rsync远程同步

一、rsync介绍

1.1 rsync简介

rsync（Remote Sync，远程同步）是一个开源的快速备份工具，可以在不同主机之间镜像同步整个目录树，支持增量备份，并保持链接和权限，且采用优化的同步算法，传输前执行压缩，因此非常适用于异地备份、镜像服务器等应用。

rsync 的官方站点的网址是 rsync.samba.org/，目前最新版本是 3.1.3，由 Wayne Davison 进行维护。作为一种最常用的文件备份工具，rsync 往往是 Linux 和 UNIX 系统默 认安装的基本组件之一。

rsync 原理？

rsync 远程同步流程：

1. 客户端发起同步请求，与服务端建立连接。
2. 服务端验证用户身份。
3. 客户端扫描 rsync 命令输入的目录或文件，根据命令参数（--exclude、--include 等）筛选出需要同步的文件列表，发送给服务端。
4. 服务端将需要同步的文件按一定大小切分成一系列数据块，对它们进行编号，同时对每个数据块的内容计算两个校验码：32位的弱滚动校验码(rolling checksum，使用 adler-32 算法)和128位的MD5强校验码。每个数据块包含五个属性：数据块编号、起始偏移地址、长度、32位的弱滚动校验码、128位的MD5强校验码。将所有这些数据块信息发送给客户端。
5. 客户端收到数据块信息集合后，对每个数据块的32位的弱滚动校验码计算得到16位长度的 hash 值。

碰撞概率大小：

hash 值 > 32位的弱滚动校验码 > 128位的MD5强校验码

6. 客户端将对应的本地文件进行处理与对比，具体流程如下：

（1）从文件的第一个字节开始取相同大小的数据块，计算出32位的弱滚动校验码，再计算得到该验证码的16位长度的 hash 值；

（2）根据 hash 值在 hash table 中查找匹配项，若找不到匹配项则说明该数据块与服务端数据块不同，执行第（5）步骤，若找到了匹配项则进行下一步比较；

（3）将该数据块的32位的弱滚动校验码与 hash table 匹配项中的32位的弱滚动校验码进行比较，若找不到相同的，则说明该数据块与服务端数据块不同，执行第（5）步骤，若找到了匹配项则进行下一步比较；

（4）对该数据块计算得到128位的MD5强校验码，与 hash table 匹配项中的128位的MD5强校验码进行比较，若相同则说明其内容与服务端数据块内容完全相同，否则说明存在差异；

（5）若比较结果为相同，将该数据块信息发送给服务端，然后跳过该数据块，从该数据块的结尾偏移地址处继续取下一个数据块进行匹配比较；若比较结果为不同，则只跳过该数据块的第一个字节，从第二个字节开始取相同大小的数据块进行匹配比较，找到匹配的数据块之后，将之前不匹配的数据逐字节发送给服务端。

7. 服务端收到客户端的反馈信息之后，会创建一个临时文件，接收来自客户端的差异数据，相同的数据块则从本地对应的文件中读取，重组成一个新的文件，然后修改这个临时文件的属性信息，重命名该文件替换掉原来对应的文件，至此同步完成。

如果数据块大小太小，则数据块的数量就太多，需要计算和匹配的数据块校验码就太多，性能就差，而且出现hash值重复、rolling checksum重复的可能性也增大；如果数据块大小太大，则可能会出现很多数据块都无法匹配的情况，导致这些数据块都被传输，降低了增量传输的优势。默认情况下，rsync会根据文件大小自动判断数据块大小，但rsync命令的"-B"(或"--block-size")选项支持手动指定大小，如果手动指定，官方建议大小在500-1000字节之间。

每个文件的同步过程是持续进行的：

客户端并不是对比完所有数据块之后才将相关数据发送给服务端的，而是每搜索出一个匹配数据块，就会立即将匹配块的相关信息以及当前匹配块和上一个匹配块中间的非匹配数据发送给服务端，并开始处理下一个数据块，服务端也是每收到一段数据后就会立即将其重组到临时文件中。

rsync 的优缺点与适用场景？

1. 优点

（1）可以镜像保存整个目录树和文件系统。
（2）可以很容易做到保持原来文件的权限、时间、软硬链接等等。
（3）无须特殊权限即可安装。
（4）快速：第一次同步时 rsync 会复制全部内容，但在下一次只传输修改过的文件。rsync 在传输数据的过程中可以实行压缩及解压缩操作，因此可以使用更少的带宽。
（5）安全：可以使用scp、ssh等方式来传输文件，当然也可以通过直接的socket连接。
（6）支持匿名传输，以方便进行网站镜像。
（7）跨平台：可在不同操作系统之间同步数据。

