

关于本文档

文档名称：《网络文件共享和实时同步》

使用协议：《知识共享公共许可协议(CCPL)》

贡献者

贡献者名称	贡献度	文档变更记录	个人主页
马哥（马永亮）	主编		http://github.com/iKubernetes/
王晓春	作者	26页	http://www.wangxiaochun.com

文档协议

署名要求：使用本系列文档，您必须保留本页中的文档来源信息，具体请参考《知识共享 (Creative Commons) 署名4.0公共许可协议国际版》。

非商业化使用：遵循《知识共享公共许可协议(CCPL)》，并且您不能将本文档用于马哥教育相关业务之外的其他任何商业用途。

您的权利：遵循本协议后，在马哥教育相关业务之外的领域，您将有以下使用权限：

共享 — 允许以非商业性质复制本作品。

改编 — 在原基础上修改、转换或以本作品为基础进行重新编辑并用于个人非商业使用。

致谢

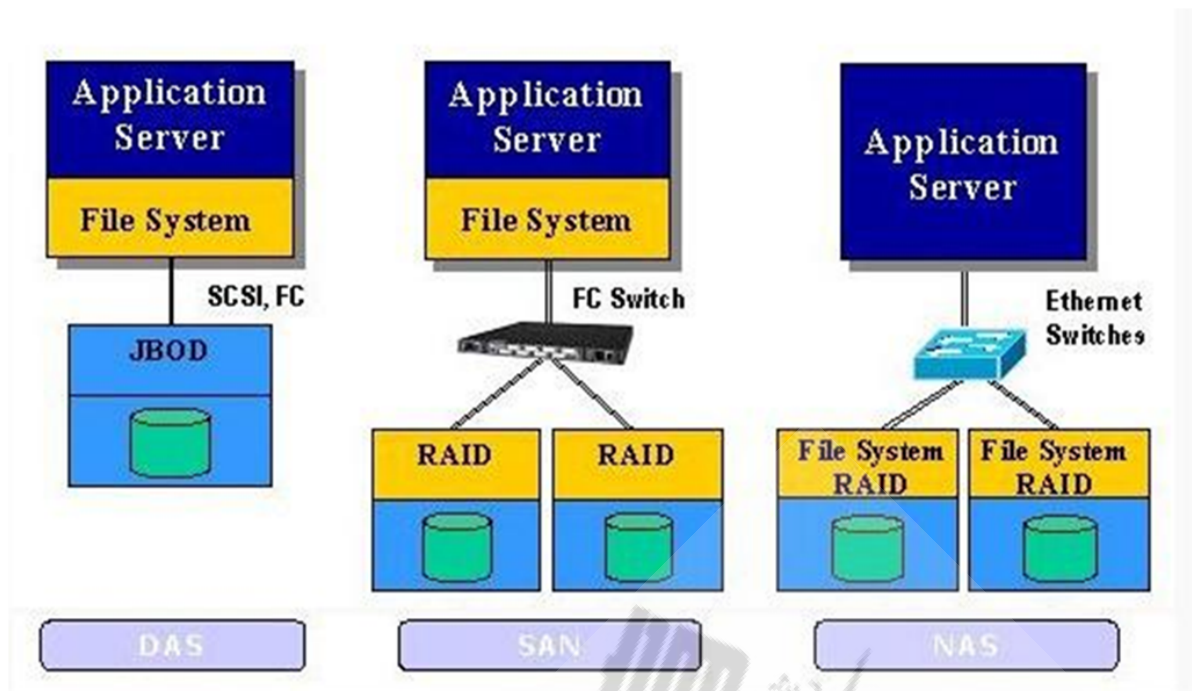
本文档中，部分素材参考了相关项目的文档，以及通过搜索引擎获得的内容，这里先一并向相关的贡献者表示感谢。

网络文件共享服务和数据同步

本章内容

- 存储类型
- NFS服务
- 网络数据同步

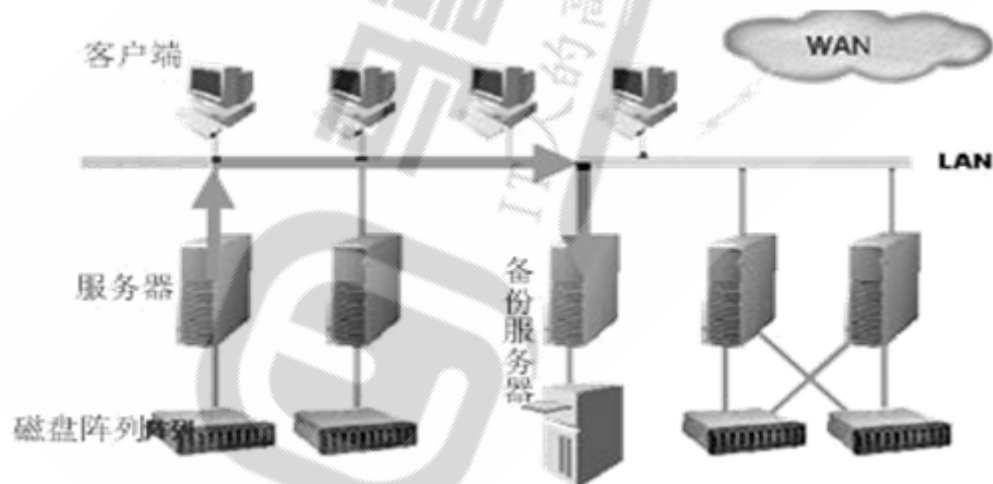
1 存储类型



存储类型分为三种

- 直连式存储：Direct-Attached Storage，简称 DAS
- 存储区域网络：Storage Area Network，简称 SAN
- 网络附加存储：Network-Attached Storage，简称 NAS

1.1 DAS 存储



DAS存储是最常见的一种存储方式，尤其是在中小企业应用中。PC中的硬盘或只有一个外部SCSI接口的JBOD都属于DAS架构。DAS是指存储设备直接连接到服务器总线上，存储设备只与一台独立的主机连接，其他主机不能使用这个存储设备。DAS存储设备与服务器主机之间的连接通道通常采用SCSI连接，DAS存储设备主要是磁盘阵列（RAID: Redundant Arrays of Independent Disks）、磁盘簇（JBOD: Just a Bunch Of Disks）等。

1.2 NAS 存储



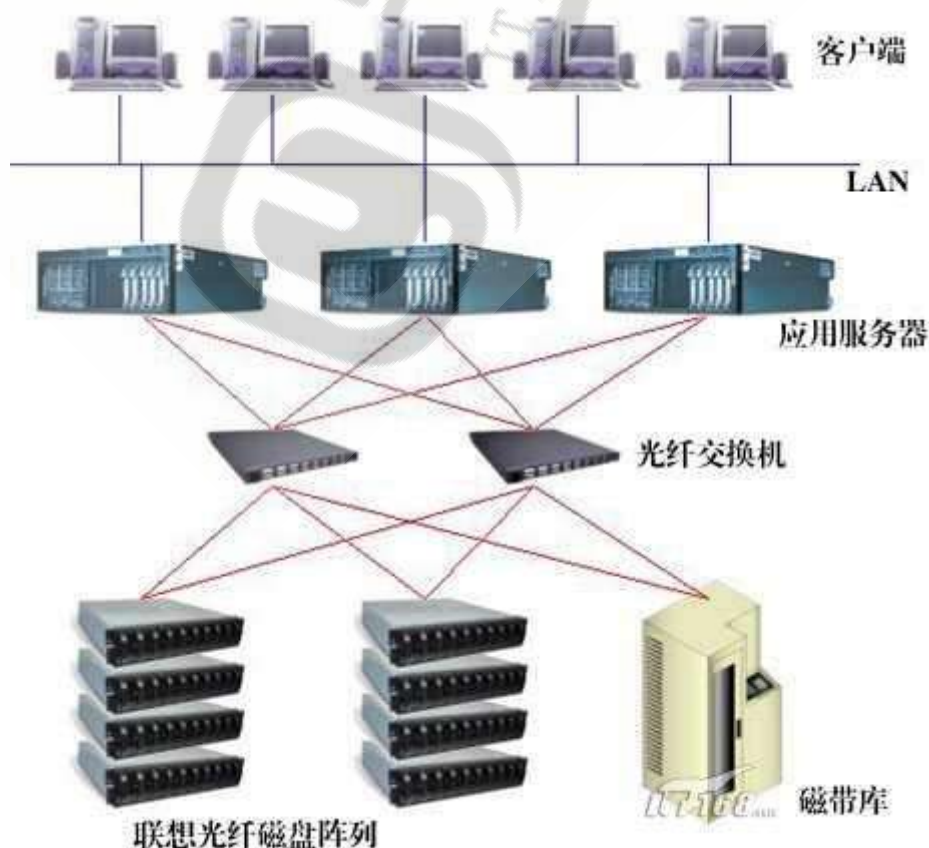
NAS存储就是存储设备通过标准的网络拓扑结构(比如以太网)添加到一群计算机上。与DAS以及SAN不同，NAS是文件级的存储方法。采用NAS较多的功能是用来进行文件共享。

