目前业内比较靠谱的同步解决方案有：

rsync+inotify-tools，Openduckbill+inotify-tools 和 rsync+sersync

前两者由于是基于脚本语言编写，所以规范程度，执行效率相对 rsync+sersync 就稍微弱一些。

sersync 是使用 c++编写，基于 boost1.43.0,inotify api,rsync command 开发，主要用于服务器同步，web 镜像等功能。其对 linux 系统文件系统产生的临时文件和重复的文件操作能够进行过滤，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,更适合线上使用。

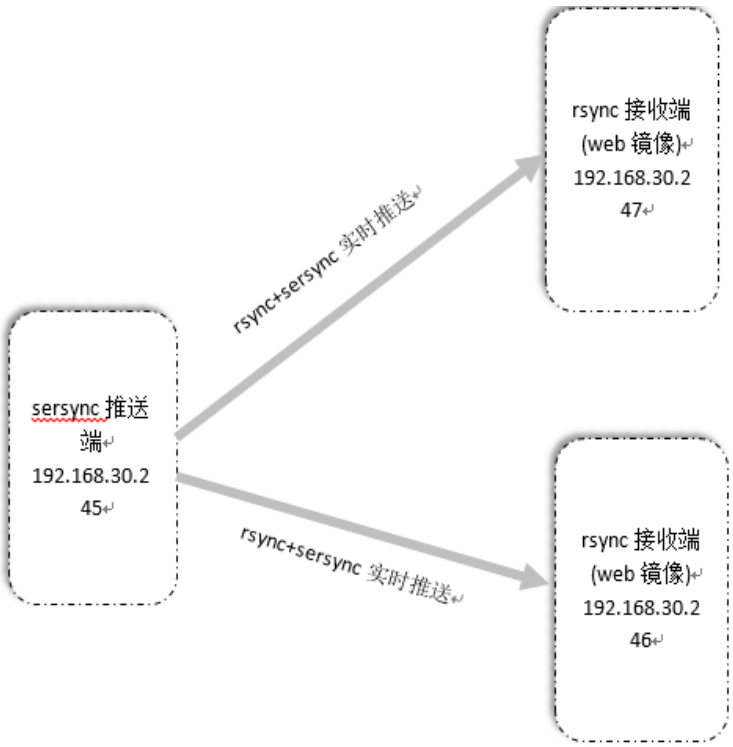
本文为实现将 sersync 推送端/data 下的数据实时同步到 rsync 接收端/data 目录下，实现 rsync 服务器为 sersync 的镜像服务器

注：使用 rsync+crontab 做定时同步时，主服务器端开启 rsync 守护进程，而镜像服务器是运行 rsync 客户端，平时一般会利用 crontab 定时获取 rsync 服务器上的数据。

但使用 rsync+sersync 做实时同步时，用于推送文件的服务器运行 sersync 服务，用于接收文件的服务器则运行 rsync 守护进程，简单来说就是 sersync 会利用 rsync 命令将文件推送到 rsync 服务器，实际线上使用一般会把 sersync 作为主服务器，rsync 作为镜像服务器，实现数据同步备份，web 镜像等功能

解决方案：

操作系统:　centos6.5 x86_64
rsync:　centos 自带 yum 源
sersync:　sersync2.5_64bit_binary_stable_final



sersync 推送端 192.168.30.245
rsync 接收端 192.168.30.247；192.168.30.246

环境搭建：(接收端,推送端)

一.首先关闭 selinux 与 iptables
vi /etc/sysconfig/selinux

SELINUX=disabled
---
service iptables stop

在使用 sersync 之前，我们必须安装配置好 rsync 服务器

rsync (接收端)

一.安装 rsync
yum install rsync -y
yum install xinetd -y

二.启动 rsync 依赖服务
/etc/init.d/xinetd start
chkconfig xinetd on

三.配置：

```
# vi /etc/rsyncd.conf
--------------------
uid = qiandao
gid = qiandao
use chroot = no
max connections = 10
strict modes = yes
port = 873
address = 192.168.30.245          #localhost 地址

[cms]                             # rsync 模块名，后面配置 sersync 会用到
path = /var/www/html/aa                    #  该同步目录只要 uid 所指定的用户有写权限即可
comment = mirror for test
ignore errors
read only = no
list = no
auth users = qiandao
secrets file = /etc/rsync.pas          #  密码认证文件，必须为 600 权限，否则 rsync 传输会报错
hosts allow = 192.168.30.0/24
# hosts deny = 0.0.0.0/0
pid file = /var/run/rsyncd.pid
lock file = /var/run/rsync.lock
log file = /var/log/rsyncd.log
--------------------
```

四.创建或指定同步目录

```
# mkdir /var/www/html/aa
```

五.配置密码认证文件或使用 SSH 密钥

```
# echo "qiandao:123456" > /etc/rsync.pas
# chmod 600 /etc/rsync.pas            ##权限必须设置为 600
```

六.启动 xinetd 和 rsync 守护进程

```
# /etc/init.d/xinetd restart
# rsync --daemon --config=/etc/rsyncd.conf
```