2. 缺点

（1）客户端需要对多个文件数据块进行多次计算与比较验证码，对 CPU 的消耗比较大；服务端需要根据原文件和客户端传送过来的差异数据进行文件内容重组，对 IO 的消耗比较大。
（2）rsync 每次同步都需要先进行所有文件的扫描和计算、对比，最后才能进行差量传输。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。而且如果改动的只是其中很小的一部分，这就是非常低效的方式。
（3）rsync 不能实时的去监测、同步数据，虽然它可以通过 crontab 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（无法实现实时同步），而且繁忙的轮询会消耗大量的资源。

3. 适用场景

由 rsync 工作原理可知，需要同步的文件改动越频繁，则客户端需要计算和比较的数据块验证码就越多（遇到数据块内容不相同时，只能跳过一个字节继续往后计算与比较，相同则可跳过一个数据块），对 CPU 的消耗就会越大；需要同步的文件越大，服务端每次都需要从一个很大的文件中复制相同的数据块进行新文件重组，几乎相当于直接 cp 了一个大文件，对 IO 的消耗也就越大。

所以 rsync 适合对改动不频繁、大小比较小的文件进行同步，对于改动频繁的大文件，只能偶尔同步一次，相当于备份的功能，而不是同步。

何为 inotify？

Inotify 是 Linux 内核从 2.6.13 开始引入的特性，它是一个内核用于通知用户空间程序文件系统变化的机制。

Inotify 监控文件系统操作，比如读取、写入和创建，基于事件驱动，可以做到对事件的实时响应，高效，而且没有轮询造成的系统资源消耗。

rsync + inotify 问题？

1. 同一个操作会触发多个事件。
2. inotifywait 存在缺陷，当向监控目录下拷贝复杂层次目录(多层次目录中包含文件)，或者向其中拷贝大量文件时，inotifywait 经常会随机性地遗漏某些文件。
3. 并发如果大于 200 个文件（10-100K），同步会有延迟。
4. 监控到事件后，调用 rsync 同步是单线程的。

更完美的方案？ —— sersync！

sersync 是金山的周洋基于 rsync + inotify-tools 开发的工具，它克服了 inotify 的缺陷，可以过滤重复事件减轻负担，并且自带 crontab 功能、多线程调用 rsync、失败重传等功能。

当同步数据量不大时，还是建议使用 rsync + inotify，当数据量很大（几百G甚至1T以上）时，建议使用 rsync + sersync。

1.2 rsync特性

支持拷贝特殊文件，如连接文件、设备等。

可以有排除指定文件或目录同步的功能，相当于打包命令tar的排除功能。

可以做到保持原文件或目录的权限、时间、软硬链接、属主、组等所有属性均不改变 –p。

可以实现增量同步，既只同步发生变化的数据，因此数据传输效率很高（tar-N）。

可以使用rcp、rsh、ssh等方式来配合传输文件（rsync本身不对数据加密）。

可以通过socket（进程方式）传输文件和数据（服务端和客户端）。

支持匿名的活认证（无需系统用户）的进程模式传输，可以实现方便安全的进行数据备份和镜像。

二、rsync同步源服务器

在远程同步任务中，负责发起 rsync 同步操作的客户机称为发起端，而负责响应来自客户机的 rsync 同步操作的服务器称为同步源。

- 在下行同步（下载）中，同步源负责提供文档的原始位置，发起端应对该位置有读取权限。
- 在上行同步（上传）中，同步源负责提供文档的目标位置，发起端应对该位置具有写入权限。

