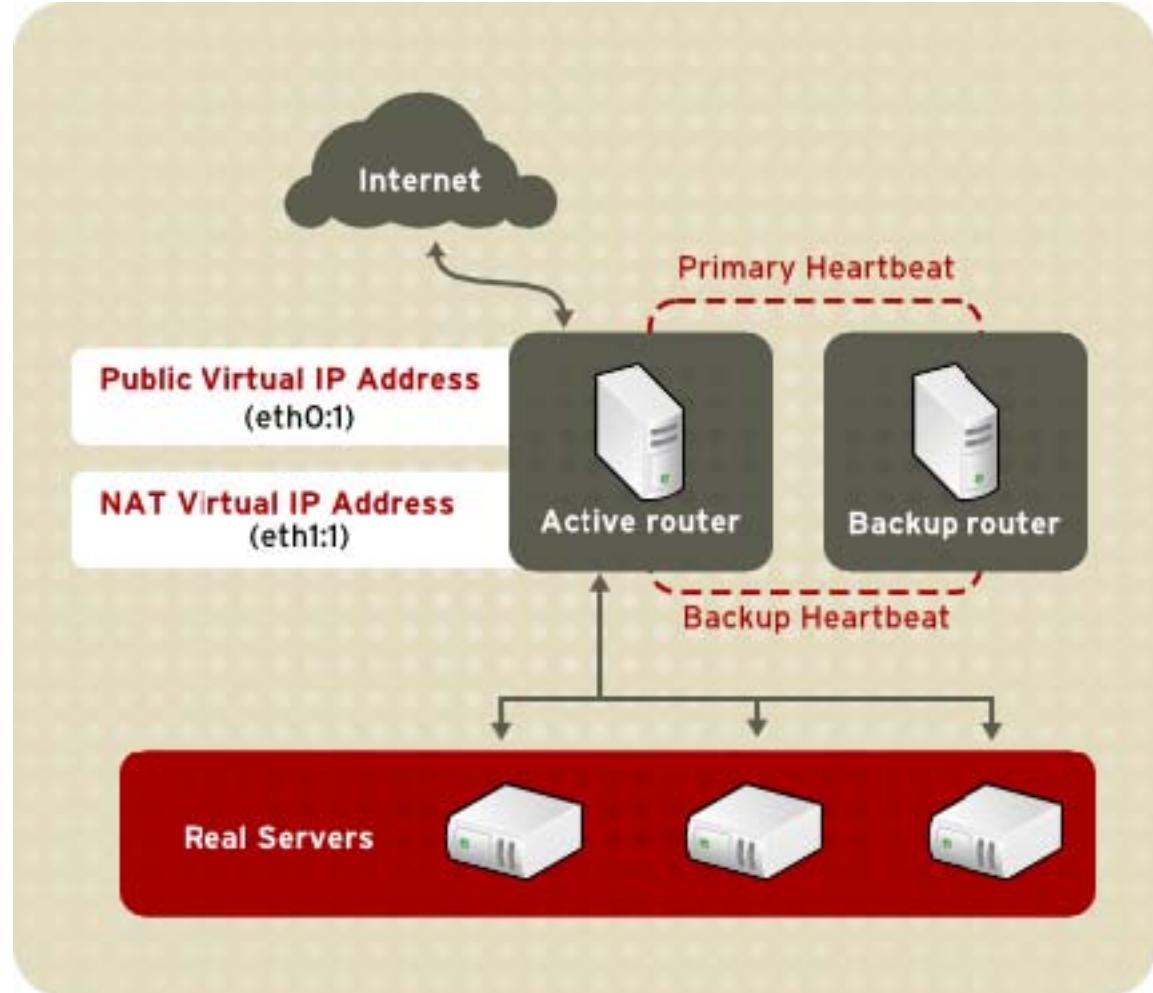
VS/TUN : 隧道模式，只有进站的数据流量经过分发器

VS/NAT

VS/NAT

实验说明：

1. KVM网络使用NAT模式
2. 调度器、Real Server都使用虚拟机或使用真实服务器
3. 在虚拟机上添加网卡，使用半虚拟化驱动
- 4.



网络拓朴：

Client: 192.168.122.0/24

Director : eth0: 192.168.122.10 VIP : 192.168.122.100

eth1: DIP: 10.10.10.1

Real Server : RIP: 10.10.10.10 10.10.10.20 10.10.10.30

DNS Server : www.tianyun.com ==> 192.168.122.100

建议：先在Real Server安装如httpd

一、VS/NAT实施

1. 准备工作（集群中所有主机） [可选]

IP, hostname, hosts, iptables, SELinux, ssh trust, ntp

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

```

127.0.0.1      localhost
10.10.10.1      director1.tianyun.com director1
10.10.10.10     node1.tianyun.com node1
10.10.10.20     node2.tianyun.com node2
10.10.10.30     node3.tianyun.com node3

```

2. RS配置

配置好网站服务器，测试所有RS

//为了看到测试效果，建议提供不同的页面

默认网关均指向Directory的DIP

3. Director分发器配置

配置VIP

[root@tianyun ~]# ip addr add dev eth0 192.168.122.100/24

[root@tianyun ~]# vim /etc/sysctl.conf

net.ipv4.ip_forward = 1

```
[root@tianyun ~]# sysctl -p //确保打开路由转发
```

定义LVS的分发策略

```
[root@tianyun ~]# yum -y install ipvsadm //确保LoadBalancer仓库可用
[root@tianyun ~]# ipvsadm -A -t 192.168.122.100:80 -s rr
[root@tianyun ~]# ipvsadm -a -t 192.168.122.100:80 -r 10.10.10.10 -m // -m masquerading (NAT)
[root@tianyun ~]# ipvsadm -a -t 192.168.122.100:80 -r 10.10.10.20 -m
[root@tianyun ~]# ipvsadm -a -t 192.168.122.100:80 -r 10.10.10.30 -m
[root@tianyun ~]# service ipvsadm save
```

```
[root@tianyun ~]# ipvsadm -L
[root@tianyun ~]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
-> RemoteAddress:Port Forward Weight ActiveConn InActConn
TCP 192.168.122.100:80 rr
-> 10.10.10.10:80 Masq    1    0      1
-> 10.10.10.20:80 Masq    1    0      1
-> 10.10.10.30:80 Masq    1    0      1
```

```
[root@tianyun ~]# ipvsadm -L -n --stats
[root@tianyun ~]# ipvsadm -L -n --rate //看速率
[root@tianyun ~]# ipvsadm -Ln -c //查看LVS的连接条目
[root@tianyun ~]# watch -n.5 'ipvsadm -Ln -c'
```

4. 测试

```
[root@client ~]# elinks -dump http://192.168.122.100/
[root@client ~]# ab -c 1000 -n 1000 http://192.168.122.100/
```

5. 小结

VS/NAT模式的原理是：当Director收到Client请求时，Director将数据包的目标IP由VIP转换为选中的Real Server的RIP来实现分发，要求RS将网关指向Director的DIP。

特点是：配置简单，所有的入站、出站数据包都经过分发器。当数据量比较大时，分发器可能会出现网络瓶颈！因而支持的RS数量少。

```
=====
```

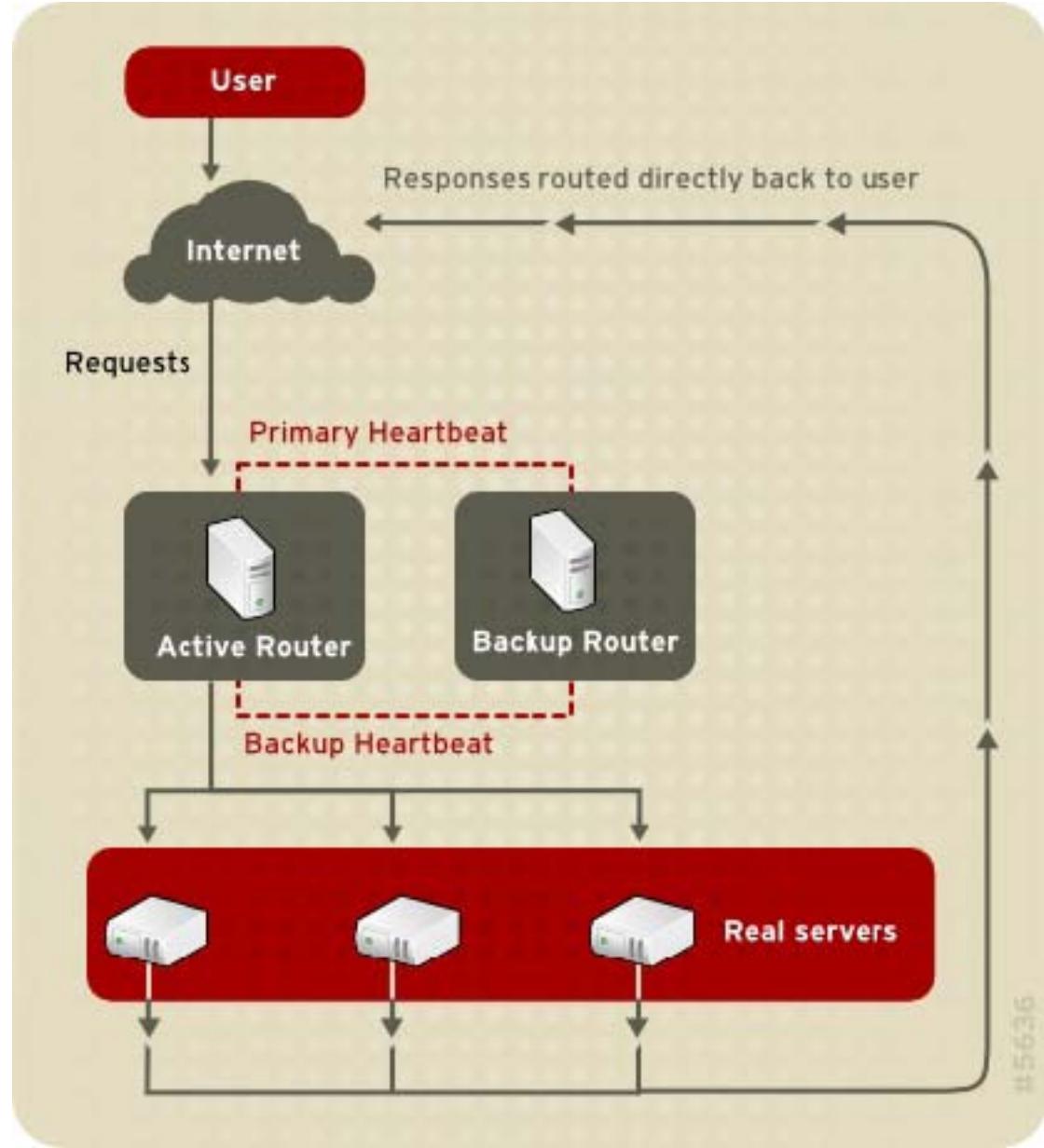
VS/DR

VS/DR

```
=====
```

实验说明：

- 1.KVM网络使用NAT模式
- 2.DR模式要求Director DIP 和 所有RealServer RIP必须在同一个网段及广播域
- 3.所有节点网关均指定真实网关



网络拓扑：

Client :

CIP : 192.168.122.1

Director :

VIP : 192.168.122.100

DIP : 192.168.122.2

Real Server : RIP : 192.168.122.10 192.168.122.20 192.168.122.30

VIP : 192.168.122.100 192.168.122.100 192.168.122.100

DNS Server : www.tianyun.com ==> 192.168.122.100

一、VS/DR实施

1. 准备工作（集群中所有主机）

IP, hostname, hosts, iptables, SELinux, ssh trust, ntp

```
[root@tianyun ~]# cat /etc/hosts
127.0.0.1      localhost
192.168.122.2      director1.tianyun.com director1
192.168.122.10      node1.tianyun.com node1
192.168.122.20      node2.tianyun.com node2
192.168.122.30      node3.tianyun.com node3
```

2. RS配置

配置好网站服务器，测试所有RS
的页面

```
[root@tianyun ~]# yum -y install httpd
```

//为了测试效果，提供不同的

```
[root@tianyun ~]# ip addr add dev lo 192.168.122.100/32          //在lo接口上绑定VIP
[root@tianyun ~]# echo 1 > /proc/sys/net/ipv4/conf/all/arp_ignore    //non-arp
[root@tianyun ~]# echo 2 > /proc/sys/net/ipv4/conf/all/arp_announce
```

3. Director分发器配置

配置VIP

```
[root@tianyun ~]# ip addr add dev eth0 192.168.122.100/32          //配置VIP
[root@tianyun ~]# yum -y install ipvsadm                           //确保LoadBalancer仓库可用
```

定义LVS分发策略

```
[root@tianyun ~]# ipvsadm -C
[root@tianyun ~]# ipvsadm -A -t 192.168.122.100:80 -s rr
[root@tianyun ~]# ipvsadm -a -t 192.168.122.100:80 -r 192.168.122.10 -g
[root@tianyun ~]# ipvsadm -a -t 192.168.122.100:80 -r 192.168.122.20 -g
[root@tianyun ~]# ipvsadm -a -t 192.168.122.100:80 -r 192.168.122.30 -g
[root@tianyun ~]# service ipvsadm save
Saving IPVS table to /etc/sysconfig/ipvsadm: [ OK ]
[root@tianyun ~]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
-> RemoteAddress:Port      Forward Weight ActiveConn InActConn
TCP 192.168.122.100:80 rr
-> 192.168.122.10:80       Route   1     0     0
-> 192.168.122.20:80       Route   1     0     0
-> 192.168.122.30:80       Route   1     0     0
```

```
[root@tianyun ~]# ipvsadm -L -n
[root@tianyun ~]# ipvsadm -L -n --stats
[root@tianyun ~]# ipvsadm -L -n --rate
[root@tianyun ~]# ipvsadm -Ln -c
[root@tianyun ~]# watch -n.5 'ipvsadm -Ln -c'
```

4. 测试

```
[root@client ~]# elinks -dump http://192.168.122.100
[root@client ~]# ab -c 1000 -n 1000 http://192.168.122.100/
```

5. 小结

VS/DR模式的原理是：当一个client发送一个请求到VIP，Director根据VIP选择对应的real-server的Pool，根据算法，在Pool中选择一台Real-server，然后将client的请求包发给选择的Real-server，最后选择的Real-server把应答包直接传给client

VS/TUN

VS/FullNAT

LVS调度算法

1. 轮叫调度 (Round Robin) (简称rr)

调度器通过“轮叫”调度算法将外部请求按顺序轮流分配到集群中的真实服务器上，它均等地对待每一台服务器，而不管服务器上实际的连接数和系统负载。

2. 加权轮叫 (Weighted Round Robin) (简称wrr)

调度器通过“加权轮叫”调度算法根据真实服务器的不同处理能力来调度访问请求。这样可以保证处理能力强的服务器能处理更多的访问流量。调度器可以自动问询真实服务器的负载情况，并动态地调整其权值。

3. 最少链接 (Least Connections) (LC)

调度器通过“最少连接”调度算法动态地将网络请求调度到已建立的链接数最少的服务器上。如果集群系统的真实服务器具有相近的系统性能，采用“最小连接”调度算法可以较好地均衡负载。

4. 加权最少链接 (Weighted Least Connections) (WLC)

在集群系统中的服务器性能差异较大的情况下，调度器采用“加权最少链接”调度算法优化负载均衡性能，具有较高权值的服务器将承受较大比例的活动连接负

载。调度器可以自动问询真实服务器的负载情况，并动态地调整其权值。

5. 基于局部性的最少链接 (Locality-Based Least Connections) (LBLC)

“基于局部性的最少链接”调度算法是针对目标IP地址的负载均衡，目前主要用于Cache集群系统。该算法根据请求的目标IP地址找出该目标IP地址最近使用的

服务器，若该服务器是可用的且没有超载，将请求发送到该服务器；若服务器不存在，或者该服务器超载且有服务器处于一半的工作负载，则用“最少链接”

的原则选出一个可用的服务器，将请求发送到该服务器。

6. 带复制的基于局部性最少链接 (Locality-Based Least Connections with Replication) (LBLCR)

“带复制的基于局部性最少链接”调度算法也是针对目标IP地址的负载均衡，目前主要用于Cache集群系统。它与LBLC算法的不同之处是它要维护从一个目标

IP地址到一组服务器的映射，而LBLC算法维护从一个目标IP地址到一台服务器的映射。该算法根据请求的目标IP地址找出该目标IP地址对应的服

务器组，按

“最小连接”原则从服务器组中选出一台服务器，若服务器没有超载，将请求发送到该服务器；若服务器超载，则按“最小连接”原则从这个集群中选出一台

服务器，将该服务器加入到服务器组中，将请求发送到该服务器。同时，当该服务器组有一段时间没有被修改，将最忙的服务器从服务器组中删

除，以降低复

制的程度。

7. 目标地址散列 (Destination Hashing) (DH)

“目标地址散列”调度算法根据请求的目标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，若该服务器是可用的且未超载，将请求发送到该服务器，否则返回空。

8. 源地址散列 (Source Hashing) (SH)

“源地址散列”调度算法根据请求的源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，若该服务器是可用的且未超载，将请求发送到该服务器，否则返回空。

9. 最短的期望的延迟 (Shortest Expected Delay Scheduling SED) (SED)

基于wlc算法。这个必须举例来说了

ABC三台机器分别权重123，连接数也分别是123。那么如果使用WLC算法的话一个新请求进入时它可能会分给ABC中的任意一个。使用sed算法后会进行这样一个运算

A(1+1)/1

B(1+2)/2

C(1+3)/3

根据运算结果，把连接交给C。

10. 最少队列调度 (Never Queue Scheduling NQ) (NQ)

无需队列。如果有台 realserver的连接数 = 0就直接分配过去，不需要在进行sed运算

持久性连接主要解决的是会话保持的问题

定义LVS的分发策略

```
[root@tianyun ~]# ipvsadm -C
[root@tianyun ~]# ipvsadm -A -t 192.168.122.100:80 -s rr -p 300
[root@tianyun ~]# ipvsadm -a -t 192.168.122.100:80 -r 192.168.122.10 -g //gatewaying (direct routing)
[root@tianyun ~]# ipvsadm -a -t 192.168.122.100:80 -r 192.168.122.20 -g
[root@tianyun ~]# ipvsadm -a -t 192.168.122.100:80 -r 192.168.122.30 -g
[root@tianyun ~]# service ipvsadm save
Saving IPVS table to /etc/sysconfig/ipvsadm: [ OK ]
```



```
[root@tianyun ~]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
    -> RemoteAddress:Port      Forward Weight ActiveConn InActConn
TCP 192.168.122.100:80 rr persistent 300
    -> 192.168.122.10:80        Route   1    0    0
    -> 192.168.122.20:80       Route   1    0    0
    -> 192.168.122.30:80       Route   1    0    0
```



```
[root@tianyun ~]# ipvsadm -Ln -c
IPVS connection entries
pro expire state      source          virtual          destination
TCP 04:09 NONE      192.168.122.1:0  192.168.122.100:80 192.168.122.30:80
TCP 00:10 FIN_WAIT  192.168.122.1:37155 192.168.122.100:80 192.168.122.20:80
```

LVS端口亲缘性

防火墙标记

对于访问同一个IP的80/tcp和443/tcp，应该作为一个虚拟服务

对于访问同一个IP的21/tcp和5000-6000/tcp，应该作为一个虚拟服务

例如VIP是192.168.122.100

Web:

```
[root@tianyun ~]# iptables -t mangle -A PREROUTING -p tcp -d 192.168.122.100 --dport 80 -j MARK --set-mark 80
[root@tianyun ~]# iptables -t mangle -A PREROUTING -p tcp -d 192.168.122.100 --dport 443 -j MARK --set-mark 80
```

FTP:

```
[root@tianyun ~]# iptables -t mangle -A PREROUTING -p tcp -d 192.168.122.100 --dport 21 -j MARK --set-mark 21
[root@tianyun ~]# iptables -t mangle -A PREROUTING -p tcp -d 192.168.122.100 --dport 50000-60000 -j MARK --set-mark 21
```

```
[root@tianyun ~]# ipvsadm -A -f 80 -s rr
[root@tianyun ~]# ipvsadm -a -f 80 -r 192.168.122.10 -g
[root@tianyun ~]# ipvsadm -a -f 80 -r 192.168.122.20 -g
[root@tianyun ~]# ipvsadm -a -f 80 -r 192.168.122.30 -g
```

```
[root@tianyun ~]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
    -> RemoteAddress:Port      Forward Weight ActiveConn InActConn
```

```
FWM 80 rr
-> 192.168.122.10:0      Route 1    0    0
-> 192.168.122.20:0      Route 1    0    0
-> 192.168.122.30:0      Route 1    0    0
```

keepalived:

```
virtual_server fwmark 80 {
    delay_loop 6
    lb_algo rr
    lb_kind DR
    nat_mask 255.255.255.0
    persistence_timeout 50
    protocol TCP
    .....
}
```

Keepalived LVS

基于LVS LB集群解决方案一：LVS + KeepAlived

KeepAlived在该项目中的功能：

1. 实现调度器的HA
2. 对RealServer做健康检查

<http://www.keepalived.org>

拓扑结构：

Client : CIP : 192.168.122.1

Director : VIP : 192.168.122.100
DIP: director1 192.168.122.2 director2 192.168.122.3

Real Server : RIP : 192.168.122.10 192.168.122.20 192.168.122.30
VIP : 192.168.122.100 192.168.122.100 192.168.122.100

DNS Server : www.tianyun.com ==> 192.168.122.100

注：主/备Directory VIP由高可用软件添加

一、实施步骤

1. 准备工作（集群中所有主机）

IP, hostname, hosts, iptables, SELinux, ssh trust, ntp

```
[root@tianyun ~]# cat /etc/hosts
127.0.0.1           localhost
192.168.122.2       director1.tianyun.com director1
192.168.122.3       director2.tianyun.com director2
192.168.122.10      node1.tianyun.com node1
192.168.122.20      node2.tianyun.com node2
192.168.122.30      node3.tianyun.com node3
```

2. RS配置

配置好网站服务器，测试所有RS
的页面

```
[root@tianyun ~]# ip addr add dev lo 192.168.122.100/32          //在lo接口上绑定VIP
```

//为了测试效果，提供不同

方法一：不对VIP的ARP请求响应

```
[root@tianyun ~]# echo 1 > /proc/sys/net/ipv4/conf/all/arp_ignore    //non-arp
[root@tianyun ~]# echo 2 > /proc/sys/net/ipv4/conf/all/arp_announce
```

方法二：不对VIP的ARP请求响应

```
[root@tianyun ~]# arptables -A IN -d 192.168.122.100 -j DROP
[root@tianyun ~]# arptables -A OUT -s 192.168.122.100 -j mangle --mangle-ip-s <real_ip>
[root@tianyun ~]# service arptables_if
save
[root@tianyun ~]# chkconfig arptables_if on
```

3. 主/备调度器安装软件

```
[root@master ~]# yum -y install ipvsadm keepalived
[root@backup ~]# yum -y install ipvsadm keepalived
```

=====

源码安装方法

ipvsadm

```
[root@tianyun ~]# yum -y install ipvsadm kernel-headers kernel-devel openssl-devel popt-devel
```

KeepAlived

```
[root@tianyun keepalived-1.2.1]# ./configure --prefix=/ --with-kernel-dir=/usr/src/kernels/2.6.32-358.el6.x86_64/
Keepalived configuration
```

```
-----
Keepalived version      : 1.2.1
Compiler                 : gcc
Compiler flags           : -g -O2 -DETHERTYPE_IPV6=0x86dd
Extra Lib                : -lpopt -lssl -lcrypto
Use IPVS Framework       : Yes
IPVS sync daemon support : Yes
Use VRRP Framework        : Yes
Use Debug flags          : No
[root@tianyun keepalived-1.2.1]# make
[root@tianyun keepalived-1.2.1]# make install
-----
```

4. 主/备调度器Keepalived配置

Master

获得Real Server测试页面的MD5SUM值 <用于对Real Server健康检查>

```
[root@tianyun ~]# genhash -s 192.168.122.10 -p 80 -u /test.html          //RS网站的主目录
MD5SUM = c5f42eddf77ef2d2b0f7263c094c20e
```

```
[root@tianyun ~]# vim /etc/keepalived/keepalived.conf
! Configuration File for keepalived
```

```
global_defs {
    router_id director1          //辅助改为director2
}

vrrp_instance VI_1 {
    state MASTER                  //辅助改为BACKUP
    interface eth0                //心跳接口，尽量单独连接心跳
    virtual_router_id 80          //MASTER,BACKUP一致
    priority 100                  //辅助改为50
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.122.100
    }
}

virtual_server 192.168.122.100 80 {
    delay_loop 6
    lb_algo rr
    lb_kind DR
    nat_mask 255.255.255.0
    persistence_timeout 50
    protocol TCP
}
```

```

real_server 192.168.122.10 80 {
    weight 1
    HTTP_GET {
        url {
            path /test.html
            digest c5f42eddf777ef2d2b0f7263c094c20e
        }
        connect_timeout 3
        nb_get_retry 3
        delay_before_retry 3
    }
}

real_server 192.168.122.20 80 {
    weight 1
    HTTP_GET {
        url {
            path /test.html
            digest c5f42eddf777ef2d2b0f7263c094c20e
        }
        connect_timeout 3
        nb_get_retry 3
        delay_before_retry 3
    }
}

real_server 192.168.122.30 80 {
    weight 1
    HTTP_GET {
        url {
            path /test.html
            digest c5f42eddf777ef2d2b0f7263c094c20e
        }
        connect_timeout 3
        nb_get_retry 3
        delay_before_retry 3
    }
}

```

BACKUP

5. 启动KeepAlived (主备均启动)

```
[root@tianyun ~]# chkconfig keepalived on
[root@tianyun ~]# service keepalived start
[root@tianyun ~]# tail -f /var/log/messages
```

```
[root@tianyun ~]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
-> RemoteAddress:Port      Forward Weight ActiveConn InActConn
TCP 192.168.122.100:80 wrr
-> 192.168.122.10:80       Route 1    0      0
-> 192.168.122.20:80       Route 3    0      0
-> 192.168.122.30:80       Route 3    0      0
```

```
[root@tianyun ~]# ipvsadm -Ln -c
IPVS connection entries
pro expire state      source          virtual          destination
TCP 01:57 FIN_WAIT 192.168.1.254:58698 192.168.122.100:80 192.168.122.10:80
TCP 01:57 FIN_WAIT 192.168.1.254:58699 192.168.122.100:80 192.168.122.20:80
TCP 01:57 FIN_WAIT 192.168.1.254:58695 192.168.122.100:80 192.168.122.20:80
TCP 01:56 FIN_WAIT 192.168.1.254:58691 192.168.122.100:80 192.168.122.20:80
TCP 01:58 FIN_WAIT 192.168.1.254:58700 192.168.122.100:80 192.168.122.20:80
TCP 01:58 FIN_WAIT 192.168.1.254:58702 192.168.122.100:80 192.168.122.10:80
TCP 01:58 FIN_WAIT 192.168.1.254:58703 192.168.122.100:80 192.168.122.20:80
TCP 01:57 FIN_WAIT 192.168.1.254:58696 192.168.122.100:80 192.168.122.20:80
TCP 01:57 FIN_WAIT 192.168.1.254:58693 192.168.122.100:80 192.168.122.20:80
```

二、测试

所有分发器和Real Server都正常

主分发器故障及恢复

Real Server故障及恢复

三、Keepalived不抢占

nopreempt 仅针对BACKUP

Piranha LVS

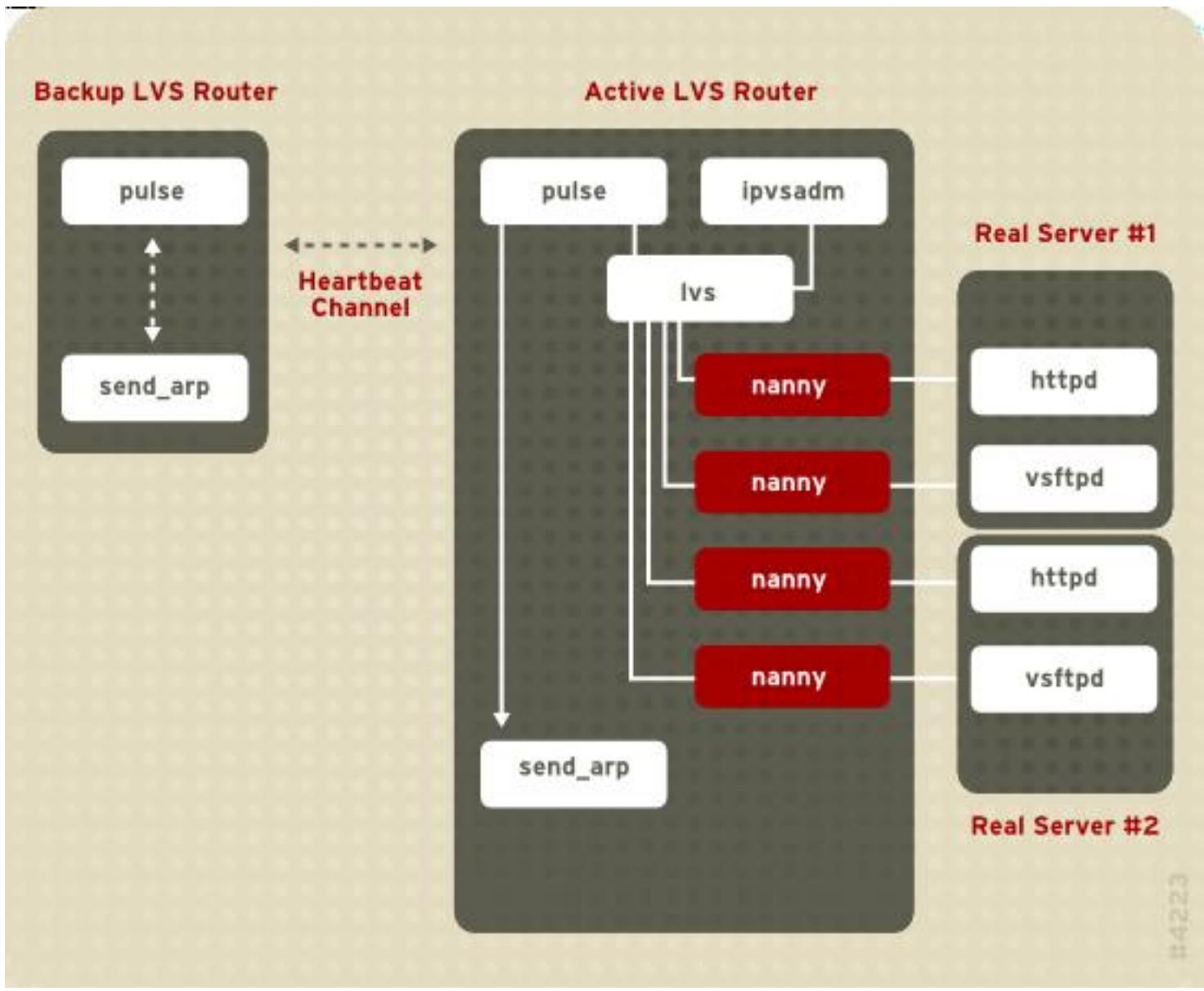
LVS 组件

基于**LVS LB**集群解决方案二：**Piranha**

该项目中的功能：

1. 实现调度器的HA
2. 对RealServer做健康检查

LVS Components



LVS 构架图:

活跃和备用 LVS 路由器中都会运行 pulse 守护进程。在备用路由器中,pulse 向活跃服务器的公共接口发送一个 heartbeat 来确定活跃路由器仍可正常工作。在活跃路由器中,pulse 启动 lvs 守护进程,并回应来自备用 LVS 路由器的 heartbeat 查询。

启动后,lvs 守护进程调用 ipvsadm 程序来配置和维护内核中的 IPVS 路由表,并为每个真实服务器中配置的虚拟服务器启动 nanny 进程,同时告知 lvs 该真实服务器中的服务是否正常工作。如果发现故障,lvs 守护进程会向 ipvsadm 发出指令将那个真实服务器从 IPVS 路由表中删除。

如果备用路由器没有收到来自活跃路由器的响应,它会通过调用 send_arp 启动失效切换来将所有虚拟 IP 地址重新分配到备用节点的 NIC 硬件地址(MAC 地址),并通过公共和专用网络接口向活跃路由器发送一个命令来关闭活跃路由器中的 lvs 守护进程,启动备用节点中的 lvs 守护进程来为配置的虚拟服务器接收请求。

pulse

活跃和备用 LVS 路由器中都会运行 pulse 守护进程。在备用路由器中,pulse 向活跃服务器的公共接口发送一个 heartbeat 来确定活跃路由器仍可正常工作。在活跃服务器中,pulse 启动 lvs 守护进程,并回应来自备用 LVS 路由器的 heartbeat 查询。

lvs

一旦被 pulse 调用,lvs 守护进程就运行在活动 LVS 路由器中。它读取配置文件/etc/sysconfig/ha/lvs.cf,调用 ipvsadm 工具来构建和维护 IPVS 路由表,并为每个配置的 LVS 服务分配 nanny 进程。如果 nanny 报告某个服务器关闭了,lvs 将指引 ipvsadm 工具从 IPVS 路由表中删除这个服务器。

ipvsadm

这个服务更新内核中的 IPVS 路由表。lvs 守护进程通过调用 ipvsadm 来添加、修改或者删除 IPVS 路由表中的条目来设置和管理 LVS。

nanny

nanny 监控运行在活跃 LVS 路由器中的守护进程。通过这个守护进程,活跃路由器可确定每个真实服务器的状态,有时还可以监控其工作负载。每个真实服务器中定义的每个服务都有一个独立进程为其运行。

Piranha Configuration Tool

这是用来监控、配置和管理 LVS 的网页工具。它是用来维护 [/etc/sysconfig/ha/lvs.cf](#) LVS 配置文件的默认工具。

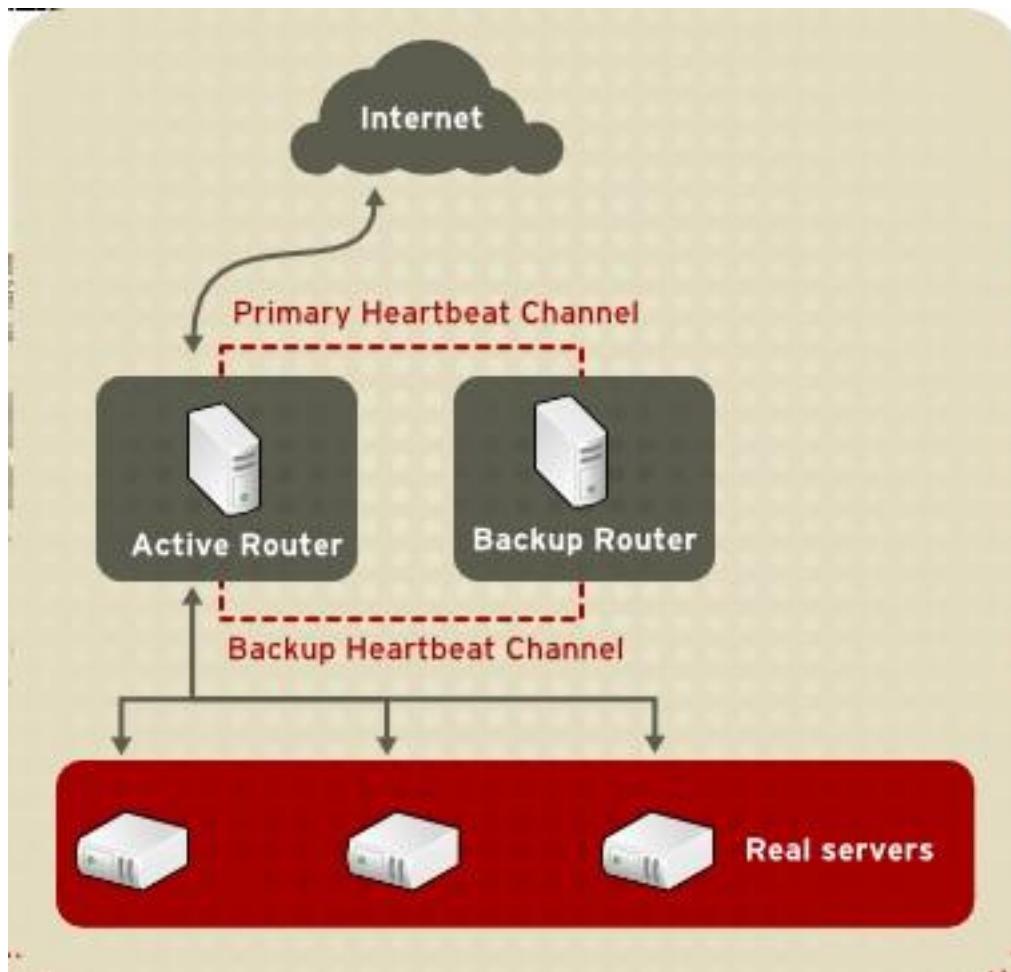
send_arp

在失效切换过程中,当浮动 IP 地址在节点间进行更改时,这个程序发送 ARP 广播。

Piranha 部署

Piranha Configuration Tool 配置 LVS (DR)

Piranha Configuration Tool 为 LVS 创建所需配置文件 —/etc/sysconfig/ha/lvs.cf



拓扑结构：

Client:

192.168.1.0/24

Director :

192.168.1.2 192.168.1.3
 master.tianyun.com backup.tianyun.com

Real Server :

192.168.1.1 192.168.1.5

VIP :

192.168.1.80

DNS Server :

www.tianyun.com ==> 192.168.1.80

一、实施步骤

1. 准备工作 (集群中所有主机)

IP, hostname, hosts, iptables, SELinux, ssh trust, ntp
[root@tianyun ~]# cat /etc/hosts
127.0.0.1 localhost
192.168.1.2 active.tianyun.com active
192.168.1.3 backup.tianyun.com backup
192.168.1.1 node1.tianyun.com node1
192.168.1.5 node2.tianyun.com node2
192.168.1.80 vip

2. RS配置

配置好网站服务器，测试所有RS
的页面

//为了测试效果，提供不同的IP

```
[root@tianyun ~]# ip addr add dev lo 192.168.1.80/32                            //在lo接口上绑定VIP  
[root@tianyun ~]# echo 1 > /proc/sys/net/ipv4/conf/all/arp_ignore    //non-arp  
[root@tianyun ~]# echo 2 > /proc/sys/net/ipv4/conf/all/arp_announce
```

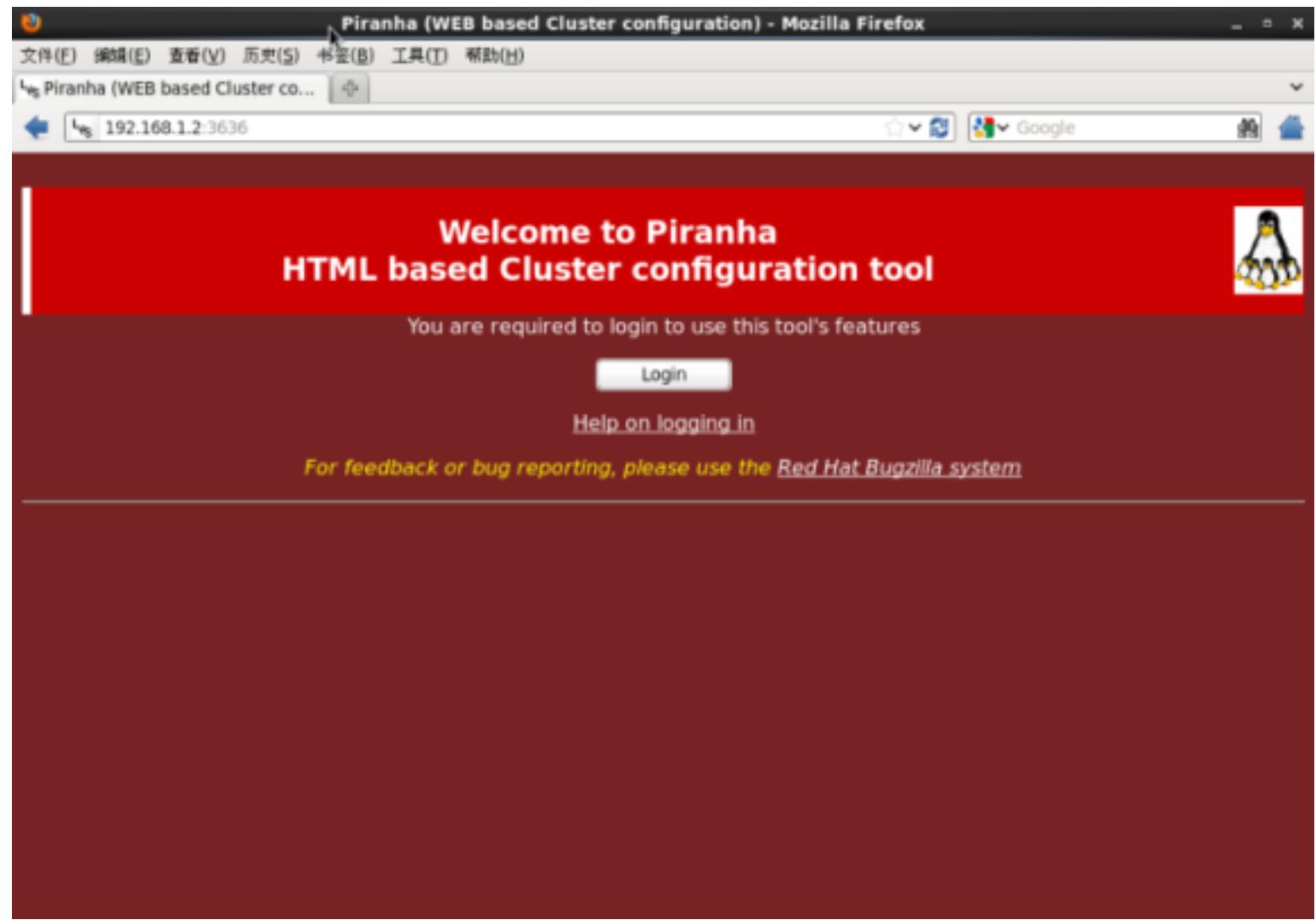
3. 主/备调度器安装Piranha

```
[root@active ~]# yum -y install piranha  
[root@active ~]# piranha-passwd  
New Password: redhat  
Verify: redhat  
Adding password for user piranha  
[root@active ~]# /etc/init.d/piranha-gui start  
[root@active ~]# chkconfig piranha-gui on  
[root@active ~]# netstat -tnlp |grep :3636  
tcp        0        0 :::3636                    ::*:*                            LISTEN      1887/httpd
```

```
[root@backup ~]# yum -y install piranha
```

4. 配置Piranha

<http://192.168.1.2:3636> (主调度器)



pulse
CONTROL/MONITORING
「控制 / 监控」面板列出了 LVS 的状态。它显示了 pulse 守护进程、LVS 路由表和 LVS 生成的 nanny 进程的状态

文件(E) 编辑(S) 查看(V) 历史(H) 书签(B) 工具(I) 帮助(H)

Piranha (Control/Monitoring)



Google



192.168.1.2:3636/secure/control.php

PIRANHA CONFIGURATION TOOL

INTRODUCTION | HELP

CONTROL / MONITORING

CONTROL/MONITORING

GLOBAL SETTINGS

REDUNDANCY

VIRTUAL SERVERS

CONTROL

Daemon: stopped

MONITOR

Auto update Update Interval: seconds

[Update information now](#)

CURRENT LVS ROUTING TABLE

CURRENT LVS PROCESSES

[CHANGE PASSWORD](#)

GLOBAL SETTINGS

The GLOBAL SETTINGS panel is where you define the networking details for the primary LVS router's public and private network interfaces.

Piranha (Global Settings) - Mozilla Firefox

文件(E) 编辑(E) 查看(V) 历史(S) 书签(B) 工具(T) 帮助(H)

Piranha (Global Settings) http://192.168.1.80/ 192.168.1.2:3636/secure/global_settings.php Google

PIRANHA CONFIGURATION TOOL INTRODUCTION | HELP

GLOBAL SETTINGS

CONTROL/MONITORING GLOBAL SETTINGS REDUNDANCY VIRTUAL SERVERS

ENVIRONMENT

Primary server public IP:

Primary server private IP:
(May be blank)

TCP timeout (seconds):

TCP FIN timeout (seconds):

UDP timeout (seconds):

Use network type:
(Current type is: **direct**) NAT Direct Routing Tunneling

ACCEPT -- Click here to apply changes on this page

注：

Primary server public IP: 主调度器公共Ip

Use network type: 选择路由方法，此处为Direct Routing即直接路由

REDUNDANCY

「冗余」面板允许您配置备用 LVS 路由器节点并设置不同的 heartbeat 监控选项

文件(E) 编辑(S) 查看(V) 历史(H) 书签(B) 工具(T) 帮助(H)

Piranha (Redundancy)

http://192.168.1.80/

Google

192.168.1.2:3636/secure/redundancy.php

PIRANHA CONFIGURATION TOOL

INTRODUCTION | HELP

REDUNDANCY

CONTROL/MONITORING

GLOBAL SETTINGS

REDUNDANCY

VIRTUAL SERVERS

Backup: active

Redundant server public IP: 192.168.1.3

Redundant server private IP:

Heartbeat interval (seconds): 6

Assume dead after (seconds): 18

Heartbeat runs on port: 539

Monitor NIC links for failures: Use sync daemon: Sync daemon interface: Sync daemon ID:

ACCEPT -- Click here to apply changes to this page

DISABLE RESET

注：

Redundant server public IP: 备调度器公共IP

Heartbeat Interval (seconds): 心跳间隔时间(秒) — 即备用节点检查主 LVS 节点功能状态的间隔时间

Assume dead after (seconds): 如果主 LVS 调度器在这段时间内没有响应，备调度器将接管

Heartbeat runs on port: 心跳端口默认为539

VIRTUAL SERVERS

「虚拟服务器」面板显示每个当前定义的虚拟服务器的信息。表格里的每个条目都显示了虚拟服务器的状态、服务器名称、分配的 IP 地址、虚拟 IP 的掩码、服务端口、使用的协议以及虚拟设备接口

添加虚拟服务

文件(E) 编辑(S) 查看(V) 历史(H) 书签(B) 工具(T) 帮助(H)

Piranha (Virtual Servers - Editin...)

Google

PIRANHA CONFIGURATION TOOL

INTRODUCTION | HELP

EDIT REAL SERVER

CONTROL/MONITORING

GLOBAL SETTINGS

REDUNDANCY

VIRTUAL SERVERS

EDIT: [VIRTUAL SERVER](#) | [REAL SERVER](#) | [MONITORING SCRIPTS](#)

STATUS

NAME

ADDRESS

 Down

[unnamed]

0.0.0.0

[ADD](#)[DELETE](#)[EDIT](#)[\(DE\)ACTIVATE](#)[CANCEL](#)

文件(E) 编辑(S) 查看(V) 历史(H) 书签(B) 工具(T) 帮助(H)

Piranha (Virtual Servers - Editin...

192.168.1.2:3636/secure/virtual_edit_virt.php?selected_host=1

Google

EDIT VIRTUAL SERVER

CONTROL/MONITORING

GLOBAL SETTINGS

REDUNDANCY

VIRTUAL SERVERS

EDIT: VIRTUAL SERVER | [REAL SERVER](#) | [MONITORING SCRIPTS](#)

Name:	<input type="text" value="http server"/>
Application port:	<input type="text" value="80"/>
Protocol:	<input type="text" value="tcp"/>
Virtual IP Address:	<input type="text" value="192.168.1.80"/>
Virtual IP Network Mask:	<input type="text" value="255.255.255.255"/>
Sorvy Server:	<input type="text"/>
Firewall Mark:	<input type="text"/>
Device:	<input type="text" value="eth0:1"/>
Re-entry Time:	<input type="text" value="15"/>
Service timeout:	<input type="text" value="6"/>
Quiesce server:	<input type="radio"/> Yes <input checked="" type="radio"/> No
Load monitoring tool:	<input type="text" value="none"/>
Scheduling:	<input type="text" value="Round robin"/>
Persistence:	<input type="text" value="300"/>
Persistence Network Mask:	<input type="text" value="Unused"/>

ACCEPT -- Click here to apply changes to this page

注：

Re-entry Time: 15 表示当Real Server失效15秒后，调度器会尝试将其接回

Service timeout: 6 表示当Real Server超时6秒，调度器会将其从调度池中删除

Persistence: 300 持久性连接时间

[激活虚拟服务](#)

文件(E) 编辑(S) 查看(V) 历史(H) 书签(B) 工具(T) 帮助(H)

Piranha (Virtual Servers)



Google



PIRANHA CONFIGURATION TOOL

INTRODUCTION | HELP

VIRTUAL SERVERS

CONTROL/MONITORING		GLOBAL SETTINGS			REDUNDANCY			VIRTUAL SERVERS
	STATUS	NAME	VIP	NETMASK	PORT	PROTOCOL	INTERFACE	
<input checked="" type="radio"/>	up	http_server	192.168.1.80	255.255.255.255	80	tcp	eth0:1	

[ADD](#) [DELETE](#) [EDIT](#) [\(DE\)ACTIVATE](#)

Note: Use the radio button on the side to select which virtual service you wish to edit before selecting 'EDIT' or 'DELETE'

[添加Real Server](#)

文件(E) 编辑(S) 查看(V) 历史(H) 书签(B) 工具(T) 帮助(H)

Piranha (Virtual servers - Editin...)



192.168.1.2:3636/secure/virtual_edit_real_edit.php?selected_host=1&selected=1



Google



PIRANHA CONFIGURATION TOOL

INTRODUCTION | HELP

EDIT REAL SERVER

CONTROL/MONITORING

GLOBAL SETTINGS

REDUNDANCY

VIRTUAL SERVERS

EDIT: [VIRTUAL SERVER](#) | [REAL SERVER](#) | [MONITORING SCRIPTS](#)Name: Address: Port: (Leave blank to default to Virtual Server's Application Port)Weight: [激活Real Server](#)

Piranha (Virtual Servers - Editing real server) - Mozilla Firefox

文件(E) 编辑(S) 查看(V) 历史(H) 书签(B) 工具(T) 帮助(H)

Piranha (Virtual Servers - Editin... +

192.168.1.2:3636/secure/virtual_edit_real.php?virtual=1&active=up&name=node1.tianyun.cc Google

PIRANHA CONFIGURATION TOOL INTRODUCTION | HELP

EDIT REAL SERVER

CONTROL/MONITORING		GLOBAL SETTINGS	REDUNDANCY	VIRTUAL SERVERS
EDIT: VIRTUAL SERVER REAL SERVER MONITORING SCRIPTS				
STATUS	NAME	ADDRESS		
<input type="radio"/> up	node1.tianyun.com	192.168.1.1		
<input checked="" type="radio"/> up	node2.tianyun.com	192.168.1.5		
ADD DELETE EDIT (DE)ACTIVATE		CANCEL		

配置对Real Server健康检查

Piranha (Virtual Servers - MONITORING SCRIPTS) - Mozilla Firefox

文件(E) 编辑(E) 查看(V) 历史(S) 书签(B) 工具(T) 帮助(H)

Piranha (Virtual Servers - MONI... +

192.168.1.2:3636/secure/virtual_edit_services.php?selected_host=1 Google

PIRANHA CONFIGURATION TOOL INTRODUCTION | HELP

EDIT MONITORING SCRIPTS

CONTROL/MONITORING **GLOBAL SETTINGS** **REDUNDANCY** **VIRTUAL SERVERS**

EDIT: VIRTUAL SERVER | REAL SERVER | MONITORING SCRIPTS

Current text Replacement text

Sending Program: NO SEND PROGRAM

Send: "GET / HTTP/1.0\r\n\r\n" BLANK SEND

Expect: "HTTP" BLANK EXPECT

Treat expect string as a regular expression

Please note: You may either use the simple send/expect mechanism built into piranha or a custom monitoring script (send program). The send program takes priority over the send string.

The send program should output a string matching the the expect string. If the argument %h is used in the send program command, it will be replaced with the ip address of the server to be checked.

ACCEPT CANCEL

This screenshot shows the 'Edit Monitoring Scripts' interface in the Piranha Configuration Tool. It includes tabs for Control/Monitoring, Global Settings, Redundancy, and Virtual Servers. The current tab is 'Control/Monitoring'. The main area has sections for 'Current text' and 'Replacement text' with fields for 'Sending Program', 'Send', and 'Expect'. There's also a checkbox for 'Treat expect string as a regular expression'. A note at the bottom explains the priority of send programs over simple strings, mentioning the use of '%h' in send programs to replace the server's IP address. At the bottom are 'ACCEPT' and 'CANCEL' buttons.