NAS存储也通常被称为附加存储，顾名思义，就是存储设备通过标准的网络拓扑结构(例如以太网)添加到一群计算机上。NAS是文件级的存储方法，它的重点在于帮助工作组和部门级机构解决迅速增加存储容量的需求。如今更多的亲们采用NAS较多的功能是用来文档共享、图片共享、电影共享等等，而且随着云计算的发展，一些NAS厂商也推出了云存储功能，大大方便了企业和亲们等个人用户的使用。

NAS产品是真正即插即用的产品。NAS设备一般支持多计算机平台，用户通过网络支持协议可进入相同的文档，因而NAS设备无需改造即可用于混合Unix/Windows NT局域网内，同时NAS的应用非常灵活。

但NAS有一个关键性问题，即备份过程中的带宽消耗。与将备份数据流从LAN中转移出去的存储区域网(SAN)不同，NAS仍使用网络进行备份和恢复。NAS的一个缺点是它将存储事务由并行SCSI连接转移到了网络上。这就是说LAN除了必须处理正常的最终用户传输流外，还必须处理包括备份操作的存储磁盘请求。

1.3 SAN 存储



存储区域网络，这个是通过光纤通道或以太网交换机连接存储阵列和服务器主机，最后成为一个专用的存储网络。SAN经过十多年历史的发展，已经相当成熟，成为业界的事实标准（但各个厂商的光纤交换技术不完全相同，其服务器和SAN存储有兼容性的要求）。

SAN提供了一种与现有LAN连接的简易方法，并且通过同一物理通道支持广泛使用的SCSI和IP协议。SAN不受现今主流的、基于SCSI存储结构的布局限制。特别重要的是，随着存储容量的爆炸性增长，SAN允许企业独立地增加它们的存储容量。SAN的结构允许任何服务器连接到任何存储阵列，这样不管数据置放在那里，服务器都可直接存取所需的数据。因为采用了光纤接口，SAN还具有更高的带宽。

如今的SAN解决方案通常会采取以下两种形式：光纤信道以及iSCSI或者基于IP的SAN，也就是FC SAN和IP SAN。光纤信道是SAN解决方案中大家最熟悉的类型，但是，最近一段时间以来，基于iSCSI的SAN解决方案开始大量出现在市场上，与光纤通道技术相比较而言，这种技术具有良好的性能，而且价格低廉。

一般通过在主机上安装HBA(Host Bus Adapter 主机总线适配器)卡,再通过光纤连接到光纤交换机,再连接至SAN存储上

SAN的优势：

随着存储容量的增长，SAN允许企业独立地增加他们的存储容量。

SAN允许任何服务器连接到任何存储阵列（好处是：不管数据放在哪里，服务器都可以直接存取所需的数据）

由于使用光纤接口，SAN具有更高的带宽。除了FC连接，SAN连接还有iSCSI（SCSI over IP）以及SAS（Serial Attached SCSI）接口。

光纤接口可以提供10公里那么长那么远的连接长度，非常容易实现物理分离的存储

1.4 三种存储比较

SAN与NAS的主要区别体现在文件系统所在的位置

	DAS	NAS	SAN
传输类型	SCSI、FC	IP	IP、FC、SAS
数据类型	数据块	文件	数据块
典型应用	任何	文件服务器	数据库应用
优点	磁盘与服务器分离， 便于统一管理	不占用应用服务器资源 广泛支持操作系统 扩展较容易 即插即用，安装简单方便	高扩展性 高可用性 数据集中，易管理
缺点	连接距离短 数据分散，共享困难 存储空间利用率不高 扩展性有限	不适合存储量大的块级应用 数据备份及恢复占用网络带宽	相比NAS成本较高 安装和升级比NAS复杂

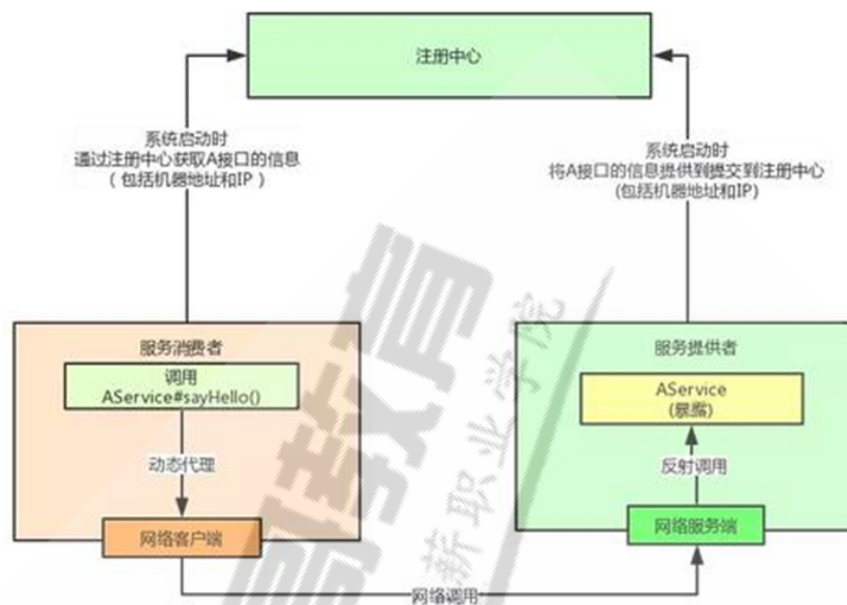
三种存储架构的应用场景

- DAS虽然比较古老了，但是还是很适用于那些数据量不大，对磁盘访问速度要求较高的中小企业

- NAS多适用于文件服务器，用来存储非结构化数据，虽然受限于以太网的速度，但是部署灵活，成本低
- SAN则适用于大型应用或数据库系统，缺点是成本高、较为复杂

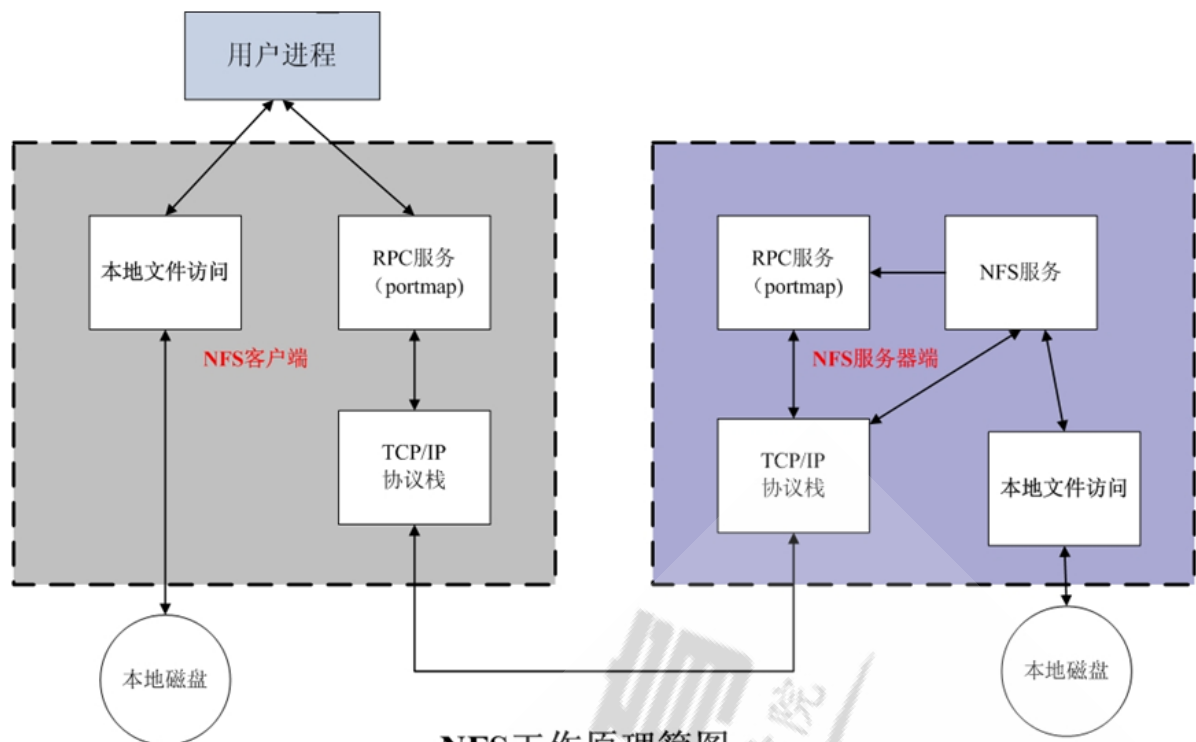
2 NFS 服务

2.1 NFS 工作原理



NFS: Network File System 网络文件系统，基于内核的文件系统。Sun 公司开发，通过使用 NFS，用户和程序可以像访问本地文件一样访问远端系统上的文件，基于 RPC（Remote Procedure Call Protocol 远程过程调用）实现

RPC采用C/S模式，客户机请求程序调用进程发送一个有进程参数的调用信息到服务进程，然后等待应答信息。在服务器端，进程保持睡眠状态直到调用信息到达为止。当一个调用信息到达，服务器获得进程参数，计算结果，发送答复信息，然后等待下一个调用信息，最后，客户端调用进程接收答复信息，获得进程结果，然后调用执行继续进行