七.设置开机启动：

```
# echo "rsync --daemon --config=/etc/rsyncd.conf" >> /etc/rc.local
```

sersync (推送端)

一.下载 sersync 源码包

wget http://sersync.googlecode.com/files/sersync2.5_64bit_binary_stable_final.tar.gz

注:若在 32 位平台安装则可下载 32 位 sersync 源码包,本例用 64 位

wget http://sersync.googlecode.com/files/sersync2.5_32bit_binary_stable_final.tar.gz

二.创建 sersync 目录结构

```
# mkdir /usr/local/sersync
# mkdir /usr/local/sersync/conf
# mkdir /usr/local/sersync/bin
# mkdir /usr/local/sersync/log
# tar zxf sersync2.5_64bit_binary_stable_final.tar.gz –C /usr/src
# cd /usr/local/GNU-Linux-x86/
# mv confxml.xml /usr/local/sersync/conf
#mv sersync2.5 /usr/local/sersync/bin/sersync2.5
```

三.配置 sersync

1.首先创建连接 rsyncd 的密码文件

```
# echo "123456" >/etc/rsync.pas
# chmod 600 /etc/rsync.pas                    #指定密码文件的权限为 600
```

2.配置 confxml.xml

```
# cd /usr/local/sersync/conf
# vi confxml.xml
```

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

<head version="2.5">

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

    <!--hostip 与 port 是针对插件的保留字段，对于同步功能没有任何作用，保留默认即可。

-->

    <debug start="false"/>

    <!--该行为 Debug 开启开关。true 为开启 debug 模式 ,会在 sersync 正在运行的控制台，

打印 inotify 事件与 rsync 同步命令，生产环境一般不开启.-->

    <fileSystem xfs="false"/>

    <!--对于 XFS 文件系统的用户，需要将这个选项开启，才能使 sersync 正常工作
```

对于 sersync 监控的文件，会默认过滤系统的临时文件（以"."开头，以"~"结尾），除了这些文件外，

在 15-20 行中，我们还可以自定义其它需要过滤的文件。

通过将 start 设置为 true 后可开启过滤功能，在 exclude 标签中可使用正则表达式。

默认给出的两个例子分别是过滤以".gz"结尾的文件与过滤监控目录下的 info 路径（监控路径/info/*），可以根据需求自己添加。

但在开启的时候，自己一定要测试下，如果正则表达式出现错误，控制台会有相应提示。

相比较使用 Rsync 的 exclude 功能，被过滤的路径，不会加入监控，大大减少 Rsync 同步的通讯量-->

```
<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="false"/>

<modify start="false"/>

</inotify>

<!--用来定义 inotify 监控参数，我们可以根据项目的特点来优化 Sersync。

对于大多数应用，可以尝试把 createFile（监控文件事件选项）设置为 false 来提高性能，进一步减少 Rsync 通讯。因为拷贝文件到监控目录会产生 create 事件与 close_write 事件，所以如果关闭 create 事件，只监控文件拷贝结束时的事件 close_write，同样可以实现文件完整同步。

注意：要使得 createFolder 保持为 true，如果将 createFolder 设为 false，则不会对产生的目录进行监控，该目录下的子文件与子目录也不会被监控，所以除非特殊需要，请开启。默认情况下对创建文件（目录）事件与删除文件（目录）事件都进行监控，如果项目中不需要删除远程目标服务器的文件（目录），则可以将 delete 参数设置为 false，则不对删除事件进行监控。
-->

<sersync>

<localpath watch="/home/wwwroot/www.dwhd.org">

<remote ip="172.16.6.101" name="home_wwwroot_www_dwhd_org"/>

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

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

</localpath>

<!--/home/wwwroot/www.dwhd.org 目录为 sersync 主服务器本地待同步的目录，ip="172.16.6.100"为从服务器的 ip 地址，如果有多个服务器，依次列出来即可。

name="home_wwwroot_www_dwhd_org"，这里的 home_wwwroot_www_dwhd_org 为 rsyncd.conf 中的模块名字，即中括号中的名称。-->

<rsync>

<commonParams params="-artuz"/>

<auth start="true" users="lookback" passwordfile="/etc/passwd.txt"/>

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

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

<ssh start="false"/>

</rsync>

<!--在 commonParams 项，我们可以自定义 rsync 的同步参数，默认是-artuz，auth start="false"设置为 true 的时候，使用 rsync 的认证模式传送，需要配置 user 与 passwordfile（-password-file=/etc/passwd.txt）来使用。userDefinedPort 当远程同步目标服务器的 rsync 端口不是默认端口的时候使用（-port=874）。timeout 设置 rsync 的 timeout 事件（-timeout=100）。<ssh start="false"/>如果开启表示 ssh 使用 rsync -e ssh 的方式进行传输。-->

<failLog path="/tmp/rsync_fail_log.sh" timeToExecute="5"/><!--default every 60mins execute once-->

<!--用来定义失败日志脚本配置,如果文件同步传输失败，会重新传送，再次失败就会写入 rsync_fail_log.sh，然后每隔一段时间（timeToExecute 进行设置）执行该脚本再次重新传送，然后清空该脚本。可以通过 path 来设置日志路径-->