5. 主/备调度器启动pulse进程

将配置文件lvs.cf同步到备调度器

```
[root@active ~]# rsync -va /etc/sysconfig/ha/lvs.cf backup:/etc/sysconfig/ha/
```

启动进程pulse

```
[root@active ~]# /etc/init.d/pulse start  
[root@active ~]# chkconfig pulse on
```

```
[root@backup ~]# /etc/init.d/pulse start  
[root@backup ~]# chkconfig pulse on
```

6. 查看集群状态

Piranha (Control/Monitoring) - Mozilla Firefox

文件(E) 编辑(S) 查看(V) 历史(H) 书签(B) 工具(I) 帮助(H)

Piranha (Control/Monitoring) [+/-](#)

192.168.1.2:3636/secure/control.php?auto_update=1&rate=10&refresh=Update+information [↑](#) [↓](#) [Google](#)

CONTROL / MONITORING

[CONTROL/MONITORING](#) [GLOBAL SETTINGS](#) [REDUNDANCY](#) [VIRTUAL SERVERS](#)

CONTROL

Daemon: running

MONITOR

Auto update Update Interval: seconds

[Update information now](#)

CURRENT LVS ROUTING TABLE

```
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
-> RemoteAddress:Port Forward Weight ActiveConn InActConn
TCP 192.168.1.80:80 rr persistent 300000 FFFFFFFF
-> 192.168.1.1:80 Route 1 0 0
-> 192.168.1.5:80 Route 1 0 3
```

CURRENT LVS PROCESSES

```
root 2611 0.0 0.1 8768 548 ? Ss 15:15 0:00 pulse
root 2615 0.0 0.1 8756 824 ? Ss 15:15 0:00 /usr/sbin/lvsd --nofork -c /etc/sysconfig/ha/lvs.cf
root 2628 0.0 0.1 8732 860 ? Ss 15:15 0:00 /usr/sbin/nanny -c -h 192.168.1.1 -p 80 -r 80 -s GET /
HTTP/1.0\r\n\r\n -x HTTP -a 15 -I /sbin/ipvsadm -t 6 -w 1 -V 192.168.1.80 -M g -U none --lvs
```

主调度器查看LVS调度策略

```
[root@active ~]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
-> RemoteAddress:Port Forward Weight ActiveConn InActConn
TCP 192.168.1.80:80 rr persistent 300
-> 192.168.1.1:80      Route 1    0      0
-> 192.168.1.5:80      Route 1    0      1
```

主调度器查看VIP

```
[root@active ~]# ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
  link/ether 52:54:00:52:f8:20 brd ff:ff:ff:ff:ff:ff
  inet 192.168.1.2/24 scope global eth0
    inet 192.168.1.80/32 brd 192.168.1.80 scope global eth0:1
    inet6 fe80::5054:ff:fe52:f820/64 scope link
      valid_lft forever preferred_lft forever
```

从同网段的客户端VIP响应

```
[root@client ~]# arping -I eth0 192.168.1.80
ARPING 192.168.1.80 from 192.168.1.254 br0
Unicast reply from 192.168.1.80 [52:54:00:52:F8:20] 1.034ms
Unicast reply from 192.168.1.80 [52:54:00:52:F8:20] 1.018ms
Unicast reply from 192.168.1.80 [52:54:00:52:F8:20] 0.944ms
Unicast reply from 192.168.1.80 [52:54:00:52:F8:20] 0.907ms
Unicast reply from 192.168.1.80 [52:54:00:52:F8:20] 0.862ms
Unicast reply from 192.168.1.80 [52:54:00:52:F8:20] 0.921ms
```

从客户端访问网站

=====一切正常！！！=====

二、集群故障测试

1. 测试网站访问正常

2. 停止主调度器（停止网卡），查看备调度器能否正常接管

以下查看均在备调度器上：

```
[root@backup ~]# tail -f /var/log/messages
May 26 15:35:17 test3 pulse[5034]: partner dead: activating lvs
May 26 15:35:17 test3 kernel: send_arp uses obsolete (PF_INET,SOCK_PACKET)
May 26 15:35:17 test3 lvsd[5038]: starting virtual service http_server active: 80
May 26 15:35:17 test3 kernel: IPVS: [rr] scheduler registered.
May 26 15:35:17 test3 lvsd[5038]: create_monitor for http_server/node1.tianyun.com running as pid 5050
May 26 15:35:17 test3 lvsd[5038]: create_monitor for http_server/node2.tianyun.com running as pid 5051
May 26 15:35:17 test3 nanny[5051]: starting LVS client monitor for 192.168.1.80:80 -> 192.168.1.5:80
May 26 15:35:17 test3 nanny[5050]: starting LVS client monitor for 192.168.1.80:80 -> 192.168.1.1:80
May 26 15:35:17 test3 nanny[5051]: [ active ] making 192.168.1.5:80 available
May 26 15:35:17 test3 nanny[5050]: [ active ] making 192.168.1.1:80 available
May 26 15:35:22 test3 pulse[5040]: gratuitous lvs arps finished
```

```
[root@backup ~]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
    -> RemoteAddress:Port      Forward Weight ActiveConn InActConn
TCP 192.168.1.80:80 rr persistent 300
    -> 192.168.1.1:80          Route   1    0      5
    -> 192.168.1.5:80          Route   1    0      1
```

```
[root@backup ~]# ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 52:54:00:58:65:cb brd ff:ff:ff:ff:ff:ff
        inet 192.168.1.3/24 brd 192.168.1.255 scope global eth0
            inet 192.168.1.80/32 brd 192.168.1.80 scope global eth0:1
                inet6 fe80::5054:ff:fe58:65cb/64 scope link
                    valid_lft forever preferred_lft forever
```

3. 主调度器重新上线

会自动从备调度器接管调度角色

```
[root@backup ~]# tail -f /var/log/messages
May 26 15:35:17 test3 kernel: send_arp uses obsolete (PF_INET,SOCK_PACKET)
May 26 15:35:17 test3 lvsd[5038]: starting virtual service http_server active: 80
May 26 15:35:17 test3 kernel: IPVS: [rr] scheduler registered.
May 26 15:35:17 test3 lvsd[5038]: create_monitor for http_server/node1.tianyun.com running as pid 5050
May 26 15:35:17 test3 lvsd[5038]: create_monitor for http_server/node2.tianyun.com running as pid 5051
May 26 15:35:17 test3 nanny[5051]: starting LVS client monitor for 192.168.1.80:80 -> 192.168.1.5:80
May 26 15:35:17 test3 nanny[5050]: starting LVS client monitor for 192.168.1.80:80 -> 192.168.1.1:80
May 26 15:35:17 test3 nanny[5051]: [ active ] making 192.168.1.5:80 available
May 26 15:35:17 test3 nanny[5050]: [ active ] making 192.168.1.1:80 available
May 26 15:35:22 test3 pulse[5040]: gratuitous lvs arps finished
May 26 15:39:05 test3 pulse[5034]: partner active: deactivating LVS
May 26 15:39:05 test3 lvsd[5038]: shutting down due to signal 15
May 26 15:39:05 test3 lvsd[5038]: shutting down virtual service http_server
May 26 15:39:05 test3 nanny[5050]: Terminating due to signal 15
May 26 15:39:05 test3 nanny[5051]: Terminating due to signal 15
```

3. 停止Real Server

调度器会自动将失效的Real Server从调度pool中删除

```
[root@active ~]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
    -> RemoteAddress:Port      Forward Weight ActiveConn InActConn
TCP 192.168.1.80:80 rr persistent 300
    -> 192.168.1.5:80          Route   1    0      25
```

4. 启用Real Server

重新启动Real Server后，调度器会自动将其加入到调度pool

```
[root@active ~]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
-> RemoteAddress:Port      Forward Weight ActiveConn InActConn
TCP 192.168.1.80:80 rr persistent 300
-> 192.168.1.1:80          Route 1    0      0
-> 192.168.1.5:80          Route 1    0      26
```

LB Nginx (L7)

Upstream模块

Nginx实现七层的负载均衡

Nginx通过Upstream模块实现负载均衡

upstream支持的负载均衡算法

轮询 (默认) : 可以通过weight指定轮询的权重，权重越大，被调度的次数越多
ip_hash : 根据每个请求IP进行调度，可以解决session的问题，不能使用weight
fair : 可以根据请求页面的大小和加载时间长短进行调度，[使用第三方的upstream_fair模块](#)
url_hash : 按请求的url的hash进行调度，从而使每个url定向到同一服务器，[使用第三方的hash模块](#)

upstream支持的状态参数 (主要用于对后端服务器的健康检查)

down : 暂停对该服务器的调度
backup: 类似于LVS [Sorry Server](#)，当所有的非backup的服务器故障
maxfails: 请求失败的次数，默认为1
fail_timeout: 在经历maxfails次失败后，暂停服务的时间

```
upstream tianyun.com {
#   ip_hash;
   server 192.168.122.10 weight=1 max_fails=2 fail_timeout=2;
   server 192.168.122.20 weight=2 max_fails=2 fail_timeout=2;
   server 192.168.122.30 max_fails=2 fail_timeout=5 down;
   server 192.168.122.200 backup;
}
```

注：当使用ip_hash时，服务器状态不可使用weight和backup

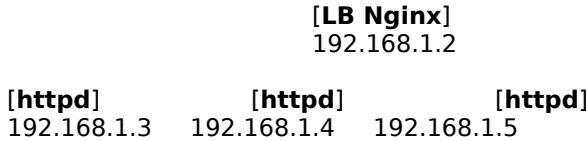
到相同后端服务器

Nginx实现七层的负载均衡

调度到同一组后端服务器

网站没按业务/版块拆分，所有后端服务器提供整站代码。

拓扑结构



实施过程

1. nginx

```
http {  
    upstream httpservers {  
        server 192.168.1.3:80 weight=1 max_fails=2 fail_timeout=2;  
        server 192.168.1.4:80 weight=2 max_fails=2 fail_timeout=2;  
        server 192.168.1.5:80 weight=2 max_fails=2 fail_timeout=2;  
        server 192.168.1.100:80 backup;  
    }  
  
    server {  
        location / {  
            proxy_pass http://httpservers;  
            proxy_set_header X-Real-IP $remote_addr;  
        }  
    }  
}
```

2. Apache LogFormat 可选

```
LogFormat "%{X-Real-IP}i %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" combined  
LogFormat "%h %l %u %t \"%r\" %>s %b" common  
LogFormat "%{Referer}i -> %U" referer  
LogFormat "%{User-agent}i" agent
```

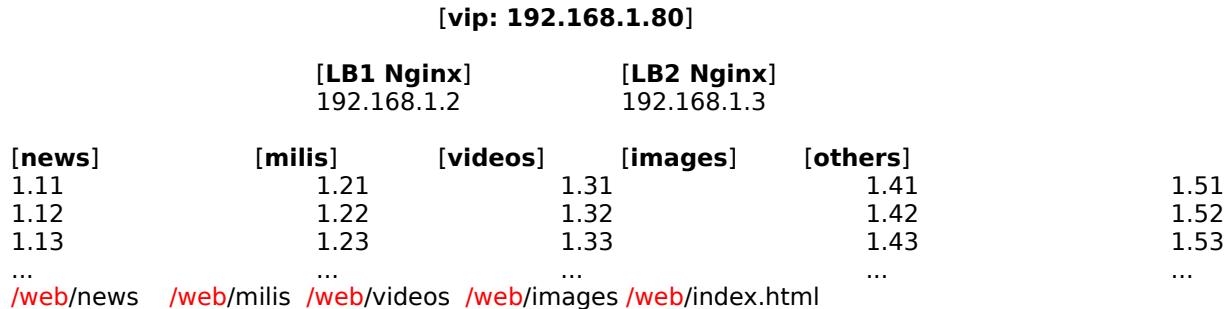
到不同后端服务器

Nginx实现七层的负载均衡

调度到不同组后端服务器

1. 动静分离
 2. 网站进行分区
-

拓扑结构



一、实施过程

1. 根据站点分区进行调度

```
http {  
    upstream news {  
        server 192.168.1.11:80 weight=1 max_fails=2 fail_timeout=2;  
        server 192.168.1.12:80 weight=2 max_fails=2 fail_timeout=2;  
        server 192.168.1.13:80 weight=2 max_fails=2 fail_timeout=2;  
    }  
  
    upstream milis {  
        server 192.168.1.21:80 weight=1 max_fails=2 fail_timeout=2;  
        server 192.168.1.22:80 weight=2 max_fails=2 fail_timeout=2;  
        server 192.168.1.23:80 weight=2 max_fails=2 fail_timeout=2;  
    }  
  
    upstream videos {  
        server 192.168.1.31:80 weight=1 max_fails=2 fail_timeout=2;  
        server 192.168.1.32:80 weight=2 max_fails=2 fail_timeout=2;  
        server 192.168.1.33:80 weight=2 max_fails=2 fail_timeout=2;  
    }  
  
    upstream images {  
        server 192.168.1.41:80 weight=1 max_fails=2 fail_timeout=2;  
        server 192.168.1.42:80 weight=2 max_fails=2 fail_timeout=2;  
        server 192.168.1.43:80 weight=2 max_fails=2 fail_timeout=2;  
    }  
  
    upstream others {  
        server 192.168.1.51:80 weight=1 max_fails=2 fail_timeout=2;  
        server 192.168.1.52:80 weight=2 max_fails=2 fail_timeout=2;  
        server 192.168.1.53:80 weight=2 max_fails=2 fail_timeout=2;  
    }  
  
    server {  
        location / {  
            proxy_pass http://others;  
        }  
  
        location /news {  
            proxy_pass http://news;  
        }  
  
        location /mili {  
            proxy_pass http://milis;  
        }  
    }  
}
```

```

proxy_pass http://milis;
}

location ~* \.(wmv|mp4|rmvb)$ {
proxy_pass http://videos;
}

location ~* \.(png|gif|jpg)$ {
proxy_pass http://images;
}

}

```

2. 根据动静分离进行调度

```

http {
upstream htmlservers {
server 192.168.1.3:80 weight=1 max_fails=2 fail_timeout=2;
server 192.168.1.4:80 weight=2 max_fails=2 fail_timeout=2;
}

upstream phpservers {
server 192.168.1.3:80 weight=1 max_fails=2 fail_timeout=2;
server 192.168.1.4:80 weight=2 max_fails=2 fail_timeout=2;
}

server {
location ~* \.html$ {
proxy_pass http://htmlservers;
}

location ~* \.php$ {
proxy_pass http://phpservers;
}
}
}

```

二、Keepalived实现调度器HA

注：主/备调度器均能够实现正常调度

1. 主/备调度器安装软件

```
[root@master ~]# yum -y install keepalived
[root@backup ~]# yum -y install keepalived
```

2. Keepalived

BACKUP1

```
[root@tianyun ~]# vim /etc/keepalived/keepalived.conf
! Configuration File for keepalived
```

```

global_defs {
  router_id director1          //辅助改为director2
}

vrrp_instance VI_1 {
  state BACKUP
  nopreempt
  interface eth0                //心跳接口，尽量单独连接心跳
  virtual_router_id 80           //整个集群的调度器一致
  priority 100                  //辅助改为50
  advert_int 1
  authentication {
    auth_type PASS
    auth_pass 1111
  }
  virtual_ipaddress {
    192.168.1.80
  }
}

BACKUP2

```

3. 启动KeepAlived (主备均启动)

```
[root@tianyun ~]# chkconfig keepalived on  
[root@tianyun ~]# service keepalived start  
[root@tianyun ~]# ip addr
```

到此：

可以解决心跳故障 keepalived
不能解决Nginx服务故障

4. 扩展对调度器Nginx健康检查 (可选)

思路：

让Keepalived以一定时间间隔执行一个外部脚本，脚本的功能是当Nginx失败，则关闭本机的Keepalived

a. script

```
[root@master ~]# cat /etc/keepalived/check_nginx_status.sh  
#!/bin/bash  
/usr/bin/curl -I http://localhost &>/dev/null  
if [ $? -ne 0 ];then  
    /etc/init.d/keepalived stop  
fi  
[root@master ~]# chmod a+x /etc/keepalived/check_nginx_status.sh
```

b. keepalived使用script

! Configuration File for keepalived

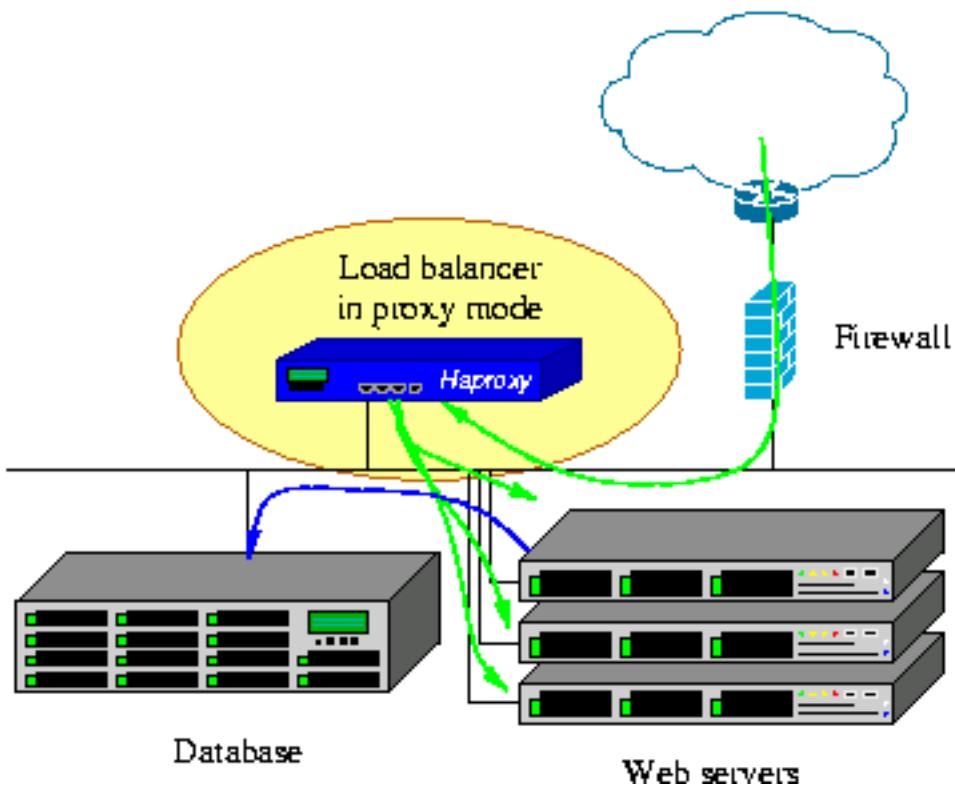
```
global_defs {  
    router_id director1  
}  
  
vrrp_script check_nginx {  
    script "/etc/keepalived/check_nginx_status.sh"  
    interval 5  
}  
  
vrrp_instance VI_1 {  
    state BACKUP  
    interface eth0  
    nopreempt  
    virtual_router_id 90  
    priority 100  
    advert_int 1  
    authentication {  
        auth_type PASS  
        auth_pass tianyun  
    }  
    virtual_ipaddress {  
        192.168.1.80  
    }  
    track_script {  
        check_nginx  
    }  
}
```

注：必须先启动nginx，再启动keepalived

Haproxy 基础

Keepalived + Haproxy

http://haproxy.lwt.eu



/etc/haproxy/haproxy.cfg

```
global
  log          127.0.0.1 local2
  chroot      /var/lib/haproxy
  pidfile     /var/run/haproxy.pid
  maxconn    4000
  user        haproxy
  group       haproxy
  daemon
```

//关于进程的全局参数

defaults、listen、frontend、backend //关于Proxy配置段

defaults 段用于为其它配置段提供默认参数

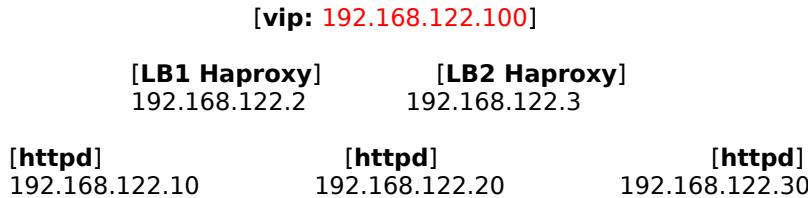
listen是frontend和backend的结合体

Haproxy L7

Keepalived + Haproxy

http://haproxy.lwt.eu

拓扑结构



一、Haproxy实施步骤

1. 准备工作（集群中所有主机）

IP, hostname, hosts, iptables, SELinux, ssh trust, ntp
[root@tianyun ~]# cat /etc/hosts
127.0.0.1 localhost
192.168.122.2 director1.tianyun.com active
192.168.122.3 director2.tianyun.com backup
192.168.122.10 node1.tianyun.com node1
192.168.122.20 node2.tianyun.com node2
192.168.122.30 node3.tianyun.com node3

2. RS配置

配置好网站服务器，测试所有RS

3. 调度器配置Haproxy（主/备）

```
[root@tianyun ~]# yum -y install haproxy
[root@tianyun ~]# cp -rf /etc/haproxy/haproxy.cfg{,.bak}
[root@tianyun ~]# sed -i -r '/^#[^#]/d;/^$/d' /etc/haproxy/haproxy.cfg
[root@tianyun ~]# vim /etc/haproxy/haproxy.cfg
global
defaults
-----配置监控-----
listen stats
    bind      *:1314
    stats     enable
    stats refresh      30s
    stats      hide-version
    stats uri   /haproxy/stats
    stats realm  Haproxy\ stats
    stats auth   tianyun:123
    stats admin  if TRUE
-----
frontend web
    mode      http
    bind      *:80
    default_backend httpservers
```

backend **httpservers**

balance roundrobin

server http1 192.168.122.10:80 maxconn 2000 weight 1 check inter 1s rise 2 fall 2

server http2 192.168.122.20:80 maxconn 2000 weight 1 check inter 1s rise 2 fall 2

server http3 192.168.122.30:80 maxconn 2000 weight 1 check inter 1s rise 2 fall 2

[root@tianyun ~]# service haproxy restart

[root@tianyun ~]# chkconfig haproxy on

4. 测试调度器(主/备)

Statistics Report for HAProxy - Mozilla Firefox

文件(F) 编辑(E) 查看(V) 历史(S) 书签(B) 工具(T) 帮助(H)

Statistics Report for HAProxy

192.168.1.3:1314/haproxy/stats

Google

HAProxy

Statistics Report for pid 22274

General process information

pid = 22274 (process #1, nprocsec = 1)
uptime = 0d 0h0m19s
system limits: memmax = unlimited; ulimit.n = 8016
maxsock = 8016; maxconn = 4000; maxpipes = 0
current connns = 1; current pipes = 0
Running tasks: 1/4

Display option: [Hide DOWN servers](#) [Refresh now](#) [CSV export](#)

External resources: [Primary site](#) [Updates \(v1.4\)](#) [Online manual](#)

Note: UP with load-balancing disabled is reported as "NOLB".

stats		Sessions												Bytes				Denied				Errors				Warnings				Server							
Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Downtime	Thrtile									
Frontend		-	1	1	-	1	1	3 000	1		0	0	0	0	0					OPEN																	
Backend	0	0	0	0	0	0	0	3 000	0	0	0	0	0	0	0	0	0	0	0	19s UP		0	0	0	0	0	0										

web		Sessions												Bytes				Denied				Errors				Warnings				Server							
Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Downtime	Thrtile									
Frontend		-	0	0	-	0	0	3 000	0		0	0	0	0	0					OPEN																	

httpservers		Sessions												Bytes				Denied				Errors				Warnings				Server							
Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Downtime	Thrtile									
<input type="checkbox"/>	http1	0	0	-	0	0	0	0	2 000	0	0	0	0	0	0	0	0	0	0	19s UP	L4OK in 0ms	1	Y	-	0	0	0s	-									
<input type="checkbox"/>	http2	0	0	-	0	0	0	0	2 000	0	0	0	0	0	0	0	0	0	0	19s UP	L4OK in 0ms	1	Y	-	0	0	0s	-									
<input type="checkbox"/>	http3	0	0	-	0	0	0	0	2 000	0	0	0	0	0	0	0	0	0	0	19s UP	L4OK in 0ms	1	Y	-	0	0	0s	-									
Backend	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	19s UP		3	3	0	0	0	0s										

Choose the action to perform on the checked servers:

Statistics Report for HAProxy - Mozilla Firefox

文件(E) 编辑(E) 查看(V) 历史(S) 书签(B) 工具(T) 帮助(H)

Statistics Report for HAProxy

192.168.1.3:1314/haproxystats Google

HAProxy

Statistics Report for pid 22274

General process information

pid = 22274 (process #1, nobjects = 1)
 uptime = 0d 0h0m22s
 system limits: memmax = unlimited; ulimit-n = 8016
 maxsock = 8016; maxcore = 4000; maxpipes = 0
 current connns = 1; current pipes = 0/0
 Running tasks: 1/4

Note: UP with load-balancing disabled is reported as "NOLB".

Display option: Hide DOWN servers Refresh now CSV export External resources: Primary site Updates (v1.4) Online manual

stats												Server																
	Queue			Session rate			Sessions			Bytes		Denied		Errors		Warnings		Status	LastChk	Wght	Act	Bck	Chk	Dwn	Downme	Throttle		
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn										Resp	Retr
Frontend				1	1	-	1	1	3 000	4	1 242	38 554	0	0	0						OPEN							
Backend	0	0		0	0		0	0	3 000	0	0	1 242	38 554	0	0	0	0	0	0	0	2m22s UP		0	0	0		0	

web												Server															
	Queue			Session rate			Sessions			Bytes		Denied		Errors		Warnings		Status	LastChk	Wght	Act	Bck	Chk	Dwn	Downme	Throttle	
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn										Resp
Frontend	0	0	-	0	0	3 000	0	0	0	0	0	0	0	0	0	0	0	0	0	OPEN							

httpservers												Server															
	Queue			Session rate			Sessions			Bytes		Denied		Errors		Warnings		Status	LastChk	Wght	Act	Bck	Chk	Dwn	Downme	Throttle	
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn										Resp
http1	0	0	-	0	0		0	0	2000	0	0	0	0	0	0	0	0	0	0	2m22s UP	L4OK in 0ms	1	Y	-	0	0	0s
http2	0	0	-	0	0		0	0	2000	0	0	0	0	0	0	0	0	0	0	2m22s UP	L4OK in 0ms	1	Y	-	0	0	0s
http3	0	0	-	0	0		0	0	2000	0	0	0	0	0	0	0	0	0	0	1s DOWN	L4CON in 0ms	1	Y	-	2	1	1s
Backend	0	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	2m22s UP		2	2	0		0	0s

Choose the action to perform on the checked servers:

二、Keepalived实现调度器HA

注：主/备调度器均能够实现正常调度

1. 主/备调度器安装软件

```
[root@master ~]# yum -y install keepalived
[root@backup ~]# yum -y install keepalived
```

2. Keepalived

Master

```
[root@tianyun ~]# vim /etc/keepalived/keepalived.conf
! Configuration File for keepalived
```

```
global_defs {
    router_id director1          //辅助改为director2
}

vrrp_instance VI_1 {
    state BACKUP
    nopreempt
    interface eth0           //心跳接口，尽量单独连接心跳
    virtual_router_id 80      //MASTER,BACKUP一致
    priority 100             //辅助改为50
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.122.100
    }
}
```

3. 启动KeepAlived (主备均启动)

```
[root@tianyun ~]# chkconfig keepalived on
[root@tianyun ~]# service keepalived start
[root@tianyun ~]# ip addr
```

4. 扩展对调度器Haproxy健康检查 (可选)

思路：
让Keepalived以一定时间间隔执行一个外部脚本，脚本的功能是当Haproxy失败，则关闭本机的Keepalived

a. script

```
[root@master ~]# cat /etc/keepalived/check_haproxy_status.sh
#!/bin/bash
/usr/bin/curl -I http://localhost &>/dev/null
if [ $? -ne 0 ];then
    /etc/init.d/keepalived stop
fi
[root@master ~]# chmod a+x /etc/keepalived/check_haproxy_status.sh
```

b. keepalived使用script

! Configuration File for keepalived

```
global_defs {
    router_id director1
}

vrrp_script check_haproxy {
    script "/etc/keepalived/check_haproxy_status.sh"
    interval 5
}

vrrp_instance VI_1 {
    state BACKUP
    interface eth0
    nopreempt
    virtual_router_id 90
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass tianyun
    }
    virtual_ipaddress {
        192.168.122.100
    }
    track_script {
        check_haproxy
    }
}
```

```
=====
listen www.tianyun.com
mode http
bind *:80
balance roundrobin
server http1 192.168.122.10:80 maxconn 2000 weight 1 check inter 1s rise 2 fall 2
server http2 192.168.122.20:80 maxconn 2000 weight 1 check inter 1s rise 2 fall 2
server http3 192.168.122.30:80 maxconn 2000 weight 1 check inter 1s rise 2 fall 2
```

Haproxy L4

Haproxy L4

```
global
  log          127.0.0.1 local2
  chroot       /var/lib/haproxy
  pidfile      /var/run/haproxy.pid
  maxconn      4000
  user         haproxy
  group        haproxy
  daemon

defaults
  mode          http
  log           global
  option        dontlognull
  retries       3
  maxconn      3000
  contimeout   50000
  clitimeout   50000
  srftimeout   50000

listen stats
  bind         *:1314
  stats        enable
  stats        hide-version
  stats uri    /haproxystats
  stats realm  Haproxy\ stats
  stats auth   tianyun:123
  stats admin  if TRUE

frontend web
  option        httplog
  option        http-server-close
  option forwardfor except 127.0.0.0/8
  #option       redispach
  mode          http
  bind         *:80
  default_backend  httpservers

backend httpservers
  balance roundrobin
  server http1 192.168.122.10:80 check maxconn 2000
  server http2 192.168.122.20:80 check maxconn 2000
  server http3 192.168.122.30:80 check maxconn 2000

listen mysql
  bind *:3306
  mode tcp
  balance roundrobin
  server mysql1 192.168.122.40:3306 weight 1 check inter 1s rise 2 fall 2
  server mysql2 192.168.122.50:3306 weight 1 check inter 1s rise 2 fall 2
  server mysql3 192.168.122.60:3306 weight 1 check inter 1s rise 2 fall 2
```

Statistics Report for HAProxy - Mozilla Firefox

文件(E) 编辑(E) 查看(V) 历史(S) 书签(B) 工具(I) 帮助(H)

Statistics Report for HAProxy

192.168.1.2:1314/haproxystats

Google

current connis = 1; current pipes = 0/0
Running tasks: 1/7

active or backup DOWN for not checked
active or backup DOWN for maintenance (MAINT)
Note: UP with load-balancing disabled is reported as "NOLB".

stats

	Queue			Session rate			Sessions			Bytes			Denied			Errors			Warnings			Server						
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Downtme
Frontend			-	1	1	-	1	1	3 000	1	0	0	0	0	0						OPEN							
Backend	0	0	-	0	0	-	0	0	3 000	0	0	0	0	0	0	0	0	0	0	0	Ss UP			0	0	0	0	0

web

	Queue			Session rate			Sessions			Bytes			Denied			Errors			Warnings			Server						
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Downtme
Frontend			-	0	0	-	0	0	3 000	0	0	0	0	0	0						OPEN							

httpservers

	Queue			Session rate			Sessions			Bytes			Denied			Errors			Warnings			Server						
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Downtme
<input type="checkbox"/> http1	0	0	-	0	0	-	0	0	2 000	0	0	0	0	0	0	0	0	0	0	Ss UP	L4OK in 0ms	1	Y	-	0	0	0s	-
<input type="checkbox"/> http2	0	0	-	0	0	-	0	0	2 000	0	0	0	0	0	0	0	0	0	0	Ss UP	L4OK in 0ms	1	Y	-	0	0	0s	-
<input type="checkbox"/> http3	0	0	-	0	0	-	0	0	2 000	0	0	0	0	0	0	0	0	0	0	Ss UP	L4OK in 0ms	1	Y	-	0	0	0s	-
Backend	0	0	-	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	Ss UP		3	3	0	0	0	0s	-

Choose the action to perform on the checked servers:

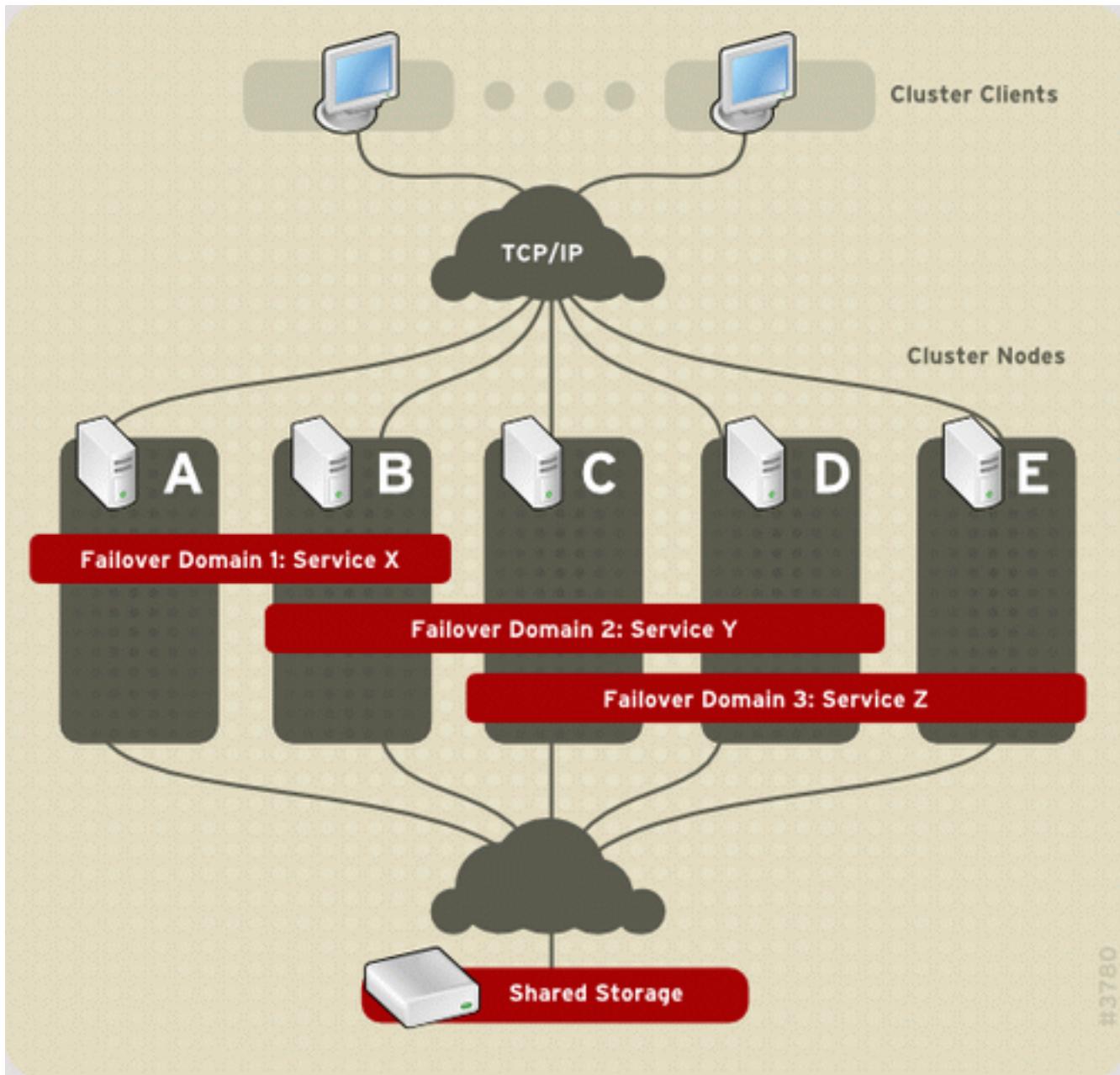
mysql

	Queue			Session rate			Sessions			Bytes			Denied			Errors			Warnings			Server						
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Downtme
Frontend			-	0	0	-	0	0	3 000	0	0	0	0	0	0						OPEN							
<input type="checkbox"/> mysql1	0	0	-	0	0	-	0	0	-	0	0	0	0	0	0	0	0	0	0	Ss UP	L4OK in 0ms	1	Y	-	0	0	0s	-
<input type="checkbox"/> mysql2	0	0	-	0	0	-	0	0	-	0	0	0	0	0	0	0	0	0	0	Ss UP	L4OK in 0ms	1	Y	-	0	0	0s	-
<input type="checkbox"/> mysql3	0	0	-	0	0	-	0	0	-	0	0	0	0	0	0	0	0	0	0	Ss UP	L4OK in 0ms	1	Y	-	0	0	0s	-
Backend	0	0	-	0	0	-	0	0	3 000	0	0	0	0	0	0	0	0	0	0	Ss UP		3	3	0	0	0	0s	-

Choose the action to perform on the checked servers:

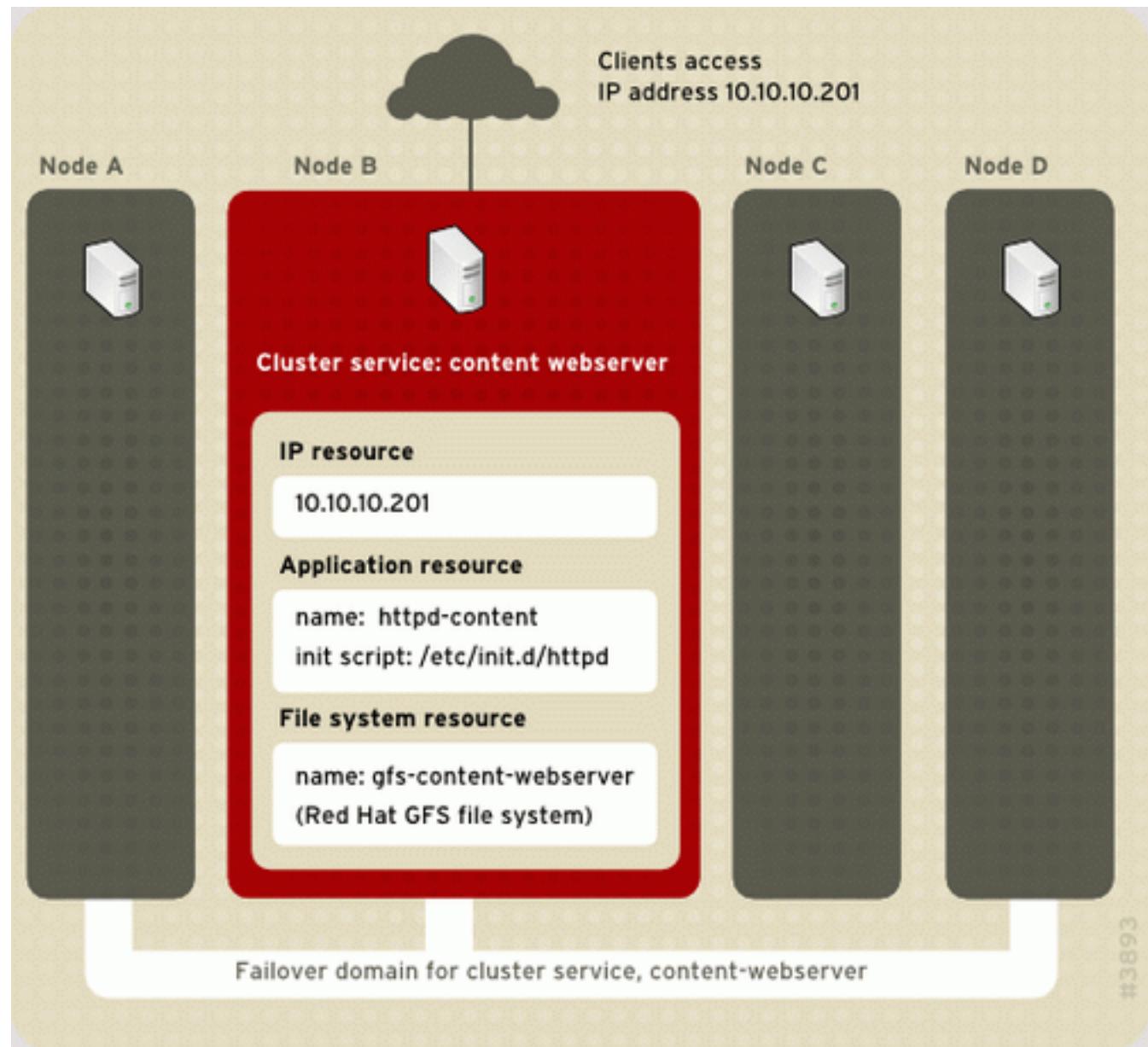
HA RHCS

RHCS基础架构

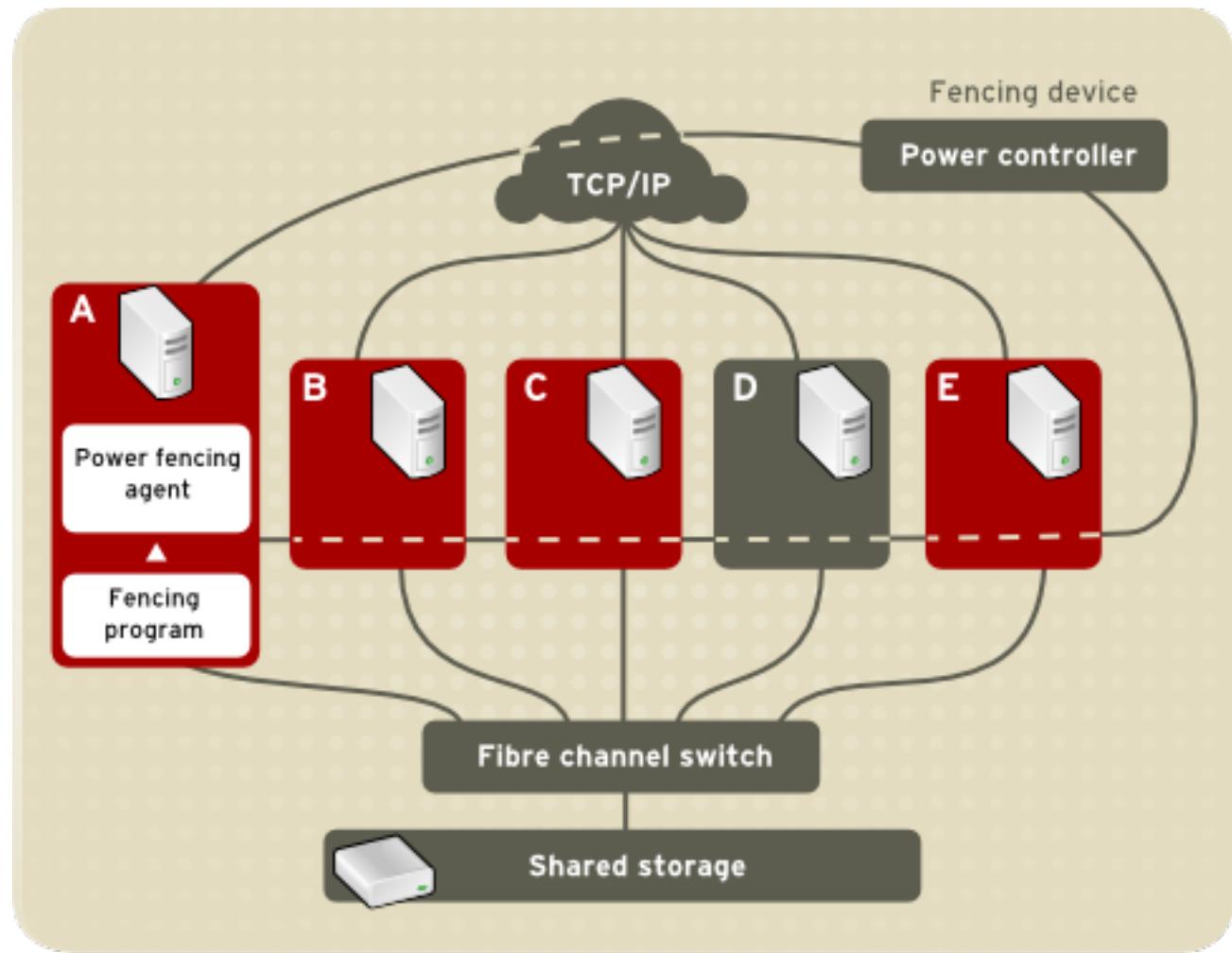


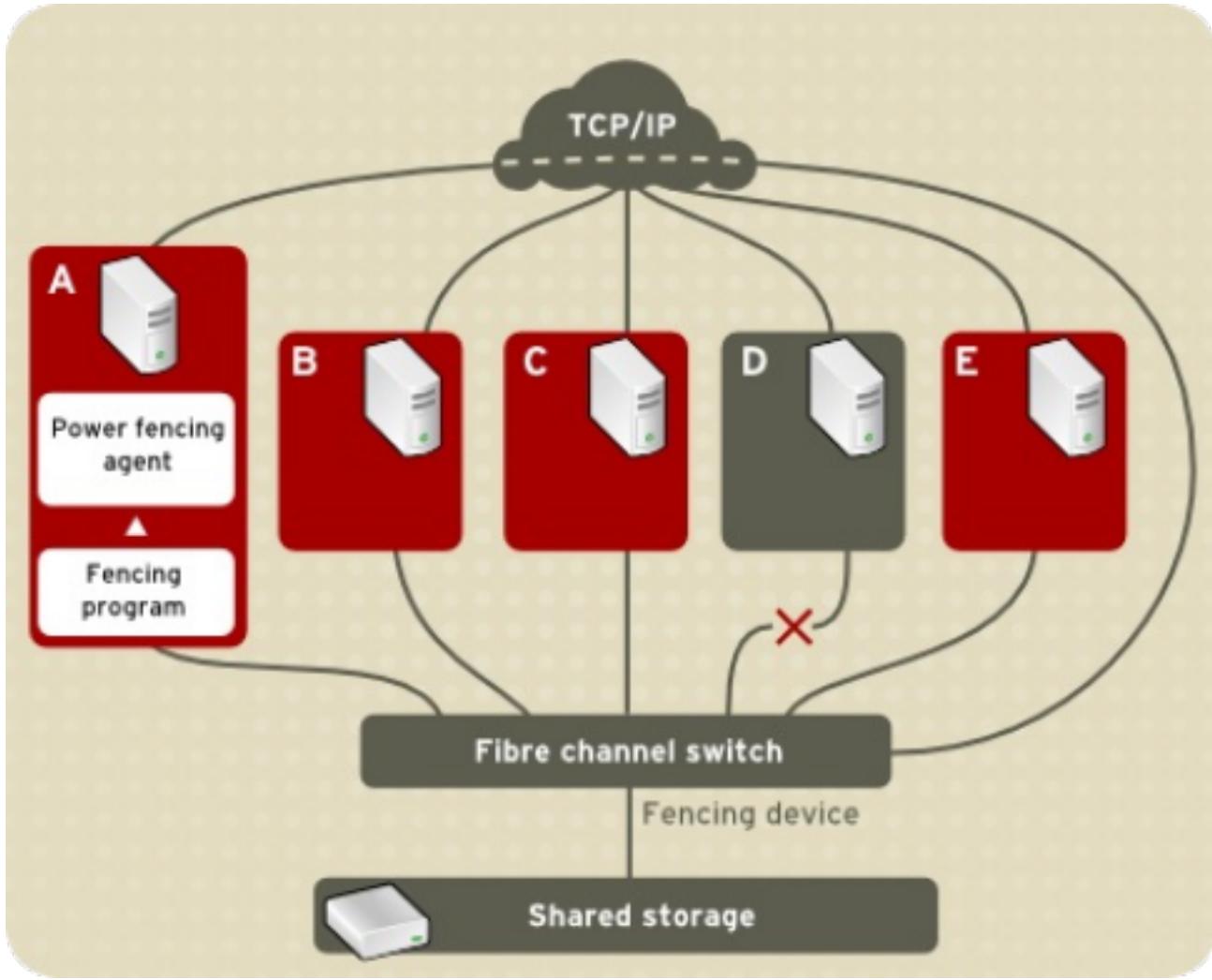
图一、Cluster Nodes、Failover Domain

#3780



图二、service





图三、Fencing Device

- 1、服务(**Service**)，是HA集群中提供的资源组，包括Float IP，FS，Script等等。
- 2、节点(**Node**)，是HA中实际运行服务提供资源的服务器。
- 3、失效域(**Failover Domain**)，是HA中提供资源的服务器的集合，当内部某个成员出现故障时，可以将服务切换到其他正常的成员服务器上。
- 4、心跳(**HeartBeat**)是HA中监控成员服务器状态的方法，一般心跳是通过网线和串口线来传输的。
- 5、单一故障点(Single Point Of Failure，**SPOF**)是指一个系统的这样的一个部件，当它失效或者停止运行，将导致整个系统不能工作。在HA中通常使用双电源，多网卡，双交换机等来避免SPOF。
- 6、仲裁(**Quorum**)是HA中为了准确的判断服务器及其提供的服务是否正常而采用的在共享磁盘中保存成员服务器信息的方法。共享的状态信息包括群集是否活跃。服务状态信息包括服务是否在运行以及哪个成员正在运行该服务。每个成员都检查这些信息来保证其它成员处于最新状态。在一个只有两个成员的群集中，每个成员都定期把一个时间戳和群集状态信息写入位于共享磁盘存储区的两个共享群集分区上。要保证正确的群集操作，如果某成员无法在启动时写入主共享群集分区和屏蔽共享群集分区，它将不会被允许加入群集。此外，如果某群集成员不更新其时间戳，或者到系统的"heartbeats"(心跳)失败了，该成员就会从群集中删除。
- 7、**Fence**设备，Fence设备的作用时在一个节点 心跳**HeartBeat** 出现问题时，另一个节点通过fence设备把出现问题的节点重新启动，这样做到了非人工的干预和防止出现问题的节点访问共享存储，造成文件系统的冲突，关于Fence 设备，有外置的比如APC的电源管理器，很多服务器都是内置的，只不过不同厂家的叫法不同而已。比如HP的称为iLo，IBM的称为BMC，Dell的称为DRAC。

LUCI——RHCS(RedHat Cluster Suite)提供了多种集群配置和管理工具，常用的有基于GUI的system-config-cluster、Conga等，也提供了基于命令行的管理工具。system-config-cluster是一个用于创建集群和配置集群节点的图形化管理工具，它有集群节点配置和集群管理两个部分组成，分别用于创建集群节点配置

文件和维护节点运行状态。一般用在RHCS早期的版本中。Conga是一种新的基于网络的集群配置工具，与system-config-cluster不同的是，Conga是通过web方式来配置和管理集群节点的。Conga有两部分组成，分别是luci和ricci，luci安装在一独立的计算机上(即服务器)，用于配置和管理集群，ricci安装在每个集群节点上，Luci通过ricci和集群中的每个节点进行通信。

RHCS主要进程

cman	RHCS集群的核心进程，维护整个集群包括心跳通信，其它进程都以cman为基础
rgmanager	管理集群资源、service
fenced	
modclusterd	
clvmd	集群级别LVM
...	
ricci	作为agent运行在集群节点，用于接受luci对集群配置和管理

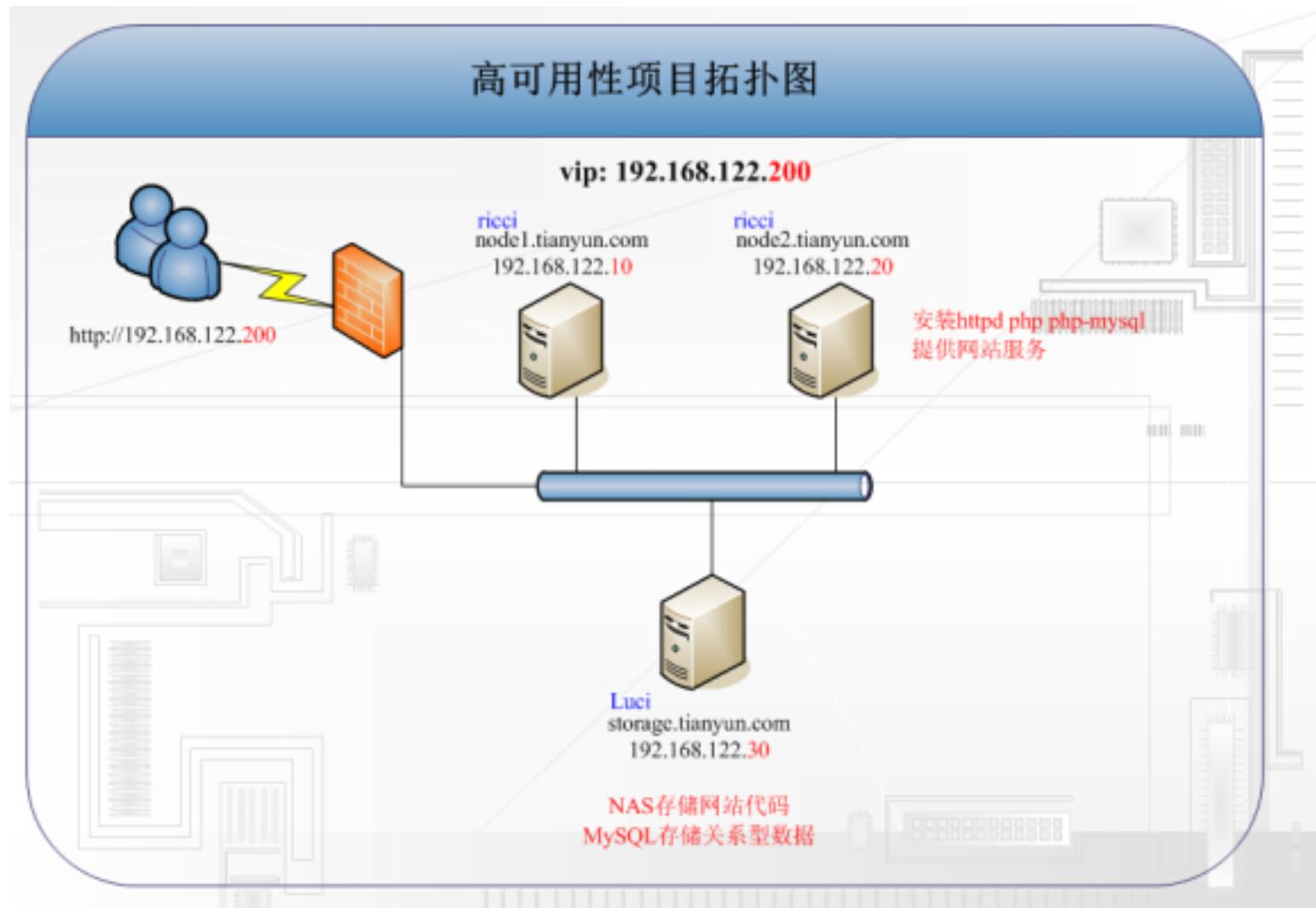
```
root@web1:~  
File Edit View Search Terminal Help  
[root@web1 ~]# /etc/init.d/cman start  
Starting cluster:  
  Checking if cluster has been disabled at boot... [ OK ]  
  Checking Network Manager... [ OK ]  
  Global setup... [ OK ]  
  Loading kernel modules... [ OK ]  
  Mounting configfs... [ OK ]  
  Starting cman... [ OK ]  
  Waiting for quorum... [ OK ]  
  Starting fenced... [ OK ]  
  Starting dlm_controld... [ OK ]  
  Tuning DLM kernel config... [ OK ]  
  Starting gfs_controld... [ OK ]  
  Unfencing self... [ OK ]  
  Joining fence domain... [ OK ]  
[root@web1 ~]#
```

Apache_HA+NAS

KVM: Apache_HA + NAS

项目环境 : KVM

拓扑结构 :



项目说明 : Conga

luci : 管理集群，包括创建集群配置文件等

ricci : 运行在集群节点上，作为agent和luci通讯

...

Services (HA浮动资源集合)

IP : 192.168.122.200

Filesystem (netfs--nfsmount) : mount -t nfs 192.168.122.30:/var/web /var/www/html

Script : /etc/init.d/httpd

一、实施准备

1. 准备工作 (集群节点 永久配置 重启节点)

IP

NetworkManager!

hostname!

hosts!

```
iptables  
SELinux  
ssh trust  
ntp  
yum (Server, HighAvailability, LoadBalancer, ResilientStorage, ScalableFileSystem)  
yum -y install httpd  
reboot
```

示例：

```
[root@tianyun ~]# cat /etc/hosts  
127.0.0.1      localhost  
192.168.122.10    node1.tianyun.com node1  
192.168.122.20    node2.tianyun.com node2
```

存储准备 [可选]

1. [NFS SERVER] : 存储端

```
[root@nfs_server ~]# yum -y install nfs-utils  
[root@nfs_server ~]# mkdir /var/web  
[root@nfs_server ~]# echo "test html" > /var/web/index.html  
[root@nfs_server ~]# vim /etc/exports  
/var/web    192.168.122.0/24(rw,sync,no_root_squash)  
[root@nfs_server ~]# service rpcbind restart  
[root@nfs_server ~]# service nfs restart  
[root@nfs_server ~]# chkconfig nfs on  
[root@nfs_server ~]# chkconfig rpcbind on
```

2. 测试 node1 node2

```
[root@node1 ~]# yum -y install nfs-utils  
[root@node1 ~]# showmount -e 192.168.122.30  
Export list for storage:  
/var/web 192.168.122.0/24  
[root@node1 ~]# mount -t nfs 192.168.122.30:/var/web /var/www/html/  
[root@node1 ~]# ls /var/www/html/  
index.html  
[root@node1 ~]# umount /var/www/html/          //测试挂载，并将其卸载
```

node2测试同上

二、配置集群

1. 安装ricci (node1.tianyun.com node2.tianyun.com)

```
[root@node1 ~]# yum -y install ricci  
[root@node1 ~]# passwd ricci  
[root@node1 ~]# service ricci start  
[root@node1 ~]# chkconfig ricci on
```

node2同上

2. 在管理机安装luci

```
[root@tianyun ~]# ping node1.tianyun.com -c1  
64 bytes from node1.tianyun.com (192.168.8.101): icmp_seq=1 ttl=64 time=0.224 ms
```

```
[root@tianyun ~]# ping node2.tianyun.com -c1  
64 bytes from node2.tianyun.com (192.168.8.102): icmp_seq=1 ttl=64 time=0.368 ms
```

```
[root@tianyun ~]# yum -y install luci  
[root@tianyun ~]# service luci start  
Point your web browser to https://tianyun:8084 to access luci
```

3. 使用Luci配置集群

a. 创建集群、添加节点

Manage Cluster - - - > Create - - - > Cluster Name

【可选】

修改组播地址，重启集群

```
[root@node1 ~]# cman_tool status  
Multicast addresses: 239.192.203.182  
[root@node1 ~]# ps aux|grep yum  
root 1920 24.6 46.3 326280 113320 ? D 10:56 0:05 /usr/bin/python /usr/bin/yum -y install cman rgmanager lvm2  
cluster sg3_utils gfs2_utils
```

b. 添加fence设备，并绑定fence到节点

参考后面：《RHEL6中使用KVM作为Fence》

c. 创建故障转移域

Prioritized	Order the nodes to which services failover.	优先级
Restricted	Service can run only on nodes specified.	限制service只能运行在指定的节点
No Fallback	Do not send service back to 1st priority node when it becomes available again.	不切回 (不抢占)

d. 添加高可用资源

IP : 192.168.122.100

NFS Mount:

Name	nfs_mount
Mount point	/var/www/html
Host	192.168.122.50
Export path	/var/web

//建议NFS Mount资源在每个节点上提前自动挂载/etc/fstab

Script : /etc/init.d/httpd

e. 添加service

Run exclusive : 是否以排它的方式运行

三、检查集群

```
[root@node1 ~]# clustat
Cluster Status for http_cluster @ Thu Jun 20 15:40:33 2013
Member Status: Quorate
```

Member Name	ID	Status
node2.tianyun.com	1	Online, rgmanager
node1.tianyun.com	2	Online, Local, rgmanager

Service Name	Owner (Last)	State
service:http_service	node2.tianyun.com	started

检查各种资源：

```
[root@node2 ~]# ip addr          //查看VIP
[root@node2 ~]# df /var/www/html/
192.168.8.103:/var/web
3428096 2612864 638272 81% /var/www/html
[root@node2 ~]# service httpd status
httpd (pid 9449) 正在运行...
```

手动切换services：

```
[root@node1 ~]# clusvcadm --help
clusvcadm -r <group> -m <member>      Relocate <group> [to <member>]
[root@node1 ~]# clusvcadm -r http_service
```

四、测试集群（模拟故障）

资源故障 - - - relocate

手动制造资源故障

umount /var/www/html/
service httpd stop

心跳故障 - - - fence - - - relocate

查看集群状态信息

tail /var/log/messages -f
clustat -i 1
cman_tool status

RHEL6中使用KVM作为Fence

1. KVM物理机

```
[root@hu ~]# yum -y install fence-virt fence-virtd fence-virtd-libvirt fence-virtd-multicast
[root@hu ~]# mkdir -p /etc/cluster
[root@hu ~]# dd if=/dev/urandom of=/etc/cluster/fence_xvm.key bs=4k count=1
```

```

[root@hu ~]# rsync -va /etc/cluster/fence_xvm.key node1:/etc/cluster/
[root@hu ~]# rsync -va /etc/cluster/fence_xvm.key node2:/etc/cluster/
[root@hu ~]# fence_virtd -c           //创建fence-virt配置文件
[root@hu ~]# cat /etc/fence_virt.conf
backends {
    libvirt {
        uri = "qemu:///system";
    }
}

listeners {
    multicast {
        interface = "virbr0";
        port = "1229";
        family = "ipv4";
        address = "225.0.0.12";
        key_file = "/etc/cluster/fence_xvm.key";
    }
}

fence_virtd {
    module_path = "/usr/lib64/fence-virt";
    backend = "libvirt";
    listener = "multicast";
}

```

[root@hu ~]# /etc/init.d/fence_virtd start
正在启动 fence_virtd : [确定]
[root@hu ~]# chkconfig fence_virtd on
[root@hu ~]# ps aux |grep fence
root 15911 0.0 0.0 140272 3528 ? Ss 12:08 0:00 /usr/sbin/fence_virtd -w

[root@tianyun ~]# setenforce 0

2. 虚拟机

```
[root@node01 ~]# yum -y install fence_virt
```

3. Luci集群中添加Fence

添加fence设备

Fence Devices ---> Add ---> Fence virt (Multicast Mode) --->

为每个节点绑定fence

Nodes ---> node1 ---> Add Fence Method

Add Fence Device(Instance) (注意虚拟机名称virsh list)

node1测试Fence功能是否正常：

```
# fence_xvm -H node2
```

五、集群管理

1. cman_tool

```
[root@web1 ~]# cman_tool status
Version: 6.2.0
Config Version: 12
Cluster Name: cluster1-web
Cluster Id: 28880
Cluster Member: Yes
Cluster Generation: 56
Membership state: Cluster-Member
Nodes: 2
Expected votes: 1
Total votes: 2
Node votes: 1
Quorum: 1
Active subsystems: 9
Flags: 2node
```

Ports Bound: 0 11 177
Node name: web1.tianyun.com
Node ID: 1
Multicast addresses: 239.192.112.65
Node addresses: 192.168.122.56

2. 手动修改集群配置文件

```
[root@node1 ~]# vim /etc/cluster/cluster.conf
[root@node1 ~]# ccs_config_validate
Configuration validates

[root@node1 ~]# cman_tool version -r
You have not authenticated to the ricci daemon on node1.zhou.com
Password:
You have not authenticated to the ricci daemon on node2.zhou.com
Password:
```

3. 通过ccs管理集群

```
[root@node1 ~]# rpm -qf /usr/sbin/ccs
ccs-0.16.2-63.el6.x86_64
[root@node1 ~]# ccs -h node1.zhou.com --stopall
[root@node1 ~]# ccs -h node1.zhou.com --startall
=====
```

Apache_HA+IP_SAN+Clvmd+EXT4

Apache_HA+IP_SAN+Clvmd+EXT4

适用于HA集群使用的共享存储

一、存储配置 (Clvmd+Ext4)

存储端 (target)

1、安装包

```
[root@target ~]# yum -y install scsi-target-utils
```

2、定义target共享

```
[root@target ~]# vim /etc/tgt/targets.conf
<target iqn.2016-10.com.example:web>
    backing-store /dev/vdb          //新添加1G磁盘
</target>
```

3、启动服务

```
[root@target ~]# service tgtd start
[root@target ~]# netstat -tnlp |grep :3260
tcp      0      0 0.0.0.0:3260          0.0.0.0:*          LISTEN      15135/tgtd
tcp      0      0 :::3260            :::*          LISTEN      15135/tgtd
```

```
[root@storage ~]# tgt-admin --show
Target 1: iqn.2015-10.com.example:mysql
    Backing store path: /dev/sdb
    ACL information:
        ALL
```

集群节点 initiator端 (node1、node2)

1、安装包

```
[root@node1 ~]# yum -y install iscsi-*
```

2、发现存储

```
[root@node1 ~]# iscsidadm -m discovery -t st -p 192.168.122.50
正在启动 iscsid : [确定]
192.168.122.50:3260,1 iqn.2015-10.com.example:mysql
```

3、登入存储

```
[root@node1 ~]# service iscsi restart
[root@node1 ~]# chkconfig iscsi on
```

4、查看设备

```
[root@node1 ~]# fdisk -cul
Disk /dev/sdb: 8589 MB, 8589934592 bytes
64 heads, 32 sectors/track, 8192 cylinders, total 16777216 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
```

二、配置集群，并使用CLVMD (node1、node2)

1. 配置集群

- a. 创建集群
- b. 配置Fence

2. 配置CLVMD

确保node1和node2 clvmd运行

```
[root@node1 ~]# /etc/init.d/clvmd status
```

clvmd (pid 1908) 正在运行...