NFS工作原理简图

NFS优势：节省本地存储空间，将常用的数据，如：/home目录，存放在NFS服务器上且可以通过网络访问，本地终端将可减少自身存储空间的使用

2.2 NFS软件介绍

软件包：

- 红帽系统: nfs-utils: 包括服务器和客户端相关工具，CentOS8 最小化安装时默认没有安装
- Ubuntu: nfs-server (nfs-kernel-server) 服务器包名,nfs-common 客户端包名

相关软件包: rpcbind (必须) , tcp_wrappers

Kernel支持: nfs.ko

端口: 2049(nfsd), 其它端口由portmap(111)分配

NFS服务主要进程：

- rpc.nfsd 最主要的NFS进程，管理客户端是否可登录
- rpc.mountd 挂载和卸载NFS文件系统，包括权限管理
- rpc.lockd 非必要，管理文件锁，避免同时写出错
- rpc.statd 非必要，检查文件一致性，可修复文件

说明：CentOS 6 开始portmap进程由rpcbind代替

日志：/var/lib/nfs/

NFS配置文件：

```
/etc/exports
/etc/exports.d/*.exports
```

范例:红帽系统安装NFS软件包

```
[root@centos8 ~]#yum -y install nfs-utils
```

#查看支持的NFS版本,注意:只有服务启动才能看此文件

```
[root@centos8 ~]#cat /proc/fs/nfsd/versions
```

```
-2 +3 +4 +4.1 +4.2
```

```
[root@centos8 ~]#systemctl enable --now nfs-server.service
```

范例: Ubuntu 安装NFS软件包

#服务器

```
[root@ubuntu2004 ~]#apt update && apt -y install nfs-kernel-server
```

```
[root@ubuntu2004 ~]#systemctl status nfs-server
```

#查看支持的NFS版本,注意:只有服务启动才能看此文件

```
[root@ubuntu2004 ~]#cat /proc/fs/nfsd/versions
```

```
-2 +3 +4 +4.1 +4.2
```

#客户端

```
[root@ubuntu2004 ~]#apt -y install nfs-common
```

2.3 NFS共享配置文件格式

```
/dir      主机1(opt1,opt2)    主机2(opt1,opt2)...
```

格式说明:

- 以#开始的行为注释
- 主机格式:

anonymous: 表示使用*通配所有客户端

单个主机: **ipv4, ipv6, FQDN**

IP networks: 两种掩码格式均支持

172.18.0.0/255.255.0.0

172.18.0.0/16

wildcards: 主机名通配,例如:*.wang.org, IP不可以

netgroups: NIS域的主机组, @group_name

- 每个条目指定目录导出的哪些主机, 及相关的权限和选项

默认选项: (ro, sync, root_squash, no_all_squash)

ro, rw 只读和读写

async 异步, 数据变化后不立即写磁盘, 先写入到缓冲区中, 过一段时间再写入磁盘, 性能高, 安全性低

sync (1.0.0后为默认) 同步, 数据在请求时立即写入共享存储磁盘, 性能低, 安全性高

root_squash (默认) 远程root映射为nfsnobody, UID为65534, CentOS8 为nobody, CentOS 7以前的版本为nfsnobody

no_root_squash 远程root映射成NFS服务器的root用户

all_squash 所有远程用户(包括root)都变成nfsnobody, CentOS8 为nobody

no_all_squash (默认) 保留共享文件的UID和GID

anonuid和**anongid** 指明匿名用户映射为特定用户UID和组GID, 而非nobody, 可配合all_squash使用, 注意: 目录需要给此用户权限, 否则无法访问

范例: NFS配置示例

```

vim /etc/exports
/myshare server.example.com
/myshare *.example.com
/myshare server?.example.com
/myshare server[0-20].example.com
/myshare 172.25.11.10
/myshare 172.25.0.0/16
/myshare 2000:472:18:b51:c32:a21
/myshare 2000:472:18:b51::/64
/myshare *.example.com 172.25.0.0/16
/myshare desktop.example.com(ro)
/myshare desktop.example.com(ro) server[0-20].example.com(rw)
/myshare diskless.example.com(rw,no_root_squash)

```

2.4 NFS工具

2.4.1 rpcinfo

rpcinfo 工具可以查看RPC相关信息

查看注册在指定主机的RPC程序

```
rpcinfo -p hostname
```

查看RPC注册程序

```
rpcinfo -s hostname
```

范例：rpcinfo

```

[root@centos8 ~]#rpcinfo -p
program vers proto  port  service
100000    4      tcp    111   portmapper
100000    3      tcp    111   portmapper
100000    2      tcp    111   portmapper
100000    4      udp    111   portmapper
100000    3      udp    111   portmapper
100000    2      udp    111   portmapper
100024    1      udp    58875 status
100024    1      tcp    49005 status
100005    1      udp    20048 mountd
100005    1      tcp    20048 mountd
100005    2      udp    20048 mountd
100005    2      tcp    20048 mountd
100005    3      udp    20048 mountd
100005    3      tcp    20048 mountd
100003    3      tcp    2049  nfs
100003    4      tcp    2049  nfs
100227    3      tcp    2049  nfs_acl
100021    1      udp    38963 nlockmgr
100021    3      udp    38963 nlockmgr
100021    4      udp    38963 nlockmgr
100021    1      tcp    35503 nlockmgr
100021    3      tcp    35503 nlockmgr
100021    4      tcp    35503 nlockmgr

```



```
[root@centos8 ~]#rpcinfo -s
```

program	version(s)	netid(s)	service	owner
100000	2,3,4	local,udp,tcp,udp6,tcp6	portmapper	superuser
100024	1	tcp6,udp6,tcp,udp	status	29

#查看远程主机

```
[root@centos7 ~]#rpcinfo -p 10.0.0.8
```

program	vers	proto	port	service
100000	4	tcp	111	portmapper
100000	3	tcp	111	portmapper
100000	2	tcp	111	portmapper
100000	4	udp	111	portmapper
100000	3	udp	111	portmapper
100000	2	udp	111	portmapper
100024	1	udp	38427	status
100024	1	tcp	34335	status
100005	1	udp	20048	mountd
100005	1	tcp	20048	mountd
100005	2	udp	20048	mountd
100005	2	tcp	20048	mountd
100005	3	udp	20048	mountd
100005	3	tcp	20048	mountd
100003	3	tcp	2049	nfs
100003	4	tcp	2049	nfs
100227	3	tcp	2049	nfs_acl
100021	1	udp	44185	nlockmgr
100021	3	udp	44185	nlockmgr
100021	4	udp	44185	nlockmgr
100021	1	tcp	46603	nlockmgr
100021	3	tcp	46603	nlockmgr
100021	4	tcp	46603	nlockmgr

```
[root@centos7 ~]#rpcinfo -s 10.0.0.8
```

program	version(s)	netid(s)	service	owner
100000	2,3,4	local,udp,tcp,udp6,tcp6	portmapper	superuser
100024	1	tcp6,udp6,tcp,udp	status	29
100005	3,2,1	tcp6,udp6,tcp,udp	mountd	superuser
100003	4,3	tcp6,tcp	nfs	superuser
100227	3	tcp6,tcp	nfs_acl	superuser
100021	4,3,1	tcp6,udp6,tcp,udp	nlockmgr	superuser

2.4.2 exportfs

exportfs:可用于管理NFS导出的文件系统

常见选项:

- v #查看本机所有NFS共享
- r #重读配置文件, 并共享目录
- a #输出本机所有共享
- au #停止本机所有共享

2.4.3 showmount

常见用法:

```
#查看远程主机的NFS共享
showmount -e hostname
```

范例:

```
[root@centos7 ~]#showmount -e 10.0.0.8
Export list for 10.0.0.8:
/data/wordpress *
```

2.4.4 mount.nfs

客户端NFS挂载

NFS相关的挂载选项: man 5 nfs

```
fg      #（默认）前台挂载
bg      #后台挂载
hard    #（默认）持续请求
soft    #非持续请求
intr    #和hard配合，请求可中断
rsize   #和wsize 一次读和写数据最大字节数，rsize=32768
_netdev #无网络服务时不挂载NFS资源
vers    #指定版本，客户端centos8默认4.2，centos7默认4.1 centos6默认4.0
```

提示: 基于安全考虑, 建议使用 nosuid,_netdev,noexec 挂载选项

范例: 临时挂载NFS共享

```
mount -o rw,nosuid,fg,hard,intr 172.16.0.1:/testdir /mnt/nfs/
```

范例:

```
[root@centos7 ~]#mkdir /mnt/nfs
[root@centos7 ~]#mount 10.0.0.8:/data/wordpress /mnt/nfs
[root@centos7 ~]#ls /mnt/nfs
index.html
[root@centos7 ~]#df -T /mnt/nfs
Filesystem                Type 1k-blocks    Used Available Use% Mounted on
10.0.0.8:/data/wordpress nfs4   52403200 398336   52004864    1% /mnt/nfs
```

范例: 开机挂载

```
#vim /etc/fstab
172.16.0.1:/public /mnt/nfs nfs defaults,_netdev 0 0
```