三、配置rsync下行同步（定时同步）

源服务器：192.168.68.111

客户机（发起端）：192.168.68.112

3.1 配置源服务器

```
 systemctl stop firewalld
 setenforce 0

rpm -q rsync                                    #一般系统已默认安装rsync

#建立/etc/rsyncd.conf 配置文件

vim /etc/rsyncd.conf                              #添加以下配置项
uid = root                                      #也可以为root（我用的是root）
gid = root                                      #也可以为root
use chroot = yes                                #禁锢在源目录
address = 192.168.68.111                          #监听地址，监听本机地址
port 873                                          #监听端口 tcp/udp 873，可通过cat /et
c/services | grep rsync查看
log file = /var/log/rsyncd.log                  #日志文件位置
pid file = /var/run/rsyncd.pid                  #存放进程 ID 的文件位置
hosts allow = 192.168.68.0/24                   #允许同步的客户机网段
[wwwroot]                                       #共享模块名称
path = /var/www/html                            #源目录的实际路径（同步的目录）
comment = Document Root of www.clj.com
read only = yes                                   #是否为只读
dont compress = *.gz *.bz2 *.tgz *.zip *.rar *.z          #同步时不再压缩的文件类型
auth users = backuper                           #授权账户，多个账号以空格分隔
secrets file = /etc/rsyncd_users.db               #存放账户信息的数据文件
-------------------------------------------------------------------------------
uid = nobody
gid = nobody
use chroot = yes
address = 192.168.68.111
port 873
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid
hosts allow = 192.168.68.0/24
[wwwroot]
path = /var/www/html
comment = Document Root of www.clj.com
read only = yes
dont compress = *.gz *.bz2 *.tgz *.zip *.rar *.z
auth users = backuper
secrets file = /etc/rsyncd_users.db
-------------------------------------------------------------------------------
--
#如采用匿名的方式，只要将其中的“auth users”和“secrets file”配置项去掉即可。
```

```
[root@localhost ~]#  systemctl stop firewalld
[root@localhost ~]#  setenforce 0
[root@localhost ~]# rpm -q rsync
rsync-3.0.9-18.el7.x86_64
[root@localhost ~]# vim /etc/rsyncd.conf
[root@localhost ~]# vim /etc/rsyncd.conf
[root@localhost ~]#
```

```
#         path = /home/ftp
#         comment = ftp export area
uid = nobody
gid = nobody
use chroot = yes
address = 192.168.68.111
port = 873
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid
hosts allow = 192.168.68.0/24
dont compress = *.gz *.bz2 *.tgz *.zip *.rar *.z

[wwwroot]
path = /var/www/html
comment = Document Root of www.tt.com
read only = yes
auth users = backuper
secrets file = /etc/rsyncd_users.db
```

```
[root@localhost ~]# vim /etc/rsyncd_users.db

backuper:abc123
~
~
~
```

```
[root@localhost ~]# yum -y install httpd
已加载插件：fastestmirror, langpacks
Loading mirror speeds from cached hostfile
正在解决依赖关系
--> 正在检查事务
---> 软件包 httpd.x86_64.0.2.4.6-67.el7.centos 将被 安装
--> 正在处理依赖关系 httpd-tools = 2.4.6-67.el7.centos，它被软件包 httpd-2.4.6-67.el7.centos.x86_6
 需要
--> 正在处理依赖关系 /etc/mime.types，它被软件包 httpd-2.4.6-67.el7.centos.x86_64 需要
--> 正在处理依赖关系 libaprutil-1.so.0()(64bit)，它被软件包 httpd-2.4.6-67.el7.centos.x86_64 需要
--> 正在处理依赖关系 libapr-1.so.0()(64bit)，它被软件包 httpd-2.4.6-67.el7.centos.x86_64 需要
--> 正在检查事务
```

```
完毕！
[root@localhost ~]# chmod +r /var/www/html/
[root@localhost ~]# ls -ld /var/www/html/
drwxr-xr-x. 2 root root 6 8月    4 2017 /var/www/html/
[root@localhost ~]#
```

```
[root@localhost ~]# rsync --daemon
[root@localhost ~]# netstat -anpt | grep rsync
tcp        0      0 192.168.68.111:873      0.0.0.0:*               LISTEN      59004/rsync

[root@localhost ~]#
```

```
[root@localhost ~]# kill $(cat /var/run/rsyncd.pid)
[root@localhost ~]# rm -rf /var/run/rsyncd.pid
[root@localhost ~]# rsync --daemon
[root@localhost ~]# netstat -anpt | grep rsync
tcp        0      0 192.168.68.111:873      0.0.0.0:*               LISTEN      59019/rsync
```

创建文件用于测试

```
cd /var/www/html
echo "this is test" >> 1.txt
cat 1.txt
```

```
[root@localhost ~]# cd /var/www/html
[root@localhost html]# echo "this is test" >> 1.txt
[root@localhost html]# cat 1.txt
this is test
```

