简介：

inotify的优点：

（1） 监控文件系统时间的变化，通过同步工具实现实时同步数据

inotify的缺点：

（1） 并发如果大于200个文件（10-100K），同步就会有延迟

（2）我们前面写的脚本，每次都是全部推送一次，但确实是增量备份额

       也可以只同步变化的文件  不变化的文件不理

（3） 监控到事件后，调用rsync同步是单线程的（加&并发）。sersync是多线程同步的

既然有了inotify-tools，为什么还要开发sersync

sersync的功能多

（1）配置文件

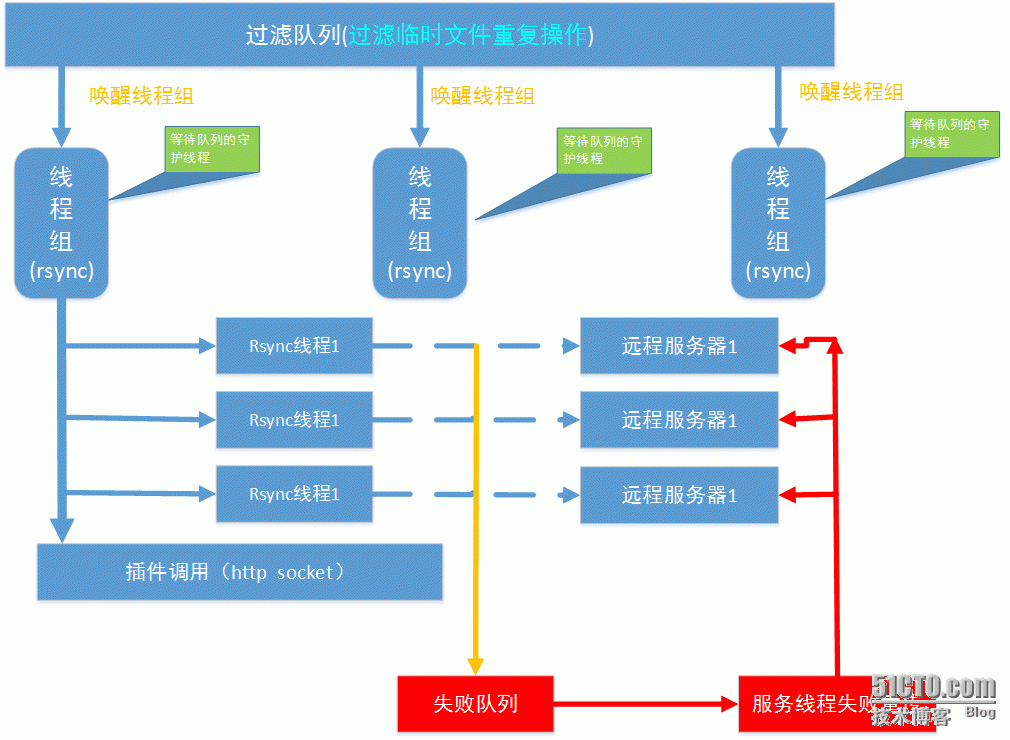
（2） 真正的守护进程socket

（3）可以对失败的文件定时重传

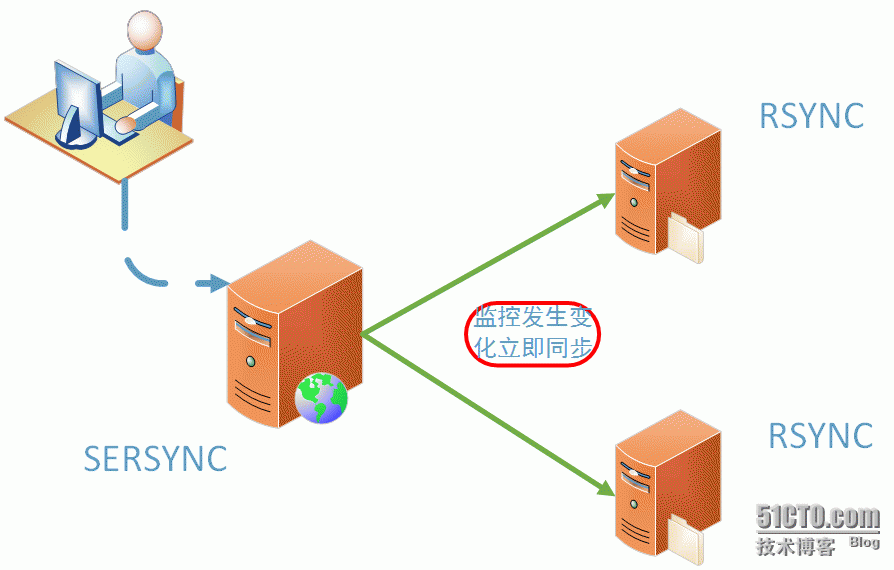
（4）第三方的http接口

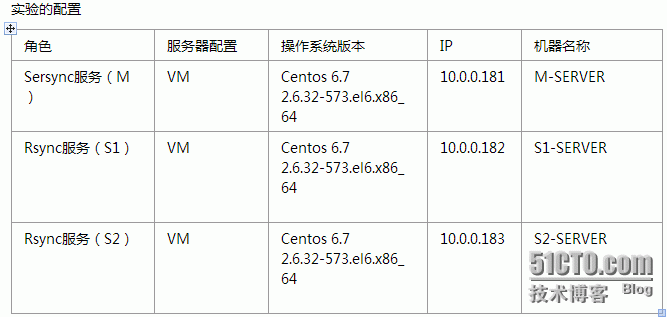
 (5) 默认是多线程的同步

sersync原理图解

[](http://s1.51cto.com/wyfs02/M01/7F/5A/wKioL1cbU8Ki22mcAAEaK-TPbgU447.png)

本次实战的图解

[](http://s4.51cto.com/wyfs02/M00/7F/5A/wKioL1cbUwvAM91xAACg10RT-4w529.png)实验配置

[](http://s5.51cto.com/wyfs02/M01/7F/5C/wKiom1cbUpzxMNyYAAA1UbPZC0o031.png)

1.Rsync服务（S1和S2的配置，都一样），这个配置文件默认是不存在的（得新建）

[[root@S2-SERVER](mailto:root@S2-SERVER) ~]# vim   /etc/rsyncd.conf

#QQ 31333741 blog:[http://oldboy.blog.51cto.com](http://oldboy.blog.51cto.com/)

##rsyncd.conf start##

uid = rsync

gid = rsync

use chroot = no

max connections = 200

timeout = 300

pid file = /var/run/rsyncd.pid

lock file = /var/run/rsync.lock

log file = /var/log/rsyncd.log

ignore errors

read only = false

list = false

hosts allow = 10.0.0.0/24

hosts deny = 0.0.0.0/32

auth users = rsync\_backup