范例: 远程的 root 映射为NFS服务器的nobody用户

```
[root@centos6 ~]#grep nobody /etc/passwd
nobody:x:99:99:Nobody:/:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin

[root@centos7 ~]#grep nobody /etc/passwd
nobody:x:99:99:Nobody:/:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin

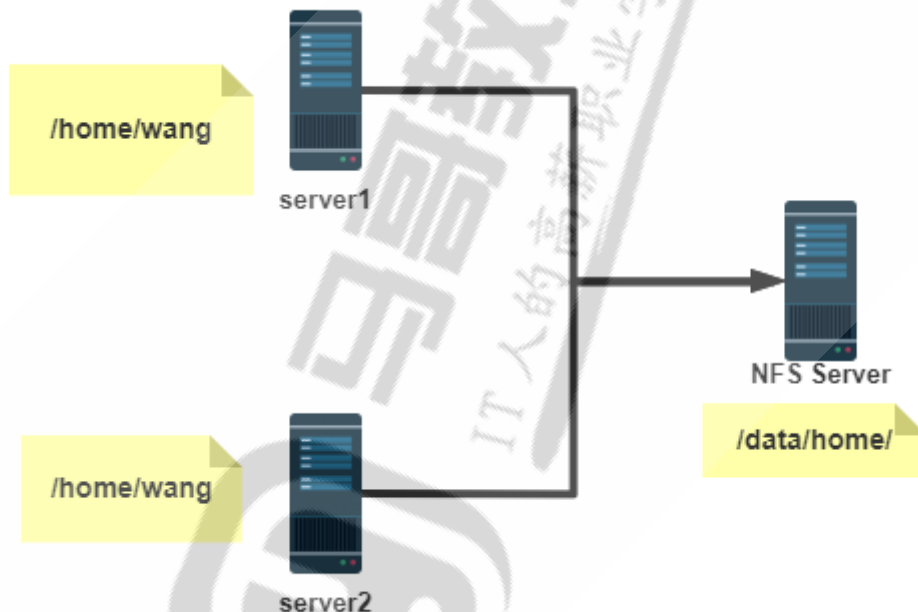
[root@centos8 ~]#grep nobody /etc/passwd
nobody:x:65534:65534:kernel overflow User:/:/sbin/nologin
```

2.5 实战案例

2.5.1 目标

将NFS的共享目录，做为远程主机用户的家目录

2.5.2 环境准备



共三台主机
一台主机 **nfs server**
IP:10.0.0.8
另两台当 **nfs client**
IP:10.0.0.7
IP:10.0.0.6

2.5.3 步骤

```
#NFS服务器创建用户和相应的家目录，将用户wang的家目录共享
[root@centos8 ~]#yum -y install nfs-utils
[root@centos8 ~]#systemctl enable --now nfs-server
[root@centos8 ~]#mkdir -pv /data/home
[root@centos8 ~]#useradd -d /data/home/wang -u 2000 wang
[root@centos8 ~]#vim /etc/exports.d/test.exports
/data/home *(rw)
```

```
[root@centos8 ~]#exportfs -r
```

#在第一台NFS客户端主机10.0.0.7上实现

```
[root@centos7 ~]#yum -y install nfs-utils
```

```
[root@centos7 ~]#useradd -u 2000 wang
```

```
[root@centos7 ~]#vim /etc/fstab
```

```
10.0.0.8:/data/home/wang /home/wang nfs _netdev 0 0
```

```
[root@centos7 ~]#mount -a
```

```
[root@centos7 ~]#su - wang
```

Last login: Fri Jul 3 16:33:34 CST 2020 on pts/0

```
[wang@centos7 ~]$pwd
```

/home/wang

```
[wang@centos7 ~]$df /home/wang -T
```

Filesystem	Type	1k-blocks	Used	Available	Use%	Mounted on
10.0.0.8:/data/home/wang	nfs4	52403200	398464	52004736	1%	/home/wang

#在第二台NFS客户端主机10.0.0.6上实现

```
[root@centos6 ~]#yum -y install nfs-utils
```

```
[root@centos6 ~]#useradd -u 2000 wang
```

```
[root@centos6 ~]#vim /etc/fstab
```

```
10.0.0.8:/data/home/wang /home/wang nfs _netdev 0 0
```

```
[root@centos6 ~]#su - wang
```

```
[wang@centos6 ~]$pwd
```

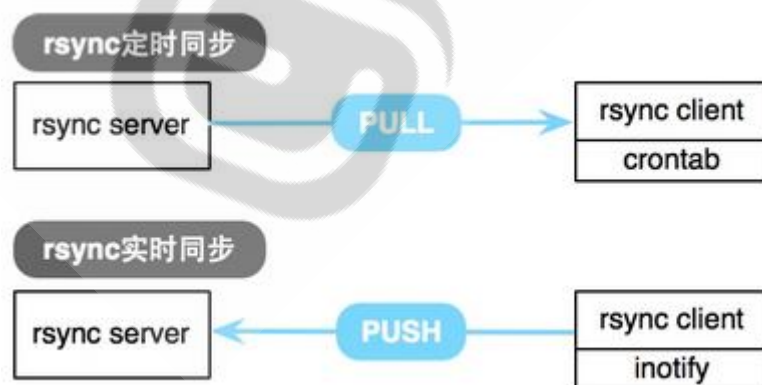
/home/wang

```
[wang@centos6 ~]$df -T /home/wang
```

Filesystem	Type	1k-blocks	Used	Available	Use%	Mounted on
10.0.0.8:/data/home/wang	nfs	52403200	398464	52004736	1%	/home/wang

3 数据的实时同步

在生产环境，有时需要两台主机的特定目录实现实时同步。比如，将NFS共享目录的数据文件，自动实时同步到备份服务器特定目录中



3.1 实时同步技术介绍

实现实时同步的方法

- inotify + rsync 方式实现数据同步,需自行编写脚本组合inotify和rsync实现
- sersync: 前金山公司周洋 (花椒直播) 在 inotify+rsync 软件基础上进行开发的, 功能更加强大

工作原理:

- 要利用监控服务 (inotify) , 监控同步数据服务器目录中信息的变化
- 发现目录中数据产生变化, 就利用rsync服务推送到备份服务器上

inotify:

异步的文件系统事件监控机制, 利用事件驱动机制, 而无须通过诸如cron等的轮询机制来获取事件, linux内核从2.6.13起支持 inotify, 通过inotify可以监控文件系统中添加、删除, 修改、移动等各种事件

```
[root@ubuntu2204 ~]#grep -i inotify /boot/config-5.15.0-52-generic
CONFIG_INOTIFY_USER=y
[root@centos8 ~]#grep -i inotify /boot/config-4.18.0-80.el8.x86_64
CONFIG_INOTIFY_USER=y
```

实现inotify软件:

- inotify-tools
- sersync
- lrsyncd

inotify+rsync使用方式

- inotify 对同步数据目录信息的监控
- rsync 完成对数据的同步
- 利用脚本进行结合

3.2 实现 inotify

3.2.1 内核支持

内核是否支持inotify

Linux支持inotify的内核最小版本为 2.6.13, 参看man 7 inotify

```
#列出下面的文件, 说明服务器内核支持inotify
[root@centos8 ~]#ls -l /proc/sys/fs/inotify
-rw-r--r-- 1 root root 0 Dec  7 10:10 max_queued_events
-rw-r--r-- 1 root root 0 Dec  7 10:10 max_user_instances
-rw-r--r-- 1 root root 0 Dec  6 05:54 max_user_watches

[root@centos8 ~]#cat /proc/sys/fs/inotify/max_queued_events
16384
[root@centos8 ~]#cat /proc/sys/fs/inotify/max_user_instances
128
[root@centos8 ~]#cat /proc/sys/fs/inotify/max_user_watches
8192

[root@ubuntu2204 ~]#cat /proc/sys/fs/inotify/max_queued_events
16384
[root@ubuntu2204 ~]#cat /proc/sys/fs/inotify/max_user_instances
128
[root@ubuntu2204 ~]#cat /proc/sys/fs/inotify/max_user_watches
14281
```

inotify 内核参数说明:

- max_queued_events: inotify 事件队列最大长度, 如值太小会出现 Event Queue Overflow 错误

默认值: 16384, 生产环境建议调大, 比如: 327679

- max_user_instances: 每个用户创建 inotify实例最大值, 默认值: 128
- max_user_watches: 可以监视的文件的总数量 (inotifywait 单进程), 默认值: 8192, 建议调大

范例:

```
[root@data-centos8 ~]# vim /etc/sysctl.conf
fs.inotify.max_queued_events=66666
fs.inotify.max_user_watches=100000
[root@centos8 ~]# sysctl -p
fs.inotify.max_queued_events = 66666
fs.inotify.max_user_watches = 100000
[root@centos8 ~]# cat /proc/sys/fs/inotify/*
66666
128
100000
```

3.2.2 inotify-tools工具

inotify-tools参考文档: <https://github.com/rvoicilas/inotify-tools/wiki>

安装inotify-tools: 基于epel源

```
[root@data-centos8 ~]# yum -y install inotify-tools
[root@data-ubuntu2004]# apt -y install inotify-tools
```

inotify-tools包主要工具:

- inotifywait: 在被监控的文件或目录上等待特定文件系统事件 (open, close, delete等) 发生, 常用于实时同步的目录监控
- inotifywatch: 收集被监控的文件系统使用的统计数据, 指文件系统事件发生的次数统计

inotifywait 命令

格式:

```
inotifywait [ options ] file1 [ file2 ] [ file3 ] [ ... ]
```

常用选项:

-m, --monitor	#始终保持事件监听
-d, --daemon	#以守护进程方式执行, 和-m相似, 配合-o使用
-r, --recursive	#递归监控目录数据信息变化
-q, --quiet	#输出少量事件信息
--exclude <pattern>	#指定排除文件或目录, 使用扩展的正则表达式匹配的模式实现
--excludei <pattern>	#和exclude相似, 不区分大小写
-o, --outfile <file>	#打印事件到文件中, 相当于标准正确输出, 注意: 使用绝对路径
-s, --syslogOutput	#发送错误到syslog相当于标准错误输出
--timefmt <fmt>	#指定时间输出格式
--format <fmt>	#指定的输出格式; 即实际监控输出内容
-e	#指定监听指定的事件, 如果省略, 表示所有事件都进行监听

inotifywait 的--timefmt 时间格式

参考 man 3 strftime

%Y	#年份信息, 包含世纪信息
%y	#年份信息, 不包括世纪信息
%m	#显示月份, 范围 01-12
%d	#每月的第几天, 范围是 01-31
%H	#小时信息, 使用 24小时制, 范围 00-23
%M	#分钟, 范围 00-59
%S	#秒, 范围 0-60

范例:

```
--timefmt "%Y-%m-%d %H:%M:%S"
```

inotifywait 的 --format 格式定义

%T	#输出时间格式中定义的时间格式信息, 通过 --timefmt option 语法格式指定时间信息
%w	#事件出现时, 监控文件或目录的名称信息, 相当于dirname
%f	#事件出现时, 将显示监控目录下触发事件的文件或目录信息, 否则为空, 相当于basename
%e	#显示发生的事件信息, 不同的事件默认用逗号分隔
%xe	#显示发生的事件信息, 不同的事件指定用X进行分隔

范例:

```
--format "%T %w%f event: %;e"  
--format '%T %w %f'
```

inotifywait -e 选项指定的事件类型

create	#文件或目录创建
delete	#文件或目录被删除
modify	#文件或目录内容被写入
attrib	#文件或目录属性改变
close_write	#文件或目录关闭, 在写入模式打开之后关闭的
close_nowrite	#文件或目录关闭, 在只读模式打开之后关闭的
close	#文件或目录关闭, 不管读或是写模式
open	#文件或目录被打开
lsdir	#浏览目录内容
moved_to	#文件或目录被移动到监控的目录中
moved_from	#文件或目录从监控的目录中被移动
move	#文件或目录不管移动到或是移出监控目录都触发事件
access	#文件或目录内容被读取
delete_self	#文件或目录被删除, 目录本身被删除
unmount	#取消挂载

范例:

```
-e create,delete,moved_to,close_write,attrib, moved_from
```

范例: 使用inotifywait

```
#监控一次性事件  
inotifywait /data/www  
Setting up watches.  
watches established.
```

```
/data/www/ CREATE f1.txt
```

#持续前台监控

```
inotifywait -mrq /data/www --exclude=".*\|\.swx\|\.swp"  
/data/www/ OPEN f1.txt  
/data/www/ ACCESS f1.txt  
/data/www/ CLOSE_NOWRITE,CLOSE f1.txt
```

#持续后台监控，并记录日志

```
inotifywait -o /root/inotify.log -drq /data/www --timefmt "%Y-%m-%d %H:%M:%S" --  
format "%T %w%f event: %e"
```

#持续前台监控特定事件

```
inotifywait -mrq /data/www --timefmt "%F %H:%M:%S" --format "%T %w%f event:  
%;e" -e create,delete,moved_to,moved_from,close_write,attrib
```

3.3 rsync 服务

rsync 常用于做为 linux系统下的数据镜像备份工具，实现远程同步，支持本地复制，或者与其他SSH、rsync主机同步数据，支持增量备份，配合任务计划，rsync能实现定时或间隔同步，配合inotify或sersync，可以实现触发式的实时数据同步

官方网站: <http://rsync.samba.org/>

服务器软件包: rsync(Ubuntu20.04), rsync-daemon (CentOS 8)

服务文件: /usr/lib/systemd/system/rsyncd.service

配置文件: /etc/rsyncd.conf

端口: 873/tcp

3.3.1 rsync命令

```
Usage: rsync [OPTION]... SRC [SRC]... DEST  
or rsync [OPTION]... SRC [SRC]... [USER@]HOST:DEST  
or rsync [OPTION]... SRC [SRC]... [USER@]HOST::DEST  
or rsync [OPTION]... SRC [SRC]... rsync://[USER@]HOST[:PORT]/DEST  
or rsync [OPTION]... [USER@]HOST:SRC [DEST]  
or rsync [OPTION]... [USER@]HOST::SRC [DEST]  
or rsync [OPTION]... rsync://[USER@]HOST[:PORT]/SRC [DEST]
```

rsync 格式

#Local: 代替 cp

```
rsync [OPTION...] SRC... [DEST]
```

#Access via remote shell:

Pull:

```
rsync [OPTION...] [USER@]HOST:SRC... [DEST]
```

Push:

```
rsync [OPTION...] SRC... [USER@]HOST:DEST
```

#Access via rsync daemon:

Pull:

```
rsync [OPTION...] [USER@]HOST::SRC... [DEST]
```

```
rsync [OPTION...] rsync://[USER@]HOST[:PORT]/SRC... [DEST]
```

Push:

```
rsync [OPTION...] SRC... [USER@]HOST::DEST
```

```
rsync [OPTION...] SRC... rsync://[USER@]HOST[:PORT]/DEST
```

The ':' usages connect via remote shell, while '::' & 'rsync://' usages connect to an rsync daemon, and require SRC or DEST to start with a module name.

rsync 有三种工作方式:

1. 本地文件系统上实现同步。命令行语法格式为上述"Local"段的格式。
2. 本地主机使用远程shell和远程主机通信。命令行语法格式为上述"Access via remote shell"段的格式。
3. 本地主机通过网络套接字连接远程主机上的 rsync daemon。命令行语法格式为上述"Access via rsync daemon"段的格式。

前两者的本质是通过本地或远程shell，而第3种方式则是让远程主机上运行rsyncd服务，使其监听在一个端口上，等待客户端的连接。

常见选项:

-v: 显示rsync过程中详细信息。可以使用"**-vvvv**"获取更详细信息。

-P: 显示文件传输的进度信息。(实际上"**-P**"="**--partial --progress**", 其中的"**--progress**"才是显示进度信息的)。

-n --dry-run : 仅测试传输，而不实际传输。常和"**-vvvv**"配合使用来查看rsync是如何工作的。

-a --archive : 归档模式，表示递归传输并保持文件属性。等同于"**-rtopgd1**"。

-r --recursive: 递归到目录中去。

-t --times: 保持mtime属性。强烈建议任何时候都加上"**-t**", 否则目标文件mtime会设置为系统时间，导致下次更新

 : 检查出mtime不同从而导致增量传输无效。

-o --owner: 保持owner属性(属主)。

-g --group: 保持group属性(属组)。

-p --perms: 保持perms属性(权限，不包括特殊权限)。

-D : 是"**--device --specials**"选项的组合，即也拷贝设备文件和特殊文件。

-l --links: 如果文件是软链接文件，则拷贝软链接本身而非软链接所指向的对象

-z : 传输时进行压缩提高效率

-R --relative: 使用相对路径。意味着将命令行中指定的全路径而非路径最尾部的文件名发送给服务端，包括它们的属性。用法见下文示例。

--size-only : 默认算法是检查文件大小和mtime不同的文件，使用此选项将只检查文件大小。

-u --update : 仅在源mtime比目标已存在文件的mtime新时才拷贝。注意，该选项是接收端判断的，不会影响删除行为。

-d --dirs : 以不递归的方式拷贝目录本身。默认递归时，如果源为"**dir1/file1**"，则不会拷贝dir1目录，使用该选项将拷贝dir1但不拷贝file1。

--max-size : 限制rsync传输的最大文件大小。可以使用单位后缀，还可以是一个小数值(例如: "**--max-size=1.5m**")

--min-size : 限制rsync传输的最小文件大小。这可以用于禁止传输小文件或那些垃圾文件。

--exclude : 指定排除规则来排除不需要传输的文件。

--delete : 以SRC为主，对DEST进行同步。多则删之，少则补之。注意"**--delete**"是在接收端执行的，所以它是在

 : **exclude/include**规则生效之后才执行的。

-b --backup : 对目标上已存在的文件做一个备份，备份的文件名后默认使用"**~**"做后缀。

--backup-dir: 指定备份文件的保存路径。不指定时默认和待备份文件保存在同一目录下。

-e : 指定所要使用的远程shell程序，默认为ssh。

--port : 连接daemon时使用的端口号，默认为873端口。

--password-file: daemon模式时的密码文件，可以从中读取密码实现非交互式。注意，这不是远程shell认证的密码，而是rsync模块认证的密码。