<crontab start="true" schedule="600"><!--600mins-->

<crontabfilter start="false">

<exclude expression="*.php"></exclude>

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

</crontabfilter>

</crontab>

<!--用来定义 Crontab 定期整体同步功能,Crontab 可以对监控路径与远程目标主机每隔一段时间进行一次整体同步，可能由于一些原因两次失败重传都失败了，这个时候如果开启了 crontab 功能，还可以进行一次保证各个服务器文件一致，如果文件量比较大，crontab 的时间间隔要设的大一些，否则可能增加通讯开销，schedule 这个参数是设置 crontab 的时间间隔，默认是 600 分钟。

如果开启了 filter 文件过滤功能，那么 crontab 整体同步也需要设置过滤，否则虽然实时同步的时候文件被过滤了，但 crontab 整体同步的时候，如果不单独设置 crontabfilter，还会将需过滤的文件同步到远程从服务器，crontab 的过滤正则与 filter 过滤的不同，也给出了两个实例分别对应与过滤文件与目录，总之如果同时开启了 filter 与 crontab，则要开启 crontab 的 crontabfilter，并按示例设置使其与 filter 的过滤一一对应。-->

<plugin start="false" name="command"/>

</sersync>

<!--此处到行尾，都是插件的相关信息。当 plugin 标签设置为 true 时候，在同步文件或路径到远程服务器之后，会调用插件。通过 name 参数指定需要执行的插件。目前支持的有 command、refreshCDN、socket、http 四种插件。其中，http 插件目前由于兼容性原因已经去除，以后会重新加入。-->

<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>
<!--socket 插件，开启该模块，则向指定 ip 与端口发送 inotify 所产生的文件路径信息-->

<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>

<!--refreshCDN 插件的相关配置,refreshCDN 用来在同步过程中将文件发送到目地服务器后，刷新 cdn 接口。如果不想使用，则将 start 属性设为 false 即可。该模块根据 chinaCDN 的协议 进行设计 ,当有文件产生的时候 ,就向 cdn 解耦发送需要刷新的路径为止。其中 localpath watch="/data0/htdocs/cms.xoyo.com/site/"是需要监控的目录。cdinfo 标签指定了 cdn 接口的域名，端口号，以及用户名与密码。sendurl 标签是需要刷新的 url 的前缀。regexurl 标签中，regex 属性为 true 时候，使用 match 属性的正则语句匹配 inotify 返回的路径信息，并将正则匹配到的部分作为 url 一部分,

上面配置文件自带的意思为，如果产生文件事件为：/data0/htdoc/cms.xoyo.com/site/jx3.xoyo.com/image/a/123.txt

经过上面的 match 正则匹配后 ,最后刷新的路径是 :http ://pic.xoyo.com/cms/a/123.txt

如果 regex 属性为 false，最后刷新的路径就是：http://pic.xoyo.com/cms/jx3.xoyo.com/images/a/123.txt-->

</head>

配置文件示例：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<head version="2.5">
    <host hostip="192.168.30.247" port="8008"></host>
    <debug start="true"/>
    <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="true"/>
        <closeWrite start="true"/>
        <moveFrom start="true"/>
        <moveTo start="true"/>
        <attrib start="false"/>
        <modify start="false"/>
    </inotify>

    <sersync>
        <localpath watch="/var/www/html/cms">
            <remote ip="192.168.30.245" name="cms"/>
            <!--<remote ip="192.168.8.39" name="tongbu"/>-->
            <!--<remote ip="192.168.8.40" name="tongbu"/>-->
        </localpath>
        <rsync>
            <commonParams params="-artuz"/>
            <auth start="true" users="qiandao" passwordfile="/etc/rsync.pas"/>
            <userDefinedPort start="false" port="874"/><!-- port=874 -->
            <timeout start="true" time="100"/><!-- timeout=100 -->
            <ssh start="false"/>
        </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>

    <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>
```

3.创建推送端 sersync 同步目录

```
# mkdir /data
```

4.设置环境变量：

```
# echo "export PATH=$PATH:/usr/local/sersync/bin/" >> /etc/profile
# source /etc/profile
```

5.启动 sersync

```
# sersync2 -r -d -o /usr/local/sersync/conf/confxml.xml
```

注：重启操作如下：

```
# killall sersync2 && sersync2 -r -d -o /usr/local/sersync/conf/confxml.xml
```

6.设置开机启动

```
# echo "sersync2 -r -d -o /usr/local/sersync/conf/confxml.xml" >> /etc/rc.local
```

验证：

(推送端)

```
# cd /var/www/html/cms
# touch 1 2 3 4 5
# echo "test sersync" > 1
```

(接收端)

```
# cd /var/www/html/aa
# ls
1  2  3  4  5
# cat 1
test sersync
```

注：这里提一个细节，如果接收端服务器本地创建或修改/var/www/html/aa 同步目录下的文件，当服务端进行目录同步时则不会对接收端服务器本地创建或修改的文件产生影响。