secrets file = /etc/rsync.password

[www]

comment = "write by  sanlang"

path = /data0/www/www

[bbs]

comment = "write by sanlang"

path = /data0/www/bbs

[blog]

comment = "write by sanlang"

path = /data0/www/blog

"/etc/rsyncd.conf" 27L, 604C 已写入

[[root@S2-SERVER](mailto:root@S2-SERVER) ~]#

2. Rsync服务（S1和S2的配置，都一样）添加用户

[[root@S2-SERVER](mailto:root@S2-SERVER) ~]# useradd  rsync -s /sbin/nologin  -M

3.修改/或者密码文件（默认是不存的）

[[root@S2-SERVER](mailto:root@S2-SERVER) ~]# vim /etc/rsync.password

rsync\_backup:oldboy

[[root@S2-SERVER](mailto:root@S2-SERVER) ~]# cat   /etc/rsync.password

rsync\_backup:oldboy

[[root@S2-SERVER](mailto:root@S2-SERVER) ~]# chmod 600  /etc/rsync.password

[[root@S2-SERVER](mailto:root@S2-SERVER) ~]#

4. Rsync服务（S1和S2的配置，都一样）创建目录，并把目录赋予相应的权限

[[root@S2-SERVER](mailto:root@S2-SERVER) ~]# mkdir /data0/www/www  -p

[[root@S2-SERVER](mailto:root@S2-SERVER) ~]# mkdir -p  /data0/www/bbs

[[root@S2-SERVER](mailto:root@S2-SERVER) ~]# mkdir -p /data0/www/blog

[[root@S2-SERVER](mailto:root@S2-SERVER) ~]# chown -R  rsync.rsync  /data0

5.以守护进程的方式启动rsync

[[root@S1-SERVER](mailto:root@S1-SERVER) ~]# rsync  --daemon

[[root@S1-SERVER](mailto:root@S1-SERVER) ~]# echo "/usr/bin/rsync  --daemon">>/etc/rc.local