-w --whole-file: rsync将不再使用增量传输，而是全量传输。在网络带宽高于磁盘带宽时，该选项比增量传输更高效。

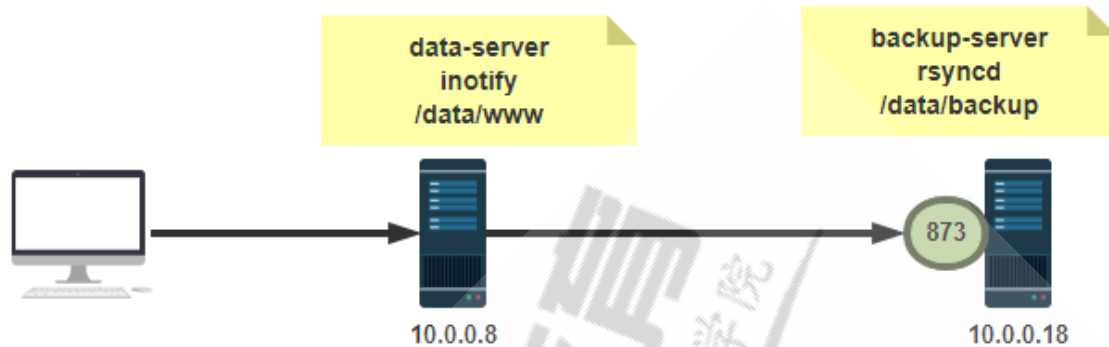
--existing : 要求只更新目标端已存在的文件，目标端还不存在的文件不传输。注意，使用相对路径时如果上层目录不存在也不会传输。

--ignore-existing: 要求只更新目标端不存在的文件。和"**--existing**"结合使用有特殊功能，见下文示例。

--remove-source-files: 要求删除源端已经成功传输的文件

--bwlimit=RATE 指定限速,单位默为MB/s

范例：两种格式访问 rsync daemon 服务



#在备份服务器启动 rsync 进程

```
[root@backup-centos8 ~]#rsync --daemon
Failed to parse config file: /etc/rsyncd.conf
[root@backup-centos8 ~]#touch /etc/rsyncd.conf
[root@backup-centos8 ~]#rsync --daemon
[root@backup-centos8 ~]#ss -ntlp|grep rsync
LISTEN 0      5            0.0.0.0:873      0.0.0.0:*
users:((("rsync",pid=2921,fd=4))
LISTEN 0      5            [::]:873        [::]:*
users:((("rsync",pid=2921,fd=5))
[root@backup-centos8 ~]#
```

```
[root@backup-centos8 ~]#cat /etc/rsyncd.conf
[backup]
path = /data/backup/
read only = no #指定可读写,默认只读
```

#指定目录给nobody权限，默认用户以nobody访问此目录

```
[root@backup-centos8 ~]#setfacl -m u:nobody:rwX /data/backup/
```

#查看rsync服务器的模块名称

```
[root@data-centos8 ~]#rsync rsync://backup-server
backup
[root@data-centos8 ~]#rsync backup-server::
backup
```

#访问rsync服务器的共享目录

#推

```
[root@data-centos8 ~]#rsync /etc/networks root@backup-server::backup #默认所有
用户都映射为nobody用户
[root@data-centos8 ~]#rsync /etc/issue wang@backup-server::backup #默认所有
用户都映射为nobody用户
[root@data-centos8 ~]#rsync /etc/passwd backup-server::backup
```



```
[root@data-centos8 ~]#rsync /etc/shells rsync://root@backup-server/backup

#拉
[root@data-server ~]#rsync backup-server::backup/* /opt
[root@data-server ~]#rsync rsync://backup-server/backup/* /mnt
```

3.3.2 以独立服务方式运行rsync并实现验证功能

范例：以独立服务方式运行 rsync

```
#Ubuntu默认提供了service文件
[root@ubuntu2204 ~]#systemctl cat rsync.service
# /lib/systemd/system/rsync.service
[Unit]
Description=fast remote file copy program daemon
ConditionPathExists=/etc/rsyncd.conf
After=network.target
Documentation=man:rsync(1) man:rsyncd.conf(5)

[Service]
ExecStart=/usr/bin/rsync --daemon --no-detach
RestartSec=1

# Citing README.md:
#
# [...] Using ssh is recommended for its security features.
#
# Alternatively, rsync can run in `daemon' mode, listening on a socket.
# This is generally used for public file distribution, [...]
#
# So let's assume some extra security is more than welcome here. We do full
# system protection (which makes /usr, /boot, & /etc read-only) and hide
# devices. To override these defaults, it's best to do so in the drop-in
# directory, often done via `systemctl edit rsync.service'. The file needs
# just the bare minimum of the right [heading] and override values.
# See systemd.unit(5) and search for "drop-in" for full details.

ProtectSystem=full
#ProtectHome=on|off|read-only
PrivateDevices=on
NoNewPrivileges=on

[Install]
WantedBy=multi-user.target

#红帽系统需要安装rsync-daemon包提供service文件
[root@backup-centos8 ~]#dnf -y install rsync-daemon
[root@backup-centos8 ~]#rpm -ql rsync-daemon
/etc/rsyncd.conf
/etc/sysconfig/rsyncd
/usr/lib/systemd/system/rsyncd.service
/usr/lib/systemd/system/rsyncd.socket
/usr/lib/systemd/system/rsyncd@.service
/usr/share/man/man5/rsyncd.conf.5.gz

#创建rsync服务器的配置文件
```

```
[root@centos8 ~]#vi /etc/rsyncd.conf
uid = root    #指定以哪个用户身份访问共享目录，默认为nobody，注意：共享目录需要给此用户权限，否则无法访问
gid = root    #指定以哪个组身份访问共享目录，默认为nobody，Ubuntu中为nogroup
#port = 874   可指定非标准端口，默认873/tcp
#use chroot = no
max connections = 0
ignore errors
exclude = lost+found/
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid
lock file = /var/run/rsyncd.lock
reverse lookup = no
#hosts allow = 10.0.0.0/24
[backup]    #每个模块名对应一个不同的path目录，如果同名后面模块生效
path = /data/backup/
comment = backup dir
read only = no    #默认是yes，即只读
auth users = rsyncuser    #默认anonymous可以访问rsync服务器
secrets file = /etc/rsync.pas
```

#配置文件内容

```
[root@ubuntu2204 ~]#cat /etc/rsyncd.conf
uid = root
gid = root
max connections = 0
ignore errors
exclude = lost+found/
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid
lock file = /var/run/rsyncd.lock
reverse lookup = no
```

```
[backup]
path = /data/backup/
comment = backup dir
read only = no
auth users = rsyncuser
secrets file = /etc/rsync.pas
```

#服务器端准备目录

```
[root@backup-centos8 ~]#mkdir -pv /data/backup
```

#服务器端生成验证文件

```
[root@backup-centos8 ~]#echo "rsyncuser:123456" > /etc/rsync.pas
[root@backup-centos8 ~]#chmod 600 /etc/rsync.pas
```

#服务器端启动rsync服务

```
[root@backup-centos8 ~]#rsync --daemon    #可加入/etc/rc.d/rc.local实现开机启动
[root@backup-centos8 ~]#systemctl start rsyncd    #CentOS 7 以上版本
```

#客户端配置密码文件

#也可将密码赋值给环境变量RSYNC_PASSWORD变量，但不安全

```
#export RSYNC_PASSWORD=123456
```

```
[root@data-centos8 ~]#echo "123456" > /etc/rsync.pas
```

```
[root@data-centos8 ~]#chmod 600 /etc/rsync.pas    #此为必要项，权限必须修改
```

```

#查看远程rsync服务器的模块信息
[root@data-server ~]#rsync rsync://rsync服务器IP
backup backup dir

#交互式验证查看具体模块内的文件
[root@data-server ~]#rsync rsync://rsyncuser@rsync服务器IP/backup
Password:

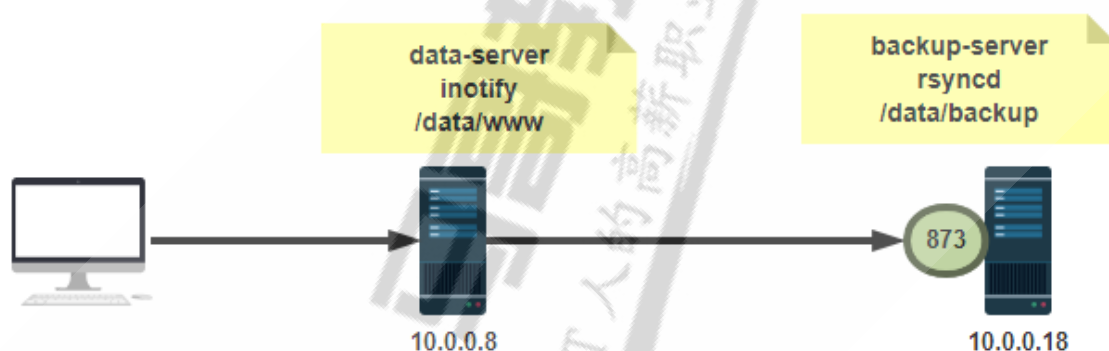
#非交互式查看共享目录
[root@data-server ~]#rsync --password-file=/etc/rsync.pas
rsync://rsyncuser@rsync服务器IP/backup

#客户端测试同步数据
[root@data-centos8 ~]#rsync -avz --delete --password-file=/etc/rsync.pas
/data/www/ rsyncuser@rsync服务器IP::backup

[root@data-centos8 ~]#rsync -avz --delete --password-file=/etc/rsync.pas
rsyncuser@rsync服务器IP::backup /data/www/