发起端配置

```
rsync  [选项]  原始位置  目标位置
```

-r递归模式，包含目录及子目录中的所有文件。
-l对于符号链接文件仍然复制为符号链接文件。
-v显示同步过程的详细（verbose）信息。
-z在传输文件时进行压缩（compress）。
-a归档模式，保留文件的权限、属性等信息，等同于组合选项“-rlptgop"。
-p保留文件的权限标记。-t保留文件的时间标记。
-g保留文件的属组标记（仅超级用户使用）。
-o保留文件的属主标记（仅超级用户使用）。
-H保留硬连接文件。
-A保留ACL属性信息。
-D保留设备文件及其他特殊文件。
--delete删除目标位置有而原始位置没有的文件。
--checksum根据校验和（而不是文件大小、修改时间）来决定是否跳过文件。

配置：

```
#将指定的资源下载到本地/opt目录下进行备份。密码abc123

格式一：#用户名@主机地址::共享模块名
rsync -avz backuper@192.168.68.111::wwwroot /opt/  #wwwroot为共享模块名，密码abc123
#backuper指的是我在同步的时候用的哪个用户身份
#wwwroot代表的是模块，模块下面会写同步的默认路径和一些特性，所以我们只需要写模块就好了
#/opt/指的是同步到本地的目录

格式二：#rsync:/用户名@主机地址/共享模块名
rsync -avz rsync://backuper@192.168.68.111/wwwroot /opt/


#免交互格式配置：
echo "abc123" > /etc/server.pass
```

```
chmod 600 /etc/server.pass     #密码文件权限必须为600，即除了属主，其他人都没有查看权限。

rsync -avz --password-file=/etc/server.pass backuper@192.168.68.111::wwwroot /opt/     #免密同步


#定时同步
crontab -e
30 22 * * * /usr/bin/rsync -az --delete --password-file=/etc/server.pass backuper@192.168.68.111::wwwroot /opt/
#为了在同步过程中不用输入密码，需要创建一个密码文件，保存backuper用户的密码，如/etc/server.pass。 在执行rsync 同步时使用选项"--password-file=/etc/server.pass"指定即可。

systemctl restart crond
systemctl enable crond
```

```
[root@localhost ~]# rsync -avz backuper@192.168.68.111::wwwroot /opt/
Password:
receiving incremental file list
./
1.txt

sent 83 bytes  received 167 bytes  9.09 bytes/sec
total size is 13  speedup is 0.05
[root@localhost ~]#
```

```
root@localhost html]# systemctl restart crond
root@localhost html]#  systemctl enable crond
root@localhost html]#
```

```
30 22 * * * /usr/bin/rsync -az --delete --password-file=/etc/server.pass backuper@192.16
wwroot /opt/
~
```

```
[root@localhost ~]# rsync -avz backuper@192.168.68.101::wwwroot /opt/
Password:
receiving incremental file list
./
1.txt

sent 83 bytes  received 163 bytes  54.67 bytes/sec
total size is 13  speedup is 0.05
[root@localhost ~]# echo "abc123" > /etc/server.pass
[root@localhost ~]# chmod 600 /etc/server.pass
[root@localhost ~]# rsync -avz --password-file=/etc/server.pass backuper@192.168.68.101::wwwroot /o
pt/
receiving incremental file list

sent 61 bytes  received 108 bytes  338.00 bytes/sec
total size is 13  speedup is 0.08
[root@localhost ~]# crontab -e
no crontab for root - using an empty one
crontab: installing new crontab
[root@localhost ~]# systemctl restart crond
[root@localhost ~]#  systemctl enable crond
[root@localhost ~]#
```

验证：（去源服务器编辑一个1.HTML）

```
[root@localhost html]# vim 1.html
[root@localhost html]#
```

```
[root@localhost opt]# rsync -avz --password-file=/etc/server.pass backuper@192.168.68.111::wwwroot
/opt/
receiving incremental file list
./
1.html

sent 83 bytes  received 173 bytes  12.49 bytes/sec
total size is 17  speedup is 0.07
```

```
[root@localhost opt]# ls
1.html  1.txt  mysql-5.7.20  mysql-boost-5.7.20.tar.gz  rh
[root@localhost opt]#
```

rsync实时同步（上行同步）

定期同步的不足

- 执行备份的时间固定，延迟明显、实时性差
- 当同步源长期不变化时，密集的定期任务是不必要的

实时同步的优点

- 一旦同步源出现变化，立即启动备份
- 只要同步源无变化，则不执行备份

Linux内核的inotify机制

- 从版本2.6.13开始提供
- 可以监控文件系统的变动情况，并做出通知响应
- 辅助软件：inotify-tools

发起端配置rsync+Inotify

- 使用inotify通知接口，可以用来监控文件系统的各种变化情况，如文件存取、删除、移动、修改等。利用这一机制，可以非常方便地实现文件异动告警、增量备份，并针对目录或文件的变化及时作出响应。
- 将inotify机制与rsync工具相结合，可以实现触发式备份（实时同步），即只要原始位置的文档发生变化，则立即启动增量备份操作；否则处于静默等待状态。
- 因为 inotify 通知机制由 Linux 内核提供，因此主要做本机监控，在触发式备份中应用时更适合上行同步。