Clustered Volume Groups: (none)

Active clustered Logical Volumes: (none)

node1创建lv

```
[root@web1 ~]# pvcreate /dev/sdb
[root@web1 ~]# vgcreate datavg /dev/sdb
[root@web1 ~]# lvcreate -L 2G -n mysql-lv datavg
[root@web1 ~]# vgs
  VG #PV #LV #SN Attr  VSize VFree
datavg  1  1  0 wz--nc 8.00g 6.00g
```

```
[root@web1 ~]# lvs
  LV   VG Attr  LSize Pool Origin Data% Move Log Cpy%Sync Convert
mysql-lv datavg -wi-a--- 2.00g
```

node2查看

```
[root@node2 ~]# vgs
  VG #PV #LV #SN Attr  VSize VFree
VolGroup 1 2 0 wz--n- 9.51g 0
gfs-vg 1 1 0 wz--nc 5.00g 3.00g
[root@node2 ~]# lvs
  LV   VG Attr  LSize Pool Origin Data% Move Log Cpy%Sync Convert
lv_root VolGroup -wi-ao--- 8.54g
lv_swap VolGroup -wi-ao--- 992.00m
gfs-lv gfs-vg -wi-a--- 2.00g
```

3、创建EXT4文件系统 node或node2

```
[root@node2 ~]# mkfs.ext4 /dev/datavg/mysql-lv
```

MySQL初始化数据库：

1. 安装

```
[root@node1 ~]# yum -y install mysql-server
node2同上
```

2. node1初始化

```
[root@node1 ~]# mount /dev/datavg/mysql-lv /var/lib/mysql/
[root@node1 ~]# service mysqld start
[root@node1 ~]# service mysqld stop
[root@node1 ~]# umount /var/lib/mysql
```

[root@web2 ~]# Apache:

```
[root@web2 ~]# ip: 192.168.122.100
[root@web2 ~]# script: /etc/init.d/httpd
[root@web2 ~]# filesystem: ----- (NAS, 由系统挂载/etc/fstab)
```

[root@web2 ~]# MySQL:

```
[root@web2 ~]# ip: 192.168.122.200
[root@web2 ~]# script: /etc/init.d/mysqld
[root@web2 ~]# filesystem: /dev/datavg/mysql-lv /var/lib/mysql (Ext4, 由集群挂载)
```

Apache_HA+IP_SAN+Clvmd+GFS2

Apache_HA+IP_SAN+Clvmd+GFS2

GFS : 全局文件系统，支持DLM锁管理，支持并发写，RHEL6不超过16个节点

适用于HA集群使用的共享存储

适用于LB集群使用的共享存储

一、存储配置

存储端 (target)

1、安装包

```
[root@target ~]# yum -y install scsi-target-utils
```

2、定义target共享

```
[root@target ~]# vim /etc/tgt/targets.conf
<target iqn.2016-10.com.example:mysql>
    backing-store /dev/sdb
</target>
```

3、启动服务

```
[root@target ~]# service tgtd start
[root@target ~]# netstat -tnlp |grep :3260
tcp      0      0 0.0.0.0:3260          0.0.0.0:*
              LISTEN    15135/tgtd
tcp      0      0 :::3260            :::*
              LISTEN    15135/tgtd
```

```
[root@storage ~]# tgt-admin --show
Target 1: iqn.2015-10.com.example:mysql
    Backing store path: /dev/sdb
    ACL information:
        ALL
```

集群节点 initiator端 (node1、node2)

1、安装包

```
[root@node1 ~]# yum -y install iscsi-*
```

2、发现存储

```
[root@node1 ~]# iscsidadm -m discovery -t st -p 192.168.122.50
正在启动 iscsid : [确定]
192.168.122.50:3260,1 iqn.2016-10.com.example:mysql
```

3、登入存储

```
[root@node1 ~]# service iscsi restart
[root@node1 ~]# chkconfig iscsi on
```

4、查看设备

```
[root@node1 ~]# fdisk -cul
Disk /dev/sdb: 8589 MB, 8589934592 bytes
64 heads, 32 sectors/track, 8192 cylinders, total 16777216 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
```

二、配置集群，并使用CLVMD (node1、node2)

1. 配置集群

a. 创建集群

2. 配置CLVMD

确保node1和node2 clvmd运行

```
[root@node1 ~]# /etc/init.d/clvmd status
clvmd (pid 1908) 正在运行...
Clustered Volume Groups: (none)
Active clustered Logical Volumes: (none)
```

node1创建lv

```
[root@web1 ~]# pvcreate /dev/sdb
[root@web1 ~]# vgcreate datavg /dev/sdb
[root@web1 ~]# lvcreate -L 2G -n mysql-lv datavg
[root@web1 ~]# vgs
  VG #PV #LV #SN Attr  VSize VFree
  datavg 1 1 0 wz--nc 8.00g 6.00g
[root@web1 ~]# lvs
  LV   VG   Attr  LSize Pool Origin Data%  Move Log Cpy%Sync Convert
  mysql-lv datavg -wi-a--- 2.00g
```

node2查看

```
[root@node2 ~]# vgs
  VG #PV #LV #SN Attr  VSize VFree
  VolGroup 1 2 0 wz-n- 9.51g 0
  gfs-vg 1 1 0 wz--nc 5.00g 3.00g
[root@node2 ~]# lvs
  LV   VG   Attr  LSize Pool Origin Data%  Move Log Cpy%Sync Convert
```

```
lv_root VolGroup -wi-ao--- 8.54g  
lv_swap VolGroup -wi-ao--- 992.00m  
gfs-lv gfs-vg -wi-a---- 2.00g
```

3、创建GFS2文件系统 node或node2

a、在node1,node2上安装gfs2的相关软件包

```
[root@node1 ~]# rpm -qa |grep gfs  
gfs2-utils-3.0.12.1-49.el6.x86_64 //提供mkfs.gfs2命令用于创建gfs2文件系统
```

```
[root@node1 ~]# modinfo -F filename gfs2  
/lib/modules/2.6.32-358.el6.x86_64/kernel/fs/gfs2/gfs2.ko
```

b、在任意一个节点上格式化

```
[root@node1 ~]# cman_tool status  
Version: 6.2.0  
Config Version: 1  
Cluster Name: http-cluster
```

```
[root@node1 ~]# mkfs.gfs2 -t http-cluster:table1 -p lock_dlm -j 3 -J 128M /dev/datavg/mysql-lv  
This will destroy any data on /dev/datavg/mysql-lv.  
It appears to contain: symbolic link to `../dm-2'
```

Are you sure you want to proceed? [y/n] y

```
Device: /dev/datavg/mysql-lv  
Blocksize: 4096  
Device Size 2.00 GB (524288 blocks)  
Filesystem Size: 2.00 GB (524288 blocks)  
Journals: 3  
Resource Groups: 8  
Locking Protocol: "lock_dlm"  
Lock Table: "http-cluster:table1"  
UUID: 98326a5a-f72c-5a1d-6ccd-bec094cab26
```

```
-t clusternamespace:locktablename  
-p lock_dlm  
-j 指定日志空间个数，此个数一定要大于等于使用gfs2文件系统存储的主机数  
-J 指定一个日志空间的大小
```

c、挂载并测试写入

```
[root@node1 ~]# vim /etc/fstab  
/dev/datavg/mysql-lv /var/www/html gfs2 defaults,_netdev 0 0  
[root@node1 ~]# mount -a
```

```
[root@node1 ~]# df -P  
/dev/mapper/datavg-mysql--lv gfs2 2.0G 410M 1.7G 20% /var/lib/mysql
```

GFS扩展知识：

1. 使用非DLM锁挂载GFS2

```
[root@node2 ~]# mount -a  
/sbin/mount.gfs2: node not a member of the default fence domain  
/sbin/mount.gfs2: error mounting lockproto lock_dlm
```

```
[root@node2 ~]# mount -o lockproto=lock_nolock /dev/datavg/mysql-lv /var/www/html/
```

2. 当集群节点扩充时，增加日志的个数

MySQL初始化数据库：

1. 安装

```
[root@node1 ~]# yum -y install mysql-server  
node2同上
```

2. node1初始化

```
[root@node1 ~]# mount /dev/datavg/mysql-lv /var/lib/mysql/  
[root@node1 ~]# service mysqld start  
[root@node1 ~]# service mysqld stop
```

```
[root@node1 ~]# umount /var/lib/mysql
```

```
[root@web2 ~]# Apache:  
[root@web2 ~]# ip: 192.168.122.100  
[root@web2 ~]# script: /etc/init.d/httpd  
[root@web2 ~]# filesystem: ----- (NAS , 由系统挂载/etc/fstab)
```

```
[root@web2 ~]# MySQL:  
[root@web2 ~]# ip: 192.168.122.200  
[root@web2 ~]# script: /etc/init.d/mysqld  
[root@web2 ~]# filesystem: /dev/datavg/mysql-lv /var/lib/mysql (GFS2 , 由集群挂载 或 由系统挂载/etc/fstab)
```

LB+Keepalived+IP_SAN+Clvmd+GFS2

LB集群构建

LVS(DR)+Keepalived+IP_SAN+Clvmd+GFS2

环境说明：

所有节点均使用KVM虚拟机，测试之前建议将物理机的ip_forward off

拓扑结构：

VIP : 192.168.1.100

Director : 192.168.122.10 192.168.122.20
 lvs-master lvs-backup

Real Server : 192.168.122.30 192.168.122.40
 web1 web2

san storage : 192.168.122.200
 storage

准备工作（集群中所有主机）

IP, hostname, hosts, iptables, SELinux, ntp

```
[root@web1 ~]# cat /etc/hosts
```

```
127.0.0.1   localhost  
192.168.122.10   lvs-master  
192.168.122.20   lvs-backup  
192.168.122.30   web1  
192.168.122.40   web2  
192.168.122.200   storage
```

web1,web2

```
[root@web1 ~]# yum -y install httpd php php-mysql
```

一、存储共享配置CLVM+GFS2

存储端 (storage)

1、安装包

```
[root@target ~]# yum -y install scsi-target-utils
```

2、准备一个分区或者一块硬盘

```
[root@target ~]# fdisk -cu /dev/sda
[root@target ~]# partx -a /dev/sda
```

3、定义target共享

```
[root@target ~]# vim /etc/tgt/targets.conf
default-driver iscsi
<target iqn.2014-09.com.tianyun:san1>
    backing-store /dev/sdb
    initiator-address 192.168.122.30
    initiator-address 192.168.122.40
</target>
```

4、启动服务

```
[root@target ~]# service tgtd start
[root@target ~]# tgt-admin --show
Target 1: iqn.2014-09.com.tianyun:san1
LUN: 1
    Type: disk
    SCSI ID: IET 00010001
    SCSI SN: beaf11
    Size: 5369 MB, Block size: 512
    Online: Yes
    Removable media: No
    Prevent removal: No
    Readonly: No
    Backing store type: rdwr
    Backing store path: /dev/sdb
    Backing store flags:
Account information:
ACL information:
    192.168.122.30
    192.168.122.40
```

```
[root@target ~]# netstat -tnlp |grep :3260
tcp      0      0 0.0.0.0:3260          0.0.0.0:*
              LISTEN      15135/tgtd
tcp      0      0 :::3260            :::*
              LISTEN      15135/tgtd
```

initiator端login存储 (web1、web2)

1、安装包

```
[root@web1 ~]# yum -y install iscsi-*
```

2、发现存储

```
[root@web1 ~]# iscsidadm -m discovery -t st -p 192.168.122.200
```

3、登入存储

```
[root@web1 ~]# service iscsi restart
[root@web1 ~]# chkconfig iscsi on
```

4、查看设备

```
[root@web1 ~]# fdisk -cul
Disk /dev/sda: 5368 MB, 5368709120 bytes
166 heads, 62 sectors/track, 1018 cylinders, total 10485760 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
```

创建配置集群

安装ricci (web1,web2)

```
[root@web1 ~]# yum -y install ricci
[root@web1 ~]# passwd ricci
[root@web1 ~]# service ricci start
[root@web1 ~]# chkconfig ricci on
```

在管理机安装luci (任意节点)

```
[root@tianyun ~]# ping web1.tianyun.com -c1
64 bytes from web1.tianyun.com (192.168.1.6): icmp_seq=1 ttl=64 time=0.224 ms
```

```
[root@tianyun ~]# ping web2.tianyun.com -c1
```

```
64 bytes from web2.tianyun.com (192.168.1.7): icmp_seq=1 ttl=64 time=0.368 ms
```

```
[root@tianyun ~]# yum -y install luci  
[root@tianyun ~]# service luci restart  
Point your web browser to https://tianyun:8084 to access luci
```

使用Luci配置集群

https://luci_ip:8084

创建集群、添加节点： Manage Cluster - - - > Create - - - > Cluster Name (gfs-cluster)

2. 配置CLVMD

确保web1和web2 clvmd运行

```
[root@web1 ~]# /etc/init.d/clvmd status  
clvmd (pid 1908) 正在运行...  
Clustered Volume Groups: (none)  
Active clustered Logical Volumes: (none)
```

web1创建lvm

```
[root@web1 ~]# pvcreate /dev/sda  
[root@web1 ~]# vgcreate gfs-vg /dev/sda  
[root@web1 ~]# lvcreate -L 2G -n gfs-lv gfs-vg
```

web2查看

```
[root@web2 ~]# vgs  
[root@web2 ~]# lvs
```

创建GFS2文件系统 (web1, web2)

1、在web1,web2上安装gfs2的相关软件包

```
[root@web1 ~]# rpm -qa |grep gfs  
gfs2-utils-3.0.12.1-49.el6.x86_64 //提供mkfs.gfs2命令用于创建gfs2文件系统
```

```
[root@web1 ~]# modinfo -F filename gfs2  
/lib/modules/2.6.32-358.el6.x86_64/kernel/fs/gfs2/gfs2.ko
```

2、在任意一个节点上格式化

```
[root@web1 ~]# cman_tool status  
Version: 6.2.0  
Config Version: 1  
Cluster Name: gfs-cluster
```

```
[root@web1 ~]# mkfs.gfs2 -t gfs-cluster:table1 -p lock_dlm -j 3 -J 128M /dev/gfs-vg/gfs-lv
```

3、挂载并测试写入

```
[root@web1 ~]# vim /etc/fstab  
/dev/gfs-vg/gfs-lv /var/www/html gfs2 defaults,_netdev 0 0  
[root@web1 ~]# mount -a
```

```
[root@web1 ~]# df -P  
/dev/mapper/gfs--vg-gfs--lv 2096912 397156 1699756 19% /var/www/html
```

二、LVS+keepalived

LB集群配置

1. RS配置(web1,web2)

配置好网站服务器，测试所有RS

```
[root@web1 ~]# echo "ip addr add dev lo 192.168.122.100/32" >> /etc/rc.local  
[root@web1 ~]# echo "net.ipv4.conf.all.arp_ignore = 1" >> /etc/sysctl.conf  
[root@web1 ~]# sysctl -p  
[root@web1 ~]# yum -y install httpd php php-mysql  
[root@web1 ~]# echo "web1..." >> /var/www/html/index.html
```

2. 主/备调度器安装软件

```
[root@lvs-master ~]# yum -y install ipvsadm keepalived  
[root@lvs-backup ~]# yum -y install ipvsadm keepalived
```

3. Keepalived

lvs-master

获得Real Server测试页面的MD5SUM值

```
[root@lvs-master ~]# genhash -s 192.168.122.30 -p 80 -u /test.html
```

```
MD5SUM = f5ac8127b3b6b85cdc13f237c6005d80
```

```
[root@lvs-master ~]# vim /etc/keepalived/keepalived.conf
```

```
! Configuration File for keepalived
```

```
global_defs {  
    router_id lvs-master      //辅助改为lvs-backup  
}
```

```
vrrp_instance VI_1 {  
    state BACKUP  
    nopreempt  
    interface eth0           //心跳接口，尽量单独连接心跳  
    virtual_router_id 80     //主备一致  
    priority 100            //辅助改为50  
    advert_int 1  
    authentication {  
        auth_type PASS  
        auth_pass 1111  
    }  
    virtual_ipaddress {  
        192.168.122.100  
    }  
}
```

```
virtual_server 192.168.122.100 80 {
```

```
    delay_loop 6  
    lb_algo rr  
    lb_kind DR  
    nat_mask 255.255.255.0  
    persistence_timeout 2  
    protocol TCP
```

```
real_server 192.168.122.30 80 {
```

```
    weight 1  
    HTTP_GET {  
        url {  
            path /test.html  
            digest f5ac8127b3b6b85cdc13f237c6005d80  
        }  
        connect_timeout 3  
        nb_get_retry 3  
        delay_before_retry 3  
    }  
}
```

```
real_server 192.168.122.40 80 {
```

```
    weight 1  
    HTTP_GET {  
        url {  
            path /test2.html  
            digest f5ac8127b3b6b85cdc13f237c6005d80  
        }  
        connect_timeout 3  
        nb_get_retry 3  
        delay_before_retry 3  
    }  
}
```

lvs-backup

5. 启动KeepAlived (主备均启动)

```
[root@lvs-master ~]# chkconfig keepalived on
```

```
[root@lvs-master ~]# service keepalived start
```

```
[root@lvs-master ~]# tail -f /var/log/messages
```

```
[root@tianyun ~]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
-> RemoteAddress:Port      Forward Weight ActiveConn InActConn
TCP 192.168.122.100:80 wrr
-> 192.168.122.30:80        Route  1    0      0
-> 192.168.122.30:80        Route  3    0      0
```

LB集群测试

所有分发器和Real Server都正常

主分发器故障及恢复

Real Server故障及恢复

三、集群整体测试

1. 上传网站代码文件，例如wordpress
2. 创建数据库并授权
3. 导入数据库表结构

<http://vip>

四、LB集群增加Real Server

1. 新增的Real Server 加入到RHCS集群
2. 新增的Real Server 发现并连接存储 vgscan
3. 新增的Real Server 安装httpd
4. 新增的Real Server fstab自动挂载GFS2
5. 新增的Real Server 配置DR模式所需vip,non-arp
6. 主备调度器 keepalived.conf增加Real Server
7. 测试集群

RHEL6 GFS文件系统最多支持16节点

RHCS 生产环境中的注意事项

1. 将ACPI配置为使用整合的Fence设备
2. Grub.conf中timeout的设置
3. 不支持在集群节点中使用 NetworkManager
4. 配置使用冗余电源的Fence (非整合电源) 例如分别连接了两台NPS

Fence Devices

Method

Name	Type/Values	Remove
pwr01	APC Power Device port : 1 option : off	X
pwr02	APC Power Device port : 1 option : off	X
pwr01	APC Power Device port : 1 option : on	X
pwr02	APC Power Device port : 1 option : on	X

[Add Fence Instance](#)

[Add Fence Method](#)

Cluster Daemons

	Status
cman	Running
rgmanager	Not running

图 3.6. 双电源 Fencing 配置

5. 配置Fence时注意Method 和 instance 的区别

两种Method: 例如备选的Fence方式

两种Instance: 例如连接双电源

6. GFS2

不支持使用 GFS2 进行部署超过 16 个节点的集群文件系统

GFS2 是基于 64 位构架，理论上可提供 8 EB 文件系统。但是，目前支持的 64 位硬件的最大 GFS2 文件系统为 100 TB，为 32 位硬件支持的最大 GFS2 文件系统为 16 TB。

7. Anywhere HA

Nginx_Director_HA
Haproxy_Director_HA

ip 资源

脚本资源 可以以如下的方式执行
/script.sh start|stop|status

HA Keepalived

LVS+KeepAlived

LVS + KeepAlived

KeepAlived在该项目中的功能：

1. 管理IPVS的路由表（包括对RealServer做健康检查）

2. 实现调度器的HA

<http://www.keepalived.org>

项目环境： VS/DR

Client 192.168.122.1

Director分发器 主192.168.122.10 备192.168.122.20 VIP **192.168.122.100**

Real Server 192.168.122.30 192.168.122.40 192.168.122.50 VIP lo**192.168.122.100**/32

实施步骤：

1. RS配置(web1,web2)

配置好网站服务器，测试所有RS

```
[root@web1 ~]# echo "ip addr add dev lo 192.168.122.100/32" >> /etc/rc.local
[root@web1 ~]# echo "net.ipv4.conf.all.arp_ignore = 1" >> /etc/sysctl.conf
[root@web1 ~]# sysctl -p
[root@web1 ~]# yum -y install httpd php php-mysql
[root@web1 ~]# echo "web1..." >> /var/www/html/index.html
```

2. 主/备调度器安装软件

```
[root@lvs-master ~]# yum -y install ipvsadm keepalived
[root@lvs-backup ~]# yum -y install ipvsadm keepalived
```

3. Keepalived

lvs-master

获得Real Server测试页面的MD5SUM值

```
[root@lvs-master ~]# genhash -s 192.168.122.30 -p 80 -u /test.html
MD5SUM = f5ac8127b3b6b85cdc13f237c6005d80
```

```
[root@lvs-master ~]# vim /etc/keepalived/keepalived.conf
! Configuration File for keepalived
```

```
global_defs {
    router_id lvs-master      //辅助改为lvs-backup
}

vrrp_instance VI_1 {
    state BACKUP
    nopreempt
    interface eth0           //心跳接口，尽量单独连接心跳
    virtual_router_id 80      //主备一致
    priority 100              //辅助改为50
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.122.100
    }
}
```

```

        }
    }

virtual_server 192.168.122.100 80 {
    delay_loop 6
    lb_algo rr
    lb_kind DR
    nat_mask 255.255.255.0
    persistence_timeout 2
    protocol TCP
}

real_server 192.168.122.30 80 {
    weight 1
    inhibit_on_failure
    HTTP_GET{
        url{
            path /test.html
            digest f5ac8127b3b6b85cdc13f237c6005d80
        }
        connect_timeout 3
        nb_get_retry 3
        delay_before_retry 3
    }
}

real_server 192.168.122.40 80 {
    weight 1
    inhibit_on_failure
    HTTP_GET{
        url{
            path /test2.html
            digest f5ac8127b3b6b85cdc13f237c6005d80
        }
        connect_timeout 3
        nb_get_retry 3
        delay_before_retry 3
    }
}

```

lvs-backup

5. 启动KeepAlived (主备均启动)

```
[root@lvs-master ~]# chkconfig keepalived on
[root@lvs-master ~]# service keepalived start
[root@lvs-master ~]# tail -f /var/log/messages
```

```
[root@tianyun ~]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port      Forward Weight ActiveConn InActConn
TCP  192.168.122.100:80 wrr
  -> 192.168.122.30:80       Route  1    0      0
  -> 192.168.122.30:80       Route  3    0      0
```

LB集群测试

所有分发器和Real Server都正常

主分发器故障及恢复

Real Server故障及恢复

Haproxy+KeepAlived

Haproxy + Keepalived

一、 Haproxy负载均衡

主/备调度器均能够实现正常调度

二、 Keepalived实现调度器HA

注：主/备调度器均能够实现正常调度

1. 主/备调度器安装软件

```
[root@master ~]# yum -y install keepalived  
[root@backup ~]# yum -y install keepalived
```

2. Keepalived

Master

```
[root@tianyun ~]# vim /etc/keepalived/keepalived.conf  
! Configuration File for keepalived
```

```
global_defs {  
    router_id director1          //辅助改为director2  
}  
  
vrrp_instance VI_1 {  
    state BACKUP  
    nopreempt  
    interface eth0                //心跳接口，尽量单独连接心跳  
    virtual_router_id 80          //MASTER,BACKUP一致  
    priority 100                 //辅助改为50  
    advert_int 1  
    authentication {  
        auth_type PASS  
        auth_pass 1111  
    }  
    virtual_ipaddress {  
        192.168.122.100  
    }  
}
```

BACKUP

3. 启动KeepAlived（主备均启动）

```
[root@tianyun ~]# chkconfig keepalived on  
[root@tianyun ~]# service keepalived start  
[root@tianyun ~]# ip addr
```

4. 扩展对调度器Haproxy健康检查（可选）

思路：

让Keepalived以一定时间间隔执行一个外部脚本，脚本的功能是当Haproxy失败，则关闭本机的Keepalived

a. script

```
[root@master ~]# cat /etc/keepalived/check_haproxy_status.sh  
#!/bin/bash  
/usr/bin/curl -I http://localhost &>/dev/null  
if [ $? -ne 0 ];then  
    /etc/init.d/keepalived stop  
fi  
[root@master ~]# chmod a+x /etc/keepalived/check_haproxy_status.sh
```

b. keepalived使用script

```
! Configuration File for keepalived
```

```
global_defs {  
    router_id director1  
}  
  
vrrp_script check_haproxy {  
    script "/etc/keepalived/check_haproxy_status.sh"  
    interval 5  
}  
  
vrrp_instance VI_1 {  
    state BACKUP  
    interface eth0  
    nopreempt  
    virtual_router_id 90  
    priority 100  
    advert_int 1  
    authentication {  
        auth_type PASS  
        auth_pass tianyun  
    }  
    virtual_ipaddress {  
        192.168.122.100  
    }  
  
    track_script {  
        check_haproxy  
    }  
}
```

Nginx+Keepalived

Nginx + Keepalived

一、Nginx负载均衡

主/备调度器均能够实现正常调度

二、Keepalived实现调度器HA

1. 主/备调度器安装软件

```
[root@master ~]# yum -y install keepalived  
[root@backup ~]# yum -y install keepalived
```

2. Keepalived

BACKUP1

```
[root@tianyun ~]# vim /etc/keepalived/keepalived.conf  
! Configuration File for keepalived
```

```
global_defs {  
    router_id director1          //辅助改为director2  
}  
  
vrrp_instance VI_1 {  
    state BACKUP  
    nopreempt  
    interface eth0               //心跳接口，尽量单独连接心跳  
    virtual_router_id 80          //整个集群的调度器一致  
    priority 100                 //辅助改为50  
    advert_int 1  
    authentication {  
        auth_type PASS  
        auth_pass 1111  
    }  
}
```

```
}
```

```
virtual_ipaddress {
```

```
    192.168.1.80
}
```

```
}
```

BACKUP2

3. 启动KeepAlived (主备均启动)

```
[root@tianyun ~]# chkconfig keepalived on
[root@tianyun ~]# service keepalived start
[root@tianyun ~]# ip addr
```

到此：

可以解决心跳故障 keepalived

不能解决Nginx服务故障

4. 扩展对调度器Nginx健康检查 (可选)

思路：

让Keepalived以一定时间间隔执行一个外部脚本，脚本的功能是当Nginx失败，则关闭本机的Keepalived

a. script

```
[root@master ~]# cat /etc/keepalived/check_nginx_status.sh
#!/bin/bash
/usr/bin/curl -I http://localhost &>/dev/null
if [ $? -ne 0 ];then
    /etc/init.d/keepalived stop
fi
[root@master ~]# chmod a+x /etc/keepalived/check_nginx_status.sh
```

b. keepalived使用script

! Configuration File for keepalived

```
global_defs {
    router_id director1
}

vrrp_script check_nginx {
    script "/etc/keepalived/check_nginx_status.sh"
    interval 5
}

vrrp_instance VI_1 {
    state BACKUP
    interface eth0
    nopreempt
    virtual_router_id 90
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass tianyun
    }
    virtual_ipaddress {
        192.168.1.80
    }
    track_script {
        check_nginx
    }
}
```

注：必须先启动nginx，再启动keepalived

Apache+Keepalive

所有的Apache节点：

挂载共享存储 (NFS , GFS2)

启动Apache服务

KeepAlived仅实现VIP的切换即可

发现对方心跳故障

应用 (Apache) 故障 ==通过执行外部脚本检查应用运行状态，如果异常则停止Keepaliced

MySQL+Keepalived

Keepalived+mysql 自动切换

项目环境：

VIP 192.168.122.100

mysql1 192.168.122.10

mysql2 192.168.122.20

一、mysql 主主同步 (不使用共享存储，数据保存本地存储)

二、安装keepalived

三、keepalived 主备配置文件

四、mysql状态检测脚本/root/bin/keepalived_check_mysql.sh

五、测试及诊断

注 keepalived之间使用vrrp组播方式通信使用的IP地址是224.0.0.18

```
=====
```

实施步骤：

一、mysql 主主同步 <略>

二、安装keepalived

```
[root@tianyun ~]# yum -y install ipvsadm kernel-headers kernel-devel openssl-devel popt-devel
[root@tianyun ~]# wget http://www.keepalived.org/software/keepalived-1.2.2.tar.gz
[root@tianyun ~]# tar zxfv keepalived-1.2.2.tar.gz
[root@tianyun ~]# cd keepalived-1.2.2
[root@tianyun ~]# ./configure --prefix=/
[root@tianyun ~]# make
[root@tianyun ~]# make install
```

三、keepalived 主备配置文件

192.168.122.10 Master配置

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

```
=====
```

```
=====
```

! Configuration File for keepalived

```
global_defs {
    router_id mysql1
}
```

```
vrrp_script check_run {
    script "/root/keepalived_check_mysql.sh"
    interval 5
}
```

```

vrrp_instance VI_1 {
    state MASTER
    interface eth0
    virtual_router_id 88
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass tianyun
    }

    track_script {
        check_run
    }

    virtual_ipaddress {
        192.168.122.100
    }
}

=====
=====
```

192.168.122.20 Slave配置

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

```

=====
=====

! Configuration File for keepalived

global_defs {
    router_id mysql2
}

vrrp_script check_run {
    script "/root/keepalived_check_mysql.sh"
    interval 5
}

vrrp_instance VI_1 {
    state BACKUP
    interface eth0
    virtual_router_id 88
    priority 50
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass tianyun
    }

    track_script {
        check_run
    }

    virtual_ipaddress {
        192.168.122.100
    }
}
```

1. 注意空格

2. 日志查看脚本是否被执行

```
[root@xen2 ~]# tail -f /var/log/messages
```

```
Jun 19 15:20:19 xen1 Keepalived_vrrp[6341]: Using LinkWatch kernel netlink reflector...
Jun 19 15:20:19 xen1 Keepalived_vrrp[6341]: VRRP sockpool: [ifindex(2), proto(112), fd(11,12)]
Jun 19 15:20:19 xen1 Keepalived_vrrp[6341]: VRRP_Script(check_run) succeeded
```

四、mysql状态检测脚本/root/keepalived_check_mysql.sh (两台MySQL同样的脚本)

版本一：简单使用：

```
#!/bin/bash
/usr/bin/mysql -uroot -p123 -e "show status" &>/dev/null
if [ $? -ne 0 ];then
    service keepalived stop
fi
```

版本二：检查多次：

```
[root@tianyun ~]# vim /root/keepalived_check_mysql.sh
#!/bin/bash
MYSQL=/usr/local/mysql/bin/mysql
MYSQL_HOST=localhost
MYSQL_USER=root
MYSQL_PASSWORD=tianyun
CHECK_TIME=3

#mysql is working MySQL_OK is 1 , mysql down MySQL_OK is 0
MYSQL_OK=1

check_mysql_health(){
    $MYSQL -h $MYSQL_HOST -u $MYSQL_USER -p${MYSQL_PASSWORD} -e "show status" &>/dev/null
    if [ $? -eq 0 ];then
        MYSQL_OK=1
    else
        MYSQL_OK=0
    fi
    return $MYSQL_OK
}

while [ $CHECK_TIME -ne 0 ]
do
    check_mysql_health
    if [ $MYSQL_OK -eq 1 ]; then
        exit 0
    fi

    if [ $MYSQL_OK -eq 0 ] && [ $CHECK_TIME -eq 1 ];then
        /etc/init.d/keepalived stop
        exit 1
    fi
    let CHECK_TIME--
    sleep 1
done
```

版本三：检查多次：

```
[root@tianyun ~]# vim /root/keepalived_check_mysql.sh
#!/bin/bash
MYSQL=/usr/local/mysql/bin/mysql
MYSQL_HOST=localhost
MYSQL_USER=root
MYSQL_PASSWORD=tianyun
CHECK_TIME=3

#mysql is working MySQL_OK is 1 , mysql down MySQL_OK is 0
MYSQL_OK=1

check_mysql_health(){
    $MYSQL -h $MYSQL_HOST -u $MYSQL_USER -p${MYSQL_PASSWORD} -e "show status" &>/dev/null
    if [ $? -eq 0 ];then
        MYSQL_OK=1
    else
        MYSQL_OK=0
    fi
    return $MYSQL_OK
}

while [ $CHECK_TIME -ne 0 ]
do
```

```

check_mysql_health
if [ $MYSQL_OK -eq 1 ] ; then
    exit 0
fi

let CHECK_TIME--
sleep 1
done

/etc/init.d/keepalived stop
exit 1
=====
=====

[root@tianyun ~]# chmod 755 /root/keepalived_check_mysql.sh

```

两边均启动keepalived

```

[root@tianyun ~]# /etc/init.d/keepalived start
[root@tianyun ~]# /etc/init.d/keepalived start
[root@tianyun ~]# chkconfig --add keepalived
[root@tianyun ~]# chkconfig keepalived on

```

Anywhere+Keepalived

所有的应用节点：

挂载共享存储（NFS，GFS2），也可能使用本地存储
启动服务，并设置开机启动

KeepAlive仅实现VIP的切换即可

发现对方心跳故障

应用故障 ==通过执行外部脚本检查应用运行状态，如果异常则停止Keepalived

HA Pacemaker

本节作业

LVS部分：

1. NAT模式：分别从客户端、调度器、RealServer上抓取访问网站的数据包，观察数据包中封装的IP
2. DR模式：分别从客户端、调度器、RealServer上抓取访问网站的数据包，观察数据包中封装的IP
3. 描述LVS NAT模式中，客户访问和收接收时数据包的流向
4. 描述LVS DR模式中，客户访问和收接收时数据包的流向

5. DR模式两台调度器，5台RealServer需要几个公网IP
6. LVS DR模式和TUN模式有什么区别？
7. 了解LVS FullNAT模式的工作原理及应用场景
8. LVS持久性连接的作用是什么？如何实现？
9. 列出至少5种LVS支持的调度算法，以及它们的工作原理及应用场景
10. LVS可以对后端的RealServer实现健康检查吗？
11. LB集群技术中四层调度和七层调度有什么区别？它们的应用场景是什么？
12. 了解什么是网站的灰度发布，例如lua+nginx做灰度发布

LB架构：

L4:

LVS + Keepalived + Nginx
LVS + Keepalived + MySQL Cluster
LVS + Keepalived + Galera MySQL
Haproxy + Keepalived + MySQL Cluster
Haproxy + Keepalived + Galera MySQL

L7:

Nginx + Keepalived + Nginx
Tengine + Keepalived + Nginx
Haproxy + keepalived + Nginx

RHCS:

1. 了解DELL、HP、IBM远程管理卡相关知识及配置
2. 了解IPMI技术及应用场景
3. 文件系统锁，单机锁，分布式锁管理
4. 了解RHCS仲裁磁盘qdisk及配置
5. ACPI的作用是什么
6. 使用ccs管理RHCS集群

Keepalived:

1. keepalived的作用是什么，举例说明应用场景。
2. keepalived主备节点之间是如何通信的，使用什么协议。
3. keepalived通信时使用的组播（多播）地址是什么？
4. 如何在一个网络中构建多套keepalived集群？
5. keepalived的virtual_route_id的范围是什么？
6. master和backup会同时发送组播吗？
7. 如果master和backup同时设置了vip，可能是什么原因造成的？
8. 写出抓取vrrp协议的tcpdump命令。
9. keepalived.conf配置文件主要分为哪几个部分，举例说明。
10. 如何设置keepalived不抢占？
11. sorry server的作用是什么，如何设置？
12. 如何设置当后端RealServer检测失败时，不从IPVS路由表中删除？
13. 举例说明keepalived中执行外部脚本的场景，如何设置外部脚本？
14. 你在项目中使用的keepalived是什么版本，目前最新的版本是什么？

Other:

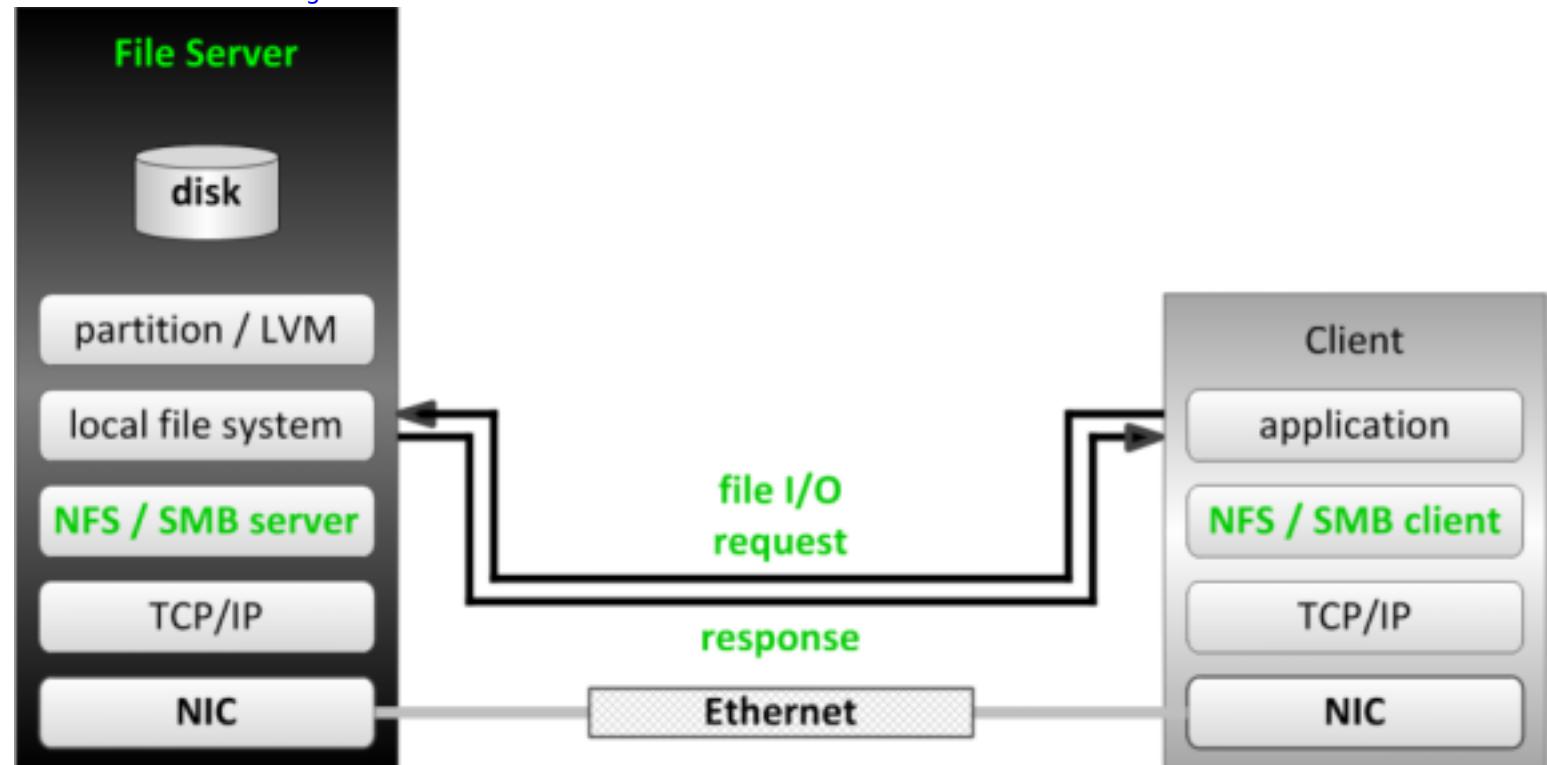
1. 了解Heartbeat HA集群技术
2. 了解Pacemaker HA集群技术
3. 了解Rose HA集群技术
4. 了解PowerHA集群技术
5. 了解Dell r630/720/730 RAID1/5/6配置，包括热备盘(hotspare)
6. 了解HP DL388G7 RAID1/5/6配置，包括热备盘(hotspare)
7. 了解IBM X3650，X3850 M2/M3 RAID1/5/6配置，包括热备盘(hotspare)
8. 了解HP、IBM刀片服务器RAID及基本配置

五、存储技术

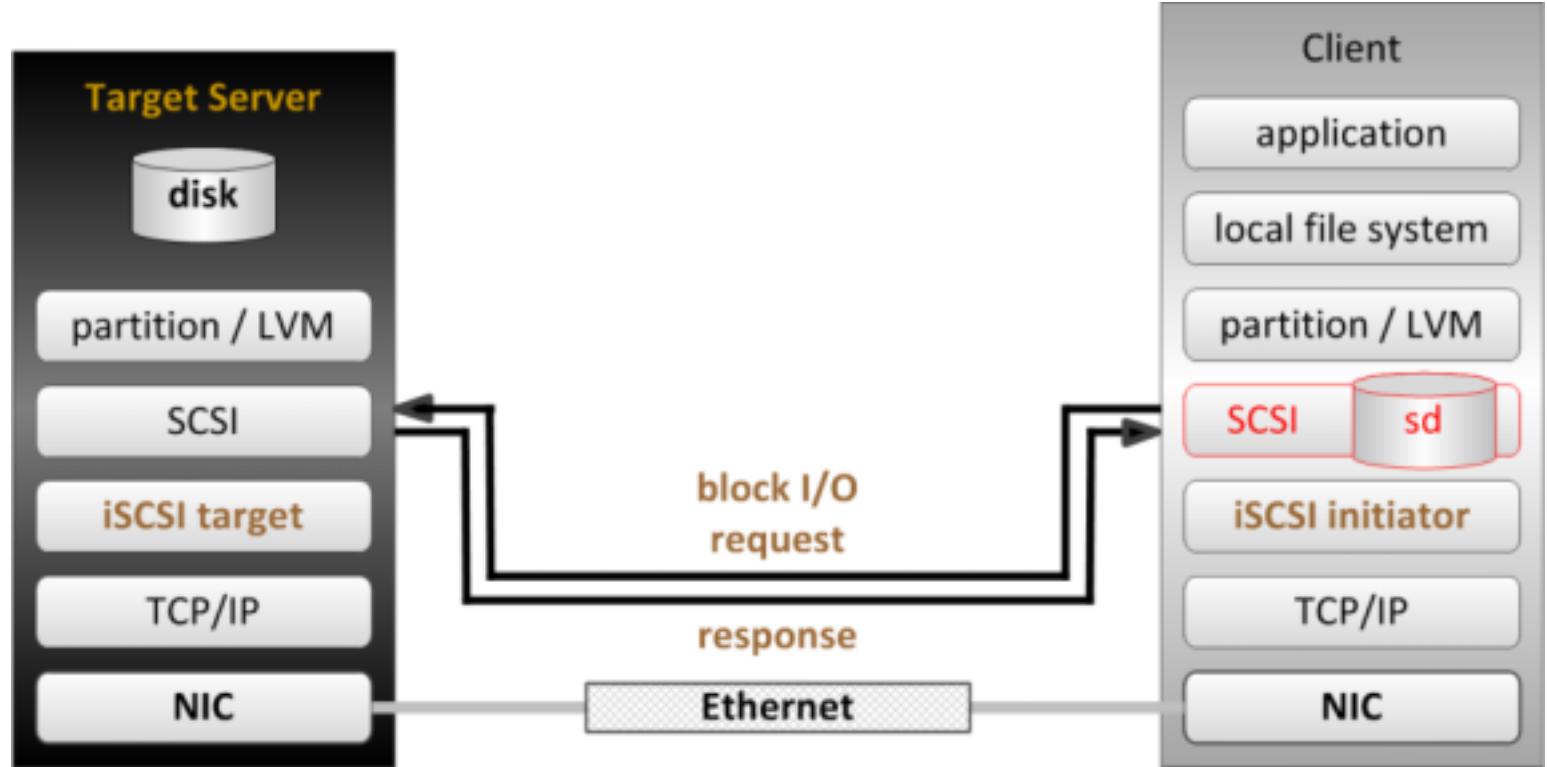
5.0 概览

存储技术概览

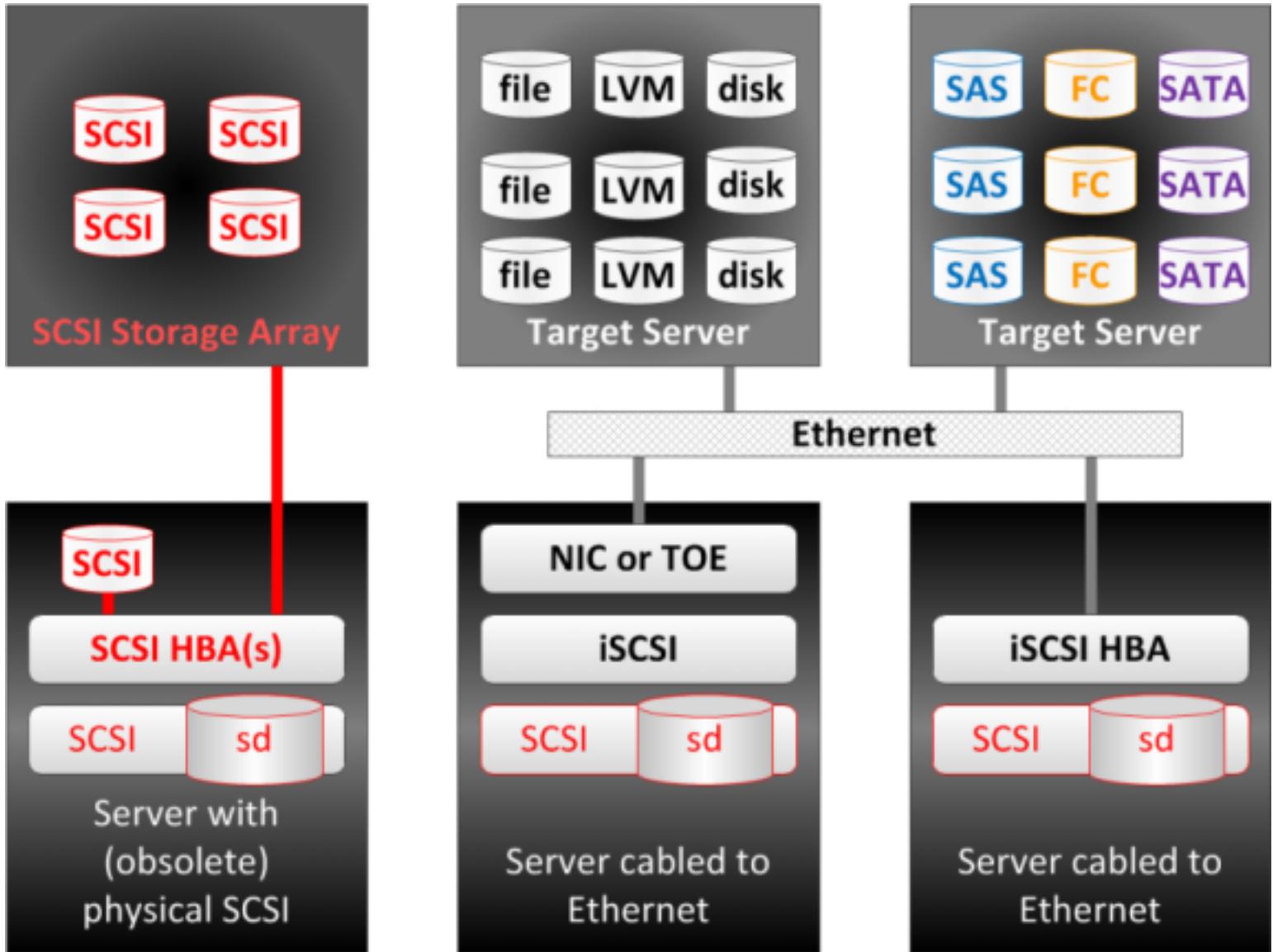
NAS File-based Storage



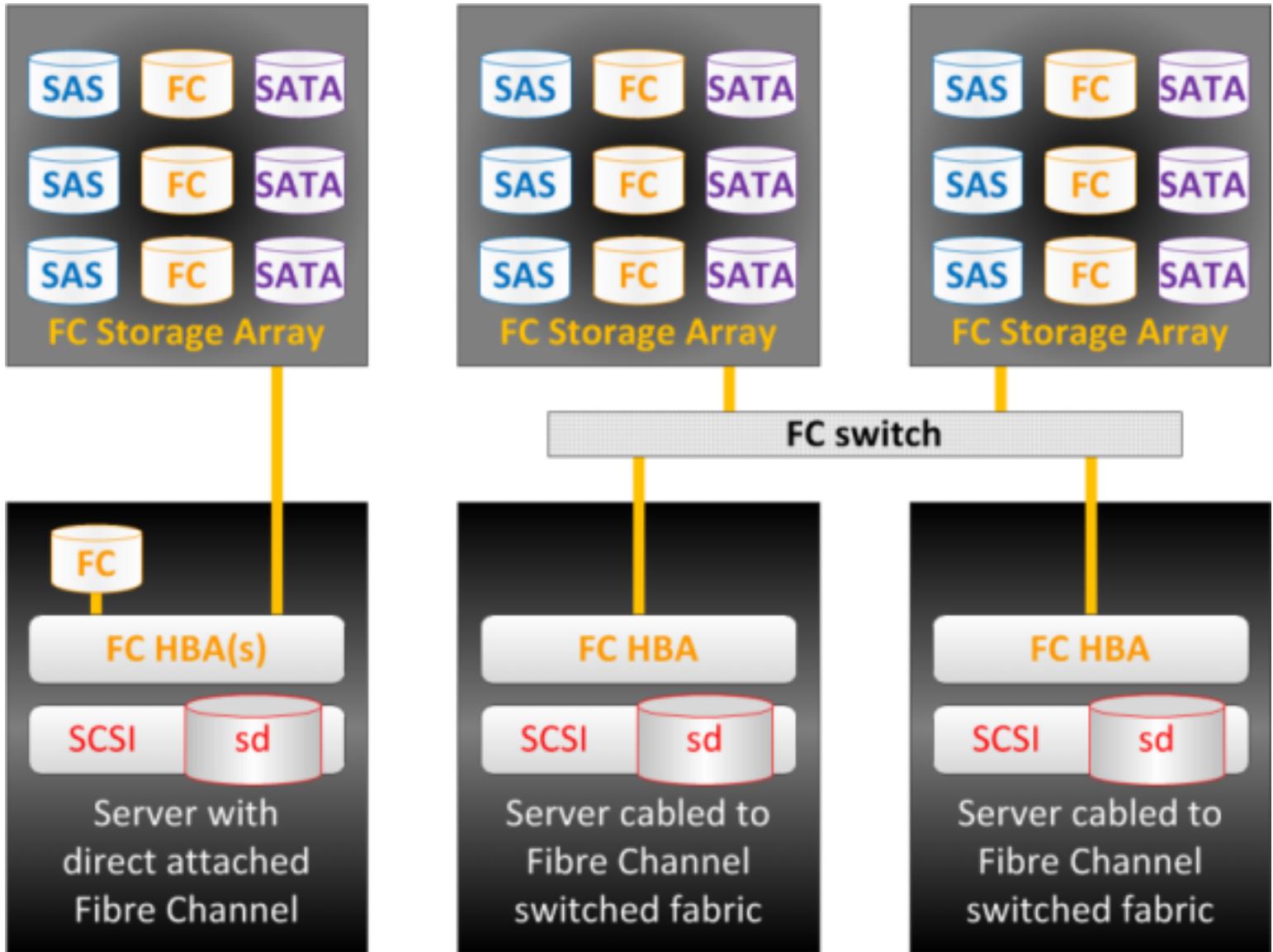
SAN Block-based Storage



IPSAN: SCSI and iSCSI block storage topologies



FC SAN: Fibre Channel block storage topologies



5.1 DAS

硬件产品：

1. 服务器通过SCSI线连接DAS设备
2. 服务器通过SAS线连接DAS设备 6Gb
2. 服务器通过FC线连接DAS设备 4Gb/8Gb

5.2 NAS

NFS

NFS: 网络文件系统，主要针对Unix/Linux客户端提供存储服务

What is NFS?

The Network File System (NFS) is a network file system commonly used by UNIX systems and network-attached storage devices to allow multiple clients to share access to files over the network. It provides access to shared directories or files from client systems.

存储端可以是：

专用的硬件存储 EMC/IBM/HP/[NATAPP](#)

IO性能较高的PC Server

Linux

/dev/sdb1	EXT4	1T	/project1
/dev/sdb2	XFS	20T	/project2

```
[root@tianyun ~]# rpm -q nfs-utils
nfs-utils-1.2.3-36.el6.x86_64
[root@tianyun ~]# vim /etc/exports
/project1    172.16.100.0/24(rw,sync,no_root_squash)
/project2    172.16.200.0/24(rw,sync,no_root_squash)
```

```
[root@tianyun ~]# exportfs -r
[root@tianyun ~]# exportfs -v
```

Client可以是：

Unix/Linux

CIFS

CIFS：通用的Internet文件系统，主要针对Windows客户端提供存储服务

存储端可以是：

专用的硬件存储 EMC/IBM/HP/[NATAPP](#)

IO性能较高的PC Server

Linux

```
[root@tianyun ~]# rpm -q samba
samba-3.6.9-151.el6.x86_64
```

Client可以是：

Windows

5.3 SAN

IP SAN

ISCSI

ISCSI 实现IP SAN

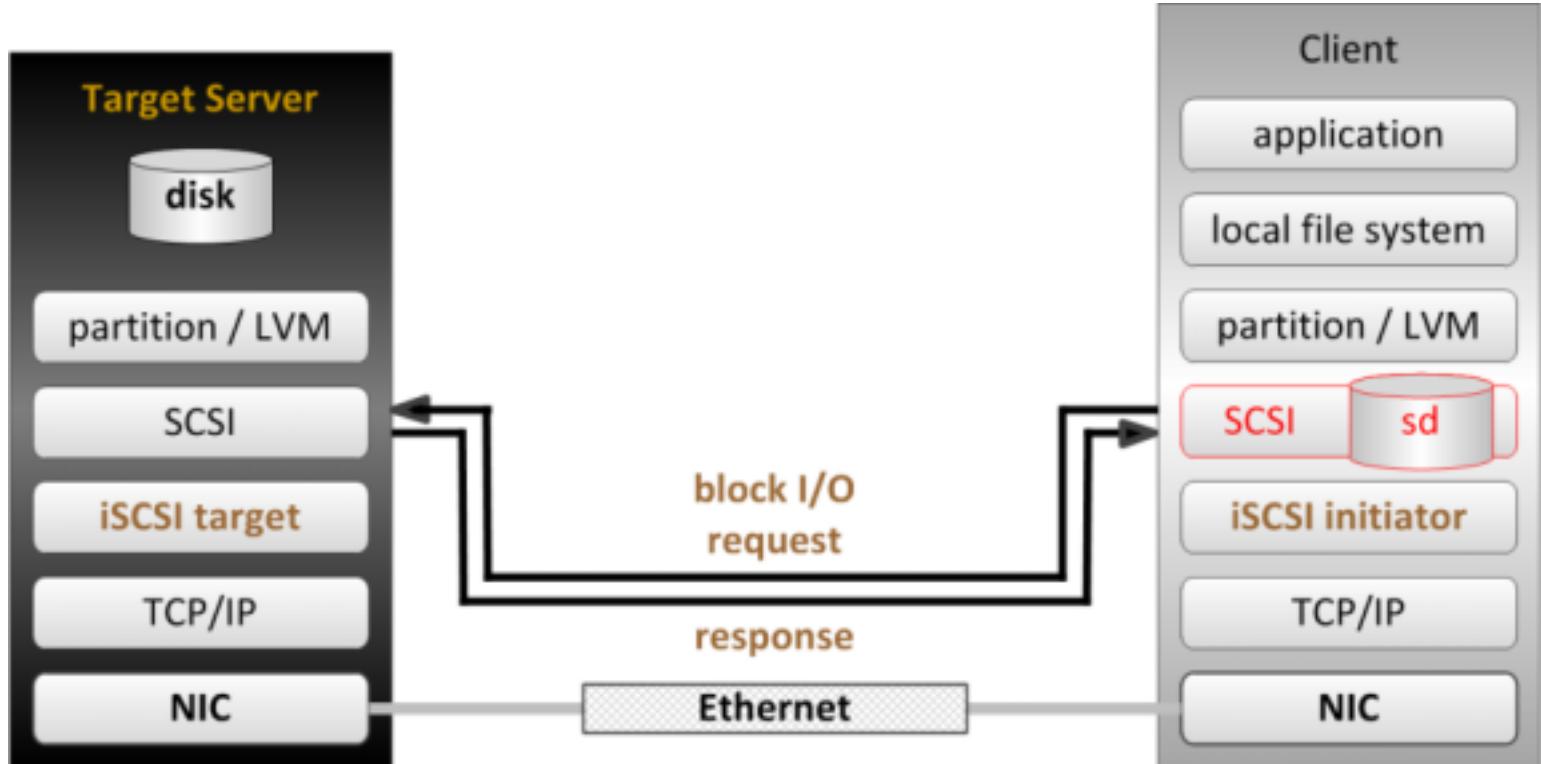
RHEL5/6 CentOS5/6

场景一：

disk1-disk2	RAID1	/dev/sda	安装系统
disk3-disk9	RAID6	/dev/sdb	提供存储
disk10			热备盘

为项目1的三台主机共享存储3G (说明：三台主机为LB的节点，需要并发写入，GFS2/OCFS...)

为项目2的两台主机共享存储1G (说明：两台主机为HA的节点，不需要并发写入，Ext3/Ext4/XFS/GFS2/OCFS...)



一、Linux系统实现ISCSI存储端(target)

1、准备一个分区或者一块硬盘(也可以是一个文件)

```
[root@target ~]# fdisk -cu /dev/vdb
[root@target ~]# partx -a /dev/vdb
[root@target ~]# ll /dev/vdb*
brw-rw----. 1 root disk 252, 16 Aug 12 06:32 /dev/vdb
brw-rw----. 1 root disk 252, 17 Aug 12 06:32 /dev/vdb1
brw-rw----. 1 root disk 252, 18 Aug 12 06:32 /dev/vdb2
```

2、安装包

```
[root@target ~]# yum -y install scsi-target-utils
```

3、定义target共享IQN

```
[root@target ~]# vim /etc/tgt/targets.conf
<target iqn.2015-08.com.tianyun:lb>
  backing-store /dev/vdb1
  initiator-address 192.168.122.134
  initiator-address 192.168.122.147
  initiator-address 192.168.122.30
</target>
```

注：

direct-store 可以共享物理硬盘
backing-store 可以共享物理硬盘、分区、卷、文件

4、启动服务

```
[root@target ~]# service tgtd start
[root@target ~]# chkconfig tgtd on
[root@target ~]# tgt-admin --show
LUN: 1
  Type: disk
  SCSI ID: IET  00010001
  SCSI SN: beaf11
  Size: 3221 MB, Block size: 512
  Online: Yes
  Removable media: No
  Prevent removal: No
  Readonly: No
  Backing store type: rdwr
  Backing store path: /dev/vdb1
  Backing store flags:
Account information:
ACL information:
  192.168.122.134
  192.168.122.147
  192.168.122.30

[root@target ~]# netstat -tnlp |grep :3260
tcp      0      0 0.0.0.0:3260          0.0.0.0:*
              LISTEN    15135/tgtd
tcp      0      0 :::3260             :::*
              LISTEN    15135/tgtd
```

5. 添加新的共享 (当前有initiator端正在使用)

```
<target iqn.2015-08.com.tianyun:ha>
  backing-store /dev/vdb2
  initiator-address 192.168.122.140
  initiator-address 192.168.122.141
</target>
```

```
[root@target ~]# tgt-admin --force --update ALL
```

二、initiator端 (使用存储的前端服务器，此处为node1、node2、node3)

1、安装包

```
[root@node1 ~]# yum -y install iscsi-initiator-utils
```

2、发现存储

```
[root@node1 ~]# iscsidadm -m discovery -t st -p 192.168.122.186:3260
Starting iscsid: [ OK ]
192.168.122.186:3260,1 iqn.2015-08.com.tianyun:lb
```

```
[root@node1 ~]# tree /var/lib/iscsi/
/var/lib/iscsi/
├── ifaces
├── isns
└── nodes
    └── iqn.2015-08.com.tianyun:lb
        └── 192.168.122.186,3260,1
            └── default
└── send_targets
    └── 192.168.122.186,3260
        └── iqn.2015-08.com.tianyun:lb,192.168.122.186,3260,1,default -> /var/lib/iscsi/nodes/
            └── iqn.2015-08.com.tianyun:lb/192.168.122.186,3260,1
                └── st_config
└── slp
└── static
```

10 directories, 2 files

3、login登入存储

方法一：登入指定的存储

```
[root@node1 ~]# iscsiadadm -m node -T iqn.2015-08.com.tianyun:lb -l
```

方法二：登入所有的 (/var/lib/iscsi/)

```
[root@node2 ~]# service iscsi restart  
[root@node2 ~]# ll /dev/sda  
brw-rw----. 1 root disk 8, 0 Aug 12 06:43 /dev/sda
```

```
[root@node2 ~]# fdisk -cul
```

```
Disk /dev/vda: 8589 MB, 8589934592 bytes  
16 heads, 63 sectors/track, 16644 cylinders, total 16777216 sectors  
Units = sectors of 1 * 512 = 512 bytes  
Sector size (logical/physical): 512 bytes / 512 bytes  
I/O size (minimum/optimal): 512 bytes / 512 bytes  
Disk identifier: 0x000bddd3
```

```
Disk /dev/sda: 3221 MB, 3221225472 bytes  
100 heads, 62 sectors/track, 1014 cylinders, total 6291456 sectors  
Units = sectors of 1 * 512 = 512 bytes  
Sector size (logical/physical): 512 bytes / 512 bytes  
I/O size (minimum/optimal): 512 bytes / 512 bytes  
Disk identifier: 0x00000000
```

注：initiator

进程iscsid： initiator iscsi的核心进程，开机启动

进程iscsi： initiator 开机时自动登入所有的 (/var/lib/iscsi/)

4、清除所有的iscsi登入 [如果需要]

```
[root@node1 ~]# tree /var/lib/iscsi/  
/var/lib/iscsi/  
├── ifaces  
├── isns  
└── nodes  
    └── iqn.2015-08.com.tianyun:lb  
        └── 192.168.122.186,3260,1  
            └── default  
└── send_targets  
    └── 192.168.122.186,3260  
        └── iqn.2015-08.com.tianyun:lb,192.168.122.186,3260,1,default -> /var/lib/iscsi/nodes/  
iqn.2015-08.com.tianyun:lb/192.168.122.186,3260,1  
    └── st_config  
└── slp  
└── static
```

```
[root@node2 ~]# iscsiadadm -m node -T iqn.2015-08.com.tianyun:lb -u
```

```
[root@node2 ~]# rm -rf /var/lib/iscsi/*
```

扩展：Iscsi 基于IQN实现访问控制

1. Target授权指定的IQN连接

2. 客户端使用被授权的IQN连接

```
[root@tianyun ~]# vim /etc/iscsi/initiatorname.iscsi  
InitiatorName=iqn.2015-08.com.tianyun:client1
```

```
[root@tianyun ~]# service iscsid restart
```

Openfiler

Openfiler 实现IP SAN

作用：可以提供IPSAN/NAS等解决方案，直接安装到裸机上。

场景一：

disk1-disk2	RAID1	/dev/sda	安装Openfiler
disk3-disk9	RAID6	/dev/sdb	提供存储
disk10			热备盘

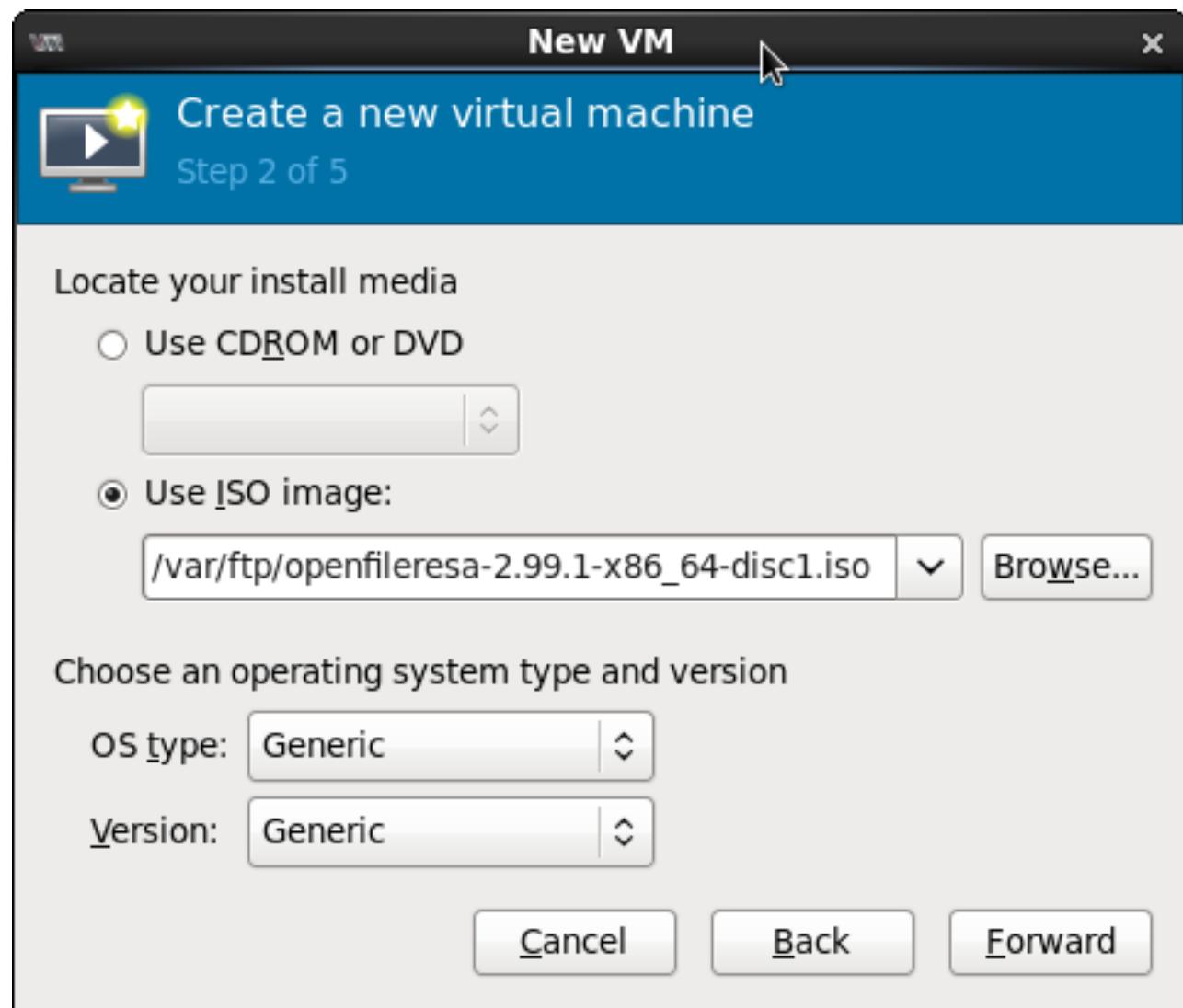
CPU核数

内存容量

硬盘速度

网络带宽

基本调优





i



openfiler Virtual Machine

File Virtual Machine View Send Key



openfiler

Installation requires partitioning of your hard drive. By default, a partitioning layout is chosen which is reasonable for most users. You can either choose to use this or create your own.