可能会出错错误：

[[root@S1-SERVER](mailto:root@S1-SERVER) ~]# rsync  --daemon

[[root@S1-SERVER](mailto:root@S1-SERVER) ~]# failed to create pid file /var/run/rsyncd.pid: File exists

解决方法：

[[root@S1-SERVER](mailto:root@S1-SERVER) ~]# rm -f /var/run/rsyncd.pid

**6.Sersync服务（M）端的配置（默认文件不存在自己指定）**

[[root@M-SERVER](mailto:root@M-SERVER) ~]# cat   /etc/rsync.password

oldboy

[[root@M-SERVER](mailto:root@M-SERVER) ~]# chmod 600   /etc/rsync.password

[[root@M-SERVER](mailto:root@M-SERVER) ~]# ll  /etc/rsync.password

-rw-------. 1 root root 7 4月  22 16:18 /etc/rsync.password

[[root@M-SERVER](mailto:root@M-SERVER) ~]#

7.测试（M-SERVER 上执行OK）

[[root@M-SERVER](mailto:root@M-SERVER) ~]# cd   /backup

[[root@M-SERVER](mailto:root@M-SERVER) backup]# ls

sanlang01.txt  sanlang03.txt  sanlang05.txt  sanlang07.txt  sanlang09.txt

sanlang02.txt  sanlang04.txt  sanlang06.txt  sanlang08.txt  sanlang10.txt

[[root@M-SERVER](mailto:root@M-SERVER) backup]# cd

[[root@M-SERVER](mailto:root@M-SERVER) ~]# rsync -avz /backup/ [rsync\_backup@10.0.0.182](mailto:rsync_backup@10.0.0.182)::www/  --password-file=/etc/rsync.password

sending incremental file list

./

sanlang01.txt

sanlang02.txt

sanlang03.txt

sanlang04.txt

sanlang05.txt

sanlang06.txt

sanlang07.txt

sanlang08.txt

sanlang09.txt

sanlang10.txt

sent 508 bytes  received 201 bytes  1418.00 bytes/sec

total size is 0  speedup is 0.00

[[root@M-SERVER](mailto:root@M-SERVER) ~]#

**6.Sersync服务（M）端的配置**（可以不加，主要是把安装包放在自己的家目录下）

[[root@M-SERVER](mailto:root@M-SERVER) ~]# useradd   oldboy

[[root@M-SERVER](mailto:root@M-SERVER) ~]# passwd  oldboy

更改用户 oldboy 的密码 。

新的 密码：

无效的密码： 过于简单化/系统化

无效的密码： 过于简单

重新输入新的 密码：

passwd： 所有的身份验证令牌已经成功更新。

[[root@M-SERVER](mailto:root@M-SERVER) ~]#

新建一个目录用于sersync的安装

[[root@M-SERVER](mailto:root@M-SERVER) ~]# mkdir -p  /home/oldboy/tools

[[root@M-SERVER](mailto:root@M-SERVER) ~]#

7.**Sersync服务（M）端开始安装sersync**

[[root@M-SERVER](mailto:root@M-SERVER) tools]# rz

rz waiting to receive.

 zmodem trl+C

[[root@M-SERVER](mailto:root@M-SERVER) tools]# rz

rz waiting to receive.

 zmodem trl+C

  100%     710 KB  710 KB/s 00:00:01       0 Errorsary\_stable\_final.tar.gz...

[[root@M-SERVER](mailto:root@M-SERVER) tools]# ls

sersync2.5.4\_64bit\_binary\_stable\_final.tar.gz

[[root@M-SERVER](mailto:root@M-SERVER) tools]#

**8.Sersync服务（M）端.解压文件（解压到/usr/local/ 下面）**

[[root@M-SERVER](mailto:root@M-SERVER) tools]# tar  -zxvf  sersync2.5.4\_64bit\_binary\_stable\_final.tar.gz  -C   /usr/local/