配置rsync实时同步（上行同步）

发起端需要配置 rsync+Inotify

修改rsync源服务器配置文件

```
vim /etc/rsyncd.conf
read only = no    #关闭只读，上行同步需要可以写

#之后重启
kill $(cat /var/run/rsyncd.pid)
rm -rf /var/run/rsyncd.pid
rsync --daemon
netstat -anpt | grep rsync

chmod 777 /var/www/html
```

```
uid = root
gid = root
use chroot = yes
address = 192.168.68.111
port = 873
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid
hosts allow = 192.168.68.0/24
dont compress = *.gz *.bz2 *.tgz *.zip *.rar *.z

[wwwroot]
path = /var/www/html
comment = Document Root of www.tt.com
read only = no
auth users = backuper
secrets file = /etc/rsyncd_users.db
```

```
[root@localhost html]# kill $(cat /var/run/rsyncd.pid)
[root@localhost html]#  rm -rf /var/run/rsyncd.pid
[root@localhost html]# rsync --daemon
[root@localhost html]#  netstat -anpt | grep rsync
tcp        0      0 192.168.68.111:873      0.0.0.0:*               LISTEN      58782/rsync

[root@localhost html]# chmod 777 /var/www/html
[root@localhost html]#
```

发起端，调整 inotify 内核参数（客户机）

在Linux内核中，默认的inotify机制提供了三个调控参数：

- max_queue_events (监控事件队列，默认值为16384)、
- max_user_instances (最多监控实例数，默认值为128)、

- max_user_watches (每个实例最多监控文件数，默认值为8192)。

当要监控的目录、文件数量较多或者变化较频繁时，建议加大这三个参数的值。

```
cat /proc/sys/fs/inotify/max_queued_events
cat /proc/sys/fs/inotify/max_user_instances
cat /proc/sys/fs/inotify/max_user_watches

vim /etc/sysctl.conf     #内核参数都在该文件中修改
fs.inotify.max_queued_events = 16384
fs.inotify.max_user_instances = 1024
fs.inotify.max_user_watches = 1048576

sysctl -p
```

```
[root@localhost opt]# ls
1.html  1.txt  mysql-5.7.20  mysql-boost-5.7.20.tar.gz  rh
[root@localhost opt]#  vim /etc/rsyncd.conf
[root@localhost opt]# cat /proc/sys/fs/inotify/max_queued_events
16384
[root@localhost opt]# cat /proc/sys/fs/inotify/max_user_instances
128
[root@localhost opt]#  cat /proc/sys/fs/inotify/max_user_watches
8192
```

```
[root@localhost opt]# vim /etc/sysctl.conf
[root@localhost opt]# sysctl -p
fs.inotify.max_queued_events = 16384
fs.inotify.max_user_instances = 1024
fs.inotify.max_user_watches = 1048576
```

发起端，安装 inotify-tools

用inotify 机制还需要安装inotify-tools， 以便提供inotifywait、 inotifywatch 辅助工具程序，用来监控、汇总改动情况。

- inotifywait：可监控modify (修改)、create (创建)、move (移动)、delete (删除)、attrib (属性更改)等各种事件，一有变动立即输出结果。
- inotifywatch：可用来收集文件系统变动情况，并在运行结束后输出汇总的变化情况。

  yum -y install gcc gcc-c++ make tar zxvf inotify-tools-3.14.tar.gz -C /opt/ cd /opt/inotify-tools-3.14 ./configure make && make install #可以先执行“inotifywait”命令，然后另外再开启一个新终端向 /data 目录下添加文件、移动文件，在原来的终端中跟踪屏幕输出结果。inotifywait -mrq -e modify,create,move,delete /data #选项“-e”：用来指定要监控哪些事件 #选项“-m”：表示持续监控 #选项“-r”：表示递归整个目录 #选项“-q”：简化输出信息