Remove all partitions on selected drives and create default layout

Select the drive(s) to use for this installation.

sda 51199 MB ATA QEMU HARDDISK

sdb 20473 MB ATA QEMU HARDDISK

Advanced storage configuration

What drive would you like to boot this installation from?

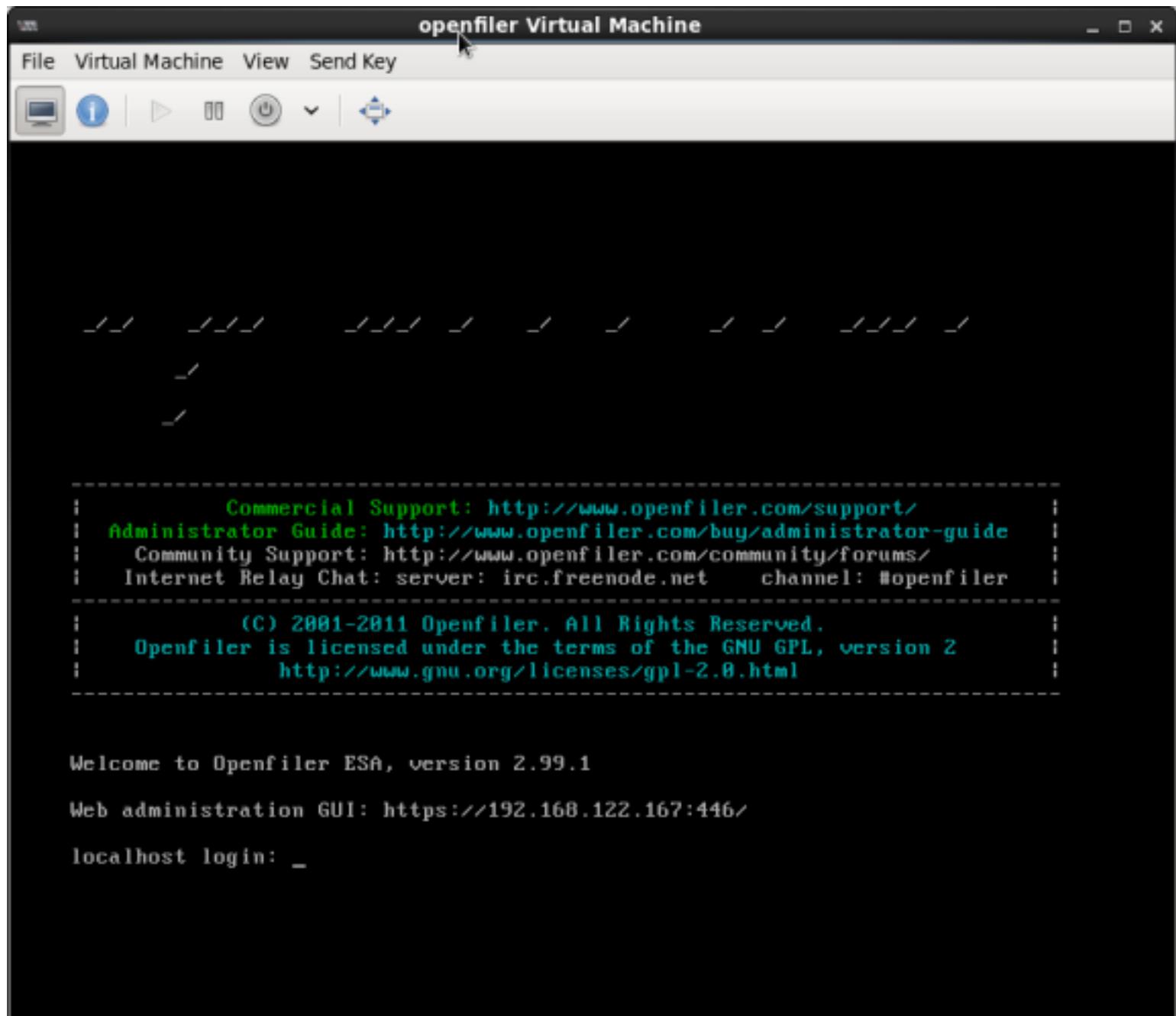
sda 51199 MB ATA QEMU HARDDISK

Review and modify partitioning layout

Release Notes

Back

Next



用户名 : openfiler
密码 : password

File Edit View History Bookmarks Tools Help

Openfiler Storage Control Center



192.168.122.167

https://192.168.122.167:446



Google



openfiler open source storage management

Username:

Password:

Distro Release: Openfiler ESA 2.99.1
GUI Version: r_15ec1bb936e555e65adce24a9b0595124c46b0ee-1-1



© 2001 - 2011 Openfiler. All rights reserved.
[Home](#) · [Documentation](#) · [Support](#) · [Website](#) · [License](#)

一、Openfiler实现 iSCSI 存储端 (target)



Network Interface Configuration

Interface	Boot Protocol	IP Address	Network Mask	Speed	MTU	Link	Edit
eth0	DHCP	192.168.122.167	255.255.255.0	100Mb/s	1500	Yes	Configure

[Create bonded interface](#)

- [Get support](#)
- [Forums](#)
- [Admin Guide](#)

Network Access Configuration

Delete	Name	Network/Host	Netmask	Type
<input type="checkbox"/>	node1	192.168.122.95	255.255.255.255	Share
<input type="checkbox"/>	node2	192.168.122.186	255.255.255.255	Share

New

© 2001 - 2011 [Openfiler](#). All rights reserved.
[Home](#) · [Documentation](#) · [Support](#) · [Website](#) · [License](#) · [Log Out](#)

Volume Group Management

Volume Group Name	Size	Allocated	Free	Members	Add physical storage	Delete VG
-------------------	------	-----------	------	---------	----------------------	-----------

Create a new volume group



Valid characters for volume group name: A-Z a-z 0-9 _ + -

Volume group name (no spaces)

 datavg

Select physical volumes to add

/dev/sdbl 19.07 GB

Add volume group

Manage Volumes

Volume Groups

Block Devices

Add Volume

iSCSI Targets

Software RAID

Support resources

Report bug

Get support

Forums

Admin Guide

© 2001 - 2011 [Openfiler](#). All rights reserved.
[Home](#) · [Documentation](#) · [Support](#) · [Website](#) · [License](#) · [Log Out](#)

Free
(100%)

Create a volume in "datavg"

Volume Name ("no spaces". Valid characters [a-z,A-Z,0-9]):	<input type="text" value="project1"/>
Volume Description:	<input type="text" value="project1"/>
Required Space (MB):	<input type="text" value="3000"/>
Filesystem / Volume type:	<input type="button" value="block (iSCSI/FC/etc)"/>

Manage Services

Service	Boot Status	Modify Boot	Current Status	Start / Stop
CIFS Server	Disabled	Enable	Stopped	Start
NFS Server	Disabled	Enable	Stopped	Start
RSync Server	Disabled	Enable	Stopped	Start
HTTP/Dav Server	Disabled	Enable	Running	Stop
LDAP Container	Disabled	Enable	Stopped	Start
FTP Server	Disabled	Enable	Stopped	Start
iSCSI Target	Enabled	Disable	Running	Stop
UPS Manager	Disabled	Enable	Stopped	Start
UPS Monitor	Disabled	Enable	Stopped	Start
iSCSI Initiator	Disabled	Enable	Stopped	Start
ACPI Daemon	Enabled	Disable	Running	Stop
SCST Target	Disabled	Enable	Stopped	Start
FC Target	Disabled	Enable	Stopped	Start
Cluster Manager	Disabled	Enable	Stopped	Start

[Manage Services](#)[SMB/CIFS Setup](#)[LDAP Setup](#)[UPS Setup](#)[Rsync Setup](#)[iSCSI Target Setup](#)[FTP Setup](#)

Support resources

[Report bug](#)[Get support](#)[Forums](#)[Admin Guide](#)

Volumes : iSCSI Targets - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Volumes : iSCSI Targets

← 192.168.122.167 https://192.168.122.167:446/admin/volumes_iscsi_targets.html ⌂ Google ⌂ Home

openfiler 08:10:40 up 29 min, 1 user, load average: 0.17, 0.14, 0.06

Log Out | Status | Update | Shutdown

Status System Volumes Cluster Quota Shares Services Accounts

Target Configuration LUN Mapping Network ACL CHAP Authentication

Add new iSCSI Target

Target IQN Add

Support resources

Report bug Get support Forums Admin Guide

© 2001 - 2011 [Openfiler](#). All rights reserved.
[Home](#) · [Documentation](#) · [Support](#) · [Website](#) · [License](#) · [Log Out](#)

Volumes : iSCSI Targets - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Volumes : iSCSI Targets

192.168.122.167 https://192.168.122.167:446/admin/volumes_iscsi_targets.html

openfiler 06:11:34 up 30 min, 1 user, load average: 0.07, 0.12, 0.05

Log Out | Status | Update | Shutdown

Status System Volumes Cluster Quota Shares Services Accounts

Target Configuration LUN Mapping Network ACL CHAP Authentication

LUNs mapped to target: iqn.2015-08.com.tianyun:project1

LUN Id.	LUN Path	R/W Mode	SCSI Serial No.	SCSI Id.	Transfer Mode	Unmap LUN
0	/dev/datavg/project1	write-back	fsBujb-2PrH-ofX9	fsBujb-2PrH-ofX9	blockio	<input type="button" value="Unmap"/>

 All mappable iSCSI LUNs for this target have been mapped.

Volumes section

-
-
-
-
-
-

Support resources

-
-
-
-

Volumes : iSCSI Targets - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Volumes : iSCSI Targets

192.168.122.167 https://192.168.122.167:446/admin/volumes_iscsi_targets.html?targetName=iqn.2015-08.com.tianyun:project1 Google

openfiler 08:12:15 up 31 min, 1 user, load average: 0.03, 0.10, 0.05 Log Out | Status | Update | Shutdown

Status System Volumes Cluster Quota Shares Services Accounts

Target Configuration LUN Mapping Network ACL CHAP Authentication

iSCSI host access configuration for target "iqn.2015-08.com.tianyun:project1"

Name	Network/Host	Netmask	Access
node1	192.168.122.95	255.255.255.255	Allow
node2	192.168.122.186	255.255.255.255	Allow

Update

Volumes section

- Manage Volumes
- Volume Groups
- Block Devices
- Add Volume
- iSCSI Targets
- Software RAID

Support resources

- Report bug
- Get support
- Forums
- Admin Guide

© 2001 - 2011 Openfiler. All rights reserved.

二、initiator端（使用存储的前端服务器，此处为node1、node2、node3）

1、安装包

```
[root@node1 ~]# yum -y install iscsi-initiator-utils
```

2、发现存储

```
[root@node1 ~]# iscsidadm -m discovery -t st -p 192.168.122.186:3260
Starting iscsid: [ OK ]
192.168.122.186:3260,1 iqn.2015-08.com.tianyun:lb
```

```
[root@node1 ~]# tree /var/lib/iscsi/
/var/lib/iscsi/
├── ifaces
├── isns
└── nodes
    └── iqn.2015-08.com.tianyun:lb
        └── 192.168.122.186,3260,1
            └── default
└── send_targets
    └── 192.168.122.186,3260
        └── iqn.2015-08.com.tianyun:lb,192.168.122.186,3260,1,default -> /var/lib/iscsi/nodes/
iqn.2015-08.com.tianyun:lb/192.168.122.186,3260,1
    └── st_config
└── static
```

10 directories, 2 files

3、login登入存储

方法一：登入指定的存储

```
[root@node1 ~]# iscsiadadm -m node -T iqn.2015-08.com.tianyun:lb -l
```

方法二：登入所有的 (/var/lib/iscsi/)

```
[root@node2 ~]# service iscsi restart  
[root@node2 ~]# ll /dev/sda  
brw-rw----. 1 root disk 8, 0 Aug 12 06:43 /dev/sda
```

```
[root@node2 ~]# fdisk -cul
```

```
Disk /dev/vda: 8589 MB, 8589934592 bytes  
16 heads, 63 sectors/track, 16644 cylinders, total 16777216 sectors  
Units = sectors of 1 * 512 = 512 bytes  
Sector size (logical/physical): 512 bytes / 512 bytes  
I/O size (minimum/optimal): 512 bytes / 512 bytes  
Disk identifier: 0x000bddf3
```

```
Disk /dev/sda: 3221 MB, 3221225472 bytes  
100 heads, 62 sectors/track, 1014 cylinders, total 6291456 sectors  
Units = sectors of 1 * 512 = 512 bytes  
Sector size (logical/physical): 512 bytes / 512 bytes  
I/O size (minimum/optimal): 512 bytes / 512 bytes  
Disk identifier: 0x00000000
```

注：initiator

进程iscsid： initiator iscsi的核心进程，开机启动

进程iscsi： initiator 开机时自动登入所有的 (/var/lib/iscsi/)

4、清除所有的iscsi登入 [如果需要]

```
[root@node1 ~]# tree /var/lib/iscsi/  
/var/lib/iscsi/  
├── ifaces  
├── isns  
└── nodes  
    └── iqn.2015-08.com.tianyun:lb  
        └── 192.168.122.186,3260,1  
            └── default  
└── send_targets  
    └── 192.168.122.186,3260  
        └── iqn.2015-08.com.tianyun:lb,192.168.122.186,3260,1,default -> /var/lib/iscsi/nodes/  
iqn.2015-08.com.tianyun:lb/192.168.122.186,3260,1  
    └── st_config  
└── slp  
└── static
```

```
[root@node2 ~]# iscsiadadm -m node -T iqn.2015-08.com.tianyun:lb -u  
[root@node2 ~]# rm -rf /var/lib/iscsi/*
```

Windows

Windows Server 2012 target

Windows Server 2008 initiator
Windows Server 2012 initiator

硬件设备

企业级：**EMC/IBM/NAT APP/HP/DELL**

FC SAN

IBM DS3400

FC SAN配置

IBM刀片服务器 + IBM DS3200

FC SAN 架构/配置/培训/技术支持 均是由服务商完成

环境：

IBM H 刀箱： 14个刀片服务器 HBA卡(4Gbps)、以太网卡

模块名	管理地址	用户名	密码	模块描述
-----	------	-----	----	------

述

AMM:	192.168.70.125	USERID		
PASSW0RD				
Ethernet				
SM:				
FC SM:	192.168.70.130	USERID	password	Brocade FC SWITCH 4Gbps
IBM DS3200: 192.168.128.101	12x72G	SAS		
15000RPM				

一、刀箱配置和管理

Blade Tasks

Power/Restart

Remote Control

I/O Module Tasks

查看、配置模块IP及网关

MM Control

AMM模块配置IP及网关

二、FC SWITCH

1. 查看Linux HBA卡WWN号

cat /sys/class/fc_host/host4/port_name

2. 查看交换机相应端口WWN号

3. 新建zone --- cfg --- save --- enable cfg

三、配置存储

1. 安装storage manager client software
3. 管理存储：
 - a) 制作raid，并创建logic driver，创建热备盘
 - b) 创建主机，主机组，并将主机添加到主机组
 - c) Mapping logic drive到主机组

四、测试

已安装FC HBA卡驱动

查看FC HBA wwn号

客户端查看fdisk -cul

Multipath

SAN: Multipath 多路径

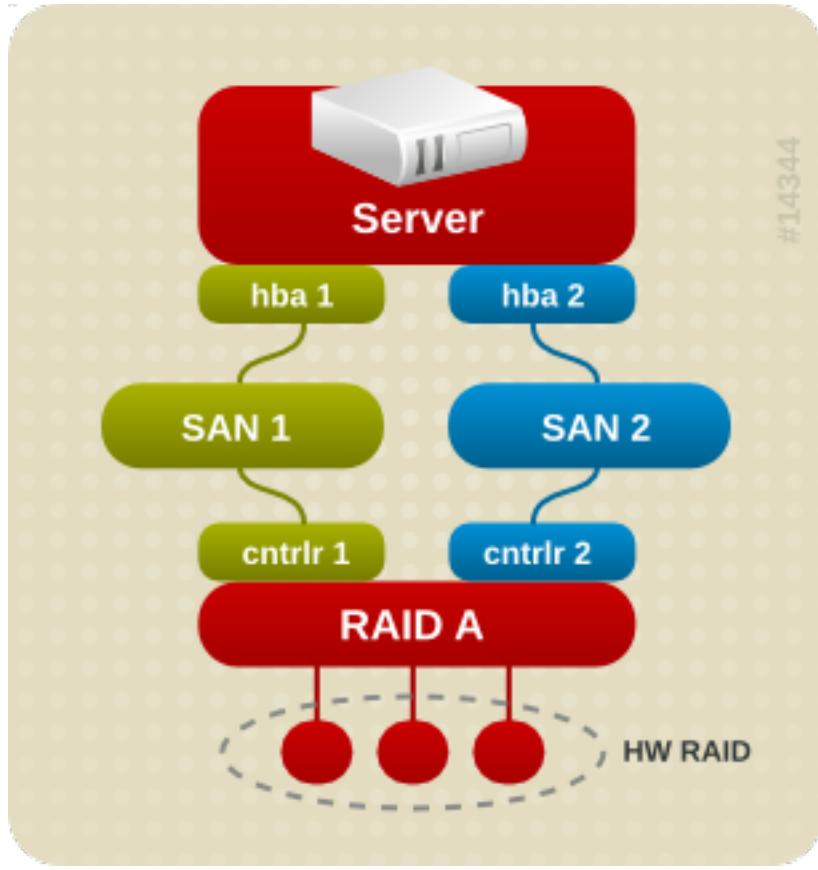
SAN存储中：

1. 解决的是多个节点发现的存储设备名不一致的问题 <在集群中保持多路径设备名称一致>
2. 解决Multipath的问题 <主要针对FC SAN>

IP SAN: 实验环境

client eth0 1.1.1.1 -----switchA-----> storage eth0 1.1.1.100
client eth1 2.2.2.1 -----switchB-----> storage eth1 2.2.2.100

FC SAN: 生产环境



#14344

RHEL5 :

1. 安装软件包
device-mapper-multipath

2. 获得wwid (全球唯一标识)
/sbin/scsi_id -g -u -s /block/sda

3. 配置multipath

```
# vim /etc/multipath.conf
# blacklist {
#     devnode "*"
#     wwid 333333          方法一
#     devnode "^sd[a-z]"    方法二
# }
```

```
defaults {
    user_friendly_names no      //mpath0
}

multipaths {
#   multipath {
#       wwid           3600508b4000156d700012000000b0000
#       alias          yellow
#       path_grouping_policy multibus
#       path_checker    readsector0
#       path_selector   "round-robin 0"
#       fallback        manual
#       rr_weight       priorities
#       no_path_retry   5
#   }
multipath {
```

```

        wwid      "1DEC____321816758474"
        alias    san1
    }
}

# /etc/init.d/multipathd start
# ls /dev/mapper/san1

# multipath -ll

# partprobe; multipath -r

```

RHEL6 :
/dev/sdb
/dev/sdc
/dev/sdd
/dev/sdf

1. 安装软件包
device-mapper-multipath

2. 获得wwid (全局唯一标识)
/lib/udev/scsi_id --whitelisted --device=/dev/sdb
1IET 00010001

3. 配置multipath
[root@node02 ~]# cp /usr/share/doc/device-mapper-multipath-0.4.9/multipath.conf /etc/
[root@node02 ~]# vim /etc/multipath.conf
defaults {
 user_friendly_names no //不使用友好的名字
}

```

multipaths {
    multipath {
        wwid      "1IET  00010001"
        alias    san1
        path_grouping_policy multibus
        path_checker   readsector0
        path_selector  "round-robin 0"
        fallback      manual
        rr_weight     priorities
        no_path_retry 5
    }
}

```

4. 启动并测试
[root@node02 ~]# /etc/init.d/multipathd start
正在启动守护进程multipathd : [确定]
[root@node02 ~]# chkconfig multipathd on

```

[r[root@node02 ~]# multipath -ll
Aug 13 03:05:34 | multipath.conf line 67, invalid keyword: path_checker
san1 (1IET 00010001) dm-2 IET,VIRTUAL-DISK
size=2.0G features='1 queue_if_no_path' hwhandler='0' wp=rw
`-- policy='round-robin 0' prio=1 status=active
|- 2:0:0:1 sdb 8:0  active ready running
`- 3:0:0:1 sdc 8:16  active ready running

```

```

[root@node02 ~]# ll /dev/mapper/san1
lrwxrwxrwx. 1 root root 7 Aug 13 03:05 /dev/mapper/san1 -> ../dm-2

```

```

pvcreate /dev/mapper/san1
vgcreate datavg /dev/mapper/san1
lvcreate -l 100%VG -n lv1 datavg
=====

```

Udev

Udev

1. 为设备设置自定义的设备名

连接到本机主板总线上的设备（存储设备/网络设备...）
通过网络连接的SAN设备

2. 可以解决多个节点发现的存储设备名不一致的问题

- 3. 没有multipath功能**
 - 4. 可以用于IP SAN/FC SAN**
-

Udev简介：

在Linux系统中，所有的设备在 Linux 里都是以设备文件的形式存在。在早期的 Linux 版本中，/dev目录包含了所有可能出现的设备的设备文件。

很难想象 Linux 用户如何在这些大量的设备文件中找到匹配条件的设备文件。现在 udev 只为那些连接到 Linux 操作系统的设备产生设备文件。并且

udev 能通过定义一个 udev 规则 (rule) 来产生匹配设备属性的设备文件，这些设备属性可以是内核设备名称、总线路径、厂商名称、型号、序列号或者磁盘大小等等。

动态管理：当设备添加 / 删除时，udev 的守护进程侦听来自内核的 uevent，以此添加或者删除 /dev下的设备文件，所以 udev 只为已经连接的设备产生设备文件，而不会在 /dev下产生大量虚无的设备文件。

自定义命名规则：通过 Linux 默认的规则文件，udev 在 /dev/ 里为所有的设备定义了内核设备名称，比如 /dev/sda、/dev/hda、/dev/fd 等等。

由于 udev 是在用户空间 (user space) 运行，Linux 用户可以通过自定义的规则文件，灵活地产生标识性强的设备文件名，比如 /dev/boot_disk、/dev/root_disk、/dev/san1 等等。

设定设备的权限和所有者/组：udev 可以按一定的条件来设置设备文件的权限和设备文件所有者/组。

Udev LAB1：使用Udev机制为SAN存储的initiator端生成自定义的设备文件（保障集群节点设备文件的一致性）

IP SAN 存储端 (target)

1、安装包

```
[root@target ~]# yum -y install scsi-target-utils
```

2、准备一个分区或整块硬盘

```
[root@target ~]# fdisk -u /dev/sda
[root@target ~]# partx -a /dev/sda
```

3、定义target共享

```
[root@target ~]# vim /etc/tgt/targets.conf
default-driver iscsi
<target iqn.2014-06.com.tianyun:san1>
    backing-store /dev/sda5
    vendor_id tianyun
    product_id uuu
    initiator-address 192.168.8.101
    initiator-address 192.168.8.102
</target>

<target iqn.2014-06.com.tianyun:san2>
    backing-store /dev/sda6
    vendor_id tianyun
    product_id yyy
    initiator-address 192.168.8.101
    initiator-address 192.168.8.102
</target>
```

4、启动服务

```
[root@target ~]# service tgtd start
[root@target ~]# tgt-admin --show
[root@target ~]# netstat -tnlp |grep :3260
```

initiator端 (使用存储的前端server , 此处为node1、node2)

1、安装包

```
[root@node1 ~]# yum -y install iscsi-*
```

2、发现存储

```
[root@node1 ~]# iscsidadm -m discovery -t st -p 192.168.8.34
```

3、登入存储

```
[root@node2 ~]# service iscsi restart
[root@node2 ~]# fdisk -cul
```

使用Udev为SAN存储设备自定义设备名或符号链接

自定义设备名
自定义符号链接
自定义设备文件属主/属组

RHEL6:

```
[root@web3 ~]# udevadm info -a -p /block/sdb
```

获得该设备的属性

```
[root@web3 ~]# cd /etc/udev/rules.d/
```

参考文件:

```
[root@web3 rules.d]# grep cdrom *
70-persistent-cd.rules:SUBSYSTEM=="block", ENV{ID_CDROM}=="?", ENV{ID_PATH}=="pci-0000:00:01.1-scsi-1:0:0:0",
SYMLINK+="cdrom", ENV{GENERATED}="1"
```

```
[root@web3 rules.d]# vim 100-san.rules
```

```
SUBSYSTEM=="block", ATTR{size}=="4194304", ATTRS{vendor}=="ibm", SYMLINK+="san1"
SUBSYSTEM=="block", ATTR{size}=="4194304", ATTRS{vendor}=="hp", SYMLINK+="san2"
```

```
[root@web3 rules.d]# start_udev
```

正在启动 udev : [确定]

```
[root@web3 rules.d]# ll /dev/san*
```

```
lrwxrwxrwx 1 root root 3 9月 25 12:17 /dev/san1 -> sdb
lrwxrwxrwx 1 root root 3 9月 25 12:17 /dev/san2 -> sda
```

```
SUBSYSTEM=="block", ATTR{size}=="10485760", ATTRS{vendor}=="IET", SYMLINK+="san1"
SUBSYSTEM=="block", ATTR{size}=="4194304", ATTRS{vendor}=="IET", MODE="0777", OWNER="alice", NAME="yang/san3"
```

单节点使用

lvm

单机文件系统

支持本地锁的文件系统：
EXT2/3/4 XFS

多节点共享

RHCS

RHCS集群 GFS2

拓朴结构：

```
[node1]           [node2]  
192.168.122.10   192.168.122.20  
  
[san_storage]  
192.168.122.50
```

IPSAN/FC SAN实施（略）

RHCS集群实施（略）

```
create cluster
```

```
[root@node1 ~]# clustat  
Cluster Status for storage @ Tue Jan 13 12:17:04 2015  
Member Status: Quorate
```

Member Name	ID	Status
node1	1	Online, Local
node2	2	Online

```
[root@node2 ~]# clustat  
Cluster Status for storage @ Mon Jan 12 20:15:23 2015  
Member Status: Quorate
```

Member Name	ID	Status
node1	1	Online
node2	2	Online, Local

```
[root@node1 ~]# service clvmd status  
clvmd (pid 2170) 正在运行...  
Clustered Volume Groups: (none)  
Active clustered Logical Volumes: (none)
```

```
[root@node2 ~]# service clvmd status  
clvmd (pid 2094) 正在运行...  
Clustered Volume Groups: (none)  
Active clustered Logical Volumes: (none)
```

CLVMD

Clvmd 集群级逻辑卷：

用于多个节点使用同一存储并使用逻辑卷

Clvmd必须有RHCS (CMAN) 环境支持

例如：LB集群中提供服务节点使用同一存储

在集群中任意节点：

```
[root@node1 ~]# pvcreate /dev/mapper/san2  
Physical volume "/dev/mapper/san2" successfully created
```

```
[root@node1 ~]# vgcreate datavg /dev/mapper/san2  
Clustered volume group "datavg" successfully created
```

```
[root@node1 ~]# lvcreate -L 1G -n cluster-lv1 datavg  
Logical volume "cluster-lv1" created
```

```
[root@node1 ~]# vgs  
VG #PV #LV #SN Attr VSize VFree  
datavg 1 1 0 wz--nc 8.00g 7.00g
```

```
[root@node1 ~]# lvs  
LV VG Attr LSize Pool Origin Data% Move Log Cpy%Sync Convert  
cluster-lv1 datavg -wi-a---- 1.00g
```

查看其它节点：

```
[root@node2 ~]# vgs
```

```
VG #PV #LV #SN Attr VSize VFree
datavg 1 1 0 wz--nc 8.00g 7.00g

[root@node2 ~]# lvs
  LV   VG Attr  LSize Pool Origin Data% Move Log Cpy%Sync Convert
cluster-lv1 datavg -wi-a---- 1.00g
```

GFS

集群级文件系统

GFS，支持DLM（分布式锁管理）

GFS，必须有RHCS（CMAN）环境支持

一、创建GFS2文件系统

1、在node1,node2上安装gfs2的相关软件包

```
[root@node1 ~]# rpm -qa |grep gfs
gfs2-utils-3.0.12.1-49.el6.x86_64 //提供mkfs.gfs2命令用于创建gfs2文件系统
```

```
[root@node1 ~]# modinfo -F filename gfs2
/lib/modules/2.6.32-358.el6_64/kernel/fs/gfs2/gfs2.ko
```

2、在任意一个节点上格式化

用法
mkfs.gfs2 -p LockProtocolName -t LockTableName -j NumberJournals BlockDevice

```
[root@node1 ~]# cman_tool status
Version: 6.2.0
Config Version: 1
Cluster Name: http-cluster
```

```
[root@node1 ~]# mkfs.gfs2 -p lock_dlm -t http-cluster:table1 -j 3 -J 128M /dev/datavg/mysql-lv
This will destroy any data on /dev/datavg/mysql-lv.
It appears to contain: symbolic link to `../dm-2'
```

Are you sure you want to proceed? [y/n] y

```
Device:          /dev/datavg/mysql-lv
Blocksize:       4096
Device Size:    2.00 GB (524288 blocks)
Filesystem Size: 2.00 GB (524288 blocks)
Journals:        3
Resource Groups: 8
Locking Protocol: "lock_dlm"
Lock Table:      "http-cluster:table1"
UUID:            98326a5a-f72c-5a1d-6ccd-bec094cab26
```

-t clustername:locktablename
-p lock_dlm
-j 指定日志空间个数，此个数一定要大于等于使用gfs2文件系统存储的主机数
-J 指定一个日志空间的大小

3、挂载并测试写入

```
[root@node1 ~]# vim /etc/fstab
/dev/datavg/mysql-lv  /var/www/html      gfs2  defaults,_netdev 0 0
[root@node1 ~]# mount -a
```

```
[root@node1 ~]# df -P  
/dev/mapper/datavg-mysql--lv gfs2 2.0G 410M 1.7G 20% /var/lib/mysql
```

二、GFS管理

1. 支持ACL

```
/dev/datavg/cluster-lv1 /mnt/gfs2 gfs2 acl,_netdev 0 0
```

2. 支持磁盘配额

```
/dev/datavg/cluster-lv1 /mnt/gfs2 gfs2 acl,quota=on,_netdev 0 0
```

3. 增大的文件系统

```
[root@node1 ~]# df /mnt/gfs2 -h -P  
Filesystem Size Used Avail Use% Mounted on  
/dev/mapper/datavg-cluster--lv1 1.0G 131M 894M 13% /mnt/gfs2
```

a. 扩展lvm

```
[root@node1 ~]# lvextend -L +2G /dev/datavg/cluster-lv1
```

b. 扩展FS

```
[root@node1 ~]# gfs2_grow /mnt/gfs2  
FS: Mount Point: /mnt/gfs2  
FS: Device: /dev/dm-2  
FS: Size: 262142 (0x3ffe)  
FS: RG size: 65529 (0xffff9)  
DEV: Size: 786432 (0xc0000)  
The file system grew by 2048MB.  
gfs2_grow complete.  
[root@node1 ~]# df /mnt/gfs2 -h -P  
Filesystem Size Used Avail Use% Mounted on  
/dev/mapper/datavg-cluster--lv1 3.0G 131M 2.9G 5% /mnt/gfs2
```

4. 在文件系统中添加日志

```
[root@node1 ~]# gfs2_tool journals /mnt/gfs2  
journal1 - 64MB  
journal0 - 64MB  
2 journal(s) found.
```

a. 新节点iscsi连接 , udev (IP SAN), multipathd (FC SAN)

b. 增加集群节点

c. 增加日志数量

```
[root@node3 ~]# mount -a  
Too many nodes mounting filesystem, no free journals
```

```
[root@node1 ~]# gfs2_jadd -j1 /mnt/gfs2 //在已挂载gfs2的节点上增加  
Filesystem: /mnt/gfs2  
Old Journals 2  
New Journals 3
```

d. 挂载GFS2文件系统

```
[root@node3 ~]# mount -a //新节点挂载
```

SAN使用总结

SAN使用总结 :

IP SAN技术

客户端使用以太网卡(bonding)

安装iscsi-initiator-utils

discovery,login(iscsi)

FC SAN技术

客户端使用FC HBA卡

SAN 存储用于HA集群节点

1. multipath 或 udev

3. ext3/ext4/xfs

SAN 存储用于LB集群Real Server

1. multipath 或 udev

2. Clvmd

3. GFS2

5.4 DFS

DFS简介

DFS：分布式文件系统

=====

分布式文件系统 DFS，可以支持大量的节点以及PB级的数量存储。新兴分布式文件系统以高IO速度、方便管理等特性著称。当前比较流行的分

布式文件系统，包括：[Lustre](#)、[Hadoop HDFS](#)、[MogileFS](#)、[FreeNAS](#)、[FastDFS](#)、[NFS](#)、[OpenAFS](#)、[MooseFS](#)、[pNFS](#)、[GoogleFS](#)、[GlusterFS](#)

以及 [TFS](#)。

TFS (Taobao FileSystem) 是一个高可扩展、高可用、高性能、面向互联网服务的分布式文件系统，主要针对海量的非结构化数据，它构筑在普

通的Linux机器集群上，可为外部提供高可靠和高并发的存储访问。TFS为淘宝提供海量小文件存储，通常文件大小不超过1M，满足了淘宝对小文

件存储的需求，被广泛地应用在 淘宝各项应用中。它采用了HA架构和平滑扩容，保证了整个文件系统的可用性和扩展性。同时扁平化的数

据组织
结构，可将文件名映射到文件的物理地址，简化了文件的访问流程，一定程度上为TFS提供了良好的读写性能。

DAS, NAS, SAN 数据集中存储

优点：

1. 集中，便于读取

2. 集中，便于管理

缺点：

1. 集中，单点故障

2. 集中，I/O压力大

3. 集中，存储容量有限

GFS：

RHEL6 节点不能超过16个

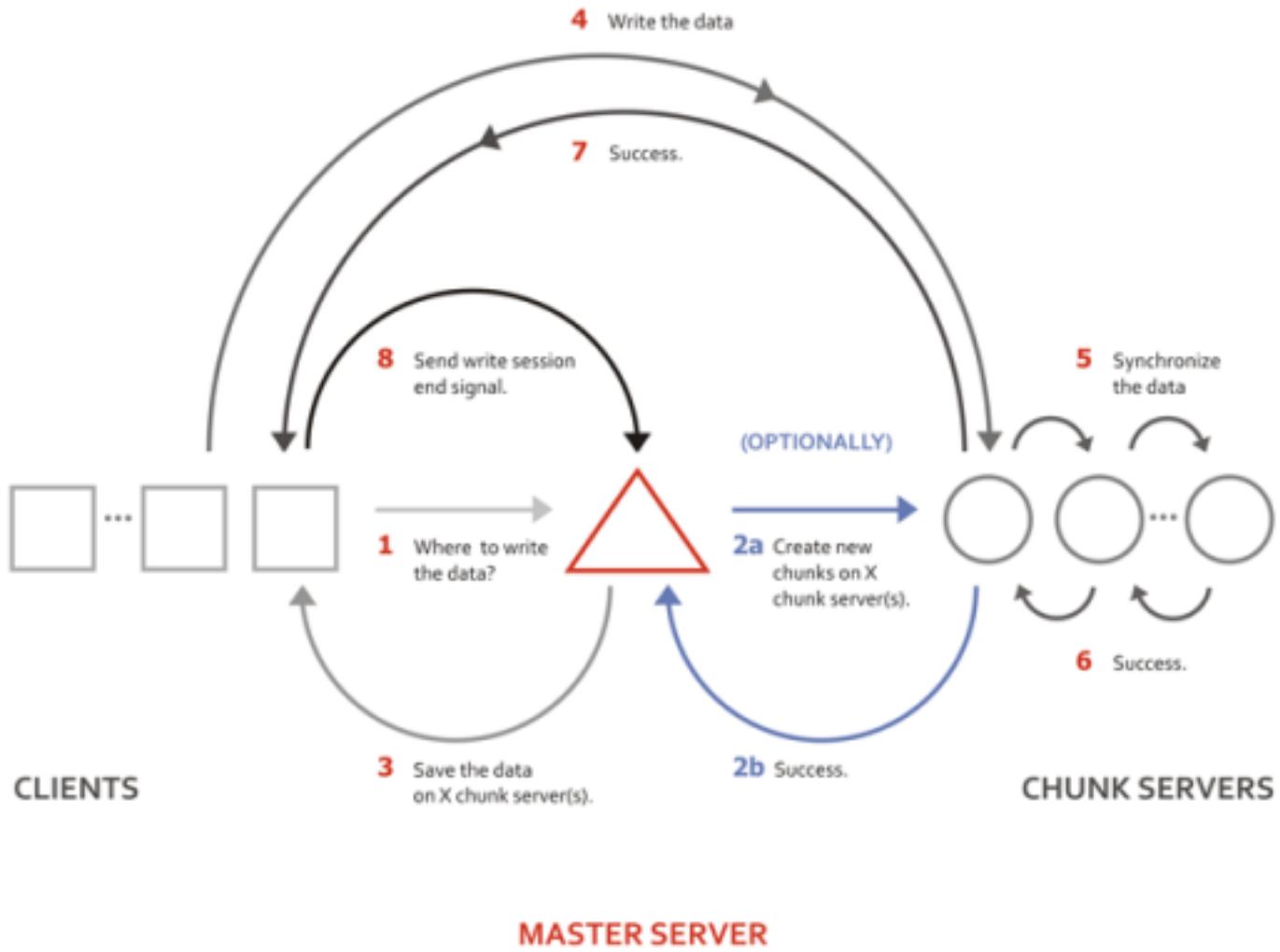
存储容量不超过100 TB

MooseFS

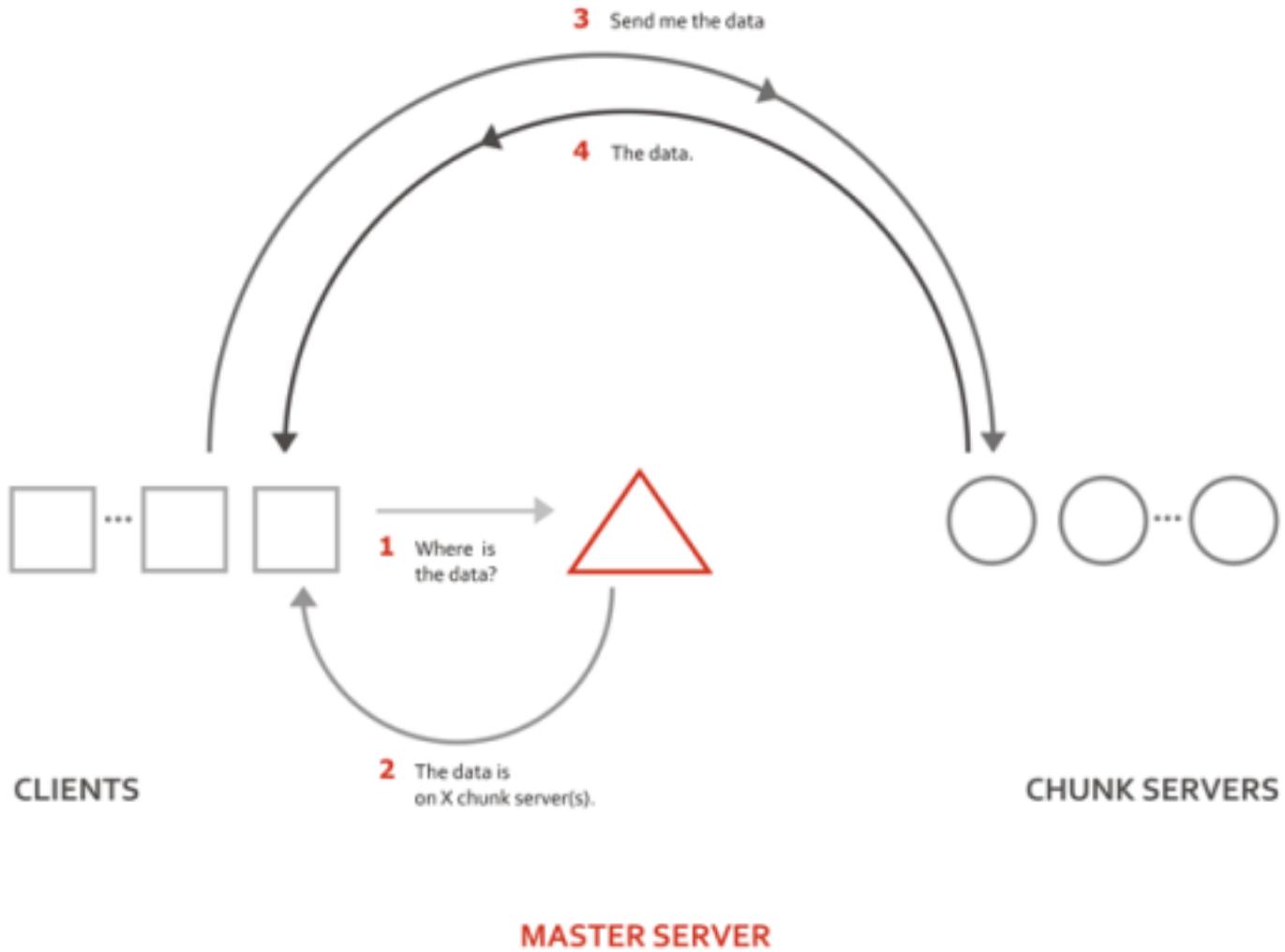
DFS : MooseFS

一、MooseFS文件系统介绍

MooseFS Write process



MooseFS Read process



MooseFS是一种分布式文件系统，MooseFS文件系统结构包括以下四种角色：

管理服务器managing server (master)

负责各个数据存储服务器的管理，文件读写调度，文件空间回收以及

恢复，多节点拷贝

元数据日志服务器Metalogger server (Metalogger)

负责备份master服务器的变化志文件，以便于在master server
出问题的时候接替其进行工作

数据存储服务器data servers (chunkservers)
数据传输

负责连接管理服务器，听从管理服务器调度，提供存储空间，并为客户提供

客户机挂载使用client computers

通过fuse内核接口挂载远程管理服务器上所管理的数据存储服务

二、部署MFS

master server:

```
[root@tianyun ~]# useradd mfs -s /sbin/nologin
```

```
[root@tianyun ~]# tar xvf mfs-1.6.17.tar.gz
[root@tianyun ~]# cd mfs-1.6.17/
[root@tianyun mfs-1.6.17]# yum -y install gcc make fuse-devel zlib-devel
[root@tianyun mfs-1.6.17]# ./configure --prefix=/usr/local/mfs --with-default-user=mfs --with-default-group=mfs
[root@tianyun mfs-1.6.17]# make && make install
[root@tianyun mfs-1.6.17]# cd /usr/local/mfs/etc
[root@tianyun etc]# mv mfsmaster.cfg.dist mfsmaster.cfg //主配置文件
[root@tianyun etc]# mv mfsexports.cfg.dist mfsexports.cfg //导出文件，权限控制文件
[root@tianyun etc]# cp /usr/local/mfs/var/mfs/metadata.mfs.empty /usr/local/mfs/var/mfs/metadata.mfs
```

```
=====
主配置文件示例 : /usr/local/mfs/etc/mfsmaster.cfg
EXPORTS_FILENAME = /usr/local/mfs/etc/mfsexports.cfg           //权限控制文件的存储位置
DATA_PATH = /usr/local/mfs/var/mfs                           //指定元数据的存储路径
MATOCS_LISTEN_PORT = 9420                                     //Master to Chunk Server 端口监听
MATOML_LISTEN_PORT = 9419                                     //Master to MetaLogger
MATOCU_LISTEN_PORT = 9421                                     //客户端挂载MFS时连接的端口
```

```
export文件配置示例 :      /usr/local/mfs/etc/mfsexports.cfg
*          /     rw,alldirs,mapall=0                                //默认配置
=====
```

```
[root@tianyun ~]# /usr/local/mfs/sbin/mfsmaster start
[root@tianyun ~]# /usr/local/mfs/sbin/mfscgiserv           //可选项，可以通过web方式查看mfs导出等信息
starting simple cgi server (host: any , port: 9425 , rootpath: /usr/local/mfs/share/mfscgi)
[root@tianyun ~]# firefox http://192.168.1.254:9425
```

matelogger server:

```
[root@tianyun ~]# useradd mfs -s /sbin/nologin
[root@tianyun ~]# tar zxvf mfs-1.6.17.tar.gz
[root@tianyun ~]# cd mfs-1.6.17/
[root@tianyun mfs-1.6.17]# ./configure --prefix=/usr/local/mfs --with-default-user=mfs --with-default-group=mfs
[root@tianyun mfs-1.6.17]# make && make install

[root@tianyun mfs-1.6.17]# cd /usr/local/mfs/etc
[root@tianyun etc]# mv mfsmetalogger.cfg.dist mfsmetalogger.cfg
```

```
=====
/usr/local/mfs/etc/mfsmetalogger.cfg
MASTER_HOST = 192.168.2.101           //指定Master服务器
MASTER_PORT = 9419
DATA_PATH = /usr/local/mfs/var/mfs       //从Master获取的日志文件存储的位置
BACK_LOGS = 50                          //超过50个备份日志则轮转
META_DOWNLOAD_FREQ = 24                 //元数据备份文件下载的频率，默认24小时
```

```
=====
[root@tianyun etc]# /usr/local/mfs/sbin/mfsmetalogger start
working directory: /usr/local/mfs/var/mfs
lockfile created and locked
initializing mfsmetalogger modules ...
mfsmetalogger daemon initialized properly
```

chunk server:

```
[root@tianyun ~]# useradd mfs -s /sbin/nologin
[root@tianyun ~]# tar zxvf mfs-1.6.17.tar.gz
[root@tianyun ~]# cd mfs-1.6.17/
[root@tianyun mfs-1.6.17]# yum -y install gcc make fuse-devel zlib-devel
[root@tianyun mfs-1.6.17]# ./configure --prefix=/usr/local/mfs --with-default-user=mfs --with-default-group=mfs
[root@tianyun mfs-1.6.17]# make && make install
```

```
[root@tianyun ~]# cd /usr/local/mfs/etc
[root@tianyun ~]# mv mfschunkserver.cfg.dist mfschunkserver.cfg
[root@tianyun ~]# mv mfshdd.cfg.dist mfshdd.cfg
[root@tianyun ~]# mkdir /data
[root@tianyun ~]# chown -R mfs:mfs /data
```

chunkserver配置示例:

```
/usr/local/mfs/etc/mfschunkserver.cfg  
MASTER_HOST = 192.168.2.101          //master server  
  
/usr/local/mfs/etc/mfshdd.cfg  
/data                                //分区的挂载点
```

MFS用户没有权限的启动报错

```
[root@tianyun ~]# /usr/local/mfs/sbin/mfschunkserver start  
working directory: /usr/local/mfs/var/mfs  
lockfile created and locked  
initializing mfschunkserver modules ...  
hdd space manager: can't create lock file '/mfsdir/.lock' (errno:13)  
init: hdd space manager failed !!!  
error occurred during initialization - exiting
```

Chunk Server正常启动

```
[root@tianyun ~]# /usr/local/mfs/sbin/mfschunkserver start  
working directory: /usr/local/mfs/var/mfs  
lockfile created and locked  
initializing mfschunkserver modules ...  
hdd space manager: scanning folder /mfsdir/ ...  
hdd space manager: /mfsdir/: 0 chunks found  
hdd space manager: scanning complete  
main server module: listen on *:9422  
no charts data file - initializing empty charts  
mfschunkserver daemon initialized properly  
=====
```

mfs client:

```
[root@client ~]# useradd mfs -s /sbin/nologin  
[root@client ~]# yum -y install fuse-devel  (user space filesystem extend)  
[root@client ~]# tar zxvf mfs-1.6.17.tar.gz.gz  
[root@client ~]# cd mfs-1.6.17/  
[root@client mfs-1.6.17]# yum -y install gcc make fuse-devel zlib-devel  
[root@client mfs-1.6.17]# ./configure --prefix=/usr/local/mfs --with-default-user=mfs --with-default-group=mfs --enable-mfsmount  
[root@client mfs-1.6.17]# make && make install
```

```
[root@client ~]# /usr/local/mfs/bin/mfsmount -H 192.168.1.254 /var/www/html  
mfsmaster accepted connection with parameters: read-write,restricted_ip ; root mapped to root:root
```

```
[root@client ~]# df -Th /var/www/html  
文件系统    类型    容量   已用   已用% 挂载点  
mfs#192.168.1.254:9421  
    fuse      47G     0    50G  0% /var/www/html
```

```
[root@client ~]# mount  
192.168.1.254:9421 on /var/www/html type fuse.mfs (rw,nosuid,nodev,allow_other,default_permissions)
```

三、客户端进行读写测试

```
[root@client ~]# vim /etc/profile  
export PATH=$PATH:/usr/local/mfs/bin  
[root@client ~]# source  
  
[root@client ~]# mfssetgoal -r 3 /var/www/html      //设置文件副本数  
[root@client ~]# cd /var/www/html  
[root@client ~]# cp -rf /etc/passwd .  
[root@client ~]# mfsgetgoal passwd                //查看文件保存的副本数  
[root@client ~]# mfsfileinfo passwd
```

四、MFS扩展

mfs的回收站设置
mfs的快照

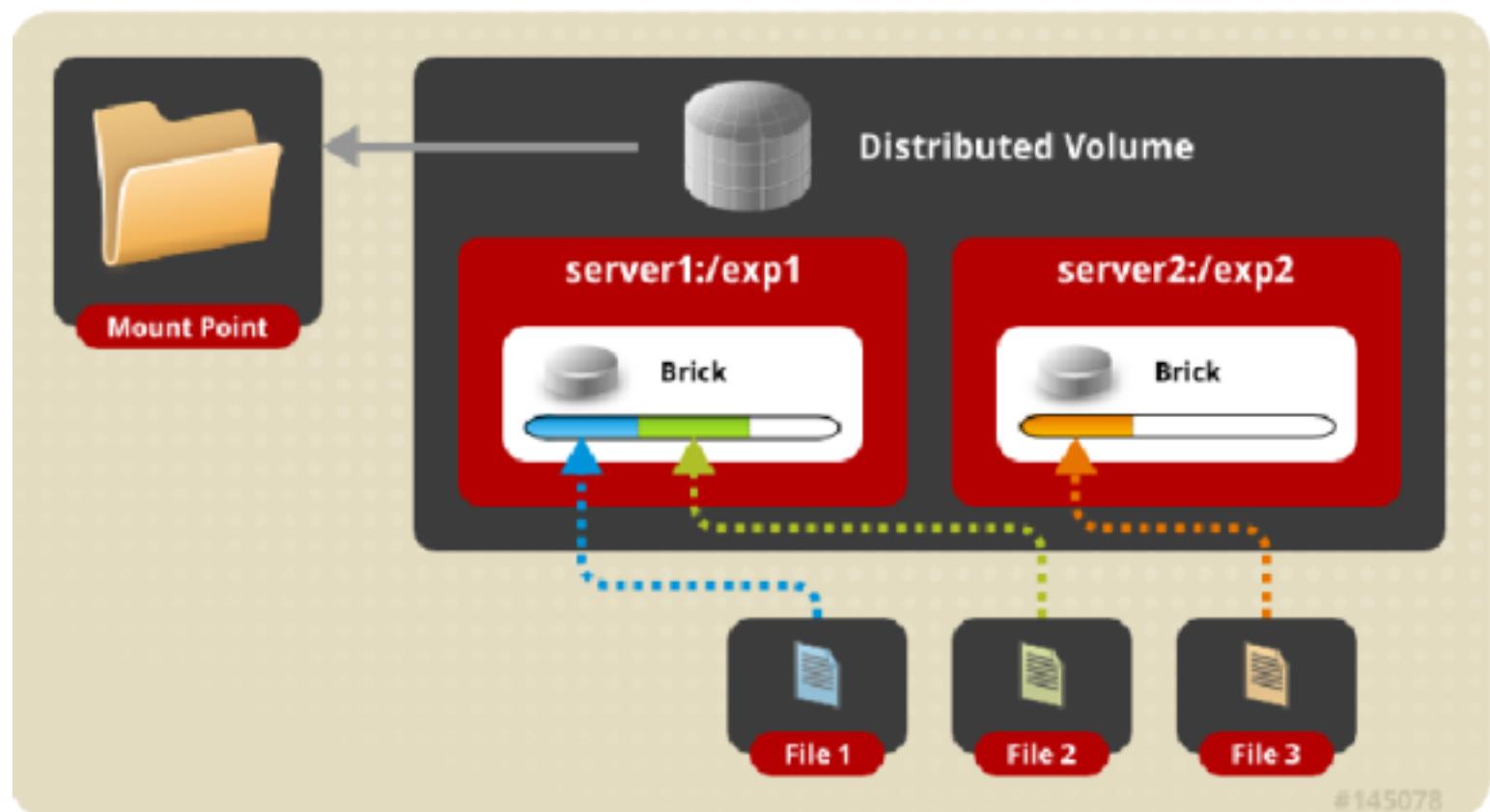
mfs利用metallogger server恢复master server
mfsmetarestore -a