GNU-Linux-x86/

GNU-Linux-x86/sersync2

GNU-Linux-x86/confxml.xml

[[root@M-SERVER](mailto:root@M-SERVER) tools]# cd /usr/local/

[[root@M-SERVER](mailto:root@M-SERVER) local]# ls

bin  etc  games  GNU-Linux-x86  include  lib  lib64  libexec  sbin  share  src

[[root@M-SERVER](mailto:root@M-SERVER) local]#

**9.Sersync服务（M）端.改名解压文件的名称**

[[root@M-SERVER](mailto:root@M-SERVER) local]# mv  GNU-Linux-x86  sersync

[[root@M-SERVER](mailto:root@M-SERVER) local]#

[[root@M-SERVER](mailto:root@M-SERVER) local]# tree sersync/

sersync/

├── confxml.xml

└── sersync2

0 directories, 2 files

[[root@M-SERVER](mailto:root@M-SERVER) local]#

**10.规范安装包（在  /usr/local/sersync 下操作）**

[[root@M-SERVER](mailto:root@M-SERVER) sersync]# mkdir conf  bin logs

[[root@M-SERVER](mailto:root@M-SERVER) sersync]# mv  confxml.xml   conf/

[[root@M-SERVER](mailto:root@M-SERVER) sersync]# mv sersync2   bin/sersync

[[root@M-SERVER](mailto:root@M-SERVER) local]# tree  sersync/

sersync/

├── bin

│   └── sersync

├── conf

│   └── confxml.xml

└── logs

3 directories, 2 files

[[root@M-SERVER](mailto:root@M-SERVER) local]#

**11.备份配置文件并且配置**

[[root@M-SERVER](mailto:root@M-SERVER) sersync]# cd  /usr/local/sersync/conf/

[[root@M-SERVER](mailto:root@M-SERVER) conf]# cp confxml.xml  confxml.xml.org

[[root@M-SERVER](mailto:root@M-SERVER) conf]# ls

confxml.xml  confxml.xml.org

[[root@M-SERVER](mailto:root@M-SERVER) conf]#

①编辑confxml.xml

 24         <localpath watch="/opt/tongbu">

 25             <remote ip="127.0.0.1" name="tongbu1"/>

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

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

 28         </localpath>

改成：

 24            <localpath watch="/data0/www/www">        #data/www/www  代表内地目录

 25             <remote ip="10.0.0.182"  name="www"/>     # www 这个是远程服务器的模块

 26             <remote ip="10.0.0.183" name="www"/>      # 10.0.0.182/183 是远程的Ip地址

 27         </localpath>

 28

 29         <localpath watch="/data0/www/bbs">

 30             <remote ip="10.0.0.182"  name="bbs"/>

 31             <remote ip="10.0.0.183" name="bbs"/>

 32         </localpath>

 33

 34         <localpath watch="/data0/www/blog">

 35             <remote ip="10.0.0.182"  name="blog"/>

 36             <remote ip="10.0.0.183" name="blog"/>

 37         </localpath>

 ②修改认证的部分

<rsync>

 40             <commonParams params="-artuz"/>

 41             <auth start="false" users="root" passwordfile="/etc/rsync.pas"/>

 42             <userDefinedPort start="false" port="874"/><!-- port=874 -->

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

 44             <ssh start="false"/>

 45         </rsync>

改成：

39         <rsync>

 40             <commonParams params="-artuz"/>

 41             <auth start="true" users="rsync\_backup" passwordfile="/etc/rsync.password"/>

 42             <userDefinedPort start="false" port="874"/><!-- port=874 -->

 43             <timeout start="true" time="100"/><!-- timeout=100 -->

 44             <ssh start="false"/>

 45         </rsync>

③修fail队列日志

<failLog path="/tmp/rsync\_fail\_log.sh" timeToExecute="60"/><!--default every 60mins execut

    e once-->

改成（这个目录是自己提前创建好的）

    <failLog path="/usr/local/sersync/logs/rsync\_fail\_log.sh" timeToExecute="60"/><!--default

    every 60mins execute once-->

最后保存配置文件

**12.配置sersync的命令**

[[root@M-SERVER](mailto:root@M-SERVER) conf]# echo "export PATH=$PATH:/usr/local/sersync/bin">>/etc/profile

[[root@M-SERVER](mailto:root@M-SERVER) conf]# tail -n 1   /etc/profile

export PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin:/usr/local/sersync/bin

[[root@M-SERVER](mailto:root@M-SERVER) conf]# which rersync

/usr/bin/which: no rersync in (/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin)