发起端，编写触发式同步脚本

在另外一个终端编写触发式同步脚本 (注意，脚本名不可包含 rsync 字符串，否则脚本可能不生效) 。

```
vim /opt/inotify.sh
#!/bin/bash

#定义inotifywait监控/opt/abc目录中文件事件的变量。attrib表示属性变化。
INOTIFY_CMD="inotifywait -mrq -e modify,create,attrib,move,delete /opt/abc"

#定义执行 rysnc 上行同步的变量。--delete保证两边目录内容一致，可以不加。
RSYNC_CMD="rsynC -azH --delete --password-file=/etc/server.pass /opt/abc backup
er@192.168.68.111::wwwroot"


#使用while、read持续获取监控结果，根据结果可以作进一步判断是否读取到输出的监控记录
```

```
$INOTIFY_CMD | while read DIRECTORY EVENT FILE
do
   #如果rsync未在执行，则立即启动
   if [ $(pgrep rsync | wc -l) -le 0 ];then
        $RSYNC_CMD
   fi
done

chmod +x /opt/inotify.sh

chmod +x /etc/rc.d/rc.local      #开机自启脚本文件
echo '/opt/inotify.sh' >> /etc/rc.d/rc.local  #加入开机自动执行

#之后运行脚本（后台运行）
cd /opt/
./inotify.sh &
#之后在发起端创建文件，查看源服务器中是否新增了
```

```
[root@localhost opt]# vim /opt/inotify.sh
[root@localhost opt]# chmod +x /opt/inotify.sh
[root@localhost opt]# chmod +x /etc/rc.d/rc.local
[root@localhost opt]# echo '/opt/inotify.sh' >> /etc/rc.d/rc.local
[root@localhost opt]# cd /opt/
[root@localhost opt]# ./inotify.sh &
[1] 24692
```

```
NOTIFY_CMD="inotifywait -mrq -e modify,create,attrib,move,delete /opt/abc"
RSYNC_CMD="rsynC -azH --delete --password-file=/etc/server.pass /opt/abc backuper@192.168.68.111::w
wwroot"
$INOTIFY_CMD | while read DIRECTORY EVENT FILE
do
if [ $(pgrep rsync | wc -l) -le 0 ];then
$RSYNC_CMD
fi
done
~
~
```

如果同步的文件比较大，同步时比较慢导致后面文件没来的及同步，则需要在脚本内添加消息队列或缓冲：

```
#!/bin/bash
#定义inotifywait监控目录中文件事件的变量
INOTIEY_CMD="inotifywait -mrq -e modify,create,attrib,move,delete /opt/abc"
#定义执行rsync上行同步的变量
RSYNC_CMD="rsync -azH --delete --password-file=/etc/server.pass /opt/abc backup
er@192.168.136.10::wwwroot"
#使用while、read持续获取监控结果，根据结果可以进一步判断是否读取到输出的监控记录
$INOTIEY_CMD | while read DIRECTORY EVENT FILE
do
       #小于等于0，则等待它执行完再去同步其他文件
       until [ $(pgrep rsync | wc -l) -le 0 ]

       do
          sleep 1
       done
       $RSYNC_CMD
done
```

验证同步效果

上述脚本用来检测本机/opt/abc目录的变动情况，一旦有更新触发rsync 同步操作，上传备份至服务器192.168.136.10 的wwwroot共享目录下。

触发式上行同步的验证过程如下:

（1）在本机运行/opt/inotify.sh 脚本程序.

（2）切换到本机的 /opt/abc 目录，执行增加、删除、修改文件等操作。

（3）查看远端服务器中的wwwroot目录下的变化情况。

```
[root@localhost abc]#: ls
111  123  123.txt  456  789
```

```
[root@localhost html]#: ls
111  123  123.txt  222  456  789
```

使用rsync快速删除大量文件

假如要在linux下删除大量文件，比如100万、1000万，像/usr/local/nginx/proxy_temp的nginx缓存等，那么rm -rf * 可能就不好用了，因为要等待很长一段时间。

在这种情况下我们可以使用rsync来巧妙处理。

rsync实际用的是替换原理。

```
mkdir /root/blank     #空文件夹
mkdir /opt/test
cd /opt/test
touch {1..1000}.txt   #模拟大缓存文件

rsync --delete-before -a -H -v --progress --stats /root/blank/ /opt/test
      #这样目标目录很快就被清空了

ls /opt/test
```

--delete-before

接收者在传输进行删除操作

-a

归档模式，表示以递归方式传输文件，并保持所有文件属性

-H

保持硬连接的文件

-v

详细输出模式

--progress

在传输时显示传输过程

--stats

给出某些文件的传输状态

登录后才能查看或发表评论，立即 登录 或者 逛逛 博客园首页