Hadoop HDFS

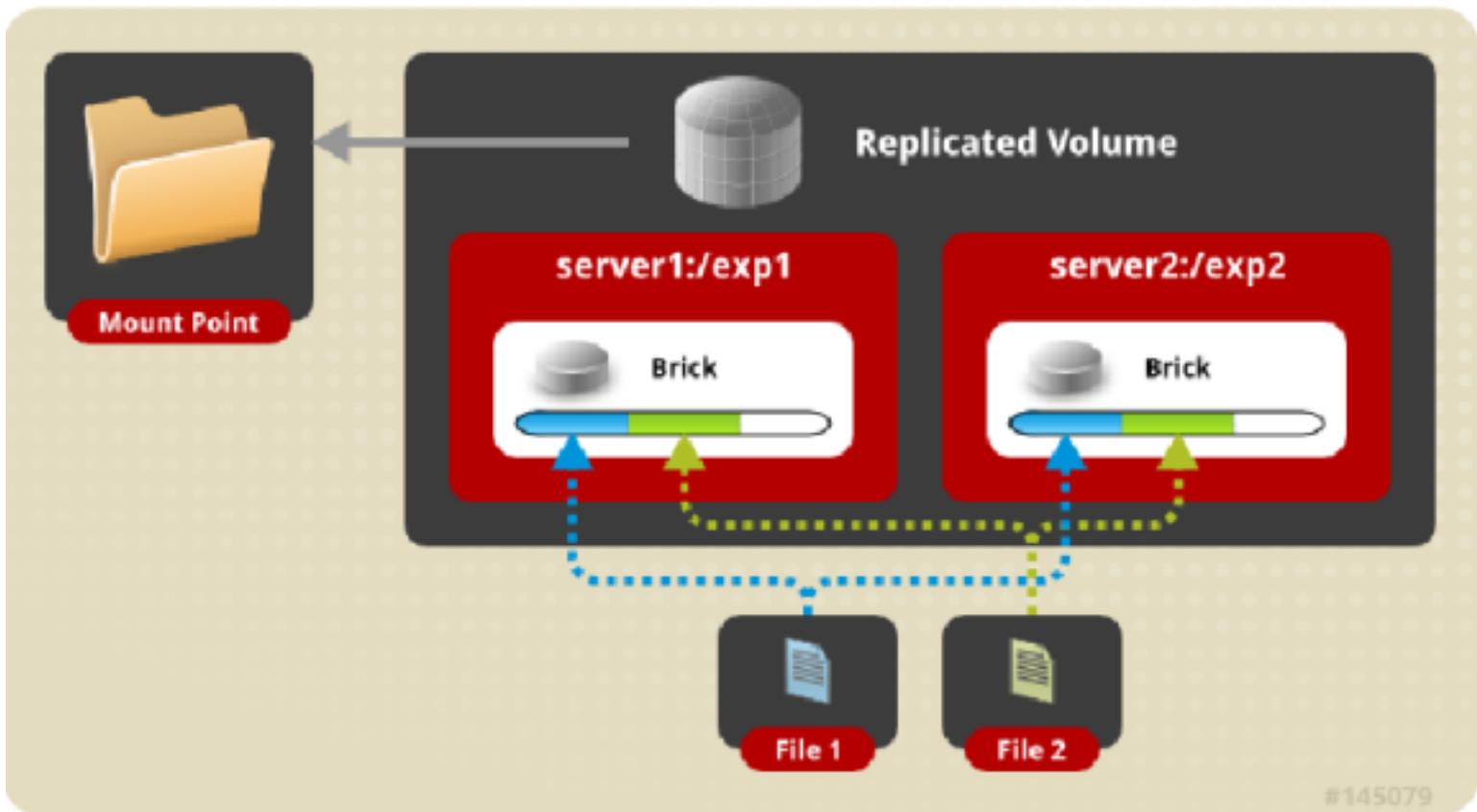
GlusterFS

GNU ClusterFile System

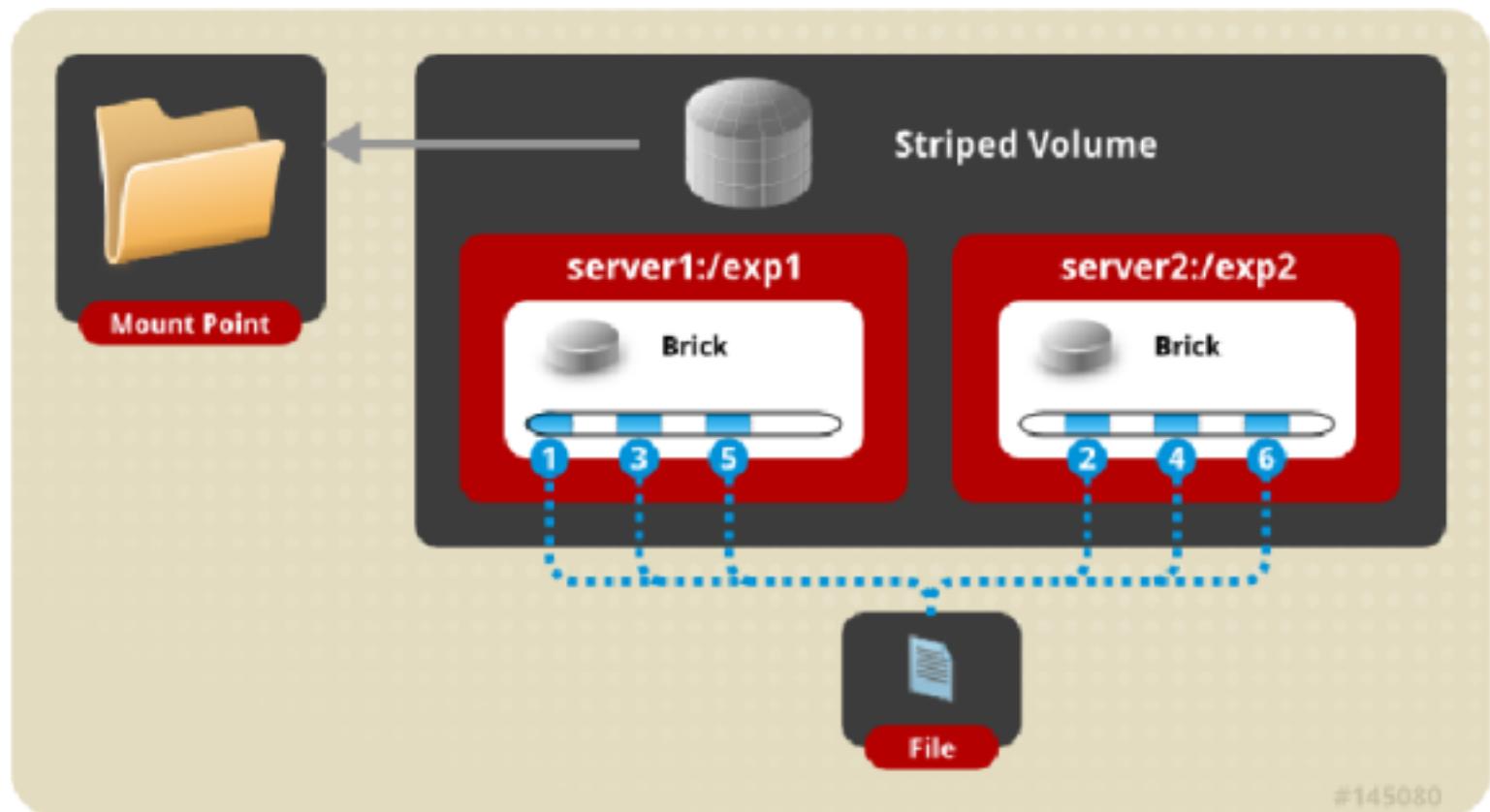
Servers: tianyun1, tianyun2, tianyun3, tianyun4
Clients: client1



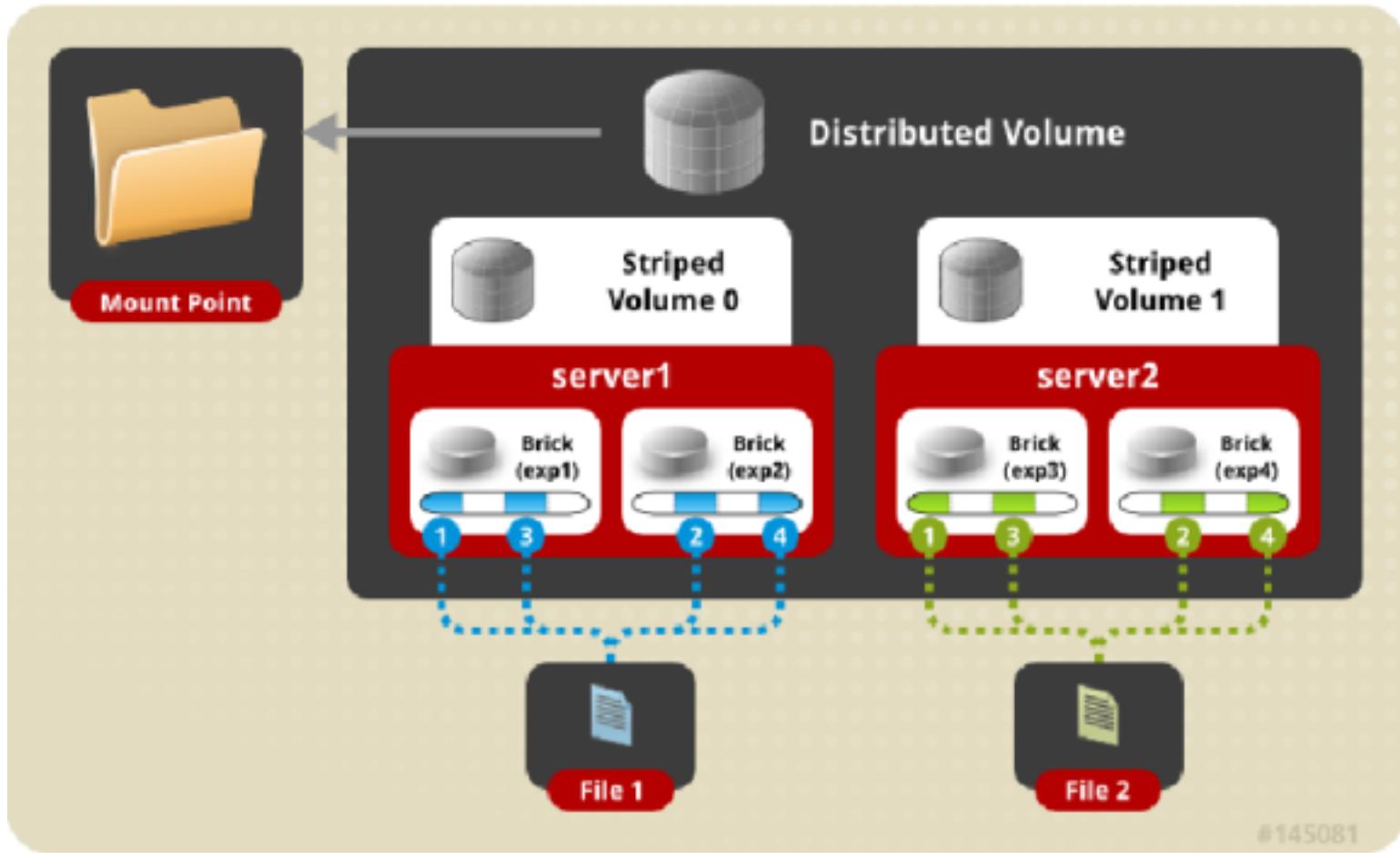
#145078



#145079



#145080



#145081

一、安装软件包tianyun{1..4}

```
[root@tianyun1 ~]# yum install -y glusterfs glusterfs-server glusterfs-fuse
[root@tianyun1 ~]# service glusterd start
[root@tianyun1 ~]# chkconfig glusterd on
```

二、添加服务器到信任存储池 Adding Servers to Trusted Storage Pool

```
[root@tianyun1 ~]# gluster peer probe tianyun2
[root@tianyun1 ~]# gluster peer probe tianyun3
[root@tianyun1 ~]# gluster peer probe tianyun4
[root@tianyun1 ~]# gluster peer status
```

三、创建GlusterFS Server Volumes

```
[root@tianyun1 ~]# gluster volume create datav1 replica 2 transport tcp tianyun1:/data1 tianyun2:/data1 tianyun3:/data1 tianyun4:/data1 force
[root@tianyun1 ~]# gluster volume start datav1
[root@tianyun1 ~]# gluster volume info
```

四、挂载GlusterFS Volumes

```
[root@client1 ~]# mkdir /mnt/v1/
[root@client1 ~]# mount -t glusterfs -o backup-volfile-servers=tianyun2:tianyun3:tianyun4 tianyun1:/datav1 /mnt/v1/
[root@client1 ~]# echo "test...." > /mnt/v1/file1
```

五、扩充数据节点

Taobao TFS [自修]

5.5 Mem

memcache原理

memcache

原理: memcached全面剖析.pdf

=====

Memcached 是一个高性能的分布式内存对象缓存系统，用于缓存数据以减轻数据库负载。它通过在内存中缓存数据和对象来减少读取数据库的次数，从而提高数据库驱动应用的速度。Memcached基于一个存储键/值对的hashmap。其守护进程 (daemon) 是用C写的，但是客户端可以用任何语言来编写，并通过memcached协议与守护进程通信。

memcached 的特征

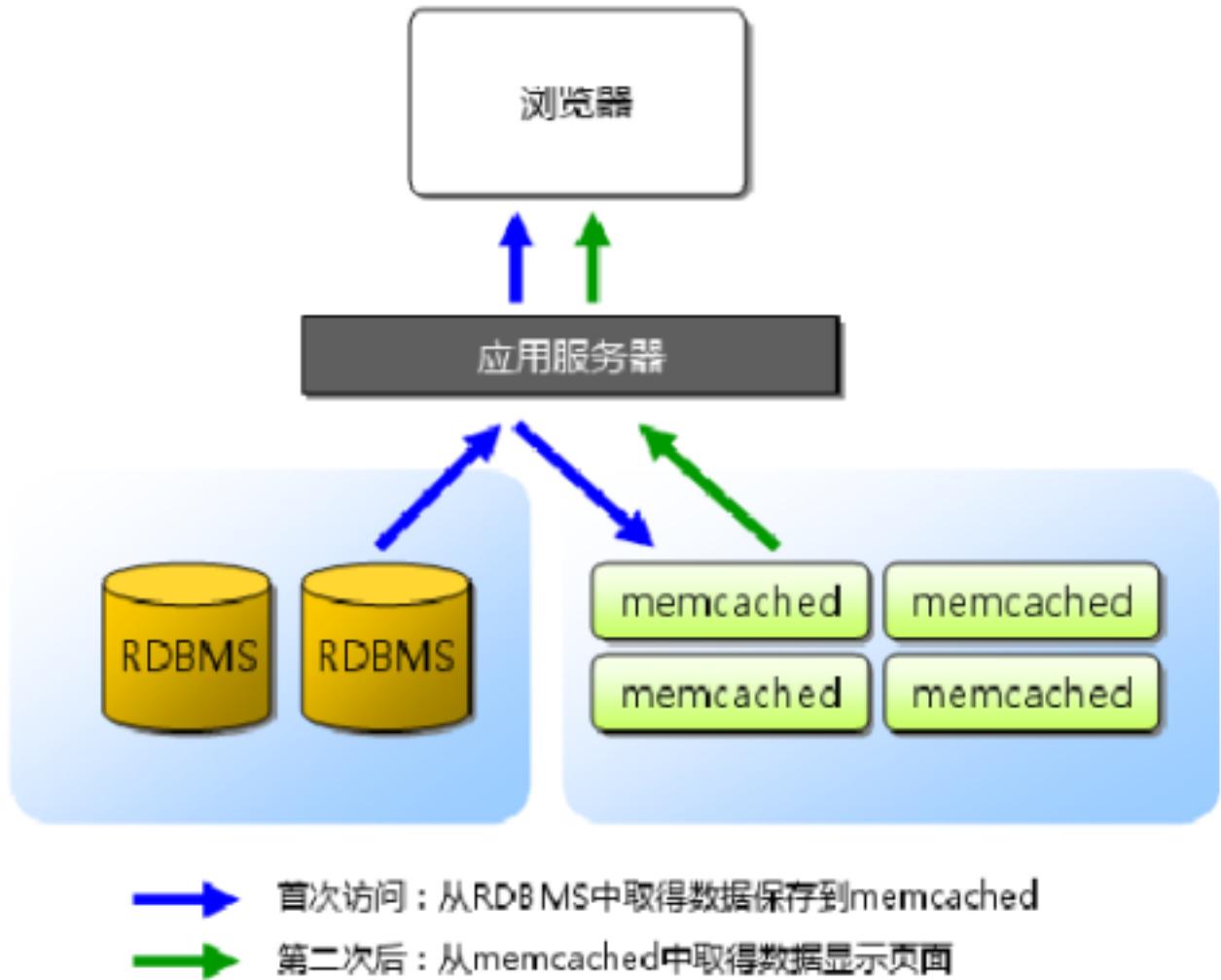
memcached 作为高速运行的分布式缓存服务器,具有以下的特点。

协议简单

基于 libevent 的事件处理

内置内存存储方式

memcached 不互相通信的分布式



1. 例如放在RDBMS（关系型数据库管理系统）的前端做cache，从而降低数据库的访问压力。
2. 使用memcached存储如session等数据，例如在LB集群中实现session共享。

memcache应用

memcache应用

拓扑结构：

[web php]:80
192.168.122.11

[memcached1]:11211 [memcached2]:11211
192.168.122.6 192.168.122.7

一、安装memcached服务器

方法一：

libevent

```
[root@localhost ~]# tar xvf libevent-2.0.21-stable.tar.gz
[root@localhost ~]# cd libevent-2.0.21-stable
[root@localhost libevent-2.0.21-stable]#./configure --prefix=/usr/local/libevent && make && make install
```

```
[root@localhost libevent-2.0.21-stable]# vim /etc/ld.so.conf.d/libevent.conf  
/usr/local/libevent/lib/  
[root@localhost memcached-1.4.10]# ldconfig -v  
  
memcached  
[root@localhost ~]# tar xvf memcached-1.4.10.tar.gz  
[root@localhost ~]# cd memcached-1.4.10  
[root@localhost memcached-1.4.10]# ./configure --prefix=/usr/local/memcache  
[root@localhost memcached-1.4.10]# make && make install  
[root@localhost memcached-1.4.10]# /usr/local/memcache/bin/memcached -u root -m 512 -p 11211 -vv  
slab class 1: chunk size 80 perslab 13107  
slab class 2: chunk size 104 perslab 10082  
slab class 3: chunk size 136 perslab 7710  
slab class 4: chunk size 176 perslab 5957  
slab class 5: chunk size 224 perslab 4681  
slab class 6: chunk size 280 perslab 3744  
slab class 7: chunk size 352 perslab 2978  
slab class 8: chunk size 440 perslab 2383
```

方法二：

```
[root@localhost ~]# yum -y install memcached  
[root@localhost ~]# /etc/init.d/memcached restart  
[root@localhost ~]# chkconfig memcached on
```

```
[alice@tianyun ~]$ /usr/bin/memcached -vv -f 2 -m 16000 -p 1121 -d
```

二、客户端使用memcache

```
1. 手动使用memcache协议命令<测试>  
[root@client ~]# telnet 192.168.122.11 11211  
Trying 192.168.122.11...  
Connected to 192.168.122.11.  
Escape character is '^]'.  
get tianyun  
END  
set tianyun 0 50 3  
abc  
STORED  
get tianyun  
VALUE tianyun 0 3s  
abc  
END
```

2. php应用程序使用memcache存储数据

方法一：给RPM安装的php 增加Memcache扩展

```
[root@client ~]# yum -y install gd httpd mysql-server php php-mysql php-gd php-pecl-memcache  
[root@client ~]# php -m  
[root@client ~]# service httpd restart
```

方法二：给源码安装php 增加Memcache扩展：

例如：php软件已安装在：/usr/local/php

a. Memcache扩展安装：

```
# tar zxvf memcache-2.2.4.tgz  
# cd memcache-2.2.4  
# /usr/local/php/bin/phpize //预先安装phpize  
# ./configure --with-php-config=/usr/local/php/bin/php-config  
# make  
# make install
```

b. 配置php

```
# ls -l /usr/local/php/lib/php/extensions/no-debug-non-zts-20060613/memcache.so  
# vim /usr/local/php/lib/php.ini  
extension_dir = "/usr/local/php/lib/php/extensions/no-debug-non-zts-20060613/"  
extension = memcache.so  
=====
```

测试一：是否支持PHP Memcache扩展

```
[root@web ~]# vim /var/www/html/index.php  
<?php  
phpinfo();
```

?>

测试二：分别测试192.168.122.6 和 192.168.122.7 (开发人员)

```
<?php
//连接Memcache
$mem = new Memcache;
$mem->connect("192.168.122.6", 11211);

//保存数据
$mem->set('key1', 'This is first value tianyun', 0, 60);
$val = $mem->get('key1');
echo "Get key1 value: " . $val . "<br>";

//替换数据
$mem->replace('key1', 'This is replace value', 0, 60);
$val = $mem->get('key1');
echo "Get key1 value: " . $val . "<br>";

//保存数组数据
$arr = array('aaa', 'bbb', 'ccc', 'ddd');
$mem->set('key2', $arr, 0, 60);
$val2 = $mem->get('key2');
echo "Get key2 value: ";
print_r($val2);
echo "<br>";

//删除数据
$mem->delete('key1');
$val = $mem->get('key1');
echo "Get key1 value: " . $val . "<br>";

//清除所有数据
$mem->flush();
$val2 = $mem->get('key2');
echo "Get key2 value: ";
print_r($val2);
echo "<br>";

//关闭连接
$mem->close();
?>
```

测试三：PHP与Memcache分布式 (开发人员)

```
<?php
//连接Memcache
$mem = new Memcache;
$mem->addServer("192.168.122.6", 11211);
$mem->addServer("192.168.122.7", 11211);

//保存数据
$mem->set('key1', 'This is first value', 0, 60);
$val = $mem->get('key1');
echo "Get key1 value: " . $val . "<br>";

//保存数据
$mem->set('tianyun', 'This is first value', 0, 60);
$val = $mem->get('tianyun');
echo "Get tianyun value: " . $val . "<br>";

//保存数组数据
$arr = array('aaa', 'bbb', 'ccc', 'ddd');
$mem->set('key2', $arr, 0, 60);
$val2 = $mem->get('key2');
echo "Get key2 value: ";
print_r($val2);
echo "<br>";

//删除数据
$mem->delete('key1');
$val = $mem->get('key1');
```

```
echo "Get key1 value: " . $val . "<br>";  
//关闭连接  
$mem->close();  
?>
```

3. php使用memcache存储session (运维人员)

```
[root@web ~]# vim /etc/php.ini  
[Session]  
session.save_handler = memcache  
session.save_path = "tcp://192.168.1.1:11211,tcp://192.168.1.7:11211"  
memcache.hash_strategy = "consistent"  
session.cookie_domain = ".tianyun.com"
```

```
<?php  
session_start();  
if (!isset($_SESSION['session_time'])) {  
    $_SESSION['session_time'] = time();  
}  
echo "session_time:". $_SESSION['session_time']. "<br />";  
echo "now_time:". time(). "<br />";  
echo "session_id:". session_id(). "<br />";  
?>  
=====
```

对于session的存储：

1. 如果网站代码中定义了session的存储位置，则按照该方法存储； [开发人员]
2. 如果网站代码中没有定义session的存储位置，则按照程序如php定义的位置存储，php程序通过主配置文件php.ini定义session存储，默认为本地。 [运维人员]

5.6 Rsync

Rsync

Rsync

用于实现数据同步：

1. 例如备份NAS或其它存储服务器上的文件；
2. 例如从[发布服务器](#)推数据到其它服务器。

注：将目标服务器配置为rsync server端，运行rsync命令端为rsync client

方法一：

基于sshd服务器，并client使用server的OS帐号同步数据，可以结合ssh公钥认证。

方法二：

Server端启动rsyncd服务，并创建专用的rsync帐号（最终也要映射到系统帐号，对应一定的权限）

=====方法一=====

Server端

```
[root@server ~]# service sshd restart  
[root@server ~]# rpm -q rsync  
rsync-3.0.6-9.el6.x86_64
```

客户端

语法 : rsync [OPTION]... [USER@]HOST:SRC [DEST]

```
[root@client ~]# rpm -qf `which rsync`  
rsync-3.0.6-4.el5_7.1
```

```
[root@client ~]# rsync -va /etc/hosts /tmp/  
[root@client ~]# mkdir /testtmp  
[root@client ~]# rsync -va /etc 192.168.1.251:/tmp  
[root@client ~]# rsync -va /etc/ 192.168.1.251:/tmp  
[root@client ~]# rsync -vza /etc 192.168.1.251:/tmp  
[root@client ~]# rsync -vza --delete /etc root@192.168.1.251:/tmp
```

```
[root@client ~]# rsync -vza 192.168.1.251:/tmp /testtmp  
[root@client ~]# rsync -vza --delete /etc/ 192.168.1.251:/
```

Rsync+inotify

Rsync+inotify 实时同步(触发式)

一、rsync+cron的优点与不足

与传统的cp、scp、tar备份方式相比，rsync具有安全性高、备份迅速、支持增量备份等优点，通过rsync可以解决对实时性要求不高的数据备份需求，例如定期的备份文件服务器数据到远端服务器，对本地磁盘定期做数据镜像等。

随着应用系统规模的不断扩大，对数据的安全性和可靠性也提出更好的要求，rsync在高端业务系统中也逐渐暴露出了很多不足，首先，rsync

同步数据时，需要扫描所有文件后进行比对，进行增量传输。如果文件数量达到了百万甚至千万量级，扫描所有文件将是非常耗时的。而且正在

发生

变化的往往是其中很少的一部分，这是非常低效的方式。其次，rsync不能实时的去监测、同步数据，虽然它可以通过linux守护进程的方式进行触发

同步，但是两次触发动作一定会有时间差，这样就导致了服务端和客户端数据可能出现不一致，无法在应用故障时完全的恢复数据

二、初识inotify

Inotify 是一种强大的、细粒度的、异步的文件系统事件监控机制，linux内核从2.6.13起，加入了Inotify支持，通过Inotify可以监控文件系统中添

加、删除，修改、移动等各种细微事件，利用这个内核接口，第三方软件就可以监控文件系统下文件的各种变化情况，而`inotify-tools`就是这样的一
个第三方软件。

rsync可以实现触发式的文件同步，但是通过crontab守护进程方式进行触发，同步的数据和实际数据会有差异，而inotify可以监控文件系统的各种变化，当文件有任何变动时，就触发rsync同步，这样刚好解决了同步数据的实时性问题。

三、安装inotify工具inotify-tools

1. 确认内核支持

```
[root@tianyun ~]# ll /proc/sys/fs/inotify
总计 0
-rw-r--r-- 1 root root 0 10-01 08:07 max_queued_events
-rw-r--r-- 1 root root 0 10-01 08:07 max_user_instances
-rw-r--r-- 1 root root 0 10-01 08:07 max_user_watches
```

2 安装rsync与inotify-tools

<http://inotify-tools.sourceforge.net>

inotify-tools是用来监控文件系统变化的工具，因此必须安装在**内容发布节点（数据源）**

```
[root@tianyun ~]# tar xf inotify-tools-3.14.tar.gz
```

```
[root@tianyun ~]# cd inotify-tools-3.14
```

```
[root@tianyun inotify-tools-3.14]# ./configure && make && make install
```

inotify-tools指令：

`inotifywait` 用于等待文件或文件集上的一个特定事件，它可以监控任何文件和目录
`inotifywatch` 用于收集被监控的文件系统统计数据，包括每个inotify事件发生多少次等信息

四、inotifywait相关参数

Inotifywait是一个监控等待事件，可以配合shell脚本使用，常用的一些参数：

`-m`，即--monitor，表示始终保持事件监听状态

`-r`，即--recursive，表示递归查询目录

`-q`，即--quiet，表示打印出监控事件

`-e`，即--event，通过此参数可以指定要监控的事件，常见的事件有modify、delete、create、attrib等

```
[root@tianyun ~]# inotifywait -mrq -e modify,delete,create,attrib /var/www/html/
```

```
[root@tianyun ~]# inotifywait -mrq --timefmt '%d/%m/%y %H:%M' --format '%T %w%f' -e modify,delete,create,attrib /var/www/html/
```

```
04/06/14 14:23 /var/www/html/file1
```

```
04/06/14 14:23 /var/www/html/file1
```

```
[root@tianyun ~]# inotifywait -mrq --format '%w%f' -e modify,delete,create,attrib /var/www/html/
/var/www/html/file2
/var/www/html/file2
```

五、配置示例

拓扑图

192.168.5.11 发布服务器(数据源)
 /var/www/html

web2 192.168.5.16 web3 192.168.5.2
/var/www/html /var/www/html

发部服务器

1. hosts

```
[root@source ~]# cat /etc/hosts
127.0.0.1       localhost
```

```
192.168.5.11 source  
192.168.5.16 web2  
192.168.5.2 web3
```

2. 配置ssh公钥认证

```
[root@source ~]# ssh-keygen  
[root@source ~]# ssh-copy-id -i web2  
[root@source ~]# ssh-copy-id -i web3  
[root@source ~]# ssh web2 date;ssh web3 date  
2015年 09月 29日 星期日 11:31:19 CST  
2015年 09月 29日 星期日 11:33:07 CST
```

3. 安装inotify-tools

```
[root@source ~]# yum -y install rsync httpd  
[root@source ~]# tar xf inotify-tools-3.14.tar.gz  
[root@source ~]# cd inotify-tools-3.14  
[root@source inotify-tools-3.14]# ./configure && make && make install
```

4. 编写脚本

```
[root@source ~]# cat /root/rsync_inotify.sh  
#!/bin/  
bash  
web1=192.168.5.16  
web2=192.168.5.2  
src=/var/www/  
html/  
dst=/var/www/  
html/  
  
inotifywait -mrq --timefmt '%d/%m/%y %H:%M' --format '%T %w%f' \  
-e modify,delete,create,attrib $src \  
| while read file  
do
```

```
    rsync -avz --delete $src root@$web1:$dst &>/dev/null  
    rsync -avz --delete $src root@$web2:$dst &>/dev/null
```

```
done
```

```
[root@source ~]# chmod a+x /root/rsync_inotify.sh  
[root@source ~]# /root/rsync_inotify.sh &
```

web2, web3

```
[root@web2 ~]# yum -y install rsync httpd  
[root@web3 ~]# yum -y install rsync httpd
```

同步测试

5.7 Device mapper

设备映射基础

Device mapper 设备映射

作用：

将一个或多个物理设备(硬盘/分区)映射成一个逻辑设备。Device mapper是实现LVM和multipath的底层技术

常见的映射类型：

linear 线性
striped 条带化
mirror 镜像
snapshot 快照
.....

linear

linear

创建线性设备

```
[root@station21 ~]# ll /dev/sda*
brw-rw---- 1 root disk 8, 0 6月 5 16:24 /dev/sda
brw-rw---- 1 root disk 8, 1 6月 5 08:29 /dev/sda1
brw-rw---- 1 root disk 8, 2 6月 5 08:28 /dev/sda2
brw-rw---- 1 root disk 8, 3 6月 5 08:29 /dev/sda3
brw-rw---- 1 root disk 8, 4 6月 5 16:24 /dev/sda4
brw-rw---- 1 root disk 8, 5 6月 5 16:24 /dev/sda5
brw-rw---- 1 root disk 8, 6 6月 5 16:24 /dev/sda6
brw-rw---- 1 root disk 8, 7 6月 5 16:24 /dev/sda7
```

```
[root@station21 ~]# pvcreate /dev/sda{5,6,7}
Physical volume "/dev/sda5" successfully created
Physical volume "/dev/sda6" successfully created
Physical volume "/dev/sda7" successfully created
```

```
[root@station21 ~]# vgcreate vg1 /dev/sda{5,6,7}
Volume group "vg1" successfully created
```

```
[root@station21 ~]# lvcreate -L 150M -n lv1 vg1
Rounding up size to full physical extent 152.00 MiB
Logical volume "lv1" created
```

查看映射表：

```
[root@station21 ~]# dmsetup table |grep lv1
vg1-lv1: 0 196608 linear 8:5 2048
```

vg1-lv1: 196608 114688 linear 8:6 2048

```
[root@station21 ~]# dmsetup ls --tree
vg01-lv_root (253:1)
└─ (8:2)
vg01-LogVol04 (253:0)
└─ (8:2)
vg1-lv1 (253:5)
└─ (8:6)
└─ (8:5)
vg01-LogVol03 (253:2)
└─ (8:2)
vg01-lv_usr (253:4)
└─ (8:2)
vg01-lv_var (253:3)
└─ (8:2)

[root@station21 ~]# ll /dev/mapper/*
crw-rw--- 1 root root 10, 58 6月 5 08:28 /dev/mapper/control
lrwxrwxrwx 1 root root 7 6月 5 08:29 /dev/mapper/vg01-LogVol03 -> ../dm-2
lrwxrwxrwx 1 root root 7 6月 5 08:28 /dev/mapper/vg01-LogVol04 -> ../dm-0
lrwxrwxrwx 1 root root 7 6月 5 08:28 /dev/mapper/vg01-lv_root -> ../dm-1
lrwxrwxrwx 1 root root 7 6月 5 08:29 /dev/mapper/vg01-lv_usr -> ../dm-4
lrwxrwxrwx 1 root root 7 6月 5 08:29 /dev/mapper/vg01-lv_var -> ../dm-3
lrwxrwxrwx 1 root root 7 6月 5 16:27 /dev/mapper/vg1-lv1 -> ../dm-5
```

测试读写：

终端一：

```
[root@station21 ~]# iostat 1 /dev/sda5 /dev/sda6
```

终端二：

```
[root@station21 ~]# mkdir /mnt/lv1
[root@station21 ~]# mount /dev/mapper/vg1-lv1 /mnt/lv1/
[root@station21 ~]# dd if=/dev/zero of=/mnt/lv1/file1
```

```
[root@station21 ~]# while :; do dd if=/dev/zero >>/mnt/lv1/file1 bs=1M count=1; usleep 30; done
```

使用了一块硬盘

```
[root@localhost ~]# dmsetup table
vg1-lv1: 0 2097152 linear 8:16 2048
```

```
[root@localhost ~]# dmsetup create tianyun1 --table "0 4194304 linear /dev/sdc 0"
[root@localhost ~]#
[root@localhost ~]# ll /dev/mapper/tianyun1
lrwxrwxrwx 1 root root 7 Nov 6 11:31 /dev/mapper/tianyun1 -> ../dm-0
[root@localhost ~]#
[root@localhost ~]# dmsetup table
tianyun1: 0 4194304 linear 8:32 0
[root@localhost ~]#
[root@localhost ~]# mkfs.ext4 /dev/mapper/tianyun1
```

striped

striped 相当于RAIDO

创建条带设备

```
[root@station21 ~]# ll /dev/sda*
brw-rw--- 1 root disk 8, 0 6月 5 16:24 /dev/sda
brw-rw--- 1 root disk 8, 1 6月 5 08:29 /dev/sda1
brw-rw--- 1 root disk 8, 2 6月 5 08:28 /dev/sda2
brw-rw--- 1 root disk 8, 3 6月 5 08:29 /dev/sda3
brw-rw--- 1 root disk 8, 4 6月 5 16:24 /dev/sda4
brw-rw--- 1 root disk 8, 5 6月 5 16:24 /dev/sda5
brw-rw--- 1 root disk 8, 6 6月 5 16:24 /dev/sda6
brw-rw--- 1 root disk 8, 7 6月 5 16:24 /dev/sda7
```

```
[root@station21 ~]# pvcreate /dev/sda{5,6,7}
Physical volume "/dev/sda5" successfully created
Physical volume "/dev/sda6" successfully created
Physical volume "/dev/sda7" successfully created
```

```
[root@station21 ~]# vgcreate vg1 /dev/sda{5,6,7}
Volume group "vg1" successfully created
```

```
[root@station21 ~]# lvcreate -i 2 -L 150M -n lv1 vg1
Using default stripesize 64.00 KiB
Rounding up size to full physical extent 152.00 MiB
Logical volume "lv1" created
```

查看映射表：

```
[root@station21 ~]# dmsetup table |grep lv1
vg1-lv1: 0 311296 striped 2 128 8:5 2048 8:6 2048
```

```
[root@station21 ~]# dmsetup ls --tree
vg01-lv_root (253:1)
└─(8:2)
vg01-LogVol04 (253:0)
└─(8:2)
vg1-lv1 (253:5)
├─(8:6)
└─(8:5)
vg01-LogVol03 (253:2)
└─(8:2)
vg01-lv_usr (253:4)
└─(8:2)
vg01-lv_var (253:3)
└─(8:2)
```

```
[root@station21 ~]# ll /dev/mapper/*
crw-rw--- 1 root root 10, 58 6月 5 08:28 /dev/mapper/control
lrwxrwxrwx 1 root root    7 6月 5 08:29 /dev/mapper/vg01-LogVol03 -> ../dm-2
lrwxrwxrwx 1 root root    7 6月 5 08:28 /dev/mapper/vg01-LogVol04 -> ../dm-0
lrwxrwxrwx 1 root root    7 6月 5 08:28 /dev/mapper/vg01-lv_root -> ../dm-1
lrwxrwxrwx 1 root root    7 6月 5 08:29 /dev/mapper/vg01-lv_usr -> ../dm-4
lrwxrwxrwx 1 root root    7 6月 5 08:29 /dev/mapper/vg01-lv_var -> ../dm-3
lrwxrwxrwx 1 root root    7 6月 5 16:27 /dev/mapper/vg1-lv1 -> ../dm-5
```

测试读写：

终端一：

```
[root@station21 ~]# iostat 1 /dev/sda5 /dev/sda6
Device:      tps  Blk_read/s  Blk_wrtn/s  Blk_read  Blk_wrtn
sda5        57.00      0.00     384.00       0       384
sda6        54.00      0.00     358.00       0       358
```

终端二：

```
[root@station21 ~]# mkdir /mnt/lv1
[root@station21 ~]# mount /dev/mapper/vg1-lv1 /mnt/lv1/
[root@station21 ~]# dd if=/dev/zero of=/mnt/lv1/file1
```

```
[root@station21 ~]# while :; do dd if=/dev/zero >>/mnt/lv1/file1 bs=1M count=1; usleep 3000; done
```

mirror

mirror 相当于RAID1

创建设备

```
[root@station21 ~]# ll /dev/sda*
brw-rw---- 1 root disk 8, 0 6月 5 16:24 /dev/sda
brw-rw---- 1 root disk 8, 1 6月 5 08:29 /dev/sda1
brw-rw---- 1 root disk 8, 2 6月 5 08:28 /dev/sda2
brw-rw---- 1 root disk 8, 3 6月 5 08:29 /dev/sda3
brw-rw---- 1 root disk 8, 4 6月 5 16:24 /dev/sda4
brw-rw---- 1 root disk 8, 5 6月 5 16:24 /dev/sda5
brw-rw---- 1 root disk 8, 6 6月 5 16:24 /dev/sda6
brw-rw---- 1 root disk 8, 7 6月 5 16:24 /dev/sda7
```

```
[root@station21 ~]# pvcreate /dev/sda{5,6,7}
Physical volume "/dev/sda5" successfully created
Physical volume "/dev/sda6" successfully created
Physical volume "/dev/sda7" successfully created
```

```
[root@station21 ~]# vgcreate vg1 /dev/sda{5,6,7}
Volume group "vg1" successfully created
```

```
[root@station21 ~]# pvs
PV      VG  Fmt Attr PSize  PFree
/dev/sda2  vg01 lvm2 a-- 390.62g  0
/dev/sda5  vg1   lvm2 a-- 96.00m 96.00m
/dev/sda6  vg1   lvm2 a-- 96.00m 96.00m
/dev/sda7  vg1   lvm2 a-- 96.00m 96.00m
```

```
[root@station21 ~]# lvcreate -m 1 -L 50M -n lv1 vg1
Rounding up size to full physical extent 52.00 MiB
Logical volume "lv1" created
```

```
[root@station21 ~]# pvs
PV      VG  Fmt Attr PSize  PFree
/dev/sda2  vg01 lvm2 a-- 390.62g  0
/dev/sda5  vg1   lvm2 a-- 96.00m 44.00m
/dev/sda6  vg1   lvm2 a-- 96.00m 44.00m
/dev/sda7  vg1   lvm2 a-- 96.00m 92.00m
```

查看映射表：

```
[root@station21 ~]# dmsetup table |grep lv1
vg1-lv1_mlog: 0 8192 linear 8:7 2048
vg1-lv1: 0 106496 mirror disk 2 253:2 1024 2 253:6 0 253:7 0 1 handle_errors
vg1-lv1_mimage_1: 0 106496 linear 8:6 2048
vg1-lv1_mimage_0: 0 106496 linear 8:5 2048
```

```
[root@station21 ~]# dmsetup ls --tree
vg01-lv_root (253:1)
└─(8:2)
vg01-LogVol04 (253:0)
└─(8:2)
vg1-lv1 (253:8)
├─vg1-lv1_mimage_1 (253:7)
└─(8:6)
├─vg1-lv1_mimage_0 (253:6)
└─(8:5)
└─vg1-lv1_mlog (253:2)
└─(8:7)
vg01-LogVol03 (253:3)
└─(8:2)
vg01-lv_usr (253:5)
└─(8:2)
vg01-lv_var (253:4)
└─(8:2)

[root@station21 ~]# ll /dev/mapper/*
crw-rw--- 1 root root 10, 58 6月 6 08:24 /dev/mapper/control
lrwxrwxrwx 1 root root 7 6月 6 08:24 /dev/mapper/vg01-LogVol03 -> ../dm-3
lrwxrwxrwx 1 root root 7 6月 6 08:24 /dev/mapper/vg01-LogVol04 -> ../dm-0
lrwxrwxrwx 1 root root 7 6月 6 08:24 /dev/mapper/vg01-lv_root -> ../dm-1
lrwxrwxrwx 1 root root 7 6月 6 08:24 /dev/mapper/vg01-lv_usr -> ../dm-5
lrwxrwxrwx 1 root root 7 6月 6 08:24 /dev/mapper/vg01-lv_var -> ../dm-4
lrwxrwxrwx 1 root root 7 6月 6 09:54 /dev/mapper/vg1-lv1 -> ../dm-8
lrwxrwxrwx 1 root root 7 6月 6 09:54 /dev/mapper/vg1-lv1_mimage_0 -> ../dm-6
lrwxrwxrwx 1 root root 7 6月 6 09:54 /dev/mapper/vg1-lv1_mimage_1 -> ../dm-7
lrwxrwxrwx 1 root root 7 6月 6 09:54 /dev/mapper/vg1-lv1_mlog -> ../dm-2
```

测试读写：

终端一：

```
[root@station21 ~]# iostat 1 /dev/sda5 /dev/sda6 /dev/sda7
Device:    tps  Blk_read/s  Blk_wrtn/s  Blk_read  Blk_wrtn
sda5       210.00      0.00   53380.00      0     53380
sda6       248.00      0.00   63106.00      0     63106
sda7        4.00      0.00    12.00          0         12
```

终端二：

```
[root@station21 ~]# mkdir /mnt/lv1
[root@station21 ~]# mount /dev/mapper/vg1-lv1 /mnt/lv1/
[root@station21 ~]# dd if=/dev/zero of=/mnt/lv1/file1
```

```
[root@station21 ~]# while :; do dd if=/dev/zero >/mnt/lv1/file1 bs=1M count=1; usleep 30; done
```

给已存在的LV增加Mirror

```
[root@station21 ~]# lvcreate -L 10M -n lv2 vg1
  Rounding up size to full physical extent 12.00 MiB
  Logical volume "lv2" created
[root@station21 ~]# pvs
PV   VG   Fmt Attr PSize  PFree
/dev/sda2  vg01 lvm2 a-- 390.62g  0
/dev/sda5  vg1  lvm2 a-- 96.00m 32.00m
/dev/sda6  vg1  lvm2 a-- 96.00m 44.00m
/dev/sda7  vg1  lvm2 a-- 96.00m 92.00m
```

```
[root@station21 ~]# dmsetup table |grep lv2
vg1-lv2: 0 24576 linear 8:5 108544
```

```
[root@station21 ~]# lvconvert -m 1 /dev/vg1/lv2
vg1/lv2: Converted: 0.0%
vg1/lv2: Converted: 100.0%
```

```
[root@station21 ~]# lvs
  LV   VG Attr  LSize Pool Origin Data% Move Log      Cpy%Sync Convert
LogVol03 vg01 -wi-ao--- 78.12g
LogVol04 vg01 -wi-ao--- 2.00g
lv_root  vg01 -wi-ao--- 97.66g
lv_usr   vg01 -wi-ao--- 87.89g
lv_var   vg01 -wi-ao--- 124.95g
lv1     vg1 mwi-aom-- 52.00m          lv1_mlog 100.00
lv2     vg1 mwi-a-m-- 12.00m          lv2_mlog 100.00
```

```
[root@station21 ~]# dmsetup ls --tree
```

```
vg01-lv_root (253:1)
  └─(8:2)
vg1-lv2 (253:9)
  ├─vg1-lv2_mimage_1 (253:12)
  │  └─(8:6)
  ├─vg1-lv2_mimage_0 (253:11)
  │  └─(8:5)
  └─vg1-lv2_mlog (253:10)
    └─(8:7)
vg01-LogVol04 (253:0)
  └─(8:2)
vg1-lv1 (253:8)
  ├─vg1-lv1_mimage_1 (253:7)
  │  └─(8:6)
  ├─vg1-lv1_mimage_0 (253:6)
  │  └─(8:5)
  └─vg1-lv1_mlog (253:2)
    └─(8:7)
vg01-LogVol03 (253:3)
  └─(8:2)
vg01-lv_usr (253:5)
  └─(8:2)
vg01-lv_var (253:4)
  └─(8:2)
```

snapshot

snapshot

创建条带设备

```
[root@station21 ~]# ll /dev/sda*
brw-rw---- 1 root disk 8, 0 6月 5 16:24 /dev/sda
brw-rw---- 1 root disk 8, 1 6月 5 08:29 /dev/sda1
brw-rw---- 1 root disk 8, 2 6月 5 08:28 /dev/sda2
brw-rw---- 1 root disk 8, 3 6月 5 08:29 /dev/sda3
brw-rw---- 1 root disk 8, 4 6月 5 16:24 /dev/sda4
brw-rw---- 1 root disk 8, 5 6月 5 16:24 /dev/sda5
brw-rw---- 1 root disk 8, 6 6月 5 16:24 /dev/sda6
brw-rw---- 1 root disk 8, 7 6月 5 16:24 /dev/sda7
```

```
[root@station21 ~]# pvcreate /dev/sda{5,6,7}
Physical volume "/dev/sda5" successfully created
Physical volume "/dev/sda6" successfully created
Physical volume "/dev/sda7" successfully created
```

```
[root@station21 ~]# vgcreate vg1 /dev/sda{5,6,7}
Volume group "vg1" successfully created
```

```
[root@station21 ~]# pvs
PV      VG  Fmt Attr PSize  PFree
/dev/sda2  vg01 lvm2 a-- 390.62g    0
/dev/sda5  vg1   lvm2 a-- 96.00m 96.00m
/dev/sda6  vg1   lvm2 a-- 96.00m 96.00m
/dev/sda7  vg1   lvm2 a-- 96.00m 96.00m
```

```
[root@station21 ~]# lvcreate -L 50M -n lv1 vg1
Rounding up size to full physical extent 52.00 MiB
Logical volume "lv1" created
[root@station21 ~]# mkfs.ext4 /dev/mapper/vg1-lv1
```

```
[root@station21 ~]# mount /dev/vg1/lv1 /mnt/lv1
[root@station21 ~]# cp /etc/hosts /mnt/lv1/
```

```
[root@station21 ~]# dmsetup ls --tree
vg01-lv_root (253:1)
  └─(8:2)
vg01-LogVol04 (253:0)
  └─(8:2)
vg1-lv1 (253:2)
  └─(8:5)
vg01-LogVol03 (253:3)
  └─(8:2)
vg01-lv_usr (253:5)
  └─(8:2)
vg01-lv_var (253:4)
  └─(8:2)
```

```
[root@station21 ~]# vgs
  VG #PV #LV #SN Attr  VSize  VFree
  vg01  1  5  0 wz--n- 390.62g    0
  vg1   3  1  0 wz--n- 288.00m 236.00m
[root@station21 ~]# lvcreate -L 20M -s -n lv1-snap /dev/vg1/lv1
Logical volume "lv1-snap" created
```

查看映射表：

```
[root@station21 ~]# dmsetup table
vg1-lv1--snap-cow: 0 40960 linear 8:5 108544
vg01-lv_root: 0 204800000 linear 8:2 430073856
vg1-lv1-real: 0 106496 linear 8:5 2048
vg01-LogVol04: 0 4194304 linear 8:2 2048
vg1-lv1: 0 106496 snapshot-origin 253:7
vg01-LogVol03: 0 163840000 linear 8:2 4196352
vg01-lv_usr: 0 184320000 linear 8:2 634873856
vg01-lv_var: 0 262037504 linear 8:2 168036352
vg1-lv1--snap: 0 106496 snapshot 253:7 253:8 P 8
```

```
[root@station21 ~]# dmsetup ls --tree
vg01-lv_root (253:1)
└ (8:2)
vg01-LogVol04 (253:0)
└ (8:2)
vg1-lv1 (253:2)
└ vg1-lv1-real (253:7)
└ (8:5)
vg01-LogVol03 (253:3)
└ (8:2)
vg01-lv_usr (253:5)
└ (8:2)
vg01-lv_var (253:4)
└ (8:2)
vg1-lv1--snap (253:6)
└ vg1-lv1--snap-cow (253:8)
└ (8:5)
└ vg1-lv1-real (253:7)
└ (8:5)
```

pv前2048扇区

PV前2048扇区

2048 sector = 1M

LVM正常

```
[root@station21 ~]# vgcreate vg1 /dev/sda{5,6,7}
  Volume group "vg1" successfully created
[root@station21 ~]# vgs
  VG #PV #LV #SN Attr  VSize  VFree
  vg01  1  5  0 wz--n- 390.62g    0
  vg1   3   0  0 wz--n- 288.00m 288.00m

[root@station21 ~]# pvs
  PV   VG Fmt Attr PSize  PFree
  /dev/sda2  vg01 lvm2 a-- 390.62g    0
  /dev/sda5  vg1  lvm2 a-- 96.00m 96.00m
  /dev/sda6  vg1  lvm2 a-- 96.00m 96.00m
  /dev/sda7  vg1  lvm2 a-- 96.00m 96.00m

[root@station21 ~]# lvcreate -L 150M -n lv1 vg1
  Rounding up size to full physical extent 152.00 MiB
  Logical volume "lv1" created
[root@station21 ~]# pvs
  PV   VG Fmt Attr PSize  PFree
  /dev/sda2  vg01 lvm2 a-- 390.62g    0
  /dev/sda5  vg1  lvm2 a-- 96.00m    0
  /dev/sda6  vg1  lvm2 a-- 96.00m 40.00m
  /dev/sda7  vg1  lvm2 a-- 96.00m 96.00m

[root@station21 ~]# mkfs.ext4 /dev/vg1/lv1

[root@station21 ~]# mount /dev/vg1/lv1 /mnt/lv1/
[root@station21 ~]# cp -rf /etc/hosts /mnt/lv1/
```

模拟PV损坏

```
[root@station21 ~]# dd if=/dev/zero of=/dev/sda6 bs=1M count=1

[root@station21 ~]# vgs
Couldn't find device with uuid 13kSnw-1Pp0-9GZt-RC37-akSJ-qYQ0-i6uvjY.
VG #PV #LV #SN Attr VSize VFree
vg01 1 5 0 wz--n- 390.62g 0
vg1 3 1 0 wz-pn- 288.00m 136.00m

[root@station21 ~]# pvs
Couldn't find device with uuid 13kSnw-1Pp0-9GZt-RC37-akSJ-qYQ0-i6uvjY.
PV VG Fmt Attr PSize PFree
/dev/sda2 vg01 lvm2 a-- 390.62g 0
/dev/sda5 vg1 lvm2 a-- 96.00m 0
/dev/sda7 vg1 lvm2 a-- 96.00m 96.00m
unknown device vg1 lvm2 a-m 96.00m 40.00m

[root@station21 ~]# umount /mnt/lv1
[root@station21 ~]# vgreduce --removemissing --force vg1
Couldn't find device with uuid 13kSnw-1Pp0-9GZt-RC37-akSJ-qYQ0-i6uvjY.
Removing partial LV lv1.
Logical volume "lv1" successfully removed
Wrote out consistent volume group vg1

[root@station21 ~]# pvs
PV VG Fmt Attr PSize PFree
/dev/sda2 vg01 lvm2 a-- 390.62g 0
/dev/sda5 vg1 lvm2 a-- 96.00m 96.00m
/dev/sda7 vg1 lvm2 a-- 96.00m 96.00m
[root@station21 ~]# vgs
VG #PV #LV #SN Attr VSize VFree
vg01 1 5 0 wz--n- 390.62g 0
vg1 2 0 0 wz--n- 192.00m 192.00m
```

本节作业

1. SAN和NAS有什么区别？
2. IP SAN和FC SAN有什么区别？
3. ISCSI默认的端口是什么？
4. initiator端iscsid和iscsi进程的作用
5. Natapp NAS设备的配置方法
6. Tomcat使用memcache
7. Php增加memcache扩展
8. 配置rsync服务器

六、系统调优

01 Understand Performance Tuning

01 Understand Performance Tuning

调优的目的：

1. 根据不同的角色调优的方法是不一样的
2. 找到性能瓶颈以及缓解这个瓶颈（CPU，内存，IO调度、网络、使用的应用程序）
3. 通过性能管理实现合理的资源分配，以及提升硬件的性价比
4. 通常做两种调优：
response time: 响应时间 Web服务器，用户感受度好
throughput: 吞吐量 文件服务器，拷贝的速度

调优需要掌握的技能：

1. 必须了解硬件和软件
2. 能够把所有的性能、指标量化，用数字说话
3. 设置一个正常期待值，比如将响应速度调到1.5秒
(企业版操作系统在出厂时已经调优，适用于普遍的应用，再根据个人的环境进行微调)
4. 建议有一定的开发能力
5. 如果想要更好的调优，让调优有艺术性，需要更多的经验的积累，从而有一定洞察力，调节时所给参数才最恰当

性能调优的效率问题：

业务级调优 尽量在业务级调，效果最明显，例如：网站一定要使用Apache吗？

例如：将原有的调度器由LVS换成F5-BigIP

能否禁用一些不必要的服务如蓝牙、smart card，makewhatis, updatadb

应用级调优 NFS, Samba, Apache, Nginx/php-fpm, MySQL, Oracle, LVS本身调优

对于日志的处理rsyslog：只要有日志产生，就会存盘fsync()，可以调整记录的日志等级或延后日志写，从而

避免大量的I/O操作

kernel级调优 操作系统系统层面，即kernel调优具有普遍性：I/O CPU Network Memory，通常在系统初始完成

```
vm.dirty_expire_centisecs = 2
net.ipv4.tcp_tw_recycle = 1
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_synack_retries = 5
```

自上往下，效果越来越不明显

优化：

= = 调整系统/应用的性能参数

= = 合理资源分配 PAM, Cgroup(Docker)

= = 架构优化，Nginx负载集群、MySQL读写分离/MHA

02 Plotting Performance Graphs

02 Plotting Performance Graphs

绘制性能图表

一、相关计量单位：

SI标准： K M G 1000 HD factory

IEC标准： Ki Mi Gi 1024 OS Bandwidth

Data Storage Unit Terminology

Bytes	Decimal Unit	Abbreviation
10^3	kilobyte	KB
10^6	megabyte	MB
10^9	gigabyte	GB
10^{12}	terabyte	TB
10^{15}	petabyte	PB
10^{18}	exabyte	EB
10^{21}	zettabyte	ZB
10^{24}	yottabyte	YB

Bytes	Binary Unit	Abbreviation
2^{10}	kibibyte	KiB
2^{20}	mebibyte	MiB
2^{30}	gibibyte	GiB
2^{40}	tebibyte	TiB
2^{50}	pebibyte	PiB
2^{60}	exbibyte	EiB
2^{70}	zebibyte	ZiB
2^{80}	yobibyte	YiB

调优时，1k必须是 2^{10} Byte

调优对数字的要求必须是非常精准的，不能大概值

fdisk -cul 查看大小，按1000算
df -h 1024
df -H 1000

示例：将1Gbp/s带宽转换为M

二、collecting data 收集数据

基本工具

rpm -ql sysstat |grep bin
/usr/bin/cifsiostat

```
/usr/bin/iostat  
/usr/bin/mpstat  
/usr/bin/pidstat  
/usr/bin/sadf  
/usr/bin/sar  
# rpm -qf `which vmstat`  
procps-3.2.8-25.el6.x86_64
```

a. vmstat 主要看内存使用的情况

```
# vmstat  
# vmstat 1  
procs -----memory----- --swap-- ----io---- --system-- ----cpu----  
r b swpd free buff cache si so bi bo in cs us sy id wa st  
0 0 0 1526080 61888 605892 0 0 51 25 126 978 3 2 95 1 0
```

procs

r CPU队列长度
dd if=/dev/zero of=/dev/null
md5sum /dev/zero

b 处于不可中断睡眠的进程数
dd if=/dev/sda of=/dev/null

memory

free
buffer cache 拿文件系统来举例：元数据inode ls -R /; find /
cache page cache 拿文件系统来举例：实际数据block cp -rf /usr /tmp/

swap

si 每秒从交换分区（磁盘）交换到内存中的量
so 每秒从内存交换到交换分区（磁盘）的量

io
bi 从磁盘读数据 blocks/s
bo 写数据到磁盘 blocks/s
echo 3 > /proc/sys/vm/drop_caches
dd if=/dev/zero of=/file

system

in 每秒中断请求数
cs 每秒上下文切换的次数

cpu
us CPU在用户层所花的时间 (user time 例如处理代码的流程，代码问题)
sy CPU在系统层所花的时间 (system time 例如kernel运算处理，算法问题)
id 空闲百分比
wa 等待时间(等I/O)
st stolen被偷走的时间

Virtual memory (VIRT) is all memory the process is using, including the resident set, shared libraries, and any mapped or swapped memory pages.
(Labeled VSZ in the ps command.)