[[root@M-SERVER](mailto:root@M-SERVER) conf]# source  /etc/profile

[[root@M-SERVER](mailto:root@M-SERVER) conf]# which  sersync

/usr/local/sersync/bin/sersync

[[root@M-SERVER](mailto:root@M-SERVER) conf]#

**13.启动命令**

[[root@M-SERVER](mailto:root@M-SERVER) ~]# sersync  -r   -d  -o  /usr/local/sersync/conf/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: -r      rsync all the local files to the remote servers before the sersync work

option: -d      run as a daemon

option: -o      config xml name：  /usr/local/sersync/conf/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 rsync\_backup

passwordfile is         /etc/rsync.password

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: 32 = 12(Thread pool nums) + 20(Sub threads)

please according your cpu ，use -n param to adjust the cpu rate

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

rsync the directory recursivly to the remote servers once

working please wait...

execute command: cd /data0/www/www && rsync -artuz -R --delete ./  --timeout=100 [rsync\_backup@10.0.0.182](mailto:rsync_backup@10.0.0.182)::www --password-file=/etc/rsync.password >/dev/null 2>&1

run the sersync:

watch path is: /data0/www/www

[[root@M-SERVER](mailto:root@M-SERVER) ~]#

-r 客户端和服务端可能第一次数据不一样 开启先做一次同步

-d 在后台以守护进程的方式启动

-o  指定xml 文件

**14.测试以后发现，在第11步骤写的模块（localpath ）只识别第一个**

这个时候进行多实例传送，每个推送数据的目录给一个配置文件下面三个文件，每个文件都有自己的单独日志和<localpath watch="/data0/www/www">  只有一个模块，这几个文件可以通过拷贝生成，修改后保存

[[root@M-SERVER](mailto:root@M-SERVER) conf]# ls

bbs\_confxml.xml  blog\_confxml.xml  tmp  www\_confxml.xml

[[root@M-SERVER](mailto:root@M-SERVER) conf]#

以blog\_confxml.xml   为例

 <localpath watch="/data0/www/bbs">

            <remote ip="10.0.0.182"  name="bbs"/>

            <remote ip="10.0.0.183" name="bbs"/>

        </localpath>

 <failLog path="/usr/local/sersync/logs/bbs\_rsync\_fail\_log.sh" timeToExecute="60"/><!--default

every 60mins execute once-->

其余的密码文件如果自己想修该的话在自行修改

**15启动三个配置文件**

先杀掉原来配置

[[root@M-SERVER](mailto:root@M-SERVER) conf]# ps  -ef |grep  rsync

root      2147     1  0 17:30 ?        00:00:00 sersync -r -d -o /usr/local/sersync/conf/confxml.xml

root      2289  2232  0 17:53 pts/1    00:00:00 grep rsync

[[root@M-SERVER](mailto:root@M-SERVER) conf]# pkill  sersync

[[root@M-SERVER](mailto:root@M-SERVER) conf]# ps  -ef |grep  rsync

root      2292  2232  0 17:53 pts/1    00:00:00 grep rsync

[[root@M-SERVER](mailto:root@M-SERVER) conf]#

多实例启动

sersync  -r   -d  -o  /usr/local/sersync/conf/www\_confxml.xml

sersync  -r   -d  -o  /usr/local/sersync/conf/bbs\_confxml.xml

sersync  -r   -d  -o  /usr/local/sersync/conf/blog\_confxml.xml

检查：可以查出总共起了三个进程，每个进行都对应一个文件

[[root@M-SERVER](mailto:root@M-SERVER) conf]# ps -ef |grep sersync

root      2300     1  0 17:57 ?        00:00:00 sersync -r -d -o /usr/local/sersync/conf/www\_confxml.xml

root      2320     1  0 17:57 ?        00:00:00 sersync -r -d -o /usr/local/sersync/conf/bbs\_confxml.xml

root      2340     1  0 17:58 ?        00:00:00 sersync -r -d -o /usr/local/sersync/conf/blog\_confxml.xml

root      2411  2232  0 18:00 pts/1    00:00:00 grep sersync

[[root@M-SERVER](mailto:root@M-SERVER) conf]#

最后主服务器**Sersync服务（M）端可能也会重启**

**所以：防止重启（吧配置文件放在/etc/rc.local中，这个时候可以不加-r文件比较大，没有必要只要第一次即可）**

[[root@M-SERVER](mailto:root@M-SERVER) tmp]# cat >>/etc/rc.local<<EOF

> sersync     -d  -o  /usr/local/sersync/conf/www\_confxml.xml

> sersync    -d  -o  /usr/local/sersync/conf/bbs\_confxml.xml

> sersync    -d  -o  /usr/local/sersync/conf/blog\_confxml.xml

> EOF

[[root@M-SERVER](mailto:root@M-SERVER) tmp]#

测试（**Sersync服务（M）端**）（测试成功）

[[root@M-SERVER](mailto:root@M-SERVER) ~]# cd /data0

[[root@M-SERVER](mailto:root@M-SERVER) data0]# ls

www

[[root@M-SERVER](mailto:root@M-SERVER) data0]# cd www

[[root@M-SERVER](mailto:root@M-SERVER) www]# ls

bbs  blog  www

[[root@M-SERVER](mailto:root@M-SERVER) www]# cd bbs

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# touch succes.log

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# cd ..