```

3.4 inotify+rsync+shell 脚本实现实时数据同步



按 5.3 搭建好 rsyncd 的备份服务器，在数据服务器上创建 inotify_rsync.sh 脚本

注意: 此脚本执行前先确保两主机初始数据处于同步状态,此脚本实现后续的数据同步

```

[root@data-centos8 ~]#vim inotify_rsync.sh
#!/bin/bash
SRC='/data/www/'          #注意最后的/
DEST='rsyncuser@rsync服务器IP::backup'
#Ubuntu20.04不支持 --password-file=/etc/rsync.pas，可以使用下面的变量实现
export RSYNC_PASSWORD=123456
#rpm -q inotify-tools &> /dev/null ||yum -y install inotify-tools
#rpm -q rsync &> /dev/null || yum -y install rsync
inotifywait -mrq --exclude=".*/.swp" --timefmt '%Y-%m-%d %H:%M:%S' --format
'%T %w %f' -e create,delete,moved_from,moved_to,close_write,attrib ${SRC} |while
read DATE TIME DIR FILE;do
    FILEPATH=${DIR}${FILE}
    rsync -az --delete $SRC $DEST && echo "At ${TIME} on ${DATE}, file
$FILEPATH was backed up via rsync" >> /var/log/changelist.log
    #rsync -az --delete --password-file=/etc/rsync.pas $SRC $DEST && echo
"At ${TIME} on ${DATE}, file $FILEPATH was backed up via rsync" >>
/var/log/changelist.log
done

```

#查看文件传输日志

```
[root@data-centos8 ~]#tail -f /var/log/changelist.log
```

3.5 sersync 实现实时数据同步

3.5.1 sersync 介绍

sersync 是前金山公司周洋在 inotify和rsync 软件基础上进行开发的，功能更加强大

sersync类似于inotify，同样用于监控，但它克服了inotify的缺点。

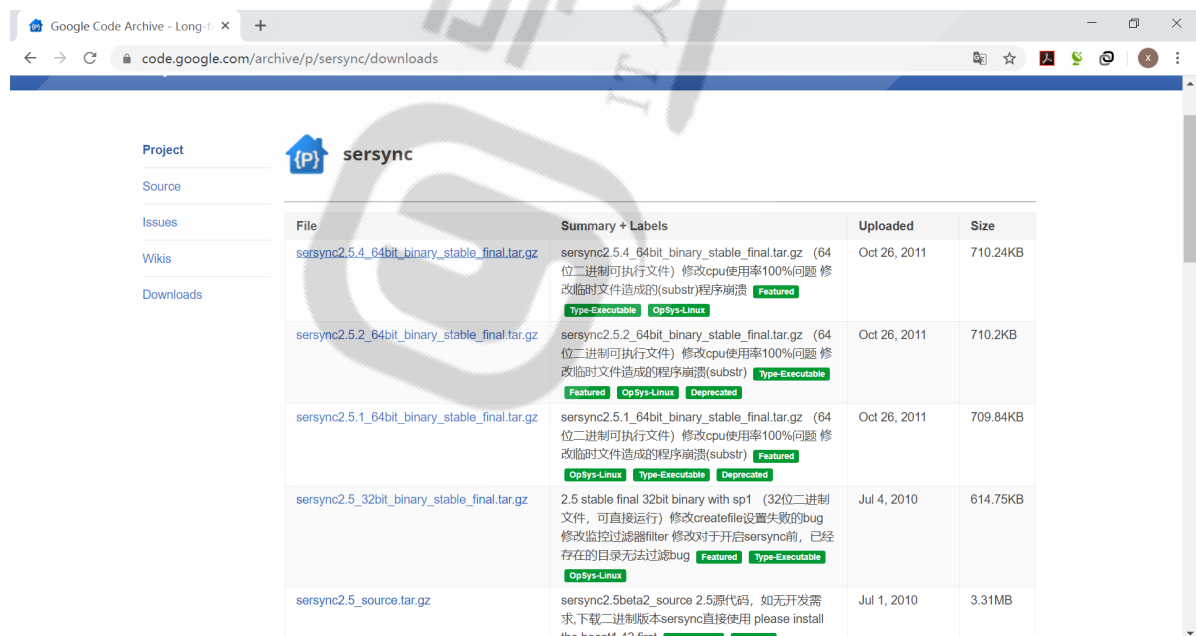
inotify最大的不足是会产生重复事件，或者同一个目录下多个文件的操作会产生多个事件，例如，当监控目录中有5个文件时，删除目录时会产生6个监控事件，从而导致重复调用rsync命令。另外比如：vim 文件时，inotify会监控到临时文件的事件，但这些事件相对于rsync来说是不应该被监控的

sersync 优点：

- sersync是使用c++编写，而且对linux系统文件系统产生的临时文件和重复的文件操作进行过滤，所以在结合rsync同步的时候，节省了运行时耗和网络资源。因此更快。
- sersync配置很简单，其中提供了静态编译好的二进制文件和xml配置文件，直接使用即可
- sersync使用多线程进行同步，尤其在同步较大文件时，能够保证多个服务器实时保持同步状态
- sersync有出错处理机制，通过失败队列对出错的文件重新同步，如果仍旧失败，则按设定时长对同步失败的文件重新同步
- sersync不仅可以实现实时同步，另外还自带crontab功能，只需在xml配置文件中开启，即也可以按要求隔一段时间整体同步一次，而无需再额外配置crontab功能
- sersync 可以二次开发

sersync项目地址：<https://code.google.com/archive/p/sersync/>

sersync下载地址：<https://code.google.com/archive/p/sersync/downloads>



The screenshot shows the Google Code Archive page for the sersync project. The page has a sidebar with links to Project, Source, Issues, Wikis, and Downloads. The main content area displays a table of available downloads.

File	Summary + Labels	Uploaded	Size
sersync2.5.4_64bit_binary_stable_final.tar.gz	sersync2.5.4_64bit_binary_stable_final.tar.gz (64位二进制可执行文件) 修改cpu使用率100%问题 修改临时文件造成的(substr)程序崩溃 Featured Type-Executable OpSys-Linux	Oct 26, 2011	710.24KB
sersync2.5.2_64bit_binary_stable_final.tar.gz	sersync2.5.2_64bit_binary_stable_final.tar.gz (64位二进制可执行文件) 修改cpu使用率100%问题 修改临时文件造成的程序崩溃(substr) Featured OpSys-Linux Deprecated Type-Executable	Oct 26, 2011	710.2KB
sersync2.5.1_64bit_binary_stable_final.tar.gz	sersync2.5.1_64bit_binary_stable_final.tar.gz (64位二进制可执行文件) 修改cpu使用率100%问题 修改临时文件造成的程序崩溃(substr) Featured OpSys-Linux Type-Executable Deprecated	Oct 26, 2011	709.84KB
sersync2.5_32bit_binary_stable_final.tar.gz	2.5 stable final 32bit binary with sp1 (32位二进制文件, 可直接运行) 修改createfile设置失败的bug 修改监控过滤器filter 修改对于开启sersync前, 已经存在的目录无法过滤bug Featured Type-Executable OpSys-Linux	Jul 4, 2010	614.75KB
sersync2.5_source.tar.gz	sersync2.5beta2_source 2.5源代码, 如无开发需求, 下载二进制版本sersync直接使用 please install the boost1.42 first OpSys-Source Featured	Jul 1, 2010	3.31MB

3.5.2 基于rsync daemon 实现 sersync

#在数据服务器上下载sersync，并拷贝至相应的目录，设置PATH变量

```
[root@data-centos8 ~]#wget https://storage.googleapis.com/google-code-archive-downloads/v2/code.google.com/sersync/sersync2.5.4_64bit_binary_stable_final.tar.gz
```

```
[root@data-centos8 ~]#tar xf sersync2.5.4_64bit_binary_stable_final.tar.gz
```

```
[root@data-centos8 ~]#cp -a GNU-Linux-x86 /usr/local/sersync
```

```
[root@data-centos8 ~]#echo 'PATH=/usr/local/sersync:$PATH' >
/etc/profile.d/sersync.sh
[root@data-centos8 ~]#source /etc/profile.d/sersync.sh
```

#sersync目录只有两个文件：一个是二进制程序文件，一个是xml格式的配置文件

```
[root@data-centos8 ~]#ls /usr/local/sersync/
confxml.xml sersync2
```

#确认安装rsync客户端工具

```
[root@data-centos8 ~]#rpm -q rsync &> /dev/null || dnf -y install rsync
```

#备份sersync配置文件

```
[root@data-centos8 ~]#cp /usr/local/sersync/confxml.xml{,.bak}
```

#修改sersync配置文件

```
[root@data-centos8 ~]#vim /usr/local/sersync/confxml.xml
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<head version="2.5">
```

```
  <host hostip="localhost" port="8008"></host>
```

```
  <debug start="false"/> # 是否开启调试模式
```

```
  <filesystem xfs="false"/>
```

```
  <filter start="false"> #不开启文件过滤功能，当为true时，以下类型的文件将不同
```