Resident memory (RES) is the physical memory used by the process, including any resident shared objects. (Labeled RSS in the ps command.)

b. iostat 用来衡量I/O的状态

```
# iostat  
# iostat -x /dev/sda 1 // -x详细  
# dd if=/dev/sda of=/dev/null // 制造负载  
# iostat -x sda 1  
rrqm/s 队列中，每秒读请求的合并数量  
wrqm/s 队列中，每秒写请求的合并数量  
r/s 每秒读请求数  
w/s 每秒写请求数  
rsec/s 每秒读扇区数 除以2，换算成k  
wsec/s 每秒写扇区数 dd if=/dev/zero of=/file2  
avgrq-sz 平均每次提交请求request的大小值，可以衡量合并的成功率，越大效率越高
```

```

avgqu-sz 平均请求的队列长度 每秒钟等待硬盘服务的有多少个请求，排队长度
await 平均响应时间，能反应出磁盘的效率
svctm
%util 磁盘的利用率 dd if=/dev/zero of=/file20 bs=1M

```

常见用法

```

iostat -d -k 1 10      #查看TPS和吞吐量信息(磁盘读写速度单位为KB)
iostat -d -m 2      #查看TPS和吞吐量信息(磁盘读写速度单位为MB)
iostat -d -x -k 1 10    #查看设备使用率 (%util) 、响应时间 (await)
iostat -c 1 10      #查看cpu状态

```

实例分析

```

iostat -d -k 1 |grep sda10
Device:      tps   kB_read/s   kB_wrtn/s   kB_read   kB_wrtn
sda10        60.72     18.95       71.53  395637647 1493241908
sda10        299.02    4266.67      129.41      4352       132
sda10        483.84    4589.90      4117.17      4544      4076
sda10        218.00    3360.00      100.00      3360       100
sda10        546.00    8784.00      124.00      8784      124
sda10        827.00   13232.00      136.00     13232      136

```

上面看到，磁盘每秒传输次数平均约400；每秒磁盘读取约5MB，写入约1MB。

```

iostat -d -x -k 1
Device: rrqm/s wrqm/s r/s w/s rsec/s wsec/s rkB/s wkB/s avgrq-sz avgqu-sz await svctm %util
sda      1.56 28.31 7.84 31.50 43.65 3.16 21.82 1.58 1.19 0.03 0.80 2.61 10.29
sda      1.98 24.75 419.80 6.93 13465.35 253.47 6732.67 126.73 32.15 2.00 4.70 2.00 85.25
sda      3.06 41.84 444.90 54.08 14204.08 2048.98 7102.04 1024.49 32.57 2.10 4.21 1.85 92.24

```

可以看到磁盘的平均响应时间<5ms，磁盘使用率>80。磁盘响应正常，但是已经很繁忙了。

```

# iostat -x 1 10
Device: rrqm/s wrqm/s r/s w/s rsec/s wsec/s avgrq-sz avgqu-sz await svctm %util
sda    0.00    3.50   0.40   2.50   5.60    48.00   18.48    0.00    0.97    0.97   0.28
sdb    0.00    0.00   0.00   0.00   0.00    0.00    0.00    0.00    0.00    0.00   0.00
sdc    0.00    0.00   0.00   0.00   0.00    0.00    0.00    0.00    0.00    0.00   0.00
sdd    0.00    0.00   0.00   0.00   0.00    0.00    0.00    0.00    0.00    0.00   0.00
sde    0.00    0.10   0.30   0.20   2.40    2.40    9.60    0.00    1.60    1.60   0.08
sdf    17.40   0.50 102.00  0.20 12095.20  5.60   118.40    0.70    6.81    2.09  21.36
sdg   232.40  1.90 379.70  0.50 76451.20 19.20   201.13    4.94   13.78   2.45  93.16

```

c. mpstat 返回CPU的详细信息

```

# taskset -c 1 dd if=/dev/zero of=/dev/null
# dd if=/dev/zero of=/dev/null          //CPU负载型
# dd if=/dev/zero of=/file10 bs=1M      //IO负载型

```

```

# mpstat 1
# mpstat -P 1 1
%iowait      io等待时间
%irq         硬中断花了CPU多少时间
%soft         软中断花了CPU多少时间
%steal        跟虚拟化相关
%guest        跟虚拟化相关

```

```
%nice      有多少时间是跑在降级模式下的，普通用户只有降优先级  
%idle     空闲百分比
```

```
d. netstat--> ss  
# ss -atnp |grep 80
```

高级的工具sar

对系统的压力非常小，**是基于历史的**

工作原理：

```
# cat /etc/cron.d/sysstat  
# Run system activity accounting tool every 10 minutes  
*/10 * * * * root /usr/lib64/sa/sa1 1 1  
# 0 * * * * root /usr/lib64/sa/sa1 600 6 &  
# Generate a daily summary of process accounting at 23:53  
53 23 * * * root /usr/lib64/sa/sa2 -A
```

每隔10分钟，用sa1从系统中抓取状态

sar查看历史的数据：`/var/log/sa`

```
# cat /etc/sysconfig/sysstat    默认存储份数  
# sar -f /var/log/sa/sa16      查看指定时间的报告
```

sar命令执行时，**是基于历史的**，当天，从开机到最后一次收集数据

```
sar          取历史  
mpstat      取现状
```

sar -q 看系统负载 Report queue length and load averages.

```
runq-sz      运行队列的长度  
plist-sz     进程的数量  
ldavg-1      1分钟内的负载  
ldavg-5      5分钟内的负载  
ldavg-15     15分钟内的负载
```

sar -b 报告I/O状况 Report I/O and transfer rate statistics.

```
tps          每秒传输的数量  
rtps         读的数量  
wtps         写的数量  
bread/s     每秒读多少个block  
bwrttn/s    每秒写多少个block  
man sar     每个block 512 bytes
```

sar 报告CPU的状态 Report per-processor statistics for the specified processor or processors.
sar -P 3

sar -n DEV 网络相关的 Report network statistics.

```
Possible keywords are DEV, EDEV, NFS, NFSD, SOCK, IP, EIP, ICMP, EICMP, TCP, ETCP,  
UDP, SOCK6, IP6, EIP6, ICMP6, EICMP6 and UDP6.
```

```
rpxck/s     接收多少个包  
txpck/s     发送多少个包  
rxkB/s      接收多少kB  
txkB/s      发送多少kB  
rxcmp/s     接收压缩的包  
txcmp/s     发送压缩的包  
rxmcst/s    接收的广播包
```

sar -n EDEV 报告错误的包

```
rxerr/s      接收错误  
txerr/s      发送错误  
coll/s       有多少个碰撞，冲突的包  
rxdrop/s    接收丢包  
txdrop/s    发送丢包  
txcarr/s    网卡产生的错误  
rxfram/s    帧产生的错误  
rxfifo/s    rxfifo/s
```

三、使用gnuplot绘图,rrdtool,zabbix

Formatting Data: Awk 格式化数据

sar -q 获得CPU负载的数据

确保格式化出来的数据文件中只有合法的数据，不包括空行和带英文字母的行： /tmp/cpudata

sar -b 获得磁盘读写的数据

注需要将读bread/s和写bwrttn/s相加

sar -b | awk '{print \$1,(\$5+\$6)/2}' 扇区数除以2，得到字节数: /tmp/diskdata

格式化数据示例：

Formatting Data for Load Average

LANG=C

LANG=C sar -q |awk '/^[^a-zA-Z]+\$/ {print \$1,\$4,\$5,\$6}'

LANG=C sar -q |awk '/^[^a-zA-Z]+\$/ {print \$1,\$4,\$5,\$6}' > /tmp/load.sar

Formatting Data for Disk I/O

LANG=C sar -b |awk '/^[^a-zA-Z]+\$/ {print \$1,(\$5+\$6)/2}'

LANG=C sar -b |awk '/^[^a-zA-Z]+\$/ {print \$1,(\$5+\$6)/2}' > /tmp/disk.sar

gnuplot绘图

[root@localhost ~]# yum -y install gnuplot

[root@localhost ~]# gnuplot

gnuplot> set xdata time

gnuplot> set grid

gnuplot> set timefmt "%H:%M:%S"

gnuplot> plot "/tmp/load.sar" using 1:2

gnuplot> plot "/tmp/load.sar" using 1:2 with lines

gnuplot> plot "/tmp/load.sar" using 1:2 title "1 min" with lines

gnuplot> plot "/tmp/load.sar" using 1:2 title "1 min" with lines, "/tmp/load.sar" using 1:3 title "5 min" with lines, "/tmp/load.sar" using 1:4 title "15 min" with lines

[root@localhost ~]# vim /tmp/load.gnuplot

set xdata time

set grid

set timefmt "%H:%M:%S"

plot "/tmp/load.sar" using 1:2 title "1 min" with lines, "/tmp/load.sar" using 1:3 title "5 min" with lines, "/tmp/load.sar" using 1:4 title "15 min" with lines

[root@localhost ~]# gnuplot -persist /tmp/load.gnuplot

Case 1:

1. 采集数据

system I/O very busy 13:00 - 17:00

#LANG=C sar -b -s 13:00:00 -e 17:00:00

2. Formatting data

LANG=C sar -b -s 13:00:00 -e 17:00:00 |awk '/^[^a-zA-Z]+\$/ {print \$1,(\$5+\$6)/2}' > /disk1.data

3. gnuplot

[root@localhost ~]# vim /diskio.gnuplot

set xdata time

set grid

set timefmt "%H:%M:%S"

set term png size 1024,768

set output "/var/www/html/sa/`date +%F`_diskIO.png"

plot "/disk1.data" using 1:2 with lines

[root@localhost ~]# gnuplot /diskio.gnuplot

=====

cron:

yum -y install httpd

service httpd start

chkconfig httpd on

```

# mkdir /var/www/html/sa/
# iptables -F
# service iptables save

# vim /diskio.sh      //采集数据脚本
LANG=C sar -b -s 13:00:00 -e 17:00:00 |awk '/^[^a-zA-Z]+$/{print $1,($5+$6)/2}' > /disk1.data
# chmod a+x /diskio.sh

# vim /diskio.gnuplot //定义绘图的方式文件
set grid
set xlabel "time"
set ylabel "DiskIO"
set xdata time
set timefmt "%H:%M:%S"
set terminal png size 1024,768
set output "/var/www/html/sa/~date +%F`_diskIO.png"
plot "/disk1.data" using 1:2 title "`date +%F`_diskIO" with lines

```

```

[root@localhost ~]# crontab -e
30 23 * * * /diskio.sh
40 23 * * * gnuplot /diskio.gnuplot
=====
```

```

[root@tianyun ~]# ss -atnp |grep 80
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.63:47801
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.104:37785
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.63:47799
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.199:37343
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.65:60547
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.65:60550
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.199:37338
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.199:37351
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.199:37348
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.199:37340
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.189:60229
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.199:37357
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.189:60227
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.199:37342
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.26:35775
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.199:37345
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.199:37360
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.199:37362
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.189:60228
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.199:37354
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.199:37350
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.65:60545
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.199:37347
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.199:37344
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.199:37352
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.237:52672
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.199:37349
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.65:60546
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.199:37359
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.199:37363
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.65:60549
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.65:60551
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.63:47798
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.199:37355
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.199:37341
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.199:37361
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.189:60226
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.199:37358
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.26:35776
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.26:35777
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.63:47800
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.199:37339
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.199:37356
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.65:60548

```

```
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.199:37346
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.26:35774
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.65:60544
TIME-WAIT 0 0 ::ffff:172.16.30.79:80 ::ffff:172.16.30.199:37353
```

03 Queueing Theory

队列理论

Little's Law $L = \lambda \cdot W$

Queue length = average arrival rate \times average wait time

Wait Time $W = S + Q$

Wait time = service time + queue time

$$L = \lambda \cdot (S + Q)$$

利特尔法则

Little's law (Little's result, theorem, lemma or formula)

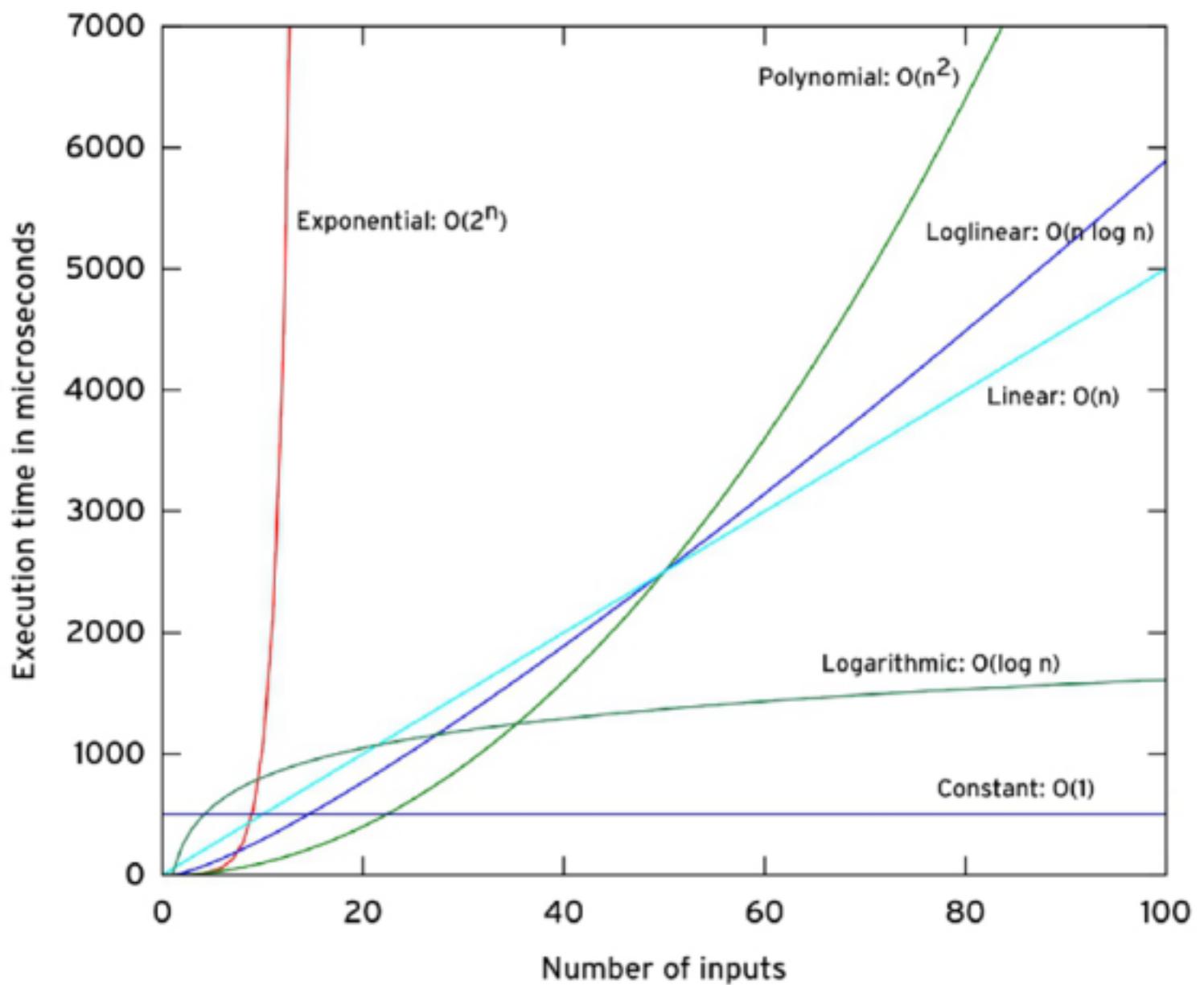
描述：在一个稳定的系统中，长时间观察到的平均顾客数量 L ，等于，长时间观察到的有效到达速率 λ 与平均每个顾客在系统中花费的时间之乘积，即 $L = \lambda W$ 。

假定我们所开发的并发服务器，并发的访问速率是：1000客户/分钟，每个客户在该服务器上将花费平均0.5分钟，根据little's law规则，在任何时刻，服务器将承担 $1000 \times 0.5 = 500$ 个客户量的业务处理。假定过了一段时间，由于客户群的增大，并发的访问速率提升为2000客户/分钟。在这样的情况下，我们该如何改进我们系统的性能？根据little's law规则，有两种方案：

第一：提高服务器并发处理的业务量，即提高到 $2000 \times 0.5 = 1000$

或者

第二：减少服务器平均处理客户请求的时间，即减少到： $500 / 2000 = 0.25$



04 Using Tuned

```
sysctl/  
00-INDEX abi.txt ctl_unnumbered.txt fs.txt kernel.txt
```

04 Using Tuned

```
=====
```

例如：在生产之中，随时随地去换成不同的角色，因为服务器角色可能在一天之中会发生变化

白天启用高性能，晚上启用低性能

```
[root@tianyun ~]# man proc      /drop_cache
```

```
[root@tianyun ~]# ls /usr/share/doc/kernel-doc-2.6.32/Documentation/sysctl/  
00-INDEX abi.txt ctl_unnumbered.txt fs.txt kernel.txt net.txt README sunrpc.txt vm.txt
```

```
[root@tianyun ~]# grep -R --color "_reuse" /usr/share/doc/kernel-doc-2.6.32/Documentation/  
/usr/share/doc/kernel-doc-2.6.32/Documentation/networking/ip-sysctl.txt:tcp_tw_reuse - BOOLEAN
```

```
[root@tianyun ~]# grep -R --color "ip_forward" /usr/share/doc/kernel-doc-2.6.32/Documentation/  
/usr/share/doc/kernel-doc-2.6.32/Documentation/networking/ip-sysctl.txt:ip_forward - BOOLEAN
```

```
[root@tianyun ~]# grep -R --color "drop_cache" /usr/share/doc/kernel-doc-2.6.32/Documentation/  
/usr/share/doc/kernel-doc-2.6.32/Documentation/sysctl/vm.txt:drop_caches
```

Using Tuned

基于以下几个目的：

低消耗

低延迟

高吞吐

可以设置自己的Tuned配置，例如针对数据库服务器、针对WEB服务器设计

安装并启动

```
# yum -y install tuned  
# chkconfig tuned on  
# chkconfig ktune on  
# service tuned start  
# service ktune start
```

注：可以tuned-adm 切换任意一个profile的时候自动启动

使用tuned

```
# tuned-adm active      查看当前tuned的状态，包括是否开启及当前使用的profile  
# tuned-adm list       列出已存在的 profile  
# tuned-adm profile server-powersave  
# tuned-adm profile default
```

系统中自带的tuned Profiles:

(红帽已经做了大量的调优)

```
# tuned-adm list  
Available profiles:  
- desktop-powersave  
- latency-performance  
- virtual-host  
- default  
- laptop-battery-powersave  
- server-powersave  
- throughput-performance  
- enterprise-storage  
- spindown-disk  
- laptop-ac-powersave  
- virtual-guest  
Current active profile: default
```

```
# cd /etc/tune-profiles  
# ls  
# cd throughput-performance
```

自定Tuned配置文件：

Case Study : noop (virtual machine or SSD disk)

```
# cd /etc/tune-profiles
# cp -a throughput-performance vm_noop
# cd vm_noop
# vim ktune.sysconfig
  ELEVATOR="noop"
# tuned-adm profile vm_noop
# cat /sys/block/vda/queue/scheduler
```

Case Study : (web server 文件打开数设置到6M)

```
# echo "6*2^20" |bc
6291456

# echo $[6*2**20]
6291456

# cd /etc/tune-profiles
# cp -a throughput-performance webserver
# cd webserver
# vim sysctl.ktune
  fs.file-max = 6291456
# vim ktune.sysconfig
  ELEVATOR="deadline"
# tuned-adm profile webserver
# cat /sys/block/sda/queue/scheduler
# sysctl -a |grep file-max
```

crond:

```
# crontab -e
00 08 * * * tuned-adm profile web-server
30 19 * * * tuned-adm profile default
```

05 Limit Resource Usage

05 Limit Resource Usage

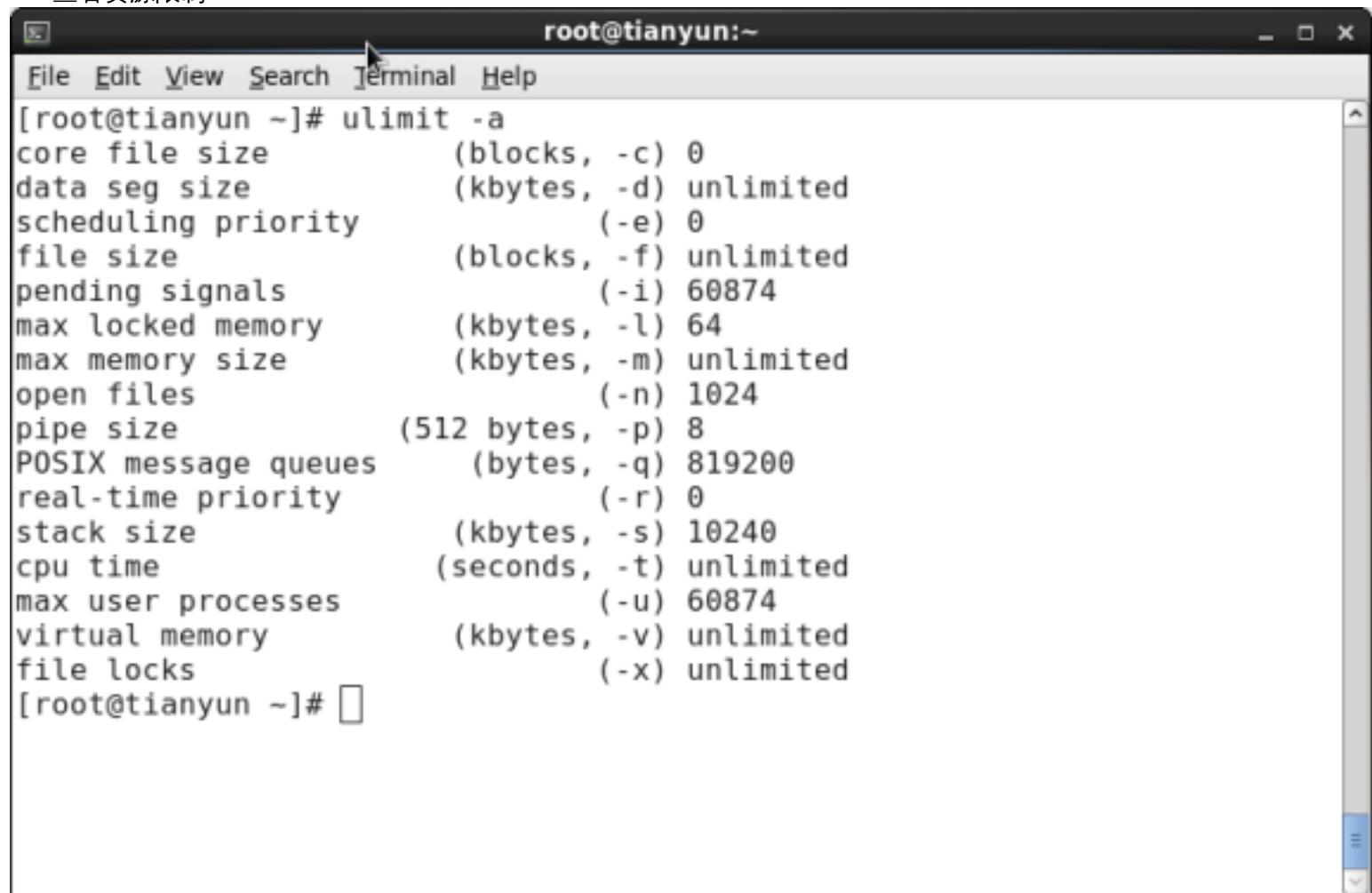
RHEL5: POSIX Resource limits PAM
RHEL6: POSIX Resource limits PAM, cgroup

一、传统：POSIX PAM

/etc/pam.d/system-auth
session required pam_limits.so +++++/etc/security/limits.conf
 +++++/etc/security/limits.d/*.conf

基于**用户**资源限制，但有很多程序都是超户在运行

==查看资源限制==



```
root@tianyun:~
```

```
File Edit View Search Terminal Help
[root@tianyun ~]# ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority     (-e) 0
file size               (blocks, -f) unlimited
pending signals          (-i) 60874
max locked memory       (kbytes, -l) 64
max memory size         (kbytes, -m) unlimited
open files              (-n) 1024
pipe size               (512 bytes, -p) 8
POSIX message queues    (bytes, -q) 819200
real-time priority      (-r) 0
stack size              (kbytes, -s) 10240
cpu time                (seconds, -t) unlimited
max user processes       (-u) 60874
virtual memory           (kbytes, -v) unlimited
file locks               (-x) unlimited
[root@tianyun ~]# 
```

LAB1：限制进程最大打开的文件数

```
[root@tianyun ~]# vim /etc/security/limits.conf
alice    hard  fsize   10240
```

```
[alice@tianyun ~]$ ulimit -f
10240
[alice@tianyun ~]$ ulimit -f -S
10240
```

```
[alice@tianyun ~]$ ulimit -f -H  
10240
```

```
[alice@tianyun ~]$ dd if=/dev/zero of=file1 bs=1M count=9  
9+0 records in  
9+0 records out  
9437184 bytes (9.4 MB) copied, 0.00569712 s, 1.7 GB/s
```

```
[alice@tianyun ~]$ dd if=/dev/zero of=file1 bs=1M count=11  
File size limit exceeded (core dumped)
```

```
[alice@tianyun ~]$ ulimit -f 102400  
-bash: ulimit: file size: cannot modify limit: Operation not permitted
```

LAB2：限制用户使用CPU的时间

该用户的单个进程最多占CPU长时间，单位MIN
jack hard cpu 1

```
[alice@tianyun ~]$ md5sum /dev/zero
```

如果用户开启100个这样的进程呢？？？

```
[alice@tianyun ~]$ for i in {1..100}  
> do  
> md5sum /dev/zero &  
> done
```

LAB3：限制用户最大运行的进程数，防shell炸弹

```
# vim bomb.sh  
bomb() {  
    bomb|bomb&  
}  
bomb
```

一行简写：

```
bomb() { bomb|bomb& }; bomb  
:(){ :;& };:
```

每次管道都会产生一个新shell

```
/etc/security/limits.d/90-nproc.conf  
* soft nproc 1024  
root soft nproc unlimited  
jack hard nproc 100
```

LAB4：限制用户使用内存

rss 驻留内存 RES
as 虚存地址，包括驻留内存，共享内存，已交换的内存 VIRT

PAM目前只能限制as，即虚存，不能限制实时内存rss

```
# vim /etc/security/limits.conf  
jack hard as 262144 //单位KB， 256M*2^10
```

```
# su - jack  
$ ulimit -a  
virtual memory 262144  
$ bigmem -v 200  
$ bigmem -v 300
```

结论：PAM limit限制

针对用户

不能控制磁盘 I/O

限制CPU和内存手段过于简单，没有办法再细化

通常用于限制进程打开的最大进程数，打开的最大文件数

```
# ulimit -n 102400
```

二、cgroups: Control Groups

基于进程的限制，而非用户，因此对于超户运行的进程也是一样

cgroup将各种子系统定义为资源，命名为controller:

可配额/可度量 - Control Groups (cgroups)

cgroups实现了对资源的配额和度量九大子系统的资源

1. blkio 限制每个块设备的输入输出控制。例如:磁盘,光盘以及usb
2. cpu 限制使用cpu比例
3. cputacct 产生cgroup任务的cpu资源报告。
4. cpuset 多核心的cpu时为cgroup任务分配单独的cpu和内存
5. devices 允许或拒绝对设备的访问。
6. freezer 暂停和恢复cgroup任务。
7. memory 设置内存限制以及产生内存资源报告。
8. net_cls 标记每个网络包。
9. ns 名称空间子系统

例如对某个进程使用内存进行限制，则需要在controller **memory**下建立cgroup
将进程分配到某个controller的控制组进行控制，没有使用controller则不会限制

cgroup安装

```
# yum -y install libcgconfig
# service cgconfig restart
# chkconfig cgconfig on

# vim /etc/cgconfig.conf          //主配置文件
# ls /cgroup                      //各种controller资源限制定义
# lssubsys -m                      //查看已mount的controller
# man cgconfig.conf
```

cgroup配置步骤：

1. 创建cgroup，定义相应的限制
2. 分配程序到cgroup

==To Control CPU Usage==

1. 使用cpu子系统创建两个cgroup

```
# cat /cgroup/cpu/cpu.shares
1024                                //1024表示能分配到的时间片为100%

# vim /etc/cgconfig.conf
group lesscpu {
    cpu{
        cpu.shares=200;
    }
}
group morecpu {
    cpu{
        cpu.shares=800;
    }
}
# service cgconfig reload
```

2. 将程序分配到相应的group

实验中，为了让两个进程抢CPU时间片，故意只留一个CPU在线

```
# lscpu
# echo 0 > /sys/devices/system/cpu/cpu0/online
# echo 0 > /sys/devices/system/cpu/cpu1/online
# lscpu
手动分配：
# cgexec -g cpu:lesscpu dd if=/dev/zero of=/dev/null
# cgexec -g cpu:morecpu md5sum /dev/zero
# top
```

==To Control Memory Usage==

```
# vim /etc/cgconfig.conf
group poormem {
    memory{
```

```

        memory.limit_in_bytes=268435465;           //物理内存限制256M
    }
}

# service cgconfig reload

//建立内存盘
# mkdir /mnt/mem_test
# mount -t tmpfs /dev/shm /mnt/mem_test
# cgexec -g memory:poormem dd if=/dev/zero of=/mnt/mem_test/file bs=1M count=200 OK
# cgexec -g memory:poormem dd if=/dev/zero of=/mnt/mem_test/file bs=1M count=500 OK
# free -m

# vim /etc/cgconfig.conf
group poormem{
    memory{
        memory.limit_in_bytes=268435465;           //物理内存限制256M
        memory.memsw.limit_in_bytes=268435465; //总内存限制，物理+SWAP
    }
}
# service cgconfig reload

# cgexec -g memory:poormem dd if=/dev/zero of=/mnt/mem_test/file bs=1M count=200 OK
# cgexec -g memory:poormem dd if=/dev/zero of=/mnt/mem_test/file bs=1M count=500 Fail
# free -m

```

==To Control Disk I/O Usage==

```

# 先看父的总值
# cat /cgroup/blkio/blkio.weight
1000

# vim /etc/cgconfig.conf
group lowio {
    blkio{
        blkio.weight=200;
    }
}

group highio {
    blkio{
        blkio.weight=800;
    }
}
# service cgconfig reload

```

准备两个大文件

```

# dd if=/dev/zero of=/file1 bs=1G count=5
# dd if=/dev/zero of=/file2 bs=1G count=5

```

注：blkio.weight works under cfq scheduler 只有cfq调度算法支持cgroup的权重

分配进程到控制组

```

# vim /etc/cgrules.conf
#<user>:<process name>      <controllers>      <destination>
*:md5sum                      blkio                  lowio/
*:sha1sum                      blkio                  highio/
*: /usr/sbin/httpd              blkio,cpu            highio/morecpu/
# service cgred start
# chkconfig cgred on
# service cgred reload

```

```

[root@tianyun ~]# echo 3 > /proc/sys/vm/drop_caches
[root@tianyun ~]# md5sum /file1
[root@tianyun ~]# sha1sum /file2
[root@tianyun ~]# iotop

```

执行多个进程对比

```
==控制器freezer==  
group stop_process {  
    freezer {  
    }  
}  
[root@yangy ~]# service cgconfig reload  
[root@yangy ~]# echo 22759 > /cgroup/freezer/stop_process/tasks  
[root@yangy ~]# echo FROZEN > /cgroup/freezer/stopitstop_process/freezer.state  
[root@yangy ~]# echo THAWED > /cgroup/freezer/stopitstop_process/freezer.state
```

练习：

设置某一个进程如dd在执行时，一定会占用较高的CPU，Mem，IO资源

06 Getting to know your hardware

06 Getting to know your hardware

获得硬件信息

硬件方面，主要要调优的对象

CPU
Memory
Storage
Networking

基本知识：所有的存储类的设备

离CPU越近会越快，离CPU越远将越慢

离CPU越近存储的容量越小，离CPU越远存储的容量越大

离CPU最近的是寄存器，寄存器的时钟周期和CPU是一样的

离CPU稍远的存储：

L1，在CPU中，使用静态内存做的，工艺也非常复杂，比寄存器要慢

L2，使用的是动态高速内存

L3

L4

Memory

Storage TB, PB

CPU的相关指标：

# cat /proc/cpuinfo	物理的，核心的，超线程的？
# lscpu	RHEL6支持
L1d cache	一级数据缓存
L1i cache	一级指令缓存
L2	二级缓存 (L2是否共享?)
Thread(s) per core:	线程数为1，不支持超线程
Core(s) per socket:	CPU核数

RHEL5/6

x86info -c
getconf -a

查看CPU缓存

```
# lscpu -p
# The following is the parsable format, which can be fed to other
# programs. Each different item in every column has an unique ID
# starting from zero.
# CPU,Core,Socket,Node,,L1d,L1i,L2,L3
0,0,0,0,,0,0,0,0
1,0,0,0,,0,0,0,0
2,1,0,0,,1,1,1,0
3,1,0,0,,1,1,1,0
```

CPU类型，HZ : 仅是影响性能的指标之一

CPU Cache: 各级缓存

CPU features : CPU的功能，架构 (x86, x86_64, Power, ARM)

主板芯片组： 跟CPU配合，也决定跟CPU通信的效率问题

物理CPU之间通信的手段： FSB , QPI(Intel), HyperTransport(AMD)

a. FSB : 前端系统总线，普通PC，笔记本

b. QPI , HyperTransport : 服务器级，服务器CPU一般不至一颗，例如4 * 8(Cores)

Memory的相关指标：

```
# cat /proc/meminfo
# free
```

Memory size:

内存效率重要指标：

a. bandwidth: 带宽，使用的是DDR几代；例如PC2100，指2100M/s的吞吐量，即带宽
b. latency: 延迟，纳秒级ns

根据CPU访问内存的方法：

a. UMA : 一致性内存访问，传统架构

MCH: 内存控制芯片，即北桥

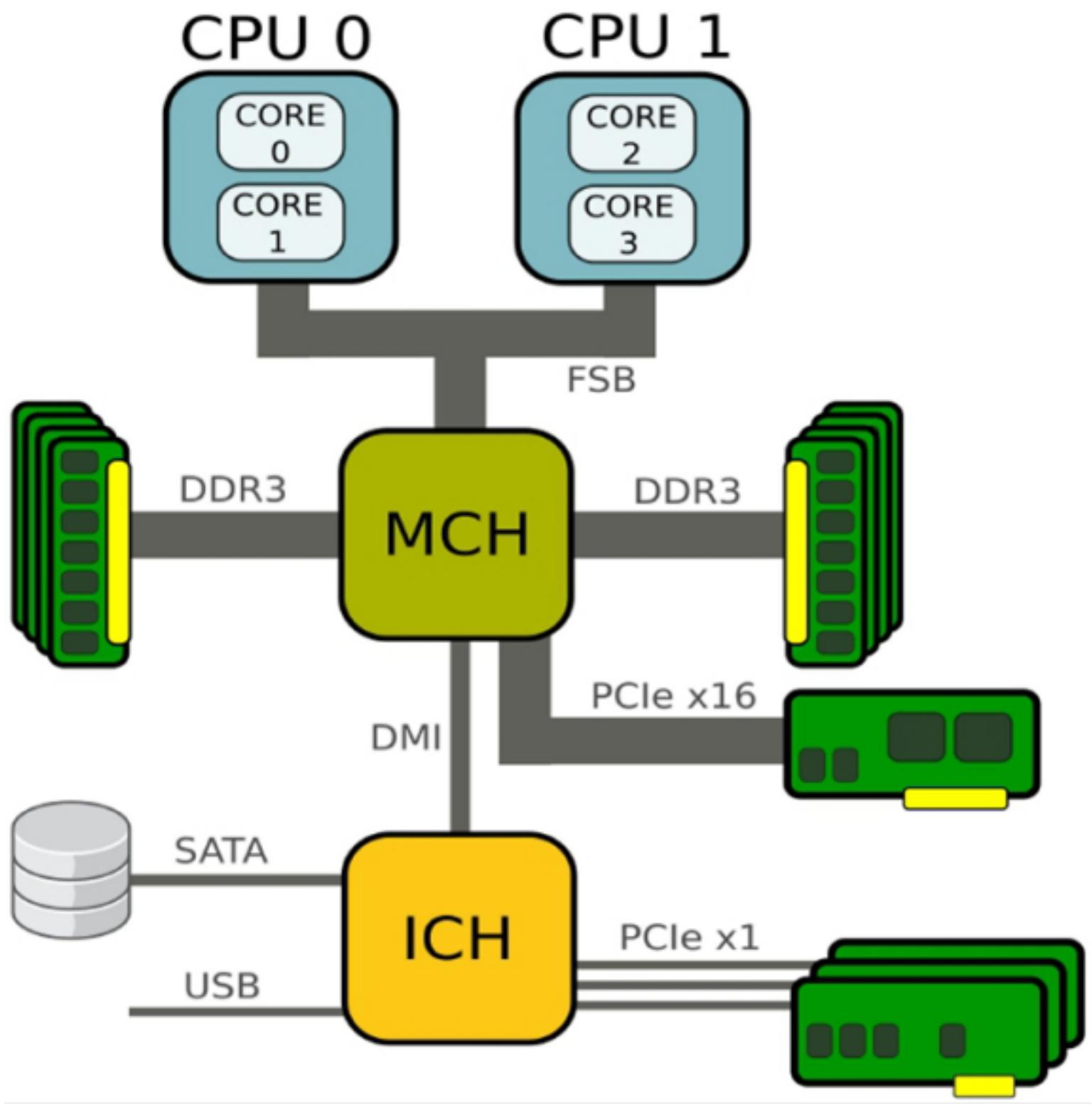
ICH : I/O控制芯片，即南桥，连接外设，硬盘，USB，慢速PCI总线设备

b. NUMA : 非一致性内存访问（内存分为多片，而不是UMA时的一片内存）

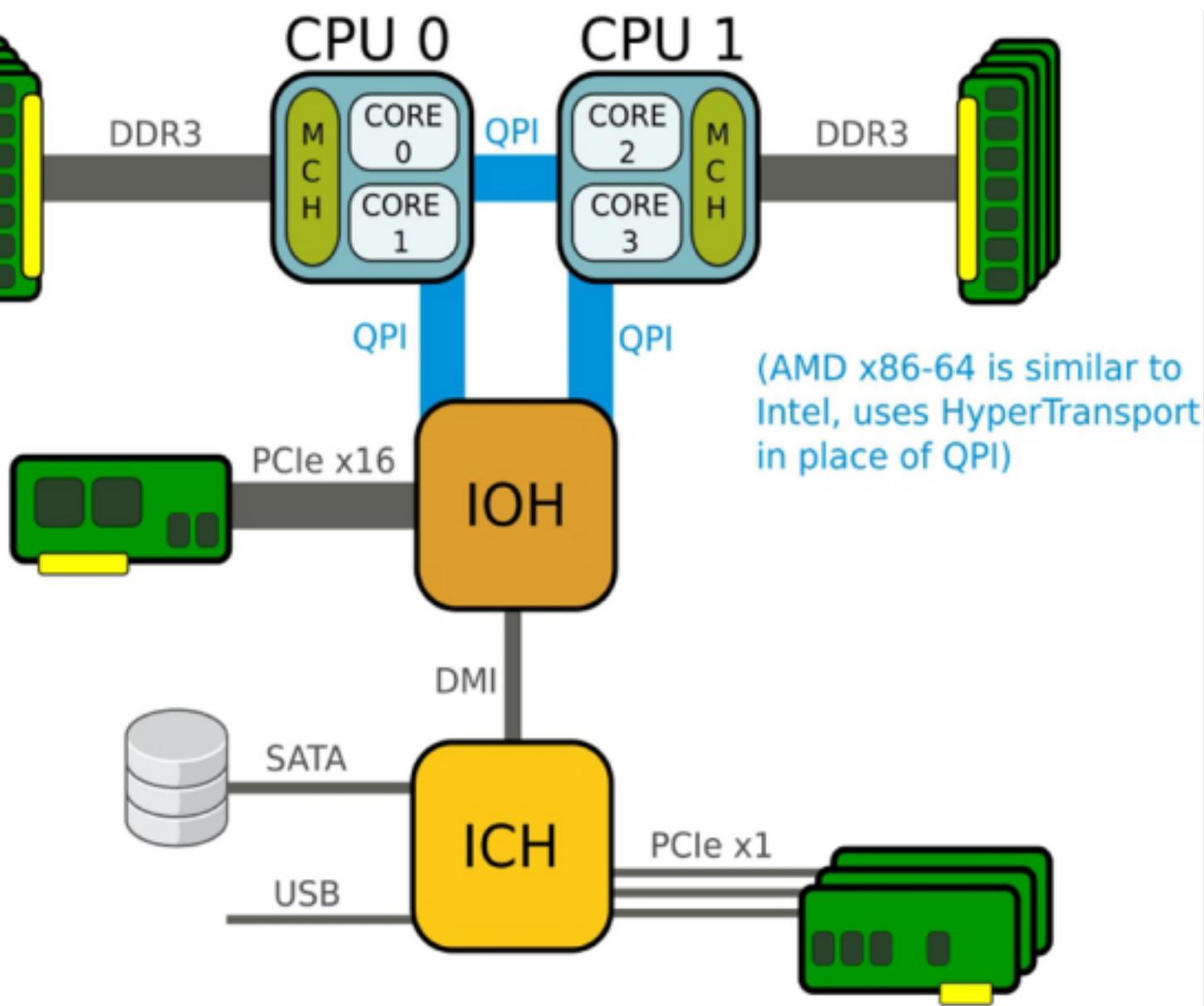
IOH : I/O HUB，也可以叫北桥

NUMA 架构显著的特色是：内存划分成片，CPU访问本区的内存的时候，速度非常快

UMA or SMP



NUMA



存储的相关性能指标：

主要的调优对像，对内存而言，特别慢

拿16G的内存存数据！

拿16G的硬盘存数据！

当前两种磁盘类型：

机械磁盘： 盘片

SSD： 固态磁盘

机械磁盘：

昂贵的寻道时间 Expensive seek time

主轴转速： 7200RPM 10000RPM(10k) 15000RPM(15k)

缺点：容易坏

优点：存储量非常高，价格低

Burst speed: 突发速率，最快读写速率，指的是顺序读写 Sequential access

average speed: 平均读写速率，生产中不能保证是顺序读写 Random access

调优：进行大量的优化合并

SSD：

Electronic disk: 电子磁盘，没有机械部件

No start-up time，读和写延迟非常低

安静，发热量小，不怕震动，而且很轻，用电少

价格高：在同等容量下比较

寿命短

连接的方式：

内部存储：**IDE (PATA)** , SATA , SCSI , **SAS** , FC

外部存储：SCSI总线连接, Fibre Channel, iSCSI, 网络带宽，延迟，多路径都会影响到外部存储的速度

4k/sector support : 4k对齐，效率更高 (默认扇区大小512B)

外圈快，内圈慢：

磁盘转一圈，外圈读的扇区数多，所以数据写读时外圈快，内圈慢

RHEL安装时，默认将`/`, `/boot`分区扔到了最快圈，交换区自动往内圈扔（它认为swap不需要经常访问）

Networking Profile

带宽

延迟

尽量把不同的网络隔离开，让其处于不同的广播域，划分VLAN

使用bonding或Team技术实现多网卡绑定，从而实现HA或LB（提高带宽）

高端网卡支持虚拟化SR-IOV

高端网卡有自己的处理芯片，网络上的数据到达时，不需要**中断**CPU，Offloads

其它获得硬件性能指标工具

dmidecode

直接查BIOS信息

dmidecode -t 0

type 0主要是硬件信息

powertop

查看最耗电进程

lspci

lsusb

ethtool

```
[root@install ~]# lspci |grep -i eth
```

```
03:00.0 Ethernet controller: Broadcom Corporation NetXtreme II BCM5708 Gigabit Ethernet (rev 12)
```

```
05:00.0 Ethernet controller: Broadcom Corporation NetXtreme II BCM5708 Gigabit Ethernet (rev 12)
```

```
[root@install ~]#
```

```
[root@install ~]# lspci |grep -i fib
```

```
12:02.0 Fibre Channel: Emulex Corporation Thor LightPulse Fibre Channel Host Adapter (rev 01)
```

```
12:02.1 Fibre Channel: Emulex Corporation Thor LightPulse Fibre Channel Host Adapter (rev 01)
```

```
[root@install ~]# dmesg |grep -i fib --color
```

```
Emulex LightPulse Fibre Channel SCSI driver 8.3.39
```

```
scsi3 : Emulex LP1050 PCI-X Fibre Channel Adapter on PCI bus 12 device 10 irq 19
```

```
scsi4 : Emulex LP1050 PCI-X Fibre Channel Adapter on PCI bus 12 device 11 irq 18
```

07 Understand IO Scheduling

07 Understand IO Scheduling

IO是调优的主要对象：最慢的设备存储，也是最容易成为整个系统的性能瓶颈

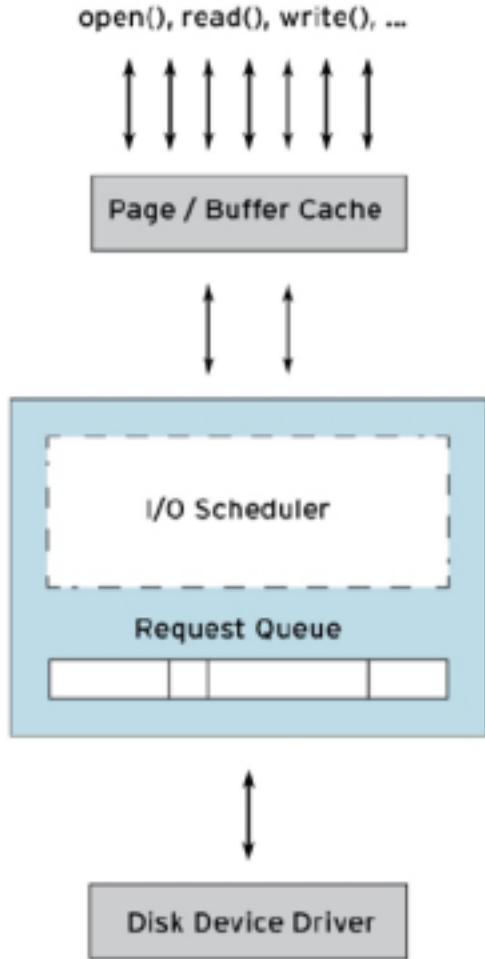
Linux通过多种手段给存储（硬盘）提速：

1. 工作方式（机械/SSD）
2. 接口类型（SCSI,SAS）
3. 连接方式
4. RAID0/1/5/6/10
5. Caching（硬盘本身的缓存，内存作为数据缓存）
程序从内存中读，写到内存
读能够被缓存，写可以延后写deferred
6. Cgroup资源分配

7. I/O Scheduling

尽量把request合并成少量的I/O请求（从队列中合并，例如一个请求扇区1，一个请求扇区6，一个请求扇区2，一个请求扇区1，合并成一个请求），从而可以使磁盘最小化机械臂的动作

I/O调度图：



直接IO direct I/O

```
# dd iflag=direct oflag=direct if=/dev/sda of=/file1 bs=1M count=1000
# free -m
# echo 3 > /proc/sys/vm/drop_caches
=====
```

I/O调度算法：

cfq

完全公平队列，比较适合于各种工作负载，绝大多数情况，具有普遍性

deadline 最后期限，预期的时间必须完成。当发出一个I/O请求，一定会在指定时间如5ms之内完成
作为虚拟机的物理机(virtualization host)，使用**deadline**可以让所有的虚拟机得到它们想要的东西

anticipatory 预料的

每次读扇区后，会稍稍等一下，看有没有人想读该扇区附近的扇区，一起读。适合于**大量的顺序读操作**，例如提供大文件下载

noop FIFO

响应很快，CPU不用费时，低开销，也就是不用调度

适合于SSD磁盘，不用合并

适合于虚拟机virtualization guests，因为写数据最终是虚拟机管理器（上层）来写，所以虚拟机不需要调度

查询和设置电梯算法：

每个磁盘可以独立的设置自己的电梯算法

```
# cat /sys/block/sda/queue/scheduler
# echo deadline > /sys/block/sda/queue/scheduler
```

适用于所有调度算法的参数：

nr_requests 队列长度

```
# cat /sys/block/sda/queue/nr_requests  
128          队列的长度，即128个请求
```

read_ahead_kb 预读技术

```
# cat /sys/block/sda/queue/read_ahead_kb  
128          单位是kb，如果kernel发现当前是顺序读，会预先读取128kb。如果kernel发现是随机读，会逐渐将其值调至0，关闭预读机制
```

开机有效：

```
[root@tianyun ~]# cat /etc/rc.local  
echo deadline > /sys/block/sda/queue/scheduler  
echo 256 > /sys/block/sda/queue/nr_requests
```

针对不同的电梯算法可调的参数：

Tuning the deadline Scheduler

```
# echo deadline > /sys/block/sda/queue/scheduler  
# cd /sys/block/sda/queue/iosched/  
# cat read_expire  
500          最后期限，保障，读操作500毫秒后响应  
# cat write_expire  
5000         最后期限，保障，写操作5秒之后内响应
```

Tuning the anticipatory Scheduler

```
# echo anticipatory > /sys/block/sda/queue/scheduler  
# cd /sys/block/sda/queue/iosched/  
# cat antic_expire  
6           等6毫秒
```

Tuning the cfq Scheduler 重点

```
# echo cfq > /sys/block/sda/queue/scheduler  
# cd /sys/block/sda/queue/iosched/
```

基于分类，优先级队列：

Class1(real-time)：实时
子优级0-7，数字越小0（最重要），越大7（最不重要）

Class2(best-effort): 最佳效果，大家都能访问 默认
子优级0-7，数字越小0（最重要），越大7（最不重要）

Class3(idle): 空闲，等磁盘没人用的时候，我再用
没有优先级

在cfq调度算法下，ionice 用于设置或更改分类

```
# ionice -p PID  
# ionice -c3 -p PID  
# ionice -c1 -n0 dd if=/dev/sda of=/dev/null
```

LAB1：

先准备三个大文件

```
# dd if=/dev/zero of=/bigfile1 bs=1M count=1000  
# dd if=/dev/zero of=/bigfile2 bs=1M count=1000  
# dd if=/dev/zero of=/bigfile3 bs=1M count=1000  
# echo 3 > /proc/sys/vm/drop_caches
```

终端一： root 监控当前谁在使用磁盘

```
# iotop  
be best-effort be  
real-time      rt  
idle          idle
```

终端二： 普通用户 user1

```
$ time ionice -c3 dd if=/bigfile3 of=/dev/null //C3没有优先级
```

终端三： 普通用户 user2

```
$ time ionice -c2 dd if=/bigfile2 of=/dev/null //c2默认，可以不写，默认为c2中的4
```

终端四：root
time ionice -c1 dd if=/bigfile1 of=/dev/null //默认为4 , c1强势

LAB2:

终端二：普通用户 user1
\$ time ionice -c2 -n0 dd if=/bigfile3 of=/dev/null

终端三：普通用户 user2
\$ time ionice -c2 -n7 dd if=/bigfile2 of=/dev/null

LAB3:

ionice 降低updatedb的IO //不希望占太多的磁盘开销

LAB4:

```
dd if=/dev/zero of=/test1.txt
ps aux |grep dd
ionice -c3 -p PID
```

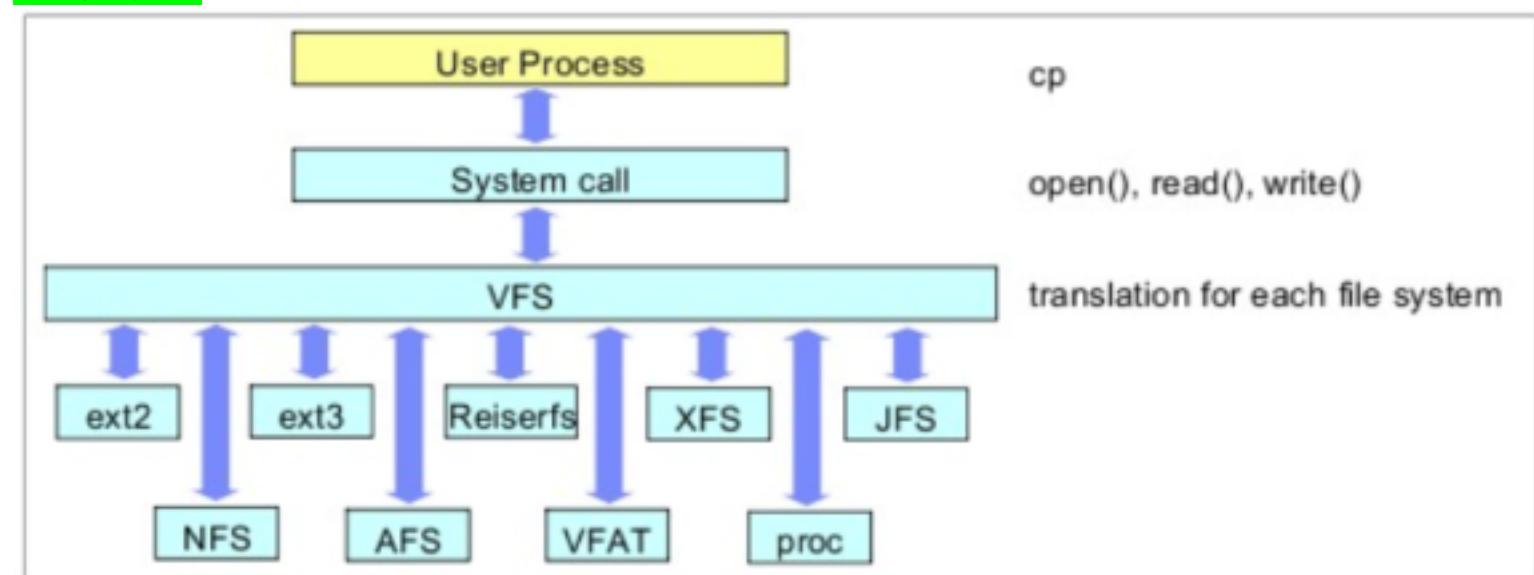
08 Tuning Filesystem

08 Tuning Filesystem

文件系统调优

一、了解文件系统

VFS虚拟文件系统



EXT3 Pros & Cons 利与弊

- Repair (fsck) 时间非常长
- 最大支持16TB

EXT4 Pros & Cons 利与弊

- Extent4 使节点表有了非常大的收缩，使节点表访问非常有效。相当于内存普通page和大页
- Extent4 不兼容EXT3, EXT2
- Faster fsck time (较ext3，速度提高10倍)
- Delayed allocation(尽量滞后写，不会立即fsync())
- 支持更大的文件系统及单个大的文件
- 最大支持RHEL6 16TB , RHEL7 50TB

XFS Pros & Cons

- 针对大存储，性能非常优异
- RHEL7 支持 500TB
- RHEL6 用的人太少， RHEL7 默认使用
- 对于大量的小文件，XFS不适合

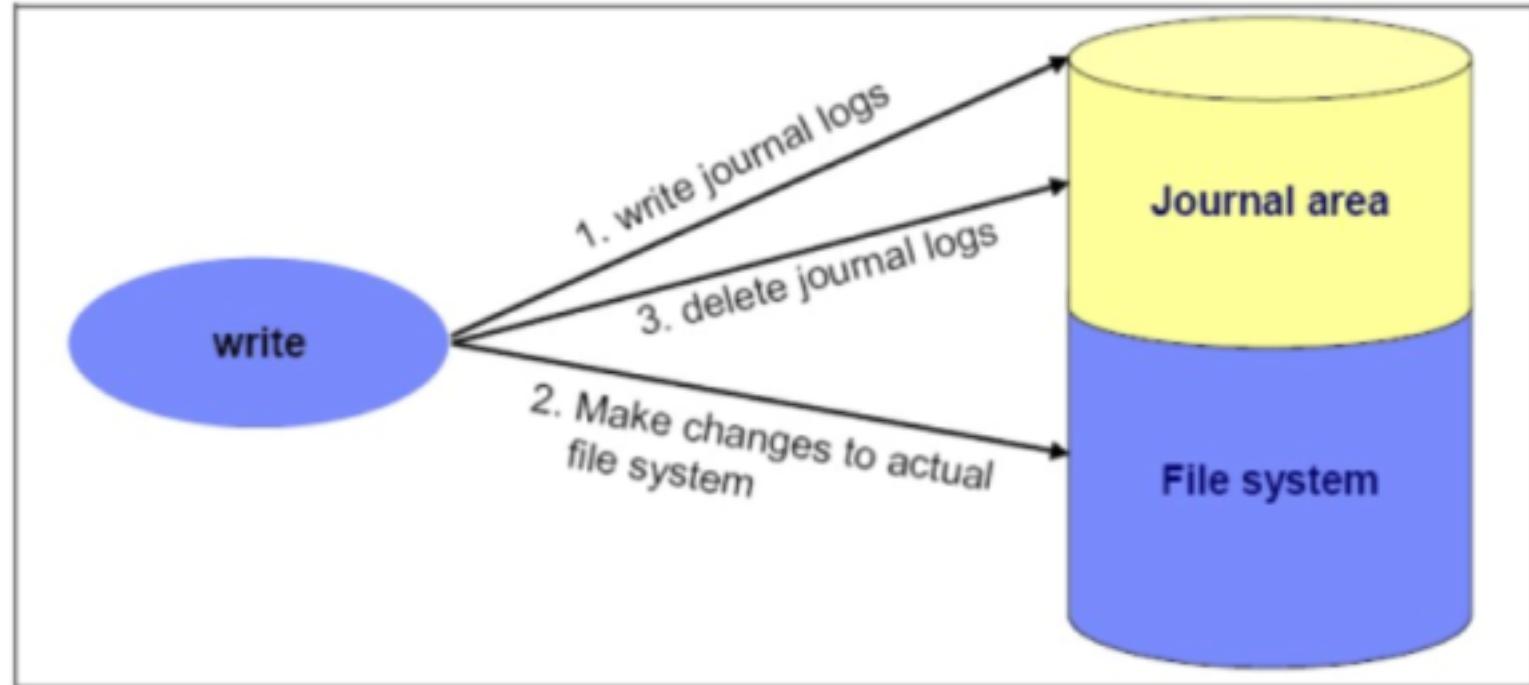
BTRFS , 从甲骨文公司推出的文件系统

- 内部就可以实现RAID和snashot
- 对于元数据和用户数据的保护是非常强的
- 支持动态拉伸和缩小
- RHEL6 Tech preview技术预览，暂不支持生产环境
- RHEL7 作为XFS 备选方案

二、文件系统日志功能 (EXT3/EXT4/XFS)

File System Journaling 文件系统日志

- 日志使恢复提速
- 日志可以保证文件系统的完整性



日志的三种模式

- data=ordered Default , 仅记录meta data到journal，而且当所有data被flushed to disk后commit
- data=writeback 仅记录metadata，速率快，但一旦发送crash
- data=journal 写两份，文件meta data和block area都记录到journal，仅适合于大量小数据存储

External Journaling 外部日志

```
# mkfs.ext4 -O journal_dev -b 4096 /dev/vdb1
# mkfs.ext4 -J device=/dev/vda1 -b 4096 /dev/vda3
# blkid
/dev/vdb1: UUID="b1615781-00c9-4b54-96f9-2f84f0f47ed4" TYPE="jbd"
```

/dev/vda3: UUID="5a5cc225-ab89-4d36-a214-c7a458384da4" EXT_JOURNAL="b1615781-00c9-4b54-96f9-2f84f0f47ed4

External Journaling on Existed Partition

```
# tune2fs -l /dev/vda1                                检查文件系统的块尺寸  
Block size: 1024  
# mkfs.ext4 -O journal_dev -b 1024 /dev/vdb2 必须和前面检查的文件块大小一致  
# umount /dev/vda1  
# tune2fs -O '^has_journal' /dev/vda1                移出存在的日志设备  
# tune2fs -j -J device=/dev/vdb2 /dev/vda1  
# mount /dev/vda1
```

三、挂载选项调优

扩展知识：Relatime 驱动器访问优化 RHEL6开始

POSIX 标准要求操作系统维护记录每个文件最后一次被访问的文件系统元数据。这个时间戳被称为 atime，维护它需要一个重复的对存储的写入操作。这些写入操作让存储设备及其连接保持忙碌和通电状态。因为很少应用程序会使用 atime 数据，所以这个存储设备活动是在浪费电力。特别是即使没有从存储中读取该文件也会发生写入存储的事件，但是从缓冲中写入。有时，Linux 内核还支持 mount 的 noatime 选项，并不在使用此选项挂载的文件系统中写入 atime。但是只是关闭这个特性是有问题的，因为有些应用程序会依赖 atime 数据，并在此数据不可用时失败。

红帽企业版 Linux 6 使用的内核之后此另一个可替换选项 -relatime。Relatime 维护 atime 数据，但不是每次访问该文件时都更改。启用这个选项，则只在上次更新 atime(mtime) 后修改该文件时，或者最后一次访问该文件是在相当长一段时间前（默认为一天）时才会将 atime 数据写入磁盘。

默认情况下，所有现在挂载的文件系统都启用 relatime。要在整个系统中限制这个特性，请使用 boot 参数 default_relatime=0。如果默认在某个系统中启用 relatime，您可以通过使用选项 norelatime 挂载某个系统来限制它在某个具体文件系统中的使用。最后，要使系统更新文件的 atime 数据的默认周期有所不同，请使用 relatime_interval= 引导参数，以秒为单位指定周期。默认值为 86400。

四、增加预读机制

```
[root@tianyun ~]# blockdev --getra /dev/sda  
256  
[root@tianyun ~]# blockdev --setra 512 /dev/sda  
[root@tianyun ~]# blockdev --getra /dev/sda  
512  
[root@tianyun ~]# echo "blockdev --setra 512 /dev/sda" >> /etc/rc.local
```

五、文件系统碎片 Fragmentation

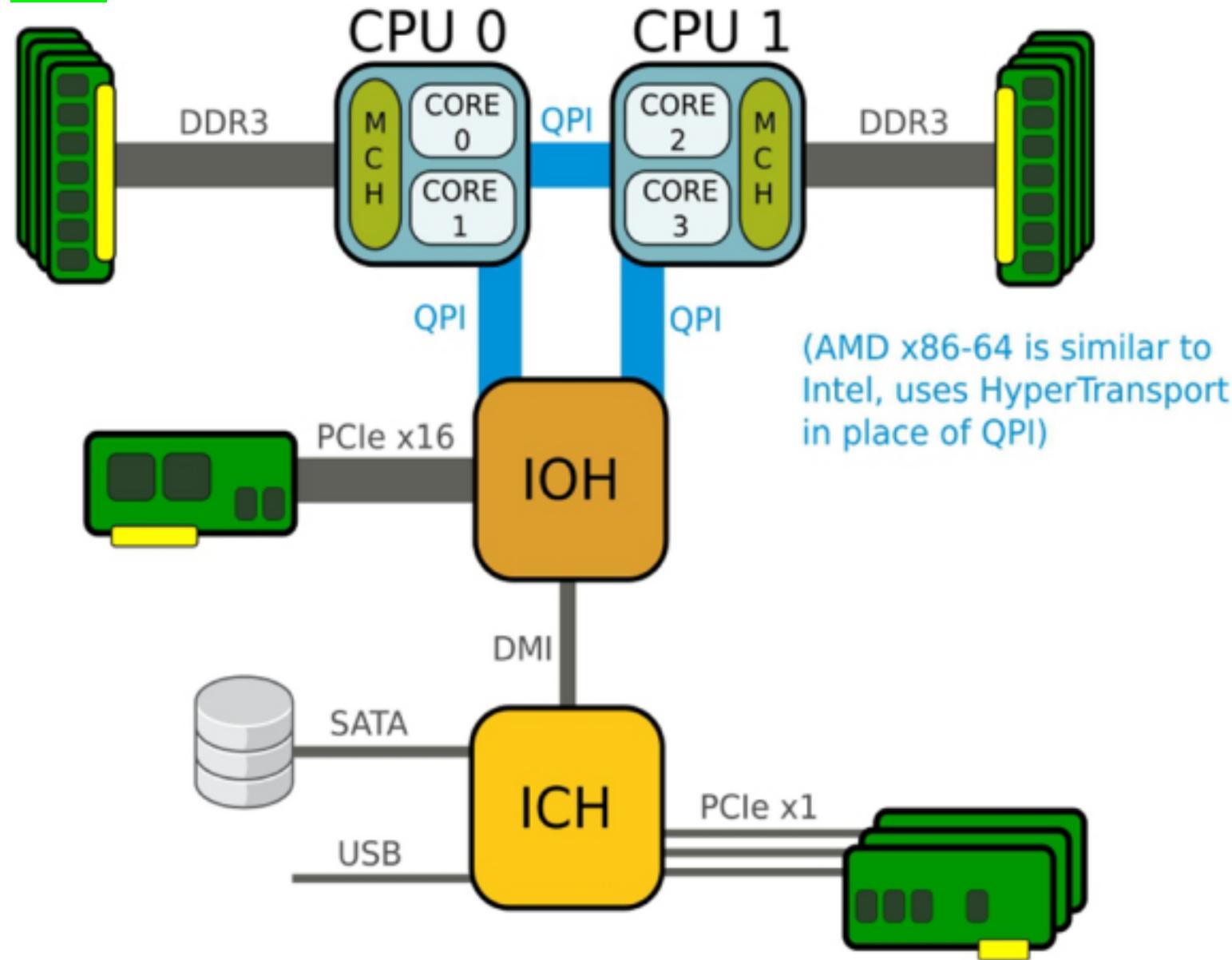
```
[root@tianyun ~]# e2freefrag /dev/sda1  
[root@tianyun ~]# filefrag /file
```