[[root@M-SERVER](mailto:root@M-SERVER) www]# cd blog/

[[root@M-SERVER](mailto:root@M-SERVER) blog]# touch success.log

[[root@M-SERVER](mailto:root@M-SERVER) blog]# cd ..

[[root@M-SERVER](mailto:root@M-SERVER) www]# cd www

[[root@M-SERVER](mailto:root@M-SERVER) www]# touch success.log

[[root@M-SERVER](mailto:root@M-SERVER) www]#

S2-SERVER端查看

[[root@S2-SERVER](mailto:root@S2-SERVER) ~]# tree  /data0

/data0

└── www

    ├── bbs

    │   ├── bbs.log

    │   ├── bbs.txt

    │   ├── hello

    │   ├── succes.log

    │   └── world

    ├── blog

    │   ├── blog.log

    │   ├── blog.txt

    │   └── success.log

    └── www

        ├── success.log

        ├── [www.log](http://www.log/)

        └── [www.txt](http://www.txt/)

4 directories, 11 files

[[root@S2-SERVER](mailto:root@S2-SERVER) ~]#

经验

2710服务器千兆网

每秒可以同步10-100K  能同步 40-50 张

大量文件同步也是受限制的

sersync的参数

-r  开启第一次进行一次完全的同步，保持一致

-o 指定confxml.xml 的文件

-n 指定默认线程池的线程总数，如果不指定默认启动线程池的数量是10，CPU使用过高，可以通过这个参数进行调整

-d 参数为后台启动

-m  不同步，只运行插件

./sersync --help

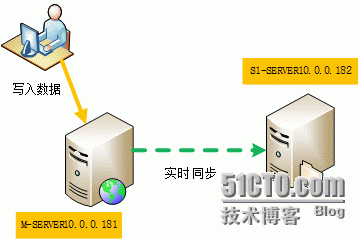
进行压力测试

监控是否同步不同步报警

本次测试的说明：

测试用户写入的数据和实时同步数据之间的差别或者说延迟，本次实验实在虚拟机上完成，仅供自己学习测试专用

1.测试图解

[](http://s5.51cto.com/wyfs02/M00/7F/60/wKioL1ccfQXhyFCDAAA1HlDuTxE745.png)

2.测试环境：

[](http://s3.51cto.com/wyfs02/M01/7F/62/wKiom1ccfHyRsiKqAAAlQQpuMrY164.png)

3.基本说明

M-SERVER10.0.0.181端的三个文件夹

[[root@M-SERVER](mailto:root@M-SERVER) ~]# tree  /data0

/data0

└── www

    ├── bbs

    ├── blog

    └── www

4 directories, 0 files

[[root@M-SERVER](mailto:root@M-SERVER) ~]#

S1-SERVER10.0.0.182端的三个文件夹

[[root@S1-SERVER](mailto:root@S1-SERVER) ~]# tree  /data0

/data0

└── www

    ├── bbs

    ├── blog

    └── www

4 directories, 0 files

[[root@S1-SERVER](mailto:root@S1-SERVER) ~]#

备注;在M-SERVER创建数据，看S1-SERVER端数据的同步的情况

4.M-SERVER 端测试脚本

备注：脚本内容：每秒往  /data0/www/bbs/    文件夹下写count个数据，count可以自己调节

[[root@M-SERVER](mailto:root@M-SERVER) scripts]# pwd

/server/scripts

[[root@M-SERVER](mailto:root@M-SERVER) scripts]# ll -h

总用量 36K

-rw-r--r--. 1 root root 10K 4月  24 14:58 10K.jpg

-rw-r--r--. 1 root root 20K 4月  24 14:58 20K.jpg

-rw-r--r--. 1 root root 297 4月  24 14:57 tp.sh

[[root@M-SERVER](mailto:root@M-SERVER) scripts]#

#!/bin/bash

count=10

while true

do

   for((i=0;i<=$count;i++))