步

```
    <exclude expression="(.)\.svn"></exclude>
```

```
    <exclude expression="(.)\.gz"></exclude>
```

```
    <exclude expression="^info/*"></exclude>
```

```
    <exclude expression="^static/*"></exclude>
```

```
  </filter>
```

```
  <inotify> # 监控事件，默认监控
```

delete/close_write/moved_from/moved_to/create folder

```
    <delete start="true"/>
```

```
    <createFolder start="true"/>
```

```
    <createFile start="false"/>
```

```
    <closewrite start="true"/>
```

```
    <moveFrom start="true"/>
```

```
    <moveTo start="true"/>
```

```
    <attrib start="true"/> #修改此行为true，文件属性变化后也会同步
```

```
    <modify start="false"/>
```

```
  </inotify>
```

```
<sersync> # rsync命令的配置段
```

```
<localpath watch="/data/www"> #修改此行，需要同步的源目录或文件，建议同步目
```

录

<remote ip="备份服务器IP" name="backup"/> #修改此行，指定备份服务器地址和rsync daemon的模块名，如果下面开启了ssh start，此时name为远程shell方式运行时的目标目录

```
<!--<remote ip="192.168.8.39" name="tongbu"/>-->
```

```
<!--<remote ip="192.168.8.40" name="tongbu"/>-->
```

```
</localpath>
```

```
<rsync>
```

```
  <commonParams params="-artuz"/> # 指定rsync选项
```

<auth start="true" users="rsyncuser" passwordfile="/etc/rsync.pas"/> #修改此行为true，指定备份服务器的rsync配置的用户和密码文件

<userDefinedPort start="false" port="874"/><!-- port=874 -->#指定rsync的非标准端口号

```
<timeout start="false" time="100"/><!-- timeout=100 -->
```

<ssh start="false"/> #默认使用rsync daemon运行rsync命令，true为使用远程shell模式

```
</rsync>
```



```

<failLog path="/tmp/rsync_fail_log.sh" timeToExecute="60"/><!--default every
60mins execute once--> #错误重传及日志文件路径
<crontab start="false" schedule="600"><!--600mins--> #不开启crontab功能
    <crontabfilter start="false"> #不开启crontab定时传输的筛选功能
        <exclude expression="*.php"></exclude>
        <exclude expression="info/*"></exclude>
    </crontabfilter>
</crontab>
<plugin start="false" name="command"/>
</sersync>

#####以下行不需要修改
#####

<plugin name="command">
    <param prefix="/bin/sh" suffix="" ignoreError="true"/> <!--prefix
/opt/tongbu/mmm.sh suffix-->
    <filter start="false">
        <include expression="(.)\.php"/>
        <include expression="(.)\.sh"/>
    </filter>
</plugin>

<plugin name="socket">
<localpath watch="/opt/tongbu">
    <deshost ip="192.168.138.20" port="8009"/>
</localpath>
</plugin>
<plugin name="refreshCDN">
<localpath watch="/data0/htdocs/cms.xoyo.com/site/">
    <cdninfo domainname="ccms.chinacache.com" port="80" username="xxxx"
passwd="xxxx"/>
    <sendurl base="http://pic.xoyo.com/cms"/>
    <regexurl regex="false" match="cms.xoyo.com/site([/a-zA-Z0-
9]*)\.xoyo.com/images"/>
</localpath>
</plugin>
</head>

```

#创建连接rsynd服务器的用户密码文件,并必须修改权限

```
[root@data-centos8 ~]#echo 123456 > /etc/rsync.pas
```

```
[root@data-centos8 ~]#chmod 600 /etc/rsync.pas
```

#查看帮助

```
[root@data-centos8 ~]#sersync2 -h
```

set the system param

```
execute: echo 50000000 > /proc/sys/fs/inotify/max_user_watches
```

```
execute: echo 327679 > /proc/sys/fs/inotify/max_queued_events
```

parse the command param

参数-d:启用守护进程模式

参数-r:在监控前,将监控目录与远程主机用rsync命令推送一遍

c参数-n:指定开启守护线程的数量,默认为10个

参数-o:指定配置文件,默认使用当前工作目录下的confxml.xml文件

参数-m:单独启用其他模块,使用 -m refreshCDN 开启刷新CDN模块

参数-m:单独启用其他模块,使用 -m socket 开启socket模块

参数-m:单独启用其他模块,使用 -m http 开启http模块

不加-m参数,则默认执行同步程序

#以后台方式执行同步

```
[root@data-centos8 ~]# sersync2 -dro /usr/local/sersync/confxml.xml
set the system param
execute: echo 50000000 > /proc/sys/fs/inotify/max_user_watches
execute: echo 327679 > /proc/sys/fs/inotify/max_queued_events
parse the command param
option: -d run as a daemon
option: -r rsync all the local files to the remote servers before the sersync
work
option: -o config xml name: /usr/local/sersync/confxml.xml
daemon thread num: 10
parse xml config file
host ip : localhost host port: 8008
daemon start, sersync run behind the console
use rsync password-file :
user is rsyncuser
passwordfile is /etc/rsync.pas
config xml parse success
please set /etc/rsyncd.conf max connections=0 Manually
sersync working thread 12 = 1(primary thread) + 1(fail retry thread) +
10(daemon sub threads)
Max threads numbers is: 22 = 12(Thread pool nums) + 10(Sub threads)
please according your cpu , use -n param to adjust the cpu rate
-----
rsync the directory recursively to the remote servers once
working please wait...
#如果同步失败,可以手动执行下面命令,观察过程
[root@data-centos8 ~]# cd /data/www && rsync -artuz -R --delete ./
rsyncuser@backup-server::backup --password-file=/etc/rsync.pas >/dev/null 2>&1
run the sersync:
watch path is: /data/www
```

#sersync支持多实例，也即监控多个目录时，只需分别配置不同配置文件，然后使用sersync2指定对应配置文件运行

```
[root@data-centos8 ~]# sersync2 -rd -o /etc/sersync.d/nginx.xml
```

3.5.3 基于远程shell 实现 sersync

#不需要配置rsync daemon,只需要配置基于key验证的ssh即可

```
[root@data-centos8 ~]# ssh-keygen
```

```
[root@data-centos8 ~]# ssh-copy-id backup-server
```

#下载sersync，并拷贝至相应的目录，设置PATH变量同5.5.2

#修改sersync配置文件

```
[root@data-centos8 ~]# cat /usr/local/sersync/confxml.xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<head version="2.5">
  <host hostip="localhost" port="8008"></host>
  <debug start="false"/>
  <filesystem xfs="false"/>
  <filter start="false">
    <exclude expression="(.)\.svn"></exclude>
    <exclude expression="(.)\.gz"></exclude>
    <exclude expression="^info/*"></exclude>
```

```

<exclude expression="\^static/*"></exclude>
</filter>
<inotify>
<delete start="true"/>
<createFolder start="true"/>
<createFile start="false"/>
<closeWrite start="true"/>
<moveFrom start="true"/>
<moveTo start="true"/>
<attrib start="true"/> #修改此行为true
<modify start="false"/>
</inotify>

<sersync>
<localpath watch="/data/www"> #修改此行,指定源数据目录
    <remote ip="备份服务器IP" name="/data/backup"/> #修改此行指定备份服务器地址和备份目标目录
    <!--<remote ip="192.168.8.39" name="tongbu"/>-->
    <!--<remote ip="192.168.8.40" name="tongbu"/>-->
</localpath>
<rsync>
    <commonParams params="-artuz"/>
    <auth start="false" users="root" passwordfile="/etc/rsync.pas"/> #必须修改此行,不启用认证start=false
    <userDefinedPort start="false" port="874"/><!-- port=874 -->
    <timeout start="false" time="100"/><!-- timeout=100 -->
    <ssh start="true"/> #修改此行为true,使用远程shell方式的rsync连接方式,无需在目标主机上配置启动rsync daemon服务

#####以下行不需要修改#####
</rsync>
<failLog path="/tmp/rsync_fail_log.sh" timeToExecute="60"/><!--default every 60mins execute once-->
<crontab start="false" schedule="600"><!--600mins-->
    <crontabfilter start="false">
        <exclude expression="*.php"></exclude>
        <exclude expression="info/*"></exclude>
    </crontabfilter>
</crontab>
<plugin start="false" name="command"/>
</sersync>
#将中间的行可以删除
</head>

```

```
[root@data-centos8 ~]#sersync2 -dro /usr/local/sersync/confxml.xml
```

set the system param

```
execute: echo 50000000 > /proc/sys/fs/inotify/max_user_watches
```

```
execute: echo 327679 > /proc/sys/fs/inotify/max_queued_events
```

parse the command param

option: -d run as a daemon

option: -r rsync all the local files to the remote servers before the sersync work

option: -o config xml name: /apps/sersync/confxml.xml

daemon thread num: 10

parse xml config file

```

host ip : localhost host port: 8008
daemon start, sersync run behind the console
config xml parse success
please set /etc/rsyncd.conf max connections=0 Manually
sersync working thread 12 = 1(primary thread) + 1(fail retry thread) +
10(daemon sub threads)
Max threads numbers is: 22 = 12(Thread pool nums) + 10(Sub threads)
please according your cpu , use -n param to adjust the cpu rate
-----
rsync the directory recursively to the remote servers once
working please wait...
execute command: cd /data/www && rsync -auz -R --delete ./ -e ssh
10.0.0.18:/data/backup >/dev/null 2>&1
run the sersync:
watch path is: /data/www

```

3.6 实战案例：实现基于分布式的LAMP架构，并将NFS实时同步到备份服务器