六、RAID优化

Stripe & Stride sizes

09 Understand CPU Scheduling

架构NUMA



CPU Scheduling

在一个时间内，一个CPU仅能执行一个进程

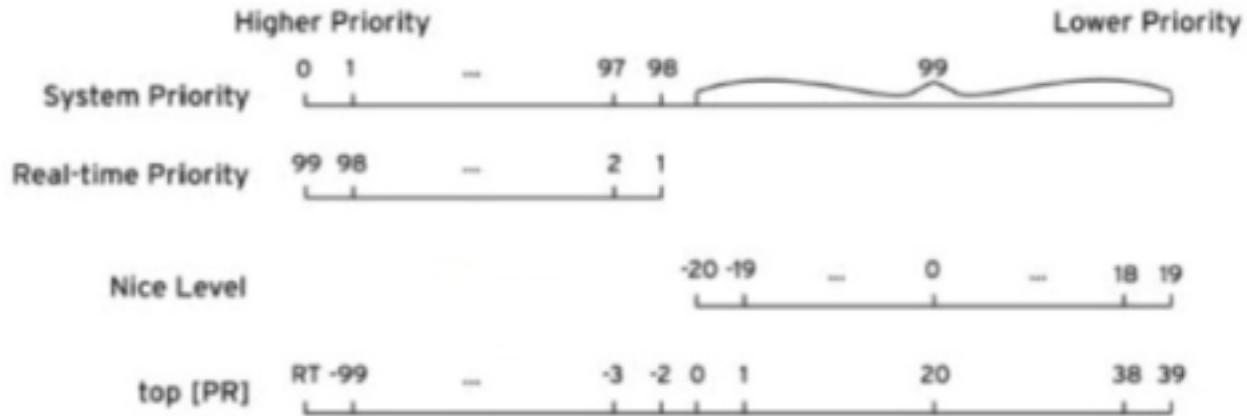
多个进程通过共享CPU时间片time slices

Kernel使用**进程调度算法**选择下一个待运行的进程

高优先级进程将获得大量的CPU time，并取代低优先级进程

CFS Scheduler Priority

CFS Scheduler Priority



Real time priorities 99 to 1 are static

Nice value -20 to 19 are dynamic(Kernel can change it by +/-5)

To use Real Time Scheduler

实时调度策略：

- SCHED_RR RR<默认>
- SCHED_FIFO FIFO

一般调度策略：

- SCHED_OTHER OTHER 该策略使用完全公平调度程序(CFS)

~~- SCHED_BATCH~~ 主要用于低优先权任务

~~- SCHED_IDLE~~ 主要用于低优先权任务

注：CFS调度算法，不是谁的优先级高，谁就拿CPU time。是谁最饥渴，谁就会拿CPU时间

CFS Scheduler

CFS： 完全公平调度，针对kernel选择下一个待运行的进程使用的调度算法

CFS是建立在Virtual time之上的

Virtual time based on: 三个因素

-- Waiting time 基于进程等待的时间，进程是不是等了很久了

-- No.of needed processes 队列中有多少个进程特别需要CPU

-- Process priority 进程的优先级只是其中一个因素

Virtual time最大的进程获得CPU时间片，然后Virtual time就减小，下次再比，如果没有进程比它大，就再执行，直到有比它大的

- 编程时调用特定函数

- chrt命令，可以在用户层使用chrt命令新运行或更改一个已运行进程调度策略

[root@tianyun ~]# chrt -h

Scheduling policies:

- b | --batch set policy to SCHED_BATCH
- f | --fifo set policy to SCHED_FIFO
- i | --idle set policy to SCHED_IDLE
- o | --other set policy to SCHED_OTHER
- r | --rr set policy to SCHED_RR (default)

[root@tianyun ~]# chrt -m
SCHED_OTHER min/max priority : 0/0

```
SCHED_FIFO min/max priority : 1/99  
SCHED_RR min/max priority : 1/99  
SCHED_BATCH min/max priority : 0/0  
SCHED_IDLE min/max priority : 0/0
```

```
[root@tianyun ~]# chrt -p 2247  
pid 2247's current scheduling policy: SCHED_OTHER  
pid 2247's current scheduling priority: 0
```

```
[root@tianyun ~]# chrt -p 7  
pid 7's current scheduling policy: SCHED_FIFO  
pid 7's current scheduling priority: 99
```

```
[root@tianyun ~]# chrt -i 0 dd if=/dev/zero of=/dev/null &  
[root@tianyun ~]# chrt -p 3451  
pid 3451's current scheduling policy: SCHED_IDLE  
pid 3451's current scheduling priority: 0
```

实时进程管理 chrt

实验环境：guest 1cpu

LAB1:

```
终端一: chrt -f 10 md5sum /dev/zero  
终端二: chrt -f 10 shasum /dev/zero
```

LAB2:

```
终端一: chrt -r 10 md5sum /dev/zero  
终端二: chrt -r 10 shasum /dev/zero
```

非实时进程 nice

LAB3: 默认为一般调度策略other

```
终端一: md5sum /dev/zero  
终端二: nice -n -15 shasum /dev/zero
```

调整终端二（后启动的进程）nice值为-20

```
=====
```

10 All about Memory

10 All about Memory

内存调优

```
=====
```

一、内存Memory

Memory Management

内存管理是kernel重要的工作

Paging(分页)机制是目前管理内存的一种手段

内存被划分成page页，对于x86，normal page 4KiB(4096 Bytes)，访问内存的最小单位

进程是不可以直接访问物理内存的，而是每个进程分配自己的虚拟地址空间VM

32-bit: 4GiB 每个进程最大使用

64-bit: 16EiB 每个进程最大使用

对于进程而言，仅能看到映射到物理内存的虚拟内存，每个进程是完全独立的

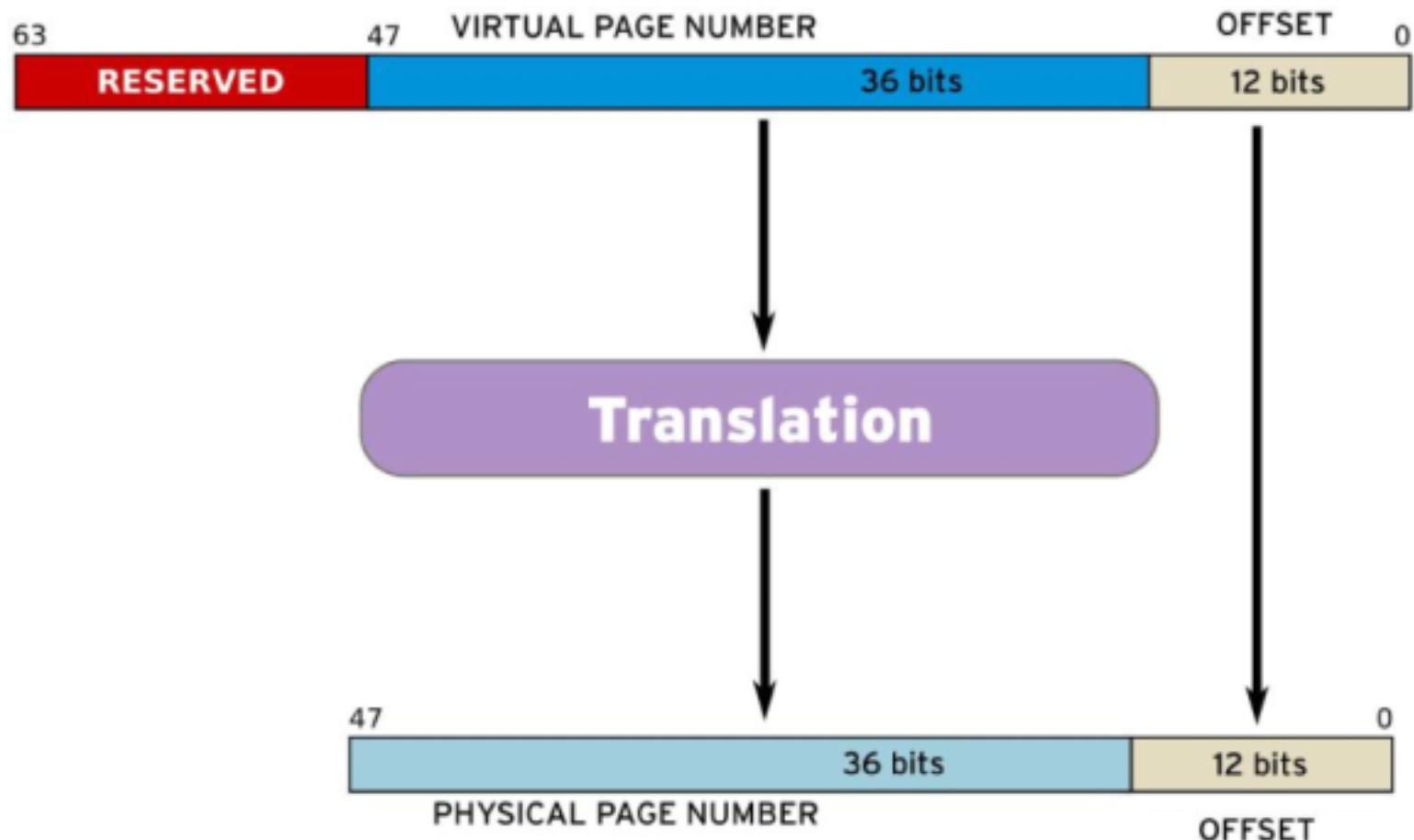
需要虚存到物理内存的转换 page table

Physical vs. Virtual

x86 4KiB

Physical Memory: 物理内存页叫做Page Frame页帧，程序不能直接访问物理内存

Virtual Memory: 虚拟内存页叫做Page，kernel给程序分配的是虚拟内存



Memory usage: VSZ vs. RSS

Virtual memory (VIRT) is all memory the process is using, including the resident set, shared libraries, and any mapped or swapped memory pages.

(Labeled VSZ in the ps command.)

Resident memory (RES) is the physical memory used by the process, including any resident shared objects. (Labeled RSS in the ps command.)

多个进程可以共享同一片物理内存，即虽然每个程序使用的虚拟是不一样的 (private virtual address space)

但都映射到了相同的physical pages，当然必须由kernel来控制实

现。

父进程fork()派生子进程：

COW

子进程的虚拟内存和父进程的虚拟内存映射到了相同的物理内存，例如子进程只是读

如果子进程需要写才为其映射到新的物理内

存

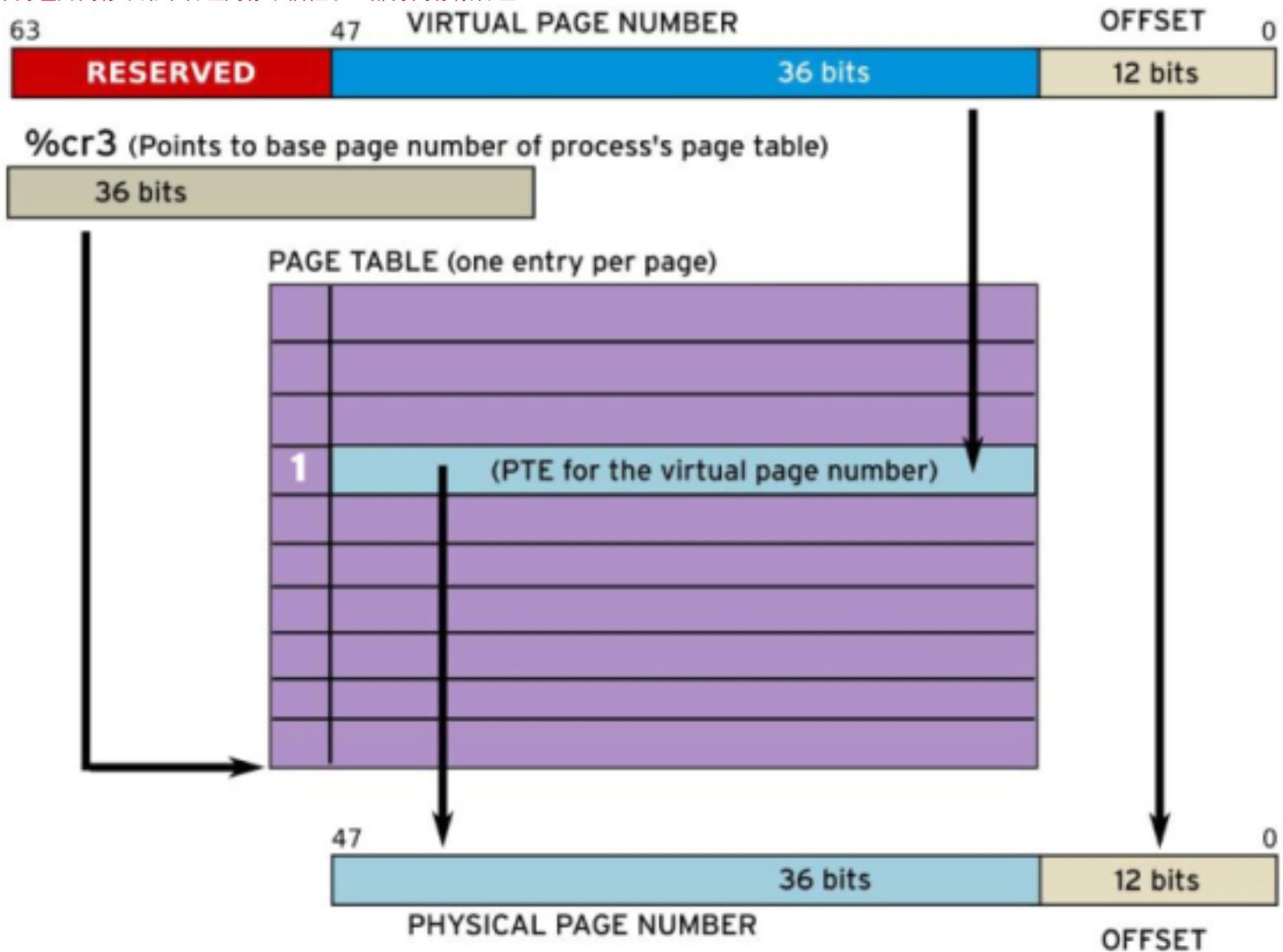
所以，在同一时间，若干个进程共享了同一片内存（物理内存）是有可能

Page Tables and the TLB

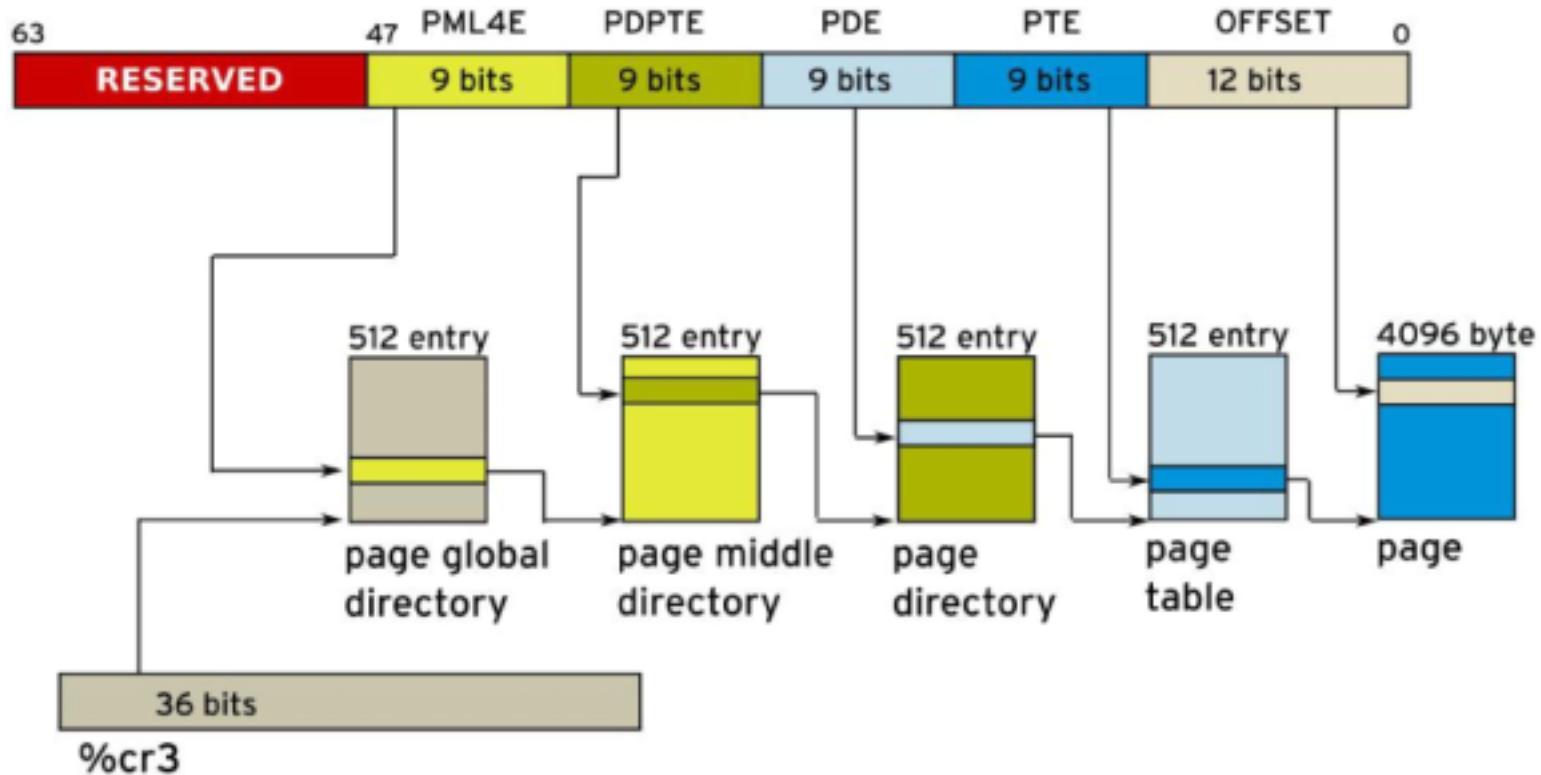
Page Tables

虚存到物理内存的映射关系是存储在 page tables(虚存到物理内存的镜像表)

页表本身也占内存，用来管理内存，牺牲了一部分内存做管理



(Page table structure is simplified)



TLB缓存buffer

CPU: TLB缓存 x86info -c 缓存虚拟到物理映射关系的条目

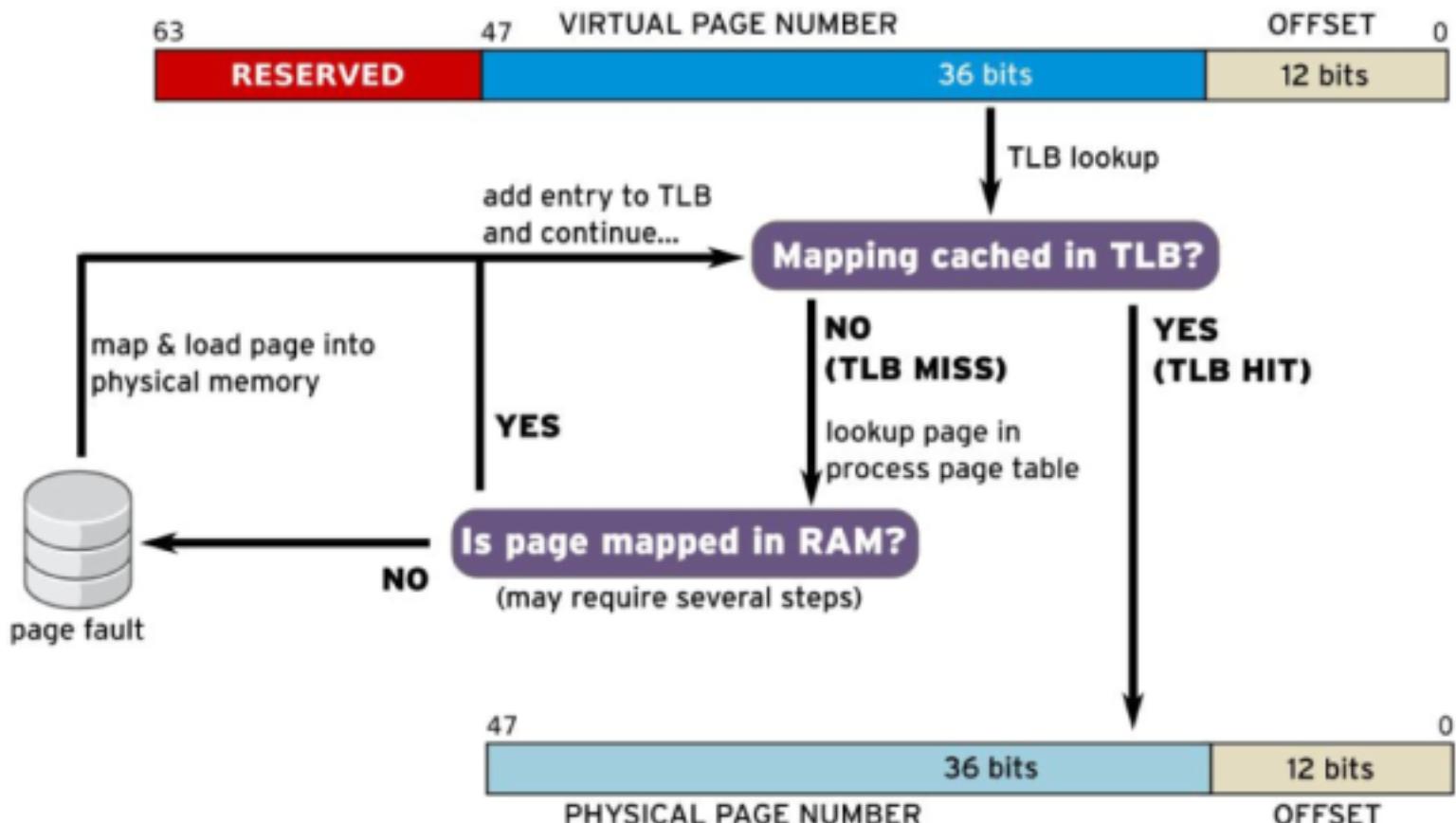
YES : 有虚拟到物理内存的镜像关系 TLB HIT , 就不用去Page Walk

NO : TLB MISS , 必须做Page Walk , 在做Page Walk时

page已映射到了RAM : 直接add entry to TLB

Page未映射到RAM : 比如之前申请的 , 但还没有映射到RAM , 产生一个page fault的行为 (映射)

如果已被交换了swap , 那么此次page fault的代价是相当高的



[查看主页及次页缺失](#)

minor page fault 次页缺失

major page fault 主页缺失

ps o pid,comm,minflt,majflt PID

ps o pid,comm,minflt,majflt `pidof vsftpd`

TLB：能带来15%的性能提升

每次上下文切换都会flush TLB

[root@tianyun ~]# x86info -c

TLB info

Instruction TLB: **4K pages**, 4-way associative, 128 entries.

Data TLB: **4KB pages**, 4-way associative, 64 entries

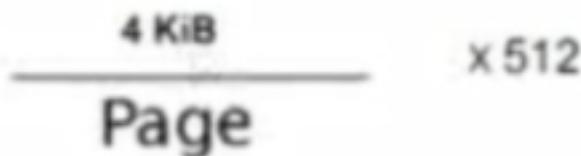
64 byte prefetching.

Found unknown cache descriptors: 5a 76 ca ff

Huge Pages Configuration

Huge Pages are pages of larger sizes

e.g. on x86 it's 2MiB



Huge Page

THP

但从RHEL6.2开始支持Transparent Huge Pages 透明式大页

应用程序不需要关心，由THP在后帮其实现

Up to 10% performance improvement

案例1：虚拟机使用Huge Pages

1. 开启Transparent Huge Pages

[root@tianyun ~]# cat /sys/kernel/mm/transparent_hugepage/enabled
[always] madvise never

2. 设置大页

[root@tianyun ~]# echo 1000 > /proc/sys/vm/nr_hugepages

[root@tianyun ~]# grep Huge /proc/meminfo

AnonHugePages: 464896 kB

HugePages_Total: 2500

```
HugePages_Free: 0  
HugePages_Rsvd: 0  
HugePages_Surp: 0  
Hugepagesize: 2048 kB  
[root@tianyun ~]# mount -t hugetlbfs hugetlbfs /dev/hugepages/
```

开机设置大页：

```
[root@tianyun ~]# vim /etc/grub.conf  
kernel parameter: hugepages=2500
```

3. guest XML

```
<memoryBacking>  
  <hugepages/>  
</memoryBacking>
```

How Does it work? 如何工作的

尽可能分配大页

可以拆成4KiB进行swap

尽可能使用大页，从而提高TLB的命中率

kernel使用的是大页

Memory Allocation 内存分配总结

1. 当父进程fork()出子进程时，并不会给孩子分配物理内存，仅是给出一个虚拟内存的承诺，
直到父亲或孩子对共享的内存做写操作，COW（写时复制）

3. 进程在申请内存时，kernel给它承诺的是虚存地址空间，而不是物理内存

4. 直到进程真正需要内存时，才会发生一个page fault页错

 . Minor 小错误，即没有分配，立即分配

 . Major 大错误，即被交换出去了，换回来

4. kernel会reclaims

pages

使用内存时，总的原则是，尽量推迟分配

二、Swap Space

Swap的作用

可以‘增加’内存

没有被使用的页可以被 paged out到磁盘上去，可以腾出更多的剩余内存，如此一来kernel就不会忙着不断去扫描有没有非活动页面需要回收
当pages被使用时，就会发生一个Major fault，从磁盘中把数据交换回来，代价很高

当发生大量的交换时，系统性能会严重下降，因为磁盘的速度太低

vmstat显示swap使用

操作系统尽量在优化（不定期的进行交换）

当si，so持续变得非常大，说明系统把交换分区真正当内存来用，说明内存不足

配置交换的建议

从一定程度上是把内存变大，适当交换可以提升系统的性能

一定程度避免了 out-of-memory，即承诺给程序如2G，但当程序真正使用时又没有2G的内存了...

Memory	Min Swap
up to 4GB	2GB
4-16GB	4GB
16-64GB	8GB
64-256GB	16GB

交换分尽量分布在不同的物理磁盘

Buffer Cache and Page Cache

Buffer Cache 主要存储文件的是：file system meta data(元数据) find / ls -aR /

Page Cache 存储文件的内容： file system block data 能否被交换？

```
[root@tianyun ~]# echo n > /proc/sys/vm/drop_caches                 man proc
```

- 1 block data

- 2 meta data

- 3 block and meta data

关于swap的kernel参数

例如当一个新进程需要申请内存，而内存空间不足，kernel有两种选择：

1. 优先reclaim pagecache
2. 交换swap

/proc/sys/vm/swappiness

The value in this file controls how aggressively the kernel will swap memory pages. Higher values increase aggressiveness, lower values decrease aggressiveness. The default value is 60.

```
[root@tianyun ~]# grep Dir /proc/meminfo
```

```
Dirty: 20 kB  
DirectMap4k: 92788 kB  
DirectMap2M: 8048640 kB
```

注：有些内存页是不会交换的，例如从磁盘上读出来的文件修改后产生的dirty，而是写回磁盘

Optimizing Swap Spaces 优化swap

1. RAID0
2. SSD

3. Mount option **pri=value** 多个交换分区（不同的物理设备）优先级相等，将会轮循使用，否则把第一个满后才用后面的

```
[root@tianyun ~]# swapon -s
```

```
[root@tianyun ~]# grep swap /etc/fstab  
/dev/sda5 swap swap defaults,pri=10 0 0  
/dev/sdb5 swap swap defaults,pri=10 0 0
```

Memory Page Status 内存的状态

```
[root@tianyun ~]# less /proc/meminfo
```

Free 未分配

Inactive Buffer Cache and Page Cache 但没有修改，这一部分也是可以被分配的

Dirty 如果从磁盘上读出来的文件修改过，但没有存盘

Active 进程正在使用

Reclaiming Dirty Pages <Dirty> 脏页回收

脏页必须定期写回到磁盘，以防止断电数据丢失

RHEL6之前，由kernel的线程pdflush来做，脏页越多，I/O越重，pdflush的进程就会越多，在后台不停的刷

现在针对每一个块设备都有一个per-BDI flush线程： **flush-MAJOR:MINOR** 主号，从号

Tune per-BDI Flush Threads 针对per-BDI Flush Threads的调优

针对per-BDI flush Threads有4个可调kernel参数：

vm.dirty_expire_centisecs 脏了多久才考虑flush，尽量滞后可以合并，单位是1/100秒，即10毫秒

vm.dirty_writeback_centisecs kernel每隔多久唤醒flush线程

vm.dirty_background_ratio(default 10) ratio(比)，当系统的内存有10%脏了，kernel会在后台刷脏页

vm.dirty_ratio(default 40) 脏页有40%，挂起所有进程的写操作，flush脏页，至到10%，例如1G内存，有400M都是脏页

查看：

```
[root@tianyun ~]# sysctl -a |grep vm.dirty  
vm.dirty_background_ratio = 10  
vm.dirty_background_bytes = 0  
vm.dirty_ratio = 20  
vm.dirty_bytes = 0  
vm.dirty_writeback_centisecs = 500  
vm.dirty_expire_centisecs = 3000
```

永久生效：

```
# vim /etc/sysctl.conf
```

Memory Overcommitment 如何对待内存超售的问题？

```
[root@tianyun ~]# sysctl -a |grep overcommit
```

vm.nr_overcommit_hugepages = 0

vm.overcommit_memory = 0

vm.overcommit_ratio = 50

-0：智能识别

-1：不做任何检查，完全可以分配

-2：不做超过能力的事

能力：

会将**swap + percentage(vm.overcommit_ratio)** of the RAM (default 50)作为分给单个进程最大的虚存的最大值

swap + 物理内存的50%(**vm.overcommit_ratio**值默认是50%，可以超过100%)

swap 8G + 物理内存4G * 50% = 10G
swap 8G + 物理内存4G * 150% = 14G

Out of Memory(OOM)

OOM发生在一个特定的条件

承诺了2G内存，当兑现的时候又无法兑现！此时，kernel会随机杀进程

[root@tianyun ~]# sysctl -a |grep vm.panic_on_oom

vm.panic_on_oom = 0

当set to 1, kernel will panic 宁可挂起

任何一个内存区域不够了，都会发生OOM，尽管free还看到有大量的空闲空间

Non-Uniform Memory Access (NUMA)

```
[root@tianyun ~]# numactl --show
policy: default
preferred node: current
physcpubind: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
cpubind: 0 1
nodebind: 0 1
membind: 0 1

[root@tianyun ~]# numactl --hardware
available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3 4 5 12 13 14 15 16 17
node 0 size: 32740 MB
node 0 free: 544 MB
node 1 cpus: 6 7 8 9 10 11 18 19 20 21 22 23
node 1 size: 32768 MB
node 1 free: 108 MB
node distances:
node 0 1
 0: 10 21
 1: 21 10
```

cgroup:

LAB1:

```
[root@tianyun ~]# vim /etc/cgconfig
group node0 {
    cpuset {
        cpuset.cpus=0-7;
        cpuset.mems=0;
    }
}
group node1 {
    cpuset {
        cpuset.cpus=8-15;
        cpuset.mems=1;
    }
}
```

LAB2:

```
[root@tianyun ~]# vim /etc/cgconfig.conf
group node0 {
    cpuset {
        cpuset.cpus=0;
        cpuset.mems=0;
    }
}
```

```
[root@tianyun ~]# vim /etc/cgrules.conf
*:top    cpuset    node0/
```

```
[root@tianyun ~]# service cgred restart
[root@tianyun ~]# service cgconfig restart
```

查看进程工作的CPU及内存节点：

```
[root@tianyun ~]# watch -n.5 ps o psr,comm,pid
```

top 按f命令，添加字段j，显示进程工作的CPU

```
[root@tianyun ~]# numastat
          node0      node1
numa_hit        68147645    102356480
numa_miss       914974     341117
numa_foreign     341117    914974
interleave_hit   20159     20155
local_node      67910717  102332020
other_node      1151902    365577
```

11 Tuning Network Performance

11 Tuning Network Performance

网络调优

=====

网络架构，网络使用的设备，连接方式都会影响到网络性能

一、measuring Network Performance 测量网络性能

qperf：C/S结构

服务器端：

```
# yum -y install qperf
# qperf
```

客户端：

```
# yum -y install qperf
# qperf 172.16.8.100 conf           //显示远程主机的信息
# qperf 172.16.8.100 tcp_bw tcp_lat //测试带宽和延迟bandwidth and latency
# qperf 172.16.8.100 udp_bw udp_lat
```

pinpoint critical point 找到关键点(临界点)

```
# qperf 172.16.8.100 -oo msg_size:1:32K:*2 tcp_bw tcp_lat
msg_size从1B到32K，每次乘以2，至到32K
```

```
# qperf 172.16.8.100 -oo msg_size:1:32K:*2 -v udp_bw udp_lat
```

观察在UDP协议下：CPU负载，小包时中断较多

二、网络优化

1. 链路聚合bonding,Team

2. 使用高端网卡，例如支持中断，虚拟化

3. 内核参数调节

Tune Network Latency

net.ipv4.tcp_low_latency=1 优化反应速度，低延时，用户的感受度好
net.ipv4.tcp_low_latency=0 优化吞吐量 (default) RHEL6就是面向高吞吐设计的

接收/发送的缓冲区调节 socket buffer

```
[root@tianyun ~]# sysctl -a |grep net.core.rmem  
net.core.rmem_max = 229376  
net.core.rmem_default = 229376  
[root@tianyun ~]# sysctl -a |grep net.core.wmem  
net.core.wmem_max = 229376  
net.core.wmem_default = 229376
```

关于TIME_WAIT调节

```
[root@tianyun ~]# sysctl -a |grep net.ipv4.tcp_tw  
net.ipv4.tcp_tw_recycle = 0      //是否允许快速回收  
net.ipv4.tcp_tw_reuse = 0      //是否允许复用
```

最大的SYN队列长度

```
[root@tianyun ~]# sysctl -a |grep net.ipv4.tcp_max_syn  
net.ipv4.tcp_max_syn_backlog = 2048
```

轻量级防DDOS攻击

```
[root@tianyun ~]# sysctl -a |grep net.ipv4.tcp_syncook  
net.ipv4.tcp_syncookies = 1
```

重发SYN+ACK的次数，防半开连接

```
[root@tianyun ~]# sysctl -a |grep net.ipv4.tcp_synack  
net.ipv4.tcp_synack_retries = 5
```

bonding配置案例：

1. 接口配置文件

```
[root@install network-scripts]# cat ifcfg-bond0
```

```
DEVICE=bond0  
TYPE=Ethernet  
ONBOOT=yes  
NM_CONTROLLED=no  
BOOTPROTO=none  
IPADDR=172.16.8.100  
PREFIX=24  
IPV6INIT=no  
USERCTL=no  
GATEWAY=172.16.8.254
```

```
[root@install network-scripts]# cat ifcfg-eth0
```

```
DEVICE=eth0  
TYPE=Ethernet  
ONBOOT=yes  
NM_CONTROLLED=no  
BOOTPROTO=none  
IPV6INIT=no  
USERCTL=no  
MASTER=bond0  
SLAVE=yes
```

```
[root@install network-scripts]# cat ifcfg-eth1
```

```
DEVICE=eth1  
TYPE=Ethernet  
ONBOOT=yes  
NM_CONTROLLED=no  
BOOTPROTO=none  
IPV6INIT=no  
USERCTL=no  
MASTER=bond0  
SLAVE=yes
```

2. bonding参数

```
[root@install ~]# tail -2 /etc/modprobe.d/dist.conf
alias bond0 bonding
options bonding mode=0 miimon=100
```

3. 重启网络服务

```
[root@install ~]# service network restart
[root@install ~]# ip a
```

```
[root@install ~]# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.6.0 (September 26, 2009)
```

Bonding Mode: load balancing (round-robin)

MII Status: up

MII Polling Interval (ms): 100

Up Delay (ms): 0

Down Delay (ms): 0

Slave Interface: eth0

MII Status: up

Speed: 1000 Mbps

Duplex: full

Link Failure Count: 0

Permanent HW addr: 00:1b:78:39:3a:00

Slave queue ID: 0

Slave Interface: eth1

MII Status: up

Speed: 1000 Mbps

Duplex: full

Link Failure Count: 0

Permanent HW addr: 00:1b:78:39:39:fe

Slave queue ID: 0

12 Database Server Tuning

Database Server tuned Profile

```
enterprise-storage
latency-performance
```

Tune the Network for Latency

```
sysctl -w net.ipv4.tcp_low_latency=1
```

Tune SysV IPCS

```
kernel.shmmax
```

```
sysctl -w kernel.shmall=512 * 1024 K / 4 K = 131072
```

Non-Uniform Memory Access (NUMA)

Huge Pages

```
# cat /sys/kernel/mm/redhat_transparent_hugepage/enabled
[always] never
```

Tuning swappiness

注重 : page cache

本节作业

1. 你对优化是如何理解的？（意义：提升性价比，手段：系统/应用参数、资源合理分配）
2. process 和 Thread的区别？（Apache响应请求创建进程或线程举例）
3. 什么是上下文件切换 Context switching?

七、云计算

什么是云计算？

什么是云计算？

很多人会告诉你：云计算是IaaS, PaaS和SaaS...

谷歌：2006年8月，在圣何塞举办的SES（搜索引擎大会上），谷歌CEO施密特提出“云计算”的概念；

亚马逊：虽然谷歌为云计算命名，但真正明确云计算商业模式的是亚马逊。在施密特提出“云计算”这个词后的几个星期之后，亚马逊提出了EC2服务；最早时，为了让网站能支持大规模的业务，亚马逊在基础设施建设上花了很大功夫，自然也积累了很多经验。为了将平时闲置的大量计算资源也作为商品出售，贝索斯的亚马逊公司先后推出了S3（简单存储服务）和EC2等存储、计算租用服务。当亚马逊推出IaaS（基础设施即服务）之后，仿佛给互联网世界开了一扇窗，告诉人们，还可以这样来运营计算资源，还有一种新的商业模式，叫云计算。

雅虎：云计算起源于互联网公司。2000年，互联网经济处于第一波热潮中。这时全球互联网用户人数已经从1995年的几千万增至数亿，并仍在不断增长。快速增长的用户数量，使许多网站感受到了系统支撑的压力。雅虎也一样，当时雅虎经常面临的一个问题是，一个频道设计完成之后，在上线之前测试可支撑一定数量的用户（比如100万人），但第二天一上线就出现达到指标的情况。怎么办？只有增加设备，尽可能的提高系统对服务的支撑能力，因此即刻购买设备然后连夜组装机器的情况并不少见。

大系统、大数据、大用户！

对于大多数中小型组织，基于个人而言，云计算的魅力来自那些灵活、弹性和随时随地可用的云计算服务，比如亚马逊的计算资源租用服务，或者一些针对企业和个人的“云存储”服务。

云计算包括两方面的内容：**服务** 和 **平台**，云计算既是商业模式，也是技术。

服务：云计算服务代表一种新商业模式，SaaS（软件即服务）、PaaS（平台即服务）和IaaS（基础设施即服务）是这种商业模式的代表形式。

SaaS（软件即服务）：通过互联网提供软件服务，例如财务、CRM（客户关系管理）、ERP（企业资源计划）、搜索、地图

PaaS（平台即服务）：通过互联网提供应用程序运行平台服务，例如Google App Engine (GAE)

IaaS（基础设施即服务）：通过互联网提供基础设施服务，例如Amazon S3、EC2、网盘，VPS，迅雷离线下载

在SaaS、PaaS、IaaS之外，又出现了一些新的服务形式名称：

商业流程即服务（Business Process as a Service）

数据库即服务（Database as a Service）

安全即服务（Security as a Service）

平台：对于任何一种商业模式而言，除了理论上可行之外，还要保证实践上的可用。因此，伴随着云计算服务理念的发展，云计算形成了一整套**技术实现机制**，而云计算平台则是这套机制的具体体现。**云计算平台在设计上就针对了“大用户”、“大系统”和“大数据”的问题提出了解决方案。所以，以云平台支撑的云计算服务，不仅可以提高服务的效率，而且还会充分发挥平台的能力的优势。**

从技术角度而言，云计算最早的出身，应该是超大规模分布式计算。比如雅虎为了解决系统对大规模应用的支撑问题，而设计的超大规模分布式系统，目的就在于将大问题分解，由分布在不同物理地点的大量计算机共同解决。但随着技术不断的发展和完善，云计算在解决具体问题时，借鉴了不少其他技术和思想，包括虚拟化技术、SOA（面向服务架构）理念等。但云计算与这些技术有根本性的差别，不仅体现在商业应用上，还体现在实现细节上。

云计算的关键技术

分布式计算技术
分布式存储技术
虚拟化技术
自动运维
计费

为了区分云计算的部署使用方式：

公有云 (Public Cloud) : 通常指开放给公众使用的云基础设施
私有云 (Private Cloud) : 通常所面向的是一个组织机构，而非公众
混合云 (Hybrid Cloud) : 比如一些游戏服务提供商有自己的私有云，但在用户使用高峰时又会租用如亚马逊的IaaS服务

技术是互通的！

云计算服务通常应该满足三个条件：

1. 服务应该是随时随地可接入
2. 服务应该永远在线
3. 服务拥有足够大的用户群，即所谓的“多租赁”
4. 支持大系统、大数据、大用户！

云计算的典型应用场景

渲染农场
迅雷离线下载
百度网盘
阿里云

OpenStack

OpenStack简介

OpenStack



云计算鼻祖之一Amazon

创始人是Jeffery Bezos。原任职华尔街，后看到互联网急速发展的商机，选择网上书店作为其投资事业
亚马逊1995年正式上线，网站名取自于南美亚马逊河亚马逊河互联网一起快速发展，1997年在Nasdaq上市，
股价从几美元一直上升到一百多美元2001年开始实现盈利，利润500万美元（营业额10亿美元）
不仅仅是电子商务

亚马逊：从卖书到云计算

AWS : Amazon Web Services 是一组服务，它们允许通过程序访问 Amazon 的计算基础设施。

Dynamo : 一种Key-Value型NoSQL数据库，是NoSQL运动的先驱

S3 : 全名为亚马逊简易储存服务 (Amazon Simple Storage Service) , 由亚马逊公司，利用他们的亚马逊网络服务系统所提供的网络线上储存服务。经由Web服务界面，包括REST, SOAP, 与BitTorrent，提供用户能够轻易把档案储存到网络服务器上。在2006年3月开始，亚马逊公司在美国推出这项服务，2007年11月扩展到欧洲地区。亚马逊公司为这项服务收取的费用，是每个月每一个gigabyte 需要0.15 元美金，如果需要额外的网络带宽与品质，则要另外收费。

EC2 : 弹性计算云



Simple Storage Service (S3)

作用

通过提供的标准接口，将任意类型的数据存储在服务器上

目标

可靠，易用，低使用成本

横向扩展

架构在Dynamo上

非传统关系数据库

使文件操作尽量简单、高效

不支持SQL的查询等功能，降低系统复杂性

Elastic Compute Cloud (EC2)

虚拟的计算机

“无限”能力

可以根据用户需求来配置实例数量，从而改变租用的计算能力

主要特性

灵活性：允许配置实例数量、地理位置

低成本：按小时计费

安全性：一整套安全措施

易用性：自由构建应用程序，自动负载平衡

容错性：无状态和多实例配置

Amazon Machine Image (AMI)

作用

相当于虚拟机模板

可将用户的应用程序、配置等一起打包

类型

公共AMI：亚马逊提供，免费

私有AMI：本人或授权用户可进入

付费AMI：向开发者付费购买

共享AMI：开发者之间共享

Openstack：亚马逊EC2的山寨品

OpenStack 是由 Rackspace 和 NASA 共同开发的云计算平台，帮助服务商和企业内部实现类似于 Amazon EC2 和 S3 的云基础架构服务

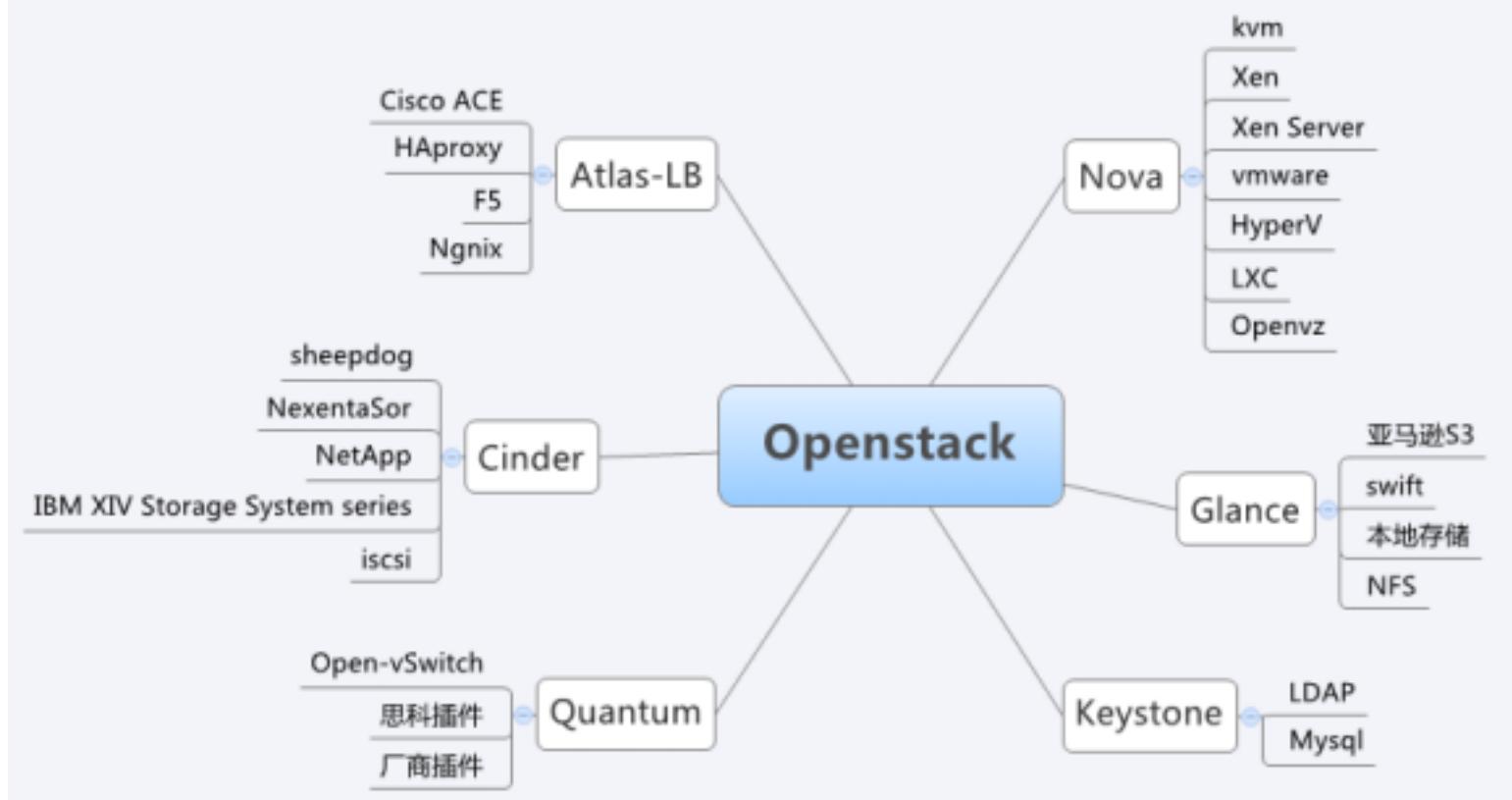
OpenStack 包含两个主要模块：**Nova** 和 **Swift**，前者是 NASA 开发的**虚拟服务器部署**和**业务计算模块**；后者是 Rackspace 开发的分布式云存储模块。主要采用Python编写源代码，使用Twisted软件框架

旨在为公共及私有云的建设与管理提供软件的开源项目，使用Apache许可证

厂商支持：Dell、Citrix、Cisco、Canonical、AMD、Intel等

Openstack官网：<http://www.openstack.org/>

Rackspace官网：<http://www.rackspace.com/>



Openstack子项目

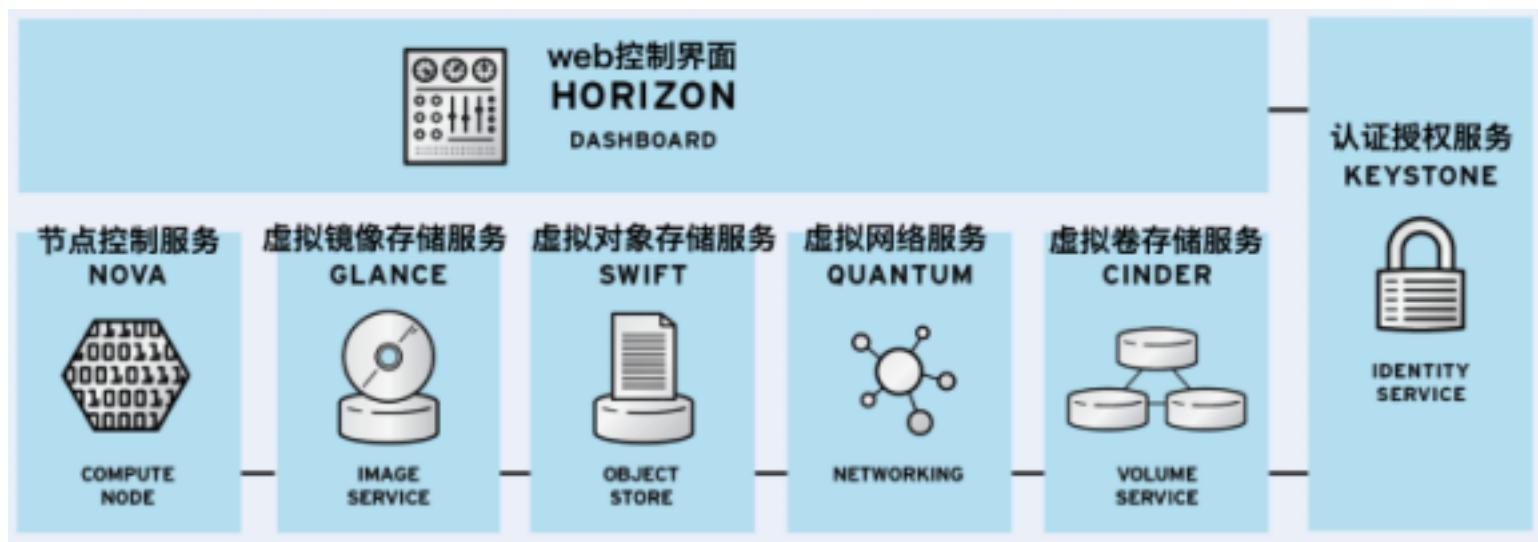
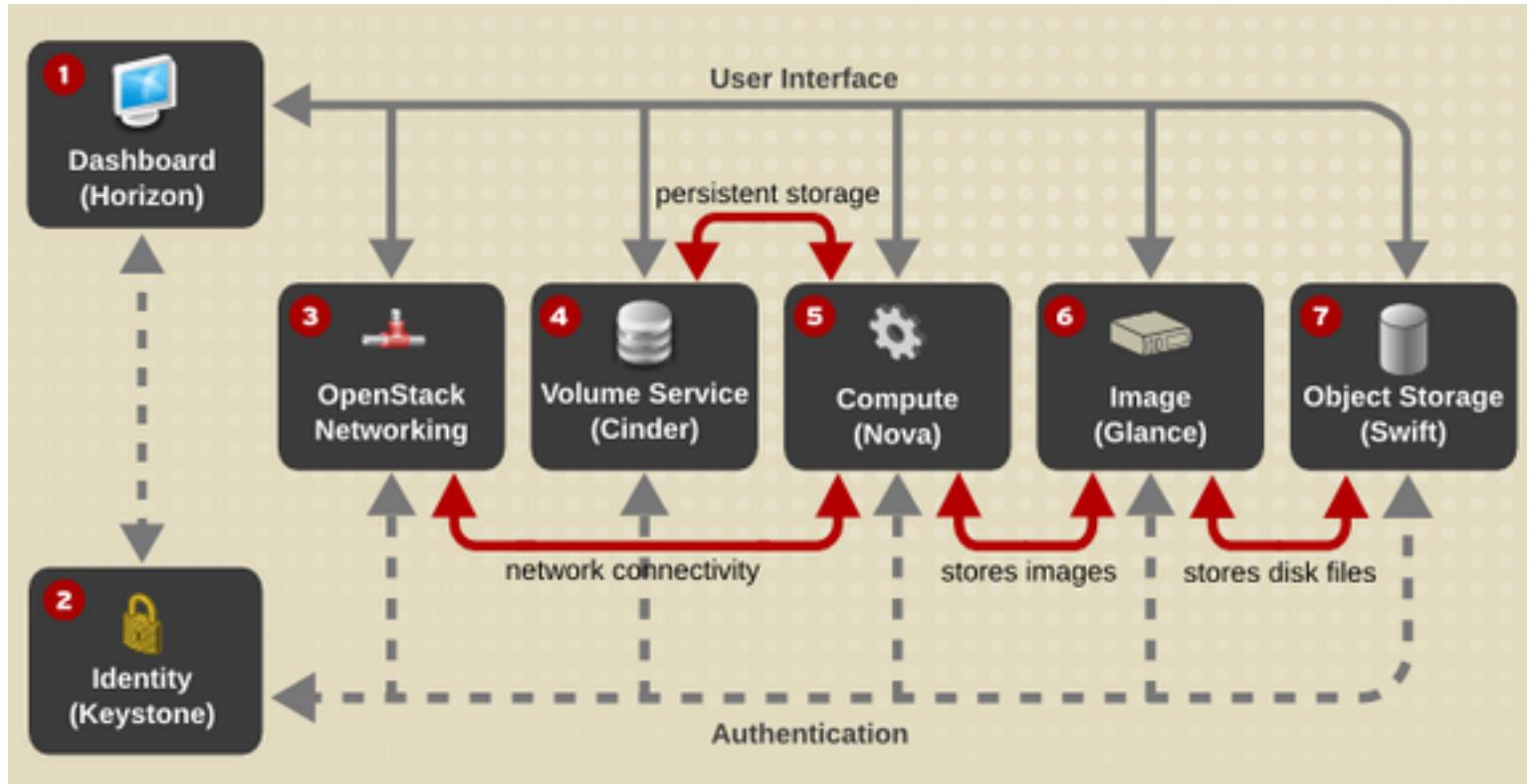
Nova (Compute)

Glance (Image Service)

Swift (Object Store)

Keystone (auth)

Horizon (Dashboard)



Openstack功能

分配和管理VM
卷管理
VM映像和文件对象等的管理
快照
用户验证，限额和租期管理

OpenStack 构建IaaS

OpenStack

实验部分参考：

tianyun_openstack.pdf

单机部署Openstack

1. 环境准备

- a. RHEL 6.4 x86_64 <建议全新安装>
- b. 磁盘预留空间>20G , 用于创建卷组cinder-volumes
- c. IP
- d. Gateway
- e. hostname: 必须是 FQDN 如 tianyun.openstack.com
- f. hosts: 解析主机名
- g. Iptables: iptables -F; service iptables save
- h. SELinux: Enforcing
- i. NetworkManager: service NetworkManager stop; chkconfig NetworkManager off

2. 安装KVM

- a. kvm install
- b. virsh iface-bridge eth0 br0
- c. 创建Linux虚拟机模板，建议镜像格式为qcow2 (可以在其它KVM上准备)
- d. 创建Windows虚拟机模板，使用半虚拟化驱动 (可以在其它KVM上准备)

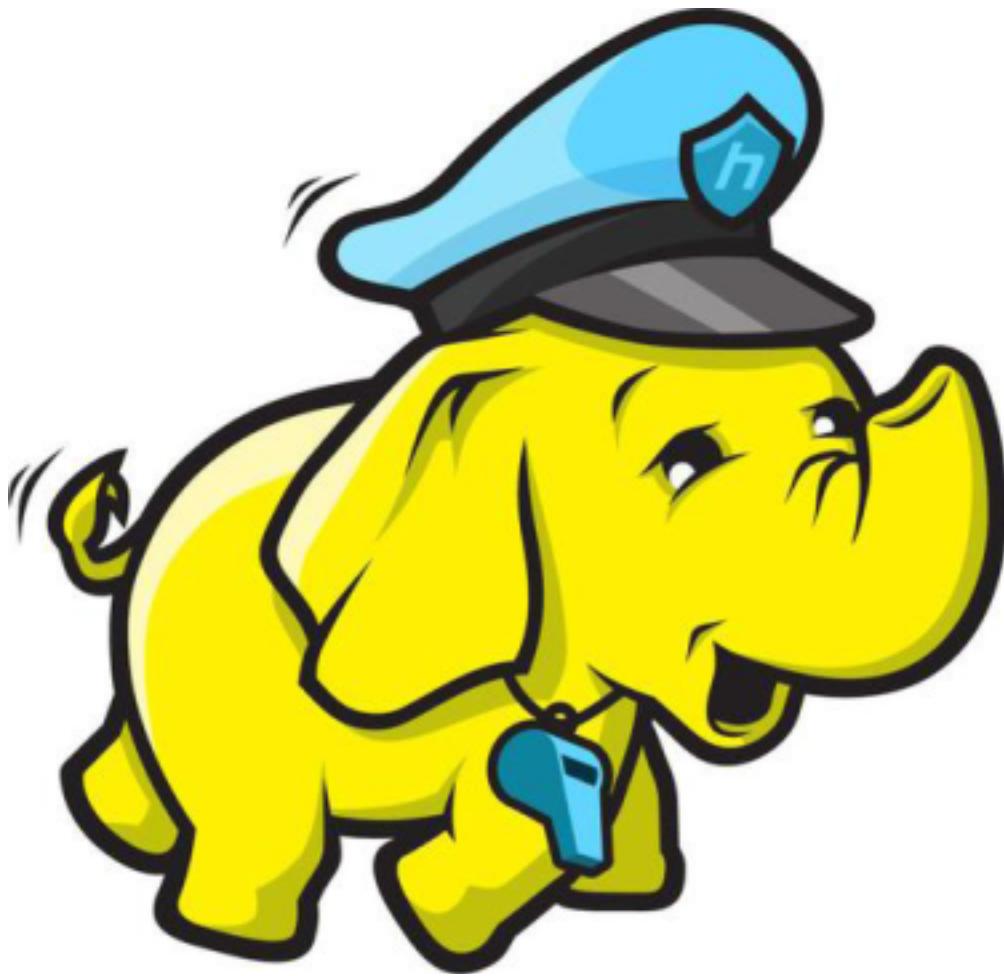
3. 部署Openstack

- a. OpenStack 安装
- b. 修改 admin 用户密码
- c. 通过 glance 管理镜像模版
- d. 创建实例(虚拟机)
- d. 为虚拟机创建快照
- e. 为虚拟机添加额外的卷
- f. 通过 nova 为虚拟机分配浮动 IP
- g. 配置虚拟机安全策略
- h. 创建自定义项目 Project

Hadoop

Hadoop简介

Hadoop



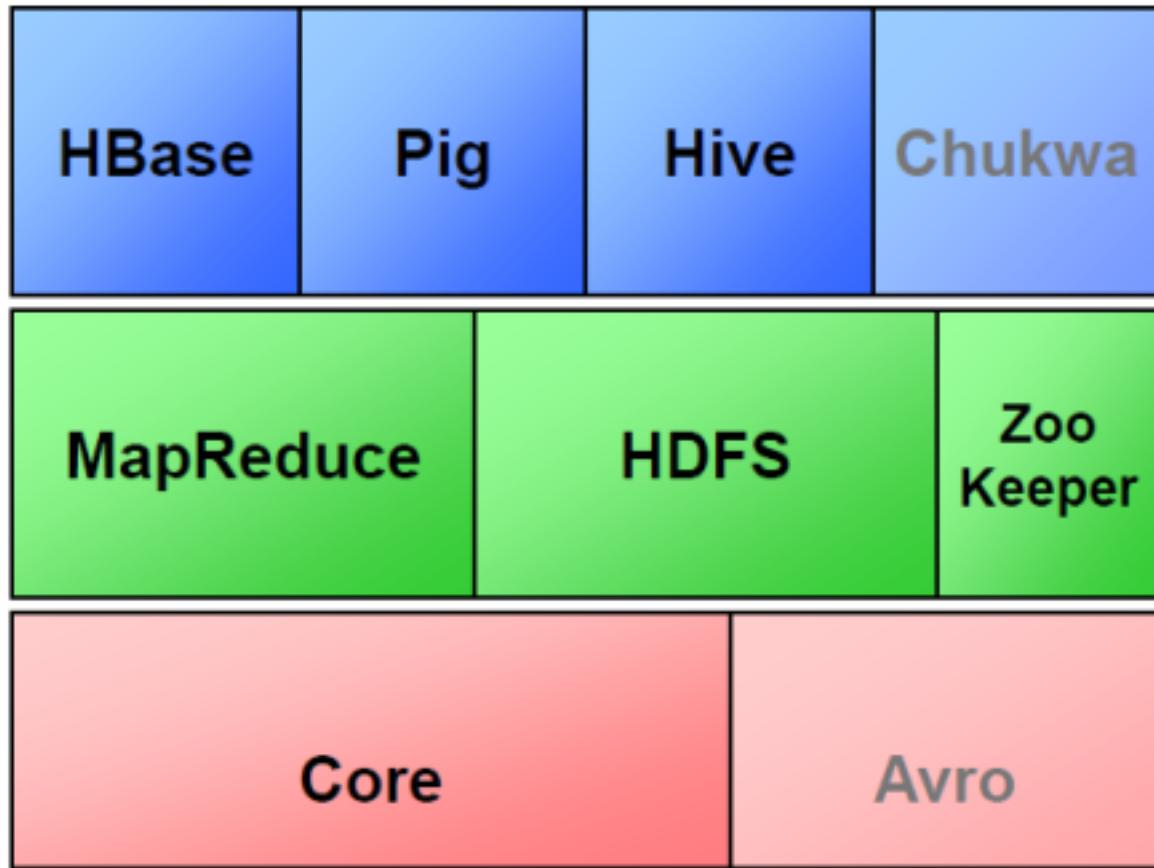
Hadoop思想之源：
Google

Google所面对的数据和计算难题
大量的网页如何存储？
搜索算法
Page-Rank计算问题
...