    do

   /bin/cp  10K.jpg   /data0/www/bbs/10K\_`echo $(date)$RANDOM|md5sum|cut -c 1-8`.jpg

    done

sleep 1

for((i=0;i<=$count;i++))

    do

   /bin/cp  20K.jpg   /data0/www/bbs/20K\_`echo $(date)$RANDOM|md5sum|cut -c 1-8`.jpg

    done

done

~

测试 在M-SERVER上执行脚本    用tree  |wc  -l（用crt将命令发送到交谈窗口）  查看 M-SERVER（可以克隆一个回话）和 S1-SERVER  文件的个数，每大约秒发送一次tree  |wc -l

测试开始：

第一次测试

M-SERVER上的数据：

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree |wc -l

140

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree |wc -l

146

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree |wc -l

168

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree |wc -l

181

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree |wc -l

190

 S1-SERVER上的数据：

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree |wc -l

135

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree |wc -l

146

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree |wc -l

168

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree |wc -l

177

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree |wc -l

190

第二次测试（调整count 的个数，每秒写20个数据，在脚本中调节）

M-SERVER上的数据

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree |wc -l

638

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree |wc -l

656

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree |wc -l

676

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree |wc -l

698

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree |wc -l

717

 S1-SERVER上的数据：

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree |wc -l

633

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree |wc -l

656

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree |wc -l

667

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree |wc -l

698

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree |wc -l

712

第三次测试（调整count 的个数，每秒写30个数据，在脚本中调节）

M-SERVER上的数据

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

1004

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

1032

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

1061

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

1061

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

1084

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

1115

[[root@M-SERVER](mailto:root@M-SERVER) bbs]#

 S1-SERVER上的数据：

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

999

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

1028

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

1058

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

1061

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

1079

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

1110

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]#

第四次测试（调整count 的个数，每秒写50个数据，在脚本中调节）

M-SERVER上的数据：

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

1542

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

1542

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

1577

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

1611

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

1642

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

1644

[[root@M-SERVER](mailto:root@M-SERVER) bbs]#

 S1-SERVER上的数据：

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

1542

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

1542

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

1572

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

1605

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

1637

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

1644

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]#

第五次测试（调整count 的个数，每秒写80个数据，在脚本中调节）

M-SERVER上的数据：

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

1948

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

1952

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

1978

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

2018

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

2049

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

2078

 S1-SERVER上的数据：

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

1943

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

1952

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

1972

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

2003

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

2038

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

2072

第六次测试（调整count 的个数，每秒写150个数据，在脚本中调节）

M-SERVER上的数据：

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

2527

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

2571

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

2624

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

2639

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

2667

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

2707

[[root@M-SERVER](mailto:root@M-SERVER) bbs]#

 S1-SERVER上的数据：

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

2521

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

2566

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

2617

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

2639

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

2649

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

2690

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]#

第七次测试（调整count 的个数，每秒写550个数据，在脚本中调节）

M-SERVER上的数据：

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

3642

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

3691

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

3739

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

3781

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

3822

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

3863

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

3900

[[root@M-SERVER](mailto:root@M-SERVER) bbs]# tree  |wc -l

3954

[[root@M-SERVER](mailto:root@M-SERVER) bbs]#

  S1-SERVER上的数据：

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

3637

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

3685

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

3731

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

3774

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

3817

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

3858

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

3896

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]# tree  |wc -l

3925

[[root@S1-SERVER](mailto:root@S1-SERVER) bbs]#

分析：

第一次测试每秒10个文件几乎无延迟    10K-20K 的文件

第二次测试每秒20个文件也是几乎无延迟  10K-20K的文件

第三次测试每秒是30个文件的时候出现了延迟  10K-20K的文件     部分延迟大约是每秒延迟5个文件

第四次测试每秒是50个文件的时候出现了延迟  10K-20K的文件     部分延迟大约是每秒延迟5个文件

第五次测试每秒是80个文件的时候出现了延迟  10K-20K的文件     几乎每秒延迟5个文件

第六次测试每秒是150个文件的时候部分出现了延迟  10K-20K的文件     部分几乎每秒延迟15个文件

第七次测试每秒是550个文件的时候部分出现了延迟  10K-20K的文件     部分几乎每秒延迟30-50个文件

...... 不在测试了