Google解决方案的关键技术和思想
GFS
Map-Reduce
Bigtable

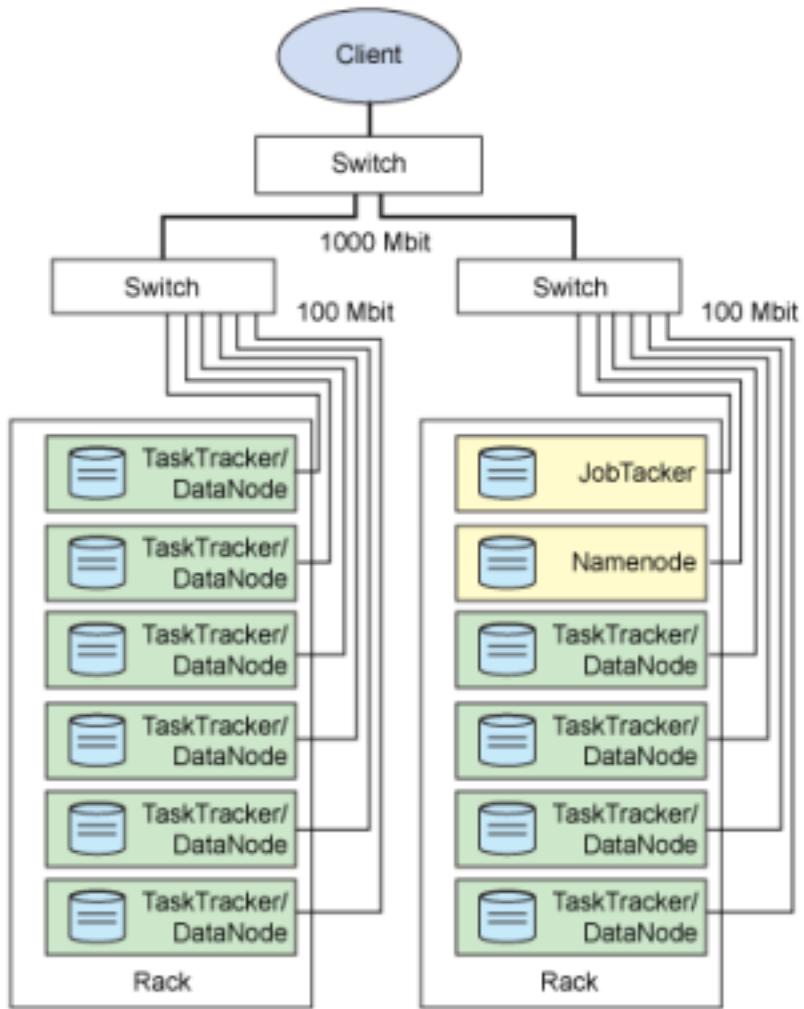
Hadoop目前达到的高度
实现云计算的事实标准开源软件
包含数十个具有强大生命力的子项目
已经能在数千节点上运行，处理数据量和排序时间不断打破世界纪录

Hadoop子项目家族



GFS -----> HDFS
Map-Reduce -----> MapReduce
Bigtable -----> HBase

Hadoop的架构



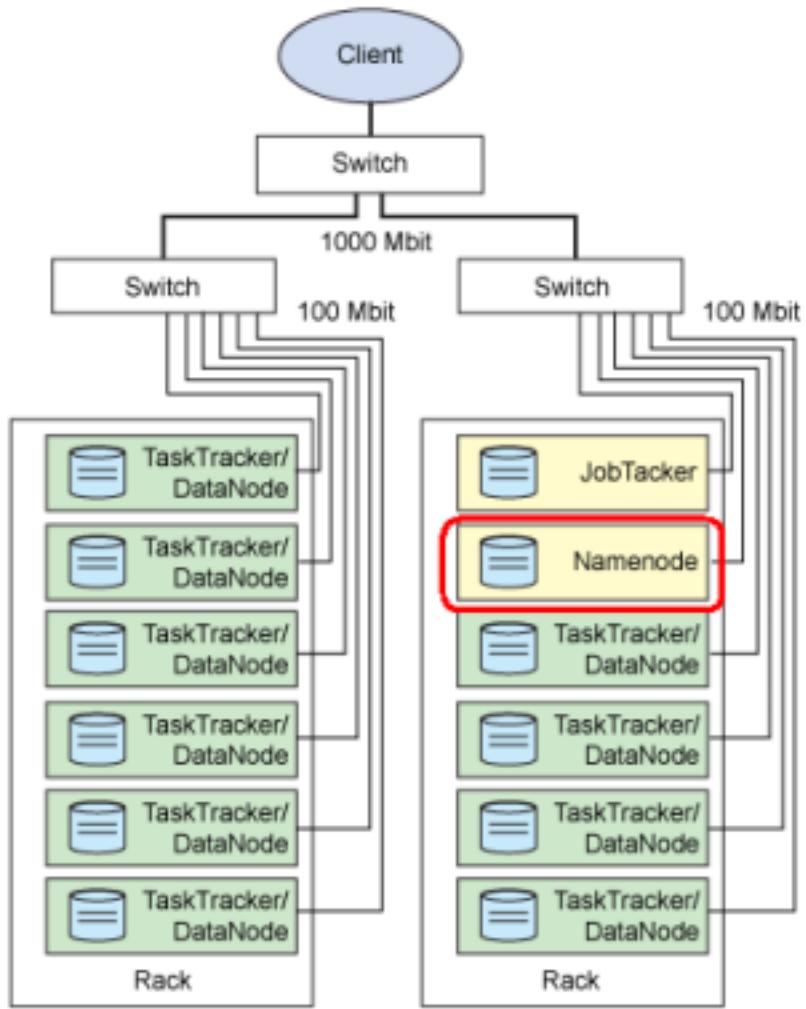
Namenode

HDFS的守护程序

记录文件是如何分割成数据块的，以及这些数据块被存储到哪些节点上

对内存和I/O进行集中管理

是个单点，发生故障将使集群崩溃



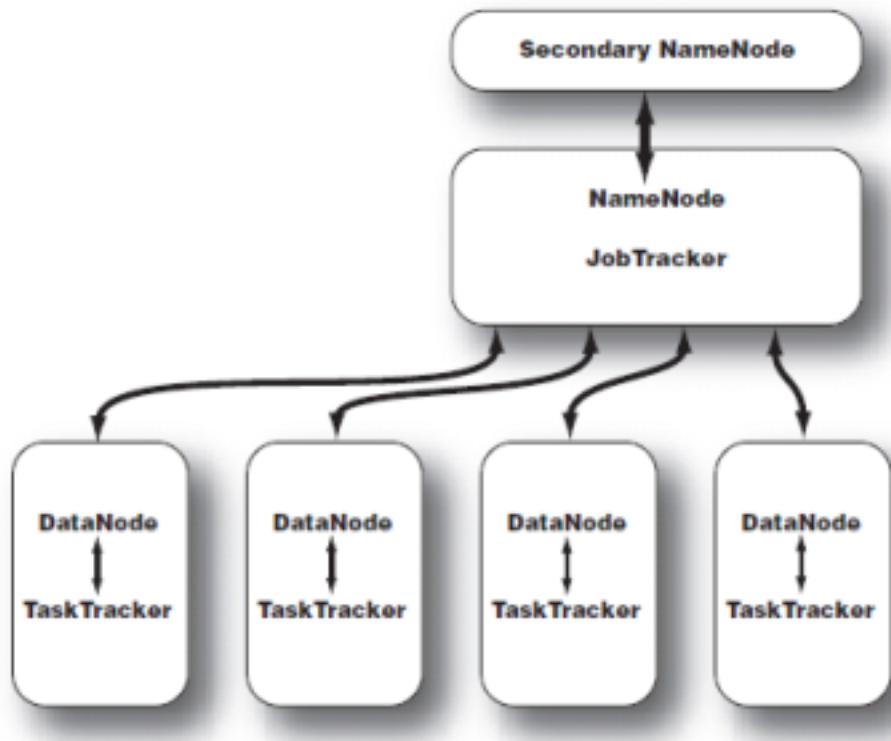
Secondary Namenode

监控HDFS状态的辅助后台程序

每个集群都有一个

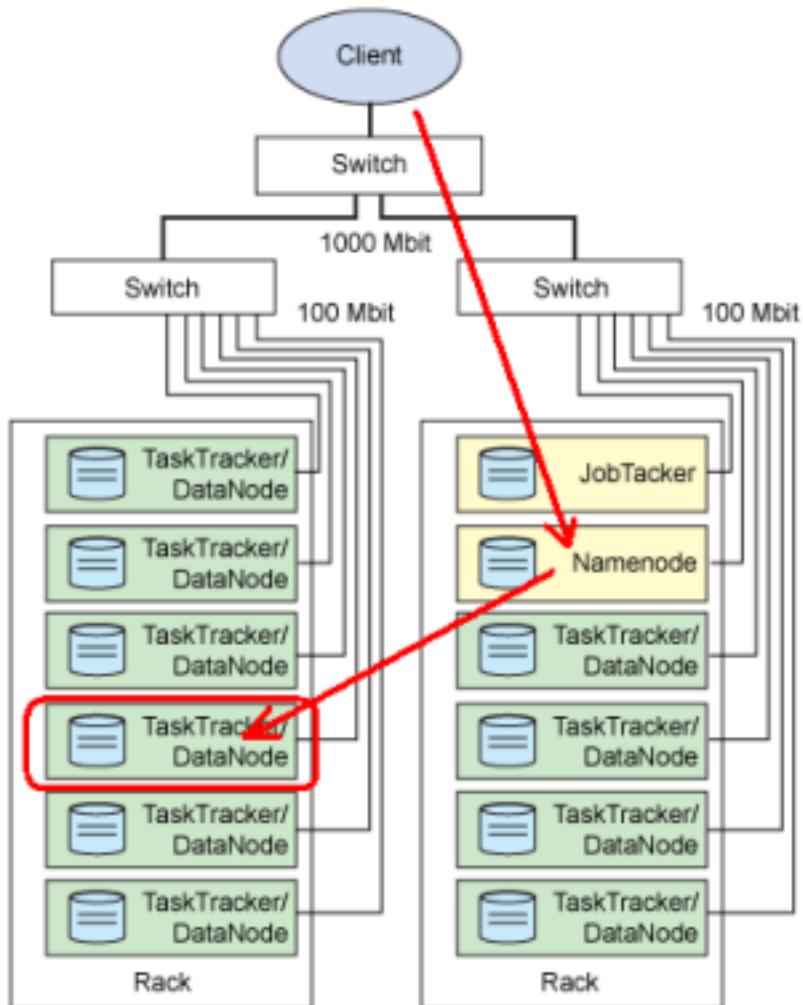
与NameNode进行通讯，定期保存HDFS元数据快照

当NameNode故障可以作为备用NameNode使用



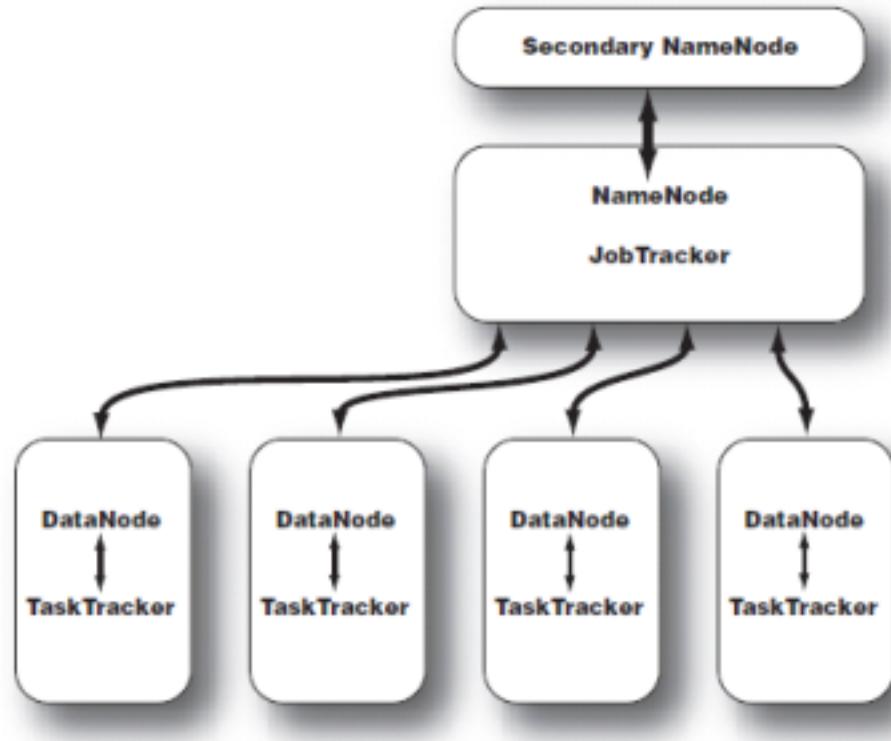
DataNode

每台从服务器都运行一个
负责把HDFS数据块读写到本地文件系统



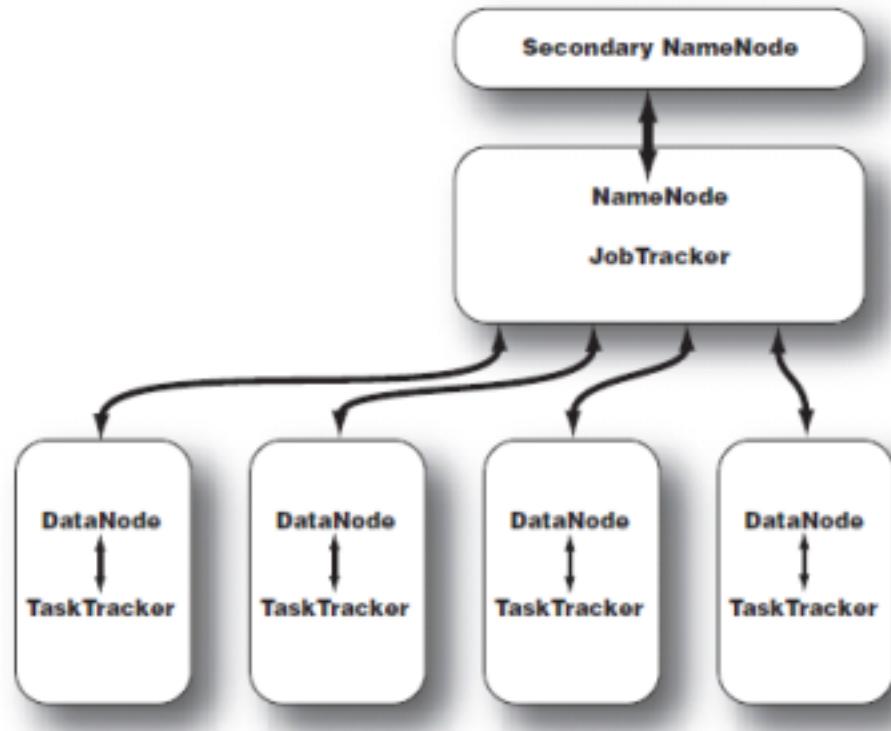
JobTracker

用于处理作业（用户提交代码）的后台程序
决定有哪些文件参与处理，然后切割task并分配节点
监控task，重启失败的task（于不同的节点）
每个集群只有一个JobTracker，位于Master节点



TaskTracker

位于slave节点上，与datanode结合（代码与数据一起的原则）
管理各自节点上的task（由jobtracker分配）
每个节点只有一个tasktracker，但一个tasktracker可以启动多个JVM，用于并行执行map或reduce任务
与jobtracker交互



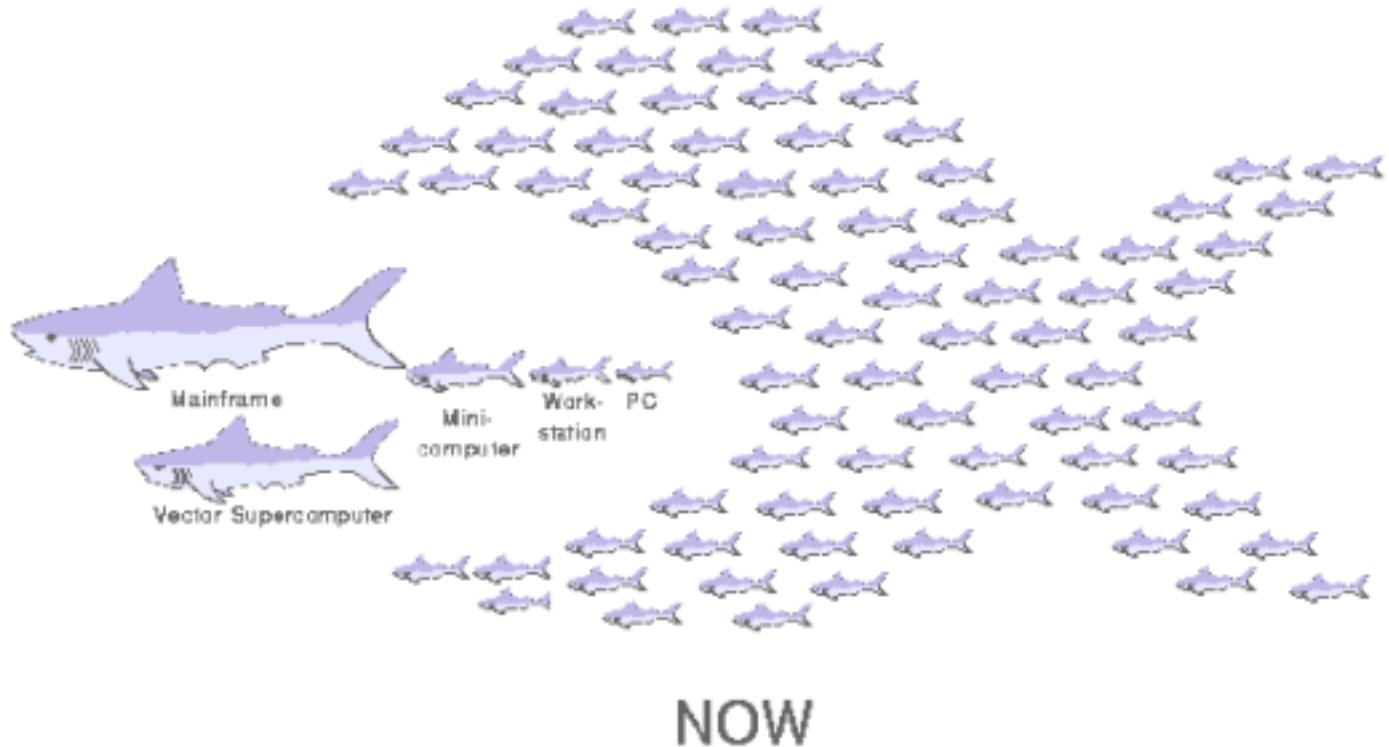
Master、Slave

Master : Namenode、Secondary Namenode、Jobtracker

Slave : Tasktracker、Datanode

Master不是唯一的

Hadoop的思想



Hadoop集群搭建

Hadoop

实验部分参考：

tianyun_openstack.pdf

一、拓扑结构

master
192.168.5.55

Master: Namenode、JobTracker

slave1
192.168.5.7

Slave: Datenode、TaskTracker

slave2
192.168.5.8

二、实现步骤

1. 配置hosts文件
2. 安装java运行环境
3. 建立hadoop运行用户
4. 配置ssh免密码连接
5. 下载并解压hadoop软件包
6. 配置hadoop (Namenode节点)
7. 复制hadoop到所有节点
8. 格式化HDFS (Namenode节点)
9. 启动hadoop (Namenode节点)
10. jps检查各个节点的进程

三、具体实现步骤 参考tianyun_hadoop.pdf

1. 配置hosts文件
2. 安装java运行环境 (所有节点)
3. 建立hadoop运行用户 (所有节点)
4. 配置ssh免密码连接 (hadoop用户)
5. 下载并解压hadoop软件包 (Namenode节点, hadoop用户)
6. 配置hadoop (Namenode节点)

hadoop-env.sh

core-site.xml

hdfs-site.xml

mapred-site.xml

masters

slaves

7. 复制hadoop到所有节点
8. 格式化HDFS (Namenode节点)
9. 启动hadoop (Namenode节点)
10. jps检查各个节点的进程

HDFS测试

HDFS文件操作

=====

查看帮助 (namenode)

```
[hadoop@master ~]$ hadoop dfs -h  
h: Unknown command  
Usage: java FsShell  
      [-ls <path>]  
      [-lsr <path>]
```

```
[-du <path>]
[-dus <path>]
[-count[-q] <path>]
[-mv <src> <dst>]
[-cp <src> <dst>]
[-rm [-skipTrash] <path>]
[-rmr [-skipTrash] <path>]
[-expunge]
[-put <localsrc> ... <dst>]
[-copyFromLocal <localsrc> ... <dst>]
[-moveFromLocal <localsrc> ... <dst>]
[-get [-ignoreCrc] [-crc] <src> <localdst>]
[-getmerge <src> <localdst> [addnl]]
[-cat <src>]
[-text <src>]
[-copyToLocal [-ignoreCrc] [-crc] <src> <localdst>]
[-moveToLocal [-crc] <src> <localdst>]
```

上传文件 (namenode)

```
[hadoop@master ~]$ hadoop dfs -ls
ls: Cannot access .: No such file or directory.

[hadoop@master ~]$ hadoop dfs -put /etc/passwd passwd.new
[hadoop@master ~]$ hadoop dfs -ls
Found 1 items
-rw-r--r-- 2 hadoop supergroup 1701 2013-10-15 12:29 /user/hadoop/passwd.new

[hadoop@master ~]$ mkdir dir1
[hadoop@master ~]$ echo "hello haoop" > dir1/file1
[hadoop@master ~]$ echo "hello tianyun" > dir1/file2
[hadoop@master ~]$ 
[hadoop@master ~]$ hadoop dfs -put dir1 dir1
[hadoop@master ~]$ hadoop dfs -ls
Found 2 items
drwxr-xr-x - hadoop supergroup 0 2013-10-15 12:30 /user/hadoop/dir1
-rw-r--r-- 2 hadoop supergroup 1701 2013-10-15 12:29 /user/hadoop/passwd.new
[hadoop@master ~]$ hadoop dfs -ls dir1
Found 2 items
-rw-r--r-- 2 hadoop supergroup 12 2013-10-15 12:30 /user/hadoop/dir1/file1
-rw-r--r-- 2 hadoop supergroup 14 2013-10-15 12:30 /user/hadoop/dir1/file2
```

查看文件cat (namenode)

```
[hadoop@master ~]$ hadoop dfs -ls
Found 2 items
drwxr-xr-x - hadoop supergroup 0 2013-10-15 12:30 /user/hadoop/dir1
-rw-r--r-- 2 hadoop supergroup 1701 2013-10-15 12:29 /user/hadoop/passwd.new
[hadoop@master ~]$ hadoop dfs -ls dir1
Found 2 items
-rw-r--r-- 2 hadoop supergroup 12 2013-10-15 12:30 /user/hadoop/dir1/file1
-rw-r--r-- 2 hadoop supergroup 14 2013-10-15 12:30 /user/hadoop/dir1/file2
[hadoop@master ~]$ hadoop dfs -cat dir1/file1
hello haoop
```

下载文件：

```
[hadoop@master ~]$ hadoop dfs -ls
Found 2 items
drwxr-xr-x - hadoop supergroup 0 2013-10-15 12:30 /user/hadoop/dir1
-rw-r--r-- 2 hadoop supergroup 1701 2013-10-15 12:29 /user/hadoop/passwd.new
[hadoop@master ~]$ hadoop dfs -get dir1 /tmp/dir1
[hadoop@master ~]$ ls /tmp/dir1/
file1 file2
```

从操作系统层面查看HDFS文件：(datanode)

```
[hadoop@slave1 ~]$ hostname
slave1
[hadoop@slave1 ~]$ ls /home/hadoop/data/
current detach in_use.lock storage tmp
[hadoop@slave1 ~]$
```

```
[hadoop@slave1 ~]$ ll /home/hadoop/data/current/
total 40
-rw-rw-r-- 1 hadoop hadoop 1701 Oct 15 12:27 blk_-4845428282999863001
-rw-rw-r-- 1 hadoop hadoop 23 Oct 15 12:27 blk_-4845428282999863001_1002.meta
-rw-rw-r-- 1 hadoop hadoop 4 Oct 15 12:20 blk_-5704500820372214196
-rw-rw-r-- 1 hadoop hadoop 11 Oct 15 12:20 blk_-5704500820372214196_1001.meta
-rw-rw-r-- 1 hadoop hadoop 14 Oct 15 12:29 blk_-6802264767937979856
-rw-rw-r-- 1 hadoop hadoop 11 Oct 15 12:29 blk_-6802264767937979856_1004.meta
-rw-rw-r-- 1 hadoop hadoop 12 Oct 15 12:29 blk_7480286643813923658
-rw-rw-r-- 1 hadoop hadoop 11 Oct 15 12:29 blk_7480286643813923658_1003.meta
-rw-rw-r-- 1 hadoop hadoop 387 Oct 15 12:40 dncp_block_verification.log.curr
-rw-rw-r-- 1 hadoop hadoop 156 Oct 15 12:19 VERSION
```

从namenode提供web接口查看文件

```
[hadoop@master ~]$ netstat -tnlp |grep :500
```

(Not all processes could be identified, non-owned process info
will not be shown, you would have to be root to see it all.)

tcp	0	0	:::50030	:::*	LISTEN	11492/java
tcp	0	0	:::50070	:::*	LISTEN	11272/java
tcp	0	0	:::50090	:::*	LISTEN	11422/java

```
[hadoop@master ~]$ firefox http://master:50070 &
```

Hadoop NameNode master:9000 - Mozilla Firefox

文件(E) 编辑(E) 查看(V) 历史(S) 书签(B) 工具(T) 帮助(H)

http://192.168.5.55:50070/dfshealth.jsp

完成

NameNode 'master:9000'

Started: Tue Oct 15 12:20:15 CST 2013

Version: 0.20.2, r911707

Compiled: Fri Feb 19 08:07:34 UTC 2010 by chrisdo

Upgrades: There are no upgrades in progress.

[Browse the filesystem](#)

[Namenode Logs](#)

Cluster Summary

13 files and directories, 4 blocks = 17 total. Heap Size is 55.81 MB / 888.94 MB (6%)

Configured Capacity :	48.83 GB
DFS Used :	120 KB
Non DFS Used :	3.07 GB
DFS Remaining :	45.76 GB
DFS Used%	0 %
DFS Remaining%	93.72 %

HDFS:/ - Mozilla Firefox

文件(E) 编辑(E) 查看(V) 历史(S) 书签(B) 工具(I) 帮助(H)

http://slave2:50075/browseDirectory.jsp?namenodeInfoPort=50070&dir=%2F

HDFS:/

Contents of directory [/](#)

Goto : go

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
home	dir				2013-10-15 12:21	rwxr-xr-x	hadoop	supergroup
user	dir				2013-10-15 12:29	rwxr-xr-x	hadoop	supergroup

[Go back to DFS home](#)

Local logs

[Log directory](#)

[Hadoop](#), 2013.

完成

HDFS:/user/hadoop - Mozilla Firefox

文件(E) 编辑(E) 查看(V) 历史(S) 书签(B) 工具(I) 帮助(H)

http://slave2:50075/browseDirectory.jsp?dir=%2Fuser%2Fhadoop&namenodeInfoPort=50070

HDFS:/user/hadoop

Contents of directory [/user/hadoop](#)

Goto : user/hadoop go

[Go to parent directory](#)

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
dir1	dir				2013-10-15 12:30	rwxr-xr-x	hadoop	supergroup
passwd.new	file	1.66 KB	2	64 MB	2013-10-15 12:29	rw-r--r--	hadoop	supergroup

[Go back to DFS home](#)

Local logs

[Log directory](#)

[Hadoop](#), 2013.

完成

Map-Reduce 测试

Map-Reduce 测试

分布式运算

```
=====
[hadoop@master ~]$ hadoop dfs -ls dir1
Found 2 items
-rw-r-r- 2 hadoop supergroup      12 2013-10-15 12:30 /user/hadoop/dir1/file1
-rw-r-r- 2 hadoop supergroup      14 2013-10-15 12:30 /user/hadoop/dir1/file2

[hadoop@master ~]$ hadoop dfs -cat dir1/file1
hello haoop
[hadoop@master ~]$ hadoop dfs -cat dir1/file2
hello tianyun
[hadoop@master ~]$ ls
hadoop-0.20.2 name tmp
[hadoop@master ~]$ ls hadoop-0.20.2/
bin    docs    ivy    NOTICE.txt
build.xml  hadoop-0.20.2-ant.jar  ivy.xml    README.txt
c++    hadoop-0.20.2-core.jar  lib    src
CHANGES.txt  hadoop-0.20.2-examples.jar  libreccordio webapps
conf    hadoop-0.20.2-test.jar  LICENSE.txt
contrib  hadoop-0.20.2-tools.jar  logs

[hadoop@master ~]$ hadoop jar hadoop-0.20.2/hadoop-0.20.2-examples.jar wordcount dir1 out1 //out1为输出目录
13/10/15 14:03:01 INFO input.FileInputFormat: Total input paths to process : 2
13/10/15 14:03:01 INFO mapred.JobClient: Running job: job_201310151220_0001
13/10/15 14:03:02 INFO mapred.JobClient: map 0% reduce 0%
13/10/15 14:03:09 INFO mapred.JobClient: map 100% reduce 0%
13/10/15 14:03:21 INFO mapred.JobClient: map 100% reduce 100%
13/10/15 14:03:23 INFO mapred.JobClient: Job complete: job_201310151220_0001
13/10/15 14:03:23 INFO mapred.JobClient: Counters: 17
13/10/15 14:03:23 INFO mapred.JobClient: Job Counters
13/10/15 14:03:23 INFO mapred.JobClient: Launched reduce tasks=1
13/10/15 14:03:23 INFO mapred.JobClient: Launched map tasks=2
13/10/15 14:03:23 INFO mapred.JobClient: Data-local map tasks=2
13/10/15 14:03:23 INFO mapred.JobClient: FileSystemCounters
13/10/15 14:03:23 INFO mapred.JobClient: FILE_BYTES_READ=56
13/10/15 14:03:23 INFO mapred.JobClient: HDFS_BYTES_READ=26
13/10/15 14:03:23 INFO mapred.JobClient: FILE_BYTES_WRITTEN=182
13/10/15 14:03:23 INFO mapred.JobClient: HDFS_BYTES_WRITTEN=26
13/10/15 14:03:23 INFO mapred.JobClient: Map-Reduce Framework
13/10/15 14:03:23 INFO mapred.JobClient: Reduce input groups=3
13/10/15 14:03:23 INFO mapred.JobClient: Combine output records=4
13/10/15 14:03:23 INFO mapred.JobClient: Map input records=2
13/10/15 14:03:23 INFO mapred.JobClient: Reduce shuffle bytes=62
13/10/15 14:03:23 INFO mapred.JobClient: Reduce output records=3
13/10/15 14:03:23 INFO mapred.JobClient: Spilled Records=8
13/10/15 14:03:23 INFO mapred.JobClient: Map output bytes=42
13/10/15 14:03:23 INFO mapred.JobClient: Combine input records=4
```

```
13/10/15 14:03:23 INFO mapred.JobClient: Map output records=4  
13/10/15 14:03:23 INFO mapred.JobClient: Reduce input records=4
```

```
[hadoop@master ~]$ hadoop dfs -ls  
Found 3 items  
drwxr-xr-x - hadoop supergroup 0 2013-10-15 12:30 /user/hadoop/dir1  
drwxr-xr-x - hadoop supergroup 0 2013-10-15 14:03 /user/hadoop/out1  
-rw-r--r-- 2 hadoop supergroup 1701 2013-10-15 12:29 /user/hadoop/passwd.new  
[hadoop@master ~]$ hadoop dfs -ls out1  
Found 2 items  
drwxr-xr-x - hadoop supergroup 0 2013-10-15 14:03 /user/hadoop/out1/_logs  
-rw-r--r-- 2 hadoop supergroup 26 2013-10-15 14:03 /user/hadoop/out1/part-r-00000  
[hadoop@master ~]$ hadoop dfs -cat out1/part-r-00000  
hadoop 1  
hello 2  
tianyun 1
```

从jobTracker提供web接口Map-Reduce的相关信息

```
[hadoop@master ~]$ firefox http://master:50030 &
```

The screenshot shows the 'master Hadoop Map/Reduce Administration' page. At the top, it displays system information: State: RUNNING, Started: Tue Oct 15 12:20:19 CST 2013, Version: 0.20.2, r911707, Compiled: Fri Feb 19 08:07:34 UTC 2010 by chrisdo, Identifier: 201310151220. Below this is a 'Cluster Summary' section showing Heap Size is 55.81 MB/888.94 MB. A table provides cluster statistics: Maps (0), Reduces (0), Total Submissions (1), Nodes (2), Map Task Capacity (4), Reduce Task Capacity (4), Avg. Tasks/Node (4.00), and Blacklisted Nodes (0). The 'Scheduling Information' section includes a table for Queue Name and Scheduling Information, which is currently empty. A status message at the bottom says '完成'.

Maps	Reduces	Total Submissions	Nodes	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes
0	0	1	2	4	4	4.00	0

Web接口查看集群

从namenode提供web接口查看文件 HDFS

```
[hadoop@master ~]$ firefox http://master:50070 &
```

从jobTracker提供web接口 Map-Reduce的相关信息 Map-Reduce

```
[hadoop@master ~]$ firefox http://master:50030 &
```

LB--

LB综合项目一

LB综合项目一

LVS (DR) + Keepalived + LAMP + GlusterFS + MySQL + Memcache

拓扑结构

Director : **VIP : 192.168.122.100**
DIP: 192.168.122.2 192.168.122.3

Real Server : RIP : 192.168.122.10 192.168.122.20 192.168.122.30

VIP : 192.168.122.100 192.168.122.100 192.168.122.100

MySQL+Memcache : 192.168.122.200

实施步骤

一、LB :

```
[root@director1 ~]# yum -y install ipvsadm keepalived  
[root@web1 ~]# yum -y install gd httpd mysql mysql-devel mysql-libs php php-mysql php-gd php-pecl-memcache
```

二、存储：Real Server

存储网站代码及用户上传文件：GlusterFS

```
[root@web1 ~]# gluster volume create web-data replica 3 transport tcp web1:/web web2:/web web3:/web force  
volume create: web-data: success: please start the volume to access data
```

```
[root@web1 ~]# mount -t glusterfs -o backup-volfile-servers=web2:web3 web1:/web-data /var/www/html/  
[root@web2 ~]# mount -t glusterfs -o backup-volfile-servers=web1:web3 web2:/web-data /var/www/html/  
[root@web3 ~]# mount -t glusterfs -o backup-volfile-servers=web1:web2 web3:/web-data /var/www/html/
```

存储关系数及Session: Memcache

```
[root@mysql-memcache ~]# yum -y install memcached  
[root@mysql-memcache ~]# service memcached start  
[root@mysql-memcache ~]# chkconfig memcached on
```

```
[root@web1 ~]# vim /etc/php.ini  
[Session]
```

```
session.save_handler = memcache  
memcache.hash_strategy = "consistent"  
session.cookie_domain = ".tianyun.com"  
session.save_path = "tcp://192.168.122.100:11211"
```

存储关系数及Session: MySQL

```
[root@mysql-memcache ~]# yum -y install mysql-server  
[root@mysql-memcache ~]# service mysqld start  
[root@mysql-memcache ~]# chkconfig mysqld on
```

mysql授权用户访问

```
mysql> create database wordpress;  
mysql> grant all on wordpress.* to wordpress@'192.168.122.%' identified by '123';  
mysql> flush privileges;
```

三、上传网站并测试

```
wordpress -----> real server: /var/www/html  
[root@tianyun ~]# ping www.tianyun.com  
PING www.tianyun.com (192.168.122.100) 56(84) bytes of data.  
64 bytes from www.tianyun.com (192.168.122.100): icmp_seq=1 ttl=64 time=0.978 ms  
  
[root@tianyun ~]# fireforx www.tianyun.com
```

LB Nginx Tomcat (七层)

Nginx实现七层的负载均衡 Tomcat

```
=====  
client  
192.168.122.1  
  
[nginx LB A] [nginx LB B]  
192.168.122.10 192.168.122.20  
  
[tomcat1] [tomcat2] [tomcat3]  
122.30 122.40 122.50  
  
[NFS/MySQL 122.60]
```

一. 准备工作 (集群中所有主机)

IP, hostname, hosts, iptables, SELinux

二、存储配置NAS (NFS)

```
[root@nfsmysql ~]# mkdir /nas  
[root@nfsmysql ~]# chmod 777 /nas/  
[root@nfsmysql ~]# vim /etc/exports  
/nas 192.168.122.0/24(rw,sync)  
[root@nfsmysql ~]# service nfs restart  
[root@nfsmysql ~]# chkconfig nfs on
```

三、数据库MySQL

```
[root@nfsmysql ~]# yum -y install mysql-server  
[root@nfsmysql ~]# service mysqld start
```

```
[root@nfsmysql ~]# /usr/bin/mysqladmin -u root password '123'  
[root@nfsmysql ~]# mysql -p123  
mysql> select user,password,host from mysql.user;  
mysql> delete from mysql.user where password='';  
mysql> create database sns;  
mysql> grant all on sns.* to sns@'192.168.122.%' IDENTIFIED BY '123';  
mysql> flush privileges;
```

四、tomcat配置 (所有节点相同操作)

1. 安装tomcat参考ULE
2. 挂载存储到tomcat网站目录

```
[root@tomcat1 ~]# showmount -e 192.168.122.60  
Export list for 192.168.122.60:  
/nas 192.168.122.0/24  
[root@tomcat1 ~]# mount 192.168.122.60:/nas /usr/local/tomcat/webapps/  
[root@tomcat1 ~]# ls /usr/local/tomcat/webapps/
```

3. 上传网站并连接数据库(在任意一个节点上)

上传网站：

```
[root@tomcat1 ~]# mkdir /usr/local/tomcat/webapps/ROOT  
[root@tomcat1 jsp]# unzip ejforum-2.3.zip  
[root@tomcat1 jsp]# cp -rf ejforum-2.3/ejforum/* /usr/local/tomcat/webapps/ROOT/
```

导入表结构：

```
[root@tomcat1 jsp]# cd ejforum-2.3/install/script/  
[root@tomcat1 script]# ls  
easyjforum_hsqldb.sql easyjforum_mysql.sql easyjforum_oracle.sql easyjforum_sqlserver.sql upgrade  
[root@tomcat1 script]# yum -y install mysql  
[root@tomcat1 script]# mysql -h 192.168.122.60 -usns -p123 sns < easyjforum_mysql.sql
```

连接数据库：

```
[root@tomcat1 script]# vim /usr/local/tomcat/webapps/ROOT/WEB-INF/conf/config.xml  
<database maxActive="10" maxIdle="10" minIdle="2" maxWait="10000"  
    username="sns" password="123"  
    driverClassName="com.mysql.jdbc.Driver"  
    url="jdbc:mysql://192.168.122.60:3306/sns?  
characterEncoding=gbk&amp;autoReconnect=true&amp;autoReconnectForPools=true&amp;zeroDateTimeBehavior=convertToNull  
    sqlAdapter="sql.MysqlAdapter"/>
```

4. 重启所有节点的tomcat

```
[root@tomcat1 script]# service tomcat stop  
[root@tomcat1 script]# service tomcat start
```

五、阶段测试

```
# firefox http://tomcat1:8080  
# firefox http://tomcat2:8080
```

六、调度器Nginx

1. 安装

```
[root@xen01 nginx-1.2.0]# yum -y install pcre-devel  
[root@nginx ~]# tar zxvf nginx-1.0.9.tar.gz  
[root@nginx nginx-1.0.9]# cd /usr/local/nginx-1.0.9  
[root@nginx nginx-1.0.9]# ./configure --prefix=/usr/local/nginx  
[root@nginx nginx-1.0.9]# make && make install
```

2. 配置使用Nginx实现LB

```
[root@xen01 ~]# sed -i '/^[*]*#/d; /^$/d' /usr/local/nginx/conf/nginx.conf  
[root@nginx conf]# cat /usr/local/nginx/conf/nginx.conf  
#user nobody;  
worker_processes 1;  
  
events {  
    worker_connections 1024;  
}  
  
http {  
    upstream tomcat_lb {  
        ip_hash;  
        server 192.168.122.30:8080;  
        server 192.168.122.40:8080;  
        server 192.168.122.50:8080;  
    }  
}
```

```

server {
    listen     80;
    server_name localhost;

    location / {
        root   html;
        index  index.jsp index.html index.htm;
        proxy_pass http://tomcat_lb;
        proxy_set_header Host $host;
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
        root   html;
    }
}

[root@nginx conf]# /usr/local/nginx/sbin/nginx -t
nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok
nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful
[root@nginx conf]# /usr/local/nginx/sbin/nginx
[root@nginx conf]# netstat -tnlp | grep :80
tcp      0      0 0.0.0.0:80          0.0.0.0:*          LISTEN    11276/nginx
tcp      0      0 0.0.0.0:8084       0.0.0.0:*          LISTEN    5214/stunnel

```

3. 测试：

4. 找客户测试结果，并在nginx上抓包，看是否能LB分发，又是如何分发的？

LB Apache+Tomcat

Apache+Tomcat集群 (session同步)

```
=====
项目环境：
Apache : 192.168.0.10
Tomcat1 : 192.168.0.100
Tomcat2 : 192.168.0.200
```

一、准备软件包

pcre-8.31.tar.gz
 apr-1.4.6.tar.gz
 apr-util-1.5.1.tar.gz
 httpd-2.4.3.tar.gz
 jdk-7u11-linux-i586.rpm
 apache-tomcat-7.0.34.tar.gz
 mysql-connector-java-5.0.8.tar.gz

二、安装Apache (LAMP可选)

Apache : 192.168.0.10
 [root@apache ~]# service httpd stop
 [root@apache ~]# chkconfig httpd off
 [root@apache ~]# netstat -tnlp |grep :80

= = 安装pcre

[root@apache ~]# tar xvf pcre-8.31.tar.gz

```

[root@apache ~]# cd pcre-8.31
[root@apache pcre-8.31]# ./configure --prefix=/usr/local/pcre
[root@apache pcre-8.31]# make && make install

== 安装Apache
[root@apache ~]# tar xf httpd-2.4.3.tar.gz
[root@apache ~]# tar xf apr-1.4.6.tar.gz
[root@apache ~]# tar xf apr-util-1.5.1.tar.gz
[root@apache ~]# mv apr-1.4.6 httpd-2.4.3/src/lib/apr
[root@apache ~]# mv apr-util-1.5.1 httpd-2.4.3/src/lib/apr-util
[root@apache ~]# cd httpd-2.4.3
[root@apache httpd-2.4.3]# ./configure \
--prefix=/usr/local/apache2 \
--enable-so \
--with-included-apr \
--enable-ssl \
--with-pcre=/usr/local/pcre
[root@apache httpd-2.4.3]# make && make install
[root@apache ~]# /usr/local/apache2/bin/apachectl start
[root@apache ~]# echo "/usr/local/apache2/bin/apachectl start" >> /etc/rc.local
[root@apache ~]#
[root@apache ~]# netstat -tnlp |grep 80
tcp      0      0 ::80                  :::*                  LISTEN      1960/httpd

```

三、安装Tomcat (支持连接Mysql)

```

Tomcat1 : 192.168.0.100
Tomcat2 : 192.168.0.200
[root@tomcat1 jsp]# rpm -qa |grep java
java-1.4.2-gcj-compat-1.4.2.0-40jpp.115
[root@tomcat1 jsp]# rpm -e --nodeps java-1.4.2-gcj-compat-1.4.2.0-40jpp.115
[root@tomcat1 jsp]# java -version
-bash: /usr/bin/java: 没有那个文件或目录

```

```

== 安装JDK
[root@tomcat1 jsp]# rpm -ivh jdk-7u11-linux-i586.rpm
[root@tomcat1 jsp]# java -version
java version "1.7.0_11"
Java(TM) SE Runtime Environment (build 1.7.0_11-b21)
Java HotSpot(TM) Client VM (build 23.6-b04, mixed mode, sharing)

```

```

[root@tomcat1 jsp]# ls /usr/java/jdk1.7.0_11
[root@tomcat1 jsp]# vim /etc/profile           //设置环境变量
JAVA_HOME=/usr/java/jdk1.7.0_11
PATH=$PATH:$JAVA_HOME/bin
export JAVA_HOME PATH
[root@tomcat1 jsp]# source /etc/profile
[root@tomcat1 jsp]# echo $PATH
/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin:/usr/java/jdk1.7.0_11/bin

```

```

== Tomcat
[root@tomcat1 jsp]# tar xvf apache-tomcat-7.0.34.tar.gz -C /usr/local/
[root@tomcat1 jsp]# cd /usr/local/
[root@tomcat1 local]# ln -sv apache-tomcat-7.0.34 tomcat
创建指向“apache-tomcat-7.0.34”的符号链接“tomcat”

```

```

[root@tomcat1 local]# vim /etc/profile           //定义Tomcat环境变量
CATALINA_HOME=/usr/local/tomcat
export CATALINA_HOME
[root@tomcat1 local]# source /etc/profile
[root@tomcat1 local]# env |grep HOME
CATALINA_HOME=/usr/local/tomcat
JAVA_HOME=/usr/java/jdk1.7.0_11
HOME=/root

```

```

[root@tomcat1 local]# /usr/local/tomcat/bin/catalina.sh start           //启动Tomcat
Using CATALINA_BASE: /usr/local/tomcat
Using CATALINA_HOME: /usr/local/tomcat
Using CATALINA_TMPDIR: /usr/local/tomcat/temp
Using JRE_HOME:      /usr/java/jdk1.7.0_11

```

Using CLASSPATH: /usr/local/tomcat/bin/bootstrap.jar:/usr/local/tomcat/bin/tomcat-juli.jar

```
[root@tomcat1 ~]# netstat -tnlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp      0      0 ::ffff:127.0.0.1:8005     :::*                  LISTEN      2920/java
tcp      0      0 :::8009                 :::*                  LISTEN      2920/java
tcp      0      0 :::8080                 :::*                  LISTEN      2920/java
tcp      0      0 :::80                  :::*                  LISTEN      2135/httpd
```

=====

System V脚本 可选项

```
# [root@tomcat1 ~]# vim /etc/init.d/tomcat
#!/bin/bash
# Init file for Tomcat server daemon
#
# chkconfig: 2345 96 14
# description: Tomcat server daemon
JAVA_OPTS='-Xms64m -Xmx128m'
JAVA_HOME=/usr/java/jdk1.7.0_11
CATALINA_HOME=/usr/local/tomcat
export JAVA_OPTS JAVA_HOME CATALINA_HOME
exec $CATALINA_HOME/bin/catalina.sh $*
```

```
[root@tomcat1 ~]# chmod a+x /etc/init.d/tomcat
```

```
[root@tomcat1 ~]# chkconfig tomcat --list
```

tomcat 服务支持 chkconfig, 但它在任何级别中都没有被引用(运行“chkconfig --add tomcat”)

```
[root@tomcat1 ~]#
```

```
[root@tomcat1 ~]# chkconfig --add tomcat
```

```
[root@tomcat1 ~]# chkconfig tomcat --list
```

tomcat 0:关闭 1:关闭 2:启用 3:启用 4:启用 5:启用 6:关闭

```
[root@tomcat1 ~]# chkconfig tomcat on
```

```
[root@tomcat1 ~]#
```

```
[root@tomcat1 ~]# service tomcat stop
```

Using CATALINA_BASE: /usr/local/tomcat

Using CATALINA_HOME: /usr/local/tomcat

Using CATALINA_TMPDIR: /usr/local/tomcat/temp

Using JRE_HOME: /usr/java/jdk1.7.0_11

Using CLASSPATH: /usr/local/tomcat/bin/bootstrap.jar:/usr/local/tomcat/bin/tomcat-juli.jar

```
[root@tomcat1 ~]# service tomcat start
```

Using CATALINA_BASE: /usr/local/tomcat

Using CATALINA_HOME: /usr/local/tomcat

Using CATALINA_TMPDIR: /usr/local/tomcat/temp

Using JRE_HOME: /usr/java/jdk1.7.0_11

Using CLASSPATH: /usr/local/tomcat/bin/bootstrap.jar:/usr/local/tomcat/bin/tomcat-juli.jar

=====

= 测试 : http://192.168.2.251:8080/

= 安装tomcat-native

```
[root@tomcat1 ~]# yum groupinstall -y "Development tools" "Development libraries"
```

```
[root@tomcat1 ~]# yum -y install openssl-devel apr-devel
```

```
[root@tomcat1 jsp]# cd /usr/local/tomcat/bin/
```

```
[root@tomcat1 bin]# ls
```

```
bootstrap.jar commons-daemon-native.tar.gz digest.bat shutdown.sh tool-wrapper.bat
catalina.bat configtest.bat digest.sh startup.bat tool-wrapper.sh
catalina.sh configtest.sh setclasspath.bat startup.sh version.bat
catalina-tasks.xml cpappend.bat setclasspath.sh tomcat-juli.jar version.sh
commons-daemon.jar daemon.sh shutdown.bat tomcat-native.tar.gz
```

```
[root@tomcat1 bin]# tar xf tomcat-native.tar.gz
```

```
[root@tomcat1 bin]# cd tomcat-native-1.1.24-src/jni/native/
```

```
[root@tomcat1 native]# ls
```

```
build build-outputs.mk include NMAKEmakefile srclib
buildconf config.layout libtcnative.dsp NMAKEmakefile.inc tcnative.dsp
build.conf configure libtcnative.dsw os tcnative.pc.in
BUILDING configure.in Makefile.in src tcnative.spec
```

```
[root@tomcat1 native]# ./configure --with-apr=/usr/ --with-ssl --with-java-home=/usr/java/jdk1.7.0_11/
```

```
[root@tomcat1 native]# make && make install
```

```
[root@tomcat1 native]# echo "/usr/local/apr/lib" > /etc/ld.so.conf.d/apr.conf
[root@tomcat1 native]# ldconfig //重新加载库文件
=====

== = 为Tomcat提供连接Mysql的文件
[root@tomcat1 jsp]# tar xf mysql-connector-java-5.0.8.tar.gz
[root@tomcat1 jsp]# cd mysql-connector-java-5.0.8
[root@tomcat1 mysql-connector-java-5.0.8]# ls
build.xml COPYING docs mysql-connector-java-5.0.8-bin.jar README.txt
CHANGES debug EXCEPTIONS-CONNECTOR-J README src
[root@tomcat1 mysql-connector-java-5.0.8]# cp mysql-connector-java-5.0.8-bin.jar /usr/local/tomcat/lib/

[root@tomcat1 ~]# service tomcat stop
[root@tomcat1 ~]# service tomcat start
```

注：节点Tomcat2配置完全相同

三、整合Apache + Tomcat

Apache : 192.168.0.10

= = 安装JK模块

```
[root@apache ~]# tar xvf tomcat-connectors-1.2.31-src.tar.gz
[root@apache ~]# cd tomcat-connectors-1.2.31-src/native/
[root@apache native]# chmod 755 buildconf.sh
[root@apache native]# ./buildconf.sh
[root@apache native]# ./configure --with-apxs=/usr/local/apache2/bin/apxs //关联Apache文件apxs
[root@apache native]# make && make install
[root@apache native]# ls /usr/local/apache2/modules/mod_jk.so
/usr/local/apache2/modules/mod_jk.so
```

```
= = /usr/local/apache2/conf/httpd.conf //Apache配置文件
DirectoryIndex index.html index.jsp
LoadModule jk_module modules/mod_jk.so //加载mod_jk.so
jkWorkersFile conf/workers.properties //JK模块配置文件
JkMount /* controller //指定那些请求交给tomcat处理,"controller"为在
workers.properties里指定的负载分配控制器名
jkLogFile logs/mod_jk.log //日志文件 可选
jkLogLevel info //日志级别 可选

= = /usr/local/apache2/conf/workers.properties //JK模块配置文件
#所有节点列表，其中controller是一个逻辑节点，负责负载均衡控制，
#如果JkMount中的URL指定给了controller就表示这个请求会被自动散列到某个物理节点上。
#注意：真正负责处理请求的tomcat的名称（这里就是tomcat1,tomcat2必须于它们在conf/server.xml文件中配置的jvmRoute的属性值是一致的！）

worker.list=controller #worker.list = controller,tomcat1,tomcat2
#=====tomcat1=====
worker.tomcat1.port=8009 #ajp13端口号，在tomcat下server.xml配置，默认8009
worker.tomcat1.host=192.168.0.100 #tomcat的主机地址
worker.tomcat1.type=ajp13
worker.tomcat1.lbfactor=1 #server的加权比重，值越高，分得的请求越多

#=====tomcat2=====
worker.tomcat2.port=8009 #ajp13 端口号，在tomcat下server.xml配置，默认8009
worker.tomcat2.host=192.168.0.200 #tomcat的主机地址
worker.tomcat2.type=ajp13
worker.tomcat2.lbfactor=1 #server的加权比重，值越高，分得的请求越多

#=====controller,负载均衡控制器=====
worker.controller.type=lb #指定分担请求的tomcat，旧版本中的balanced_workers,已不再推荐使用
worker.controller.balance_workers=tomcat1,tomcat2 #sticky_session为1表示，当某一client的session创建之后，后续由该客户
worker.controller.sticky_session_force=1 #端发起的请求，也就是这个session的所有请求都始终由第一次处理该请求
worker.controller.sticky_session=false #的结点负责处理（除非该结点挂掉）
```

其它配置参考：

worker.controller.type=lb

```
worker.controller.balanced_workers=tomcat1,tomcat2,tomcat3  
worker.controller.sticky_session=false  
worker.controller.sticky_session_force=1  
#worker.controller.sticky_session=1
```

四、配置Tomcat <server.xml>

Tomcat1 : 192.168.0.100

```
<Engine name="Catalina" defaultHost="localhost" jvmRoute="tomcat1"> //jvmRoute名必须和前面workers.properties中的节点名一致
```

```
<Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/> //去掉注释，支持集群功能
```

Tomcat1 : 192.168.0.200

```
<Engine name="Catalina" defaultHost="localhost" jvmRoute="tomcat2"> //jvmRoute名必须和前面workers.properties中的节点名一致
```

```
<Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/> //去掉注释，支持session的集群复制
```

五、在两个Tomcat节点上建立测试项目

修改conf/web.xml文件，添加 distributable 属性，表示Tomcat要为此web应用复制 Session。在web.xml文件最后加上distributable,具体配置如下：

```
= = conf/web.xml  
<welcome-file-list>  
    <welcome-file>index.html</welcome-file>  
    <welcome-file>index.htm</welcome-file>  
    <welcome-file>index.jsp</welcome-file>  
</welcome-file-list>  
<distributable/>  
</web-app>
```

index.jsp

```
=====  
<%@ page contentType="text/html; charset=GBK" %>  
<%@ page import="java.util.*" %>  
<html><head><title>Cluster App Test</title></head>  
<body>  
Server Info:  
<%  
out.println(request.getLocalAddr() + " :" + request.getLocalPort()+"<br>");%>  
<%  
out.println("<br> ID " + session.getId()+"<br>");  
String dataName = request.getParameter("dataName");  
if (dataName != null && dataName.length() > 0) {  
    String dataValue = request.getParameter("dataValue");  
    session.setAttribute(dataName, dataValue);  
}  
  
out.println("<b>Session 列表</b><br>");  
System.out.println("=====");  
Enumeration e = session.getAttributeNames();  
while (e.hasMoreElements()) {  
    String name = (String)e.nextElement();  
    String value = session.getAttribute(name).toString();  
    out.println( name + " = " + value+"<br>");  
    System.out.println( name + " = " + value);  
}  
%>  
<form action="test2.jsp" method="POST">  
    名称:<input type=text size=20 name="dataName">  
    <br>  
    值:<input type=text size=20 name="dataValue">  
    <br>  
    <input type=submit>  
</form>  
</body>  
</html>  
=====
```

PHP 存储Session到Memcached

php memcache扩展

方法一：RPM安装Memcache扩展

```
[root@client ~]# yum -y install gd httpd mysql-server php php-mysql php-gd php-pecl-memcache
[root@client ~]# php -m
[root@client ~]# vim /etc/php.ini          //php配置文件
[root@client ~]# service httpd restart
```

方法二：源码安装Memcache扩展：

例如：php软件已安装在：/usr/local/php

a. Memcache扩展安装：

```
# tar zxvf memcache-2.2.4.tgz
# cd memcache-2.2.4
# /usr/local/php/bin/phpize           //事先安装phpize
# ./configure --with-php-config=/usr/local/php/bin/php-config
# make
# make install
```

b. 配置php

```
# ls -l /usr/local/php/lib/php/extensions/no-debug-non-zts-20060613/memcache.so
# vim /usr/local/php/lib/php.ini
extension_dir = "/usr/local/php/lib/php/extensions/no-debug-non-zts-20060613/"
extension = memcache.so
```

测试：是否支持PHP Memcache扩展

```
[root@web ~]# vim /var/www/html/index.php
<?php
phpinfo();
?>
```

php使用memcache存储session (运维人员)

```
[root@web ~]# vim /etc/php.ini      //php配置文件，源码安装的PHP配置文件/usr/local/php/lib/php.ini
[Session]
session.save_handler = memcache
memcache.hash_strategy = "consistent"
session.cookie_domain = ".tianyun.com"
session.save_path = "tcp://192.168.1.1:11211,tcp://192.168.1.7:11211"
```

Tomcat 存储Session到Memcached

```
[root@tomcat1 ~]# ls /usr/local/tomcat/lib/
memcached-session-manager-1.8.1.jar
memcached-session-manager-tc7-1.8.1.jar
spymemcached-2.10.2.jar
```

```
[root@tomcat1 ~]# vi /usr/local/tomcat/conf/context.xml
<Context>
```

```
<Manager className="de.javakaffee.web.msm.MemcachedBackupSessionManager"
memcachedNodes="h1:192.168.122.50:11211"
failoverNodes="h1"
requestUrlIgnorePattern=".*\.(ico|png|gif|jpg|css|js)$"
transcoderFactoryClass="de.javakaffee.web.msm.serializer.kryo.KryoTranscoderFactory"
/>
</Context>
```

测试页面：

tomcat1: test1.jsp

```
<%@ page language= "java" %>
<html>
  <head><title>TomcatA</title></head>
  <body>
    <h1><font color= "red" >TomcatA </font></h1>
    <table align= "centre" border= "1" >
      <tr>
        <td>Session ID</td>
      <% session.setAttribute( "abc" , "abc" ); %>
        <td><%= session.getId() %></td>
      </tr>
      <tr>
        <td>Created on</td>
        <td><%= session.getCreationTime() %></td>
      </tr>
    </table>
  </body>
</html>
```

tomcat2: test2.jsp

```
<%@ page language= "java" %>
<html>
  <head><title>TomcatB</title></head>
  <body>
    <h1><font color= "blue" >TomcatB </font></h1>
    <table align= "centre" border= "1" >
      <tr>
        <td>Session ID</td>
      <% session.setAttribute( "abc" , "abc" ); %>
        <td><%= session.getId() %></td>
      </tr>
      <tr>
        <td>Created on</td>
        <td><%= session.getCreationTime() %></td>
      </tr>
    </table>
  </body>
</html>
```