第1章 为什么需要实时同步

1.实时同步的应用场景

```
1.解决NFS单点故障问题
2.备份NFS数据并且提供冗余的服务功能
```

2.实时同步实施的难点

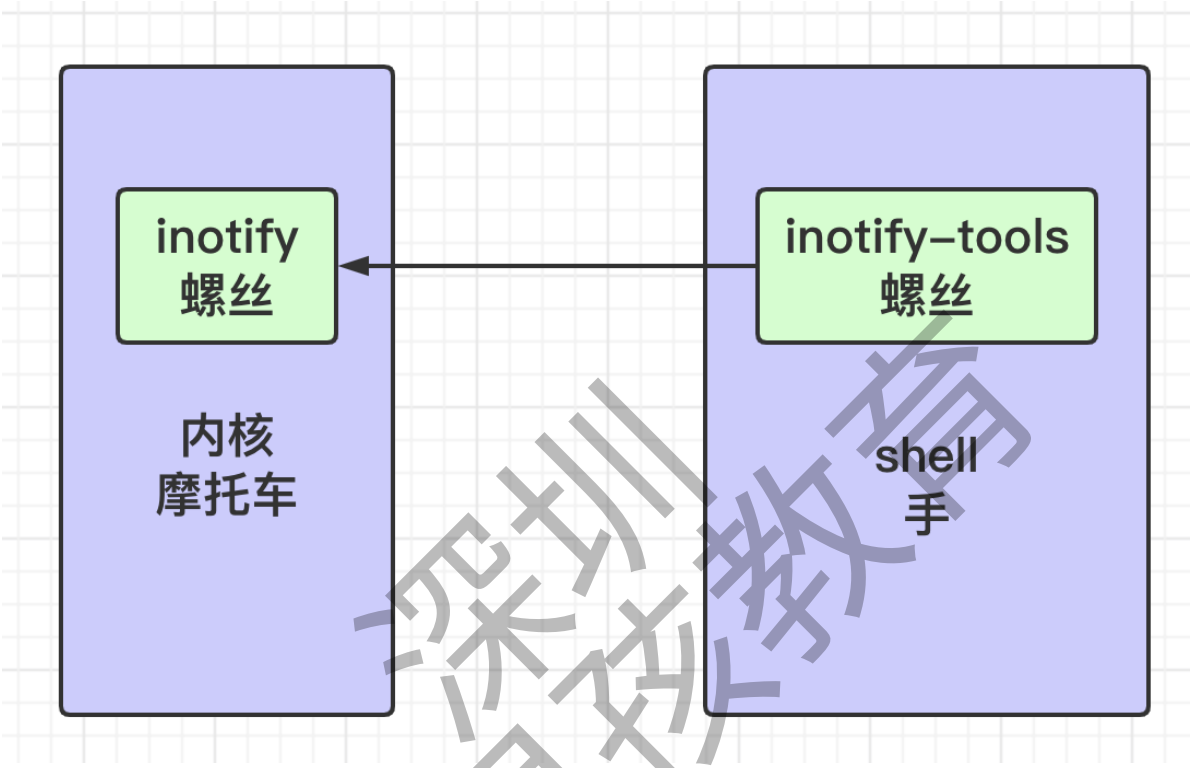
```
1.什么条件才同步？
2.同步哪些文件？
3.多久同步一次？
4.用什么工具同步？
```

第2章 inotify实现同步-了解即可

1.inotify简介

```
怎么读？爱No踢fai
Inotify是一种强大的,细粒度的,异步的文件系统事件监视机制
Linux2.6.13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正是实施这样监控的软件
```

示意图:



2.inotify实战

2.1 查看当前系统是否支持inotify

```
#1.查看命令
[root@nfs-31 ~]# uname -r
3.10.0-957.el7.x86_64
[root@nfs-31 ~]# ls -l /proc/sys/fs/inotify/
总用量 0
-rw-r--r-- 1 root root 0 4月  22 15:11 max_queued_events
-rw-r--r-- 1 root root 0 4月  22 15:11 max_user_instances
-rw-r--r-- 1 root root 0 4月  22 15:11 max_user_watches

#2.状态解释
max_queued_events        设置inotify实例事件(event)队列可容纳的事件数量
max_user_instances       设置每个用户可以运行的inotify或者inotifywatch命令的进程数
max_user_watches         设置inotifywait或者inotifywatch命令可以监视的文件数量 (单进程)
```

2.2 安装iontify-tools

```
[root@nfs-31 ~]# yum install inotify-tools -y
```

2.3 inotifywait命令参数解释

```
[root@nfs-31 ~]# inotifywait --help
inotifywait参数说明
参数名称              参数说明
-m ,-monitor          始终保持事件监听状态
-r,-recursive         递归查询目录
-q,-quiet             只打印监控事件的信息
-exclude              排除文件或目录时,不区分大小写
-t,-timeout           超时时间
--timefmt             指定时间输出格式
--format              指定输出格式
-e,event              后面指定增,删,改等事件

inotifywait events 事件说明
access                读取文件或目录内容
modify                修改文件或目录内容
attrib                文件或目录的属性改变
close                 文件被关闭
open                  文件或目录被打开
move                  移动文件或目录移动到监视目录
create                在监视目录下创建文件或目录
delete                删除监视目录下的文件或目录
umount                卸载文件系统
```

2.4 inotify监控命令测试

格式化输出参数解释:

```
%T   #调用并显示定义好的时间格式
%w   #显示发生变化的文件的绝对路径
%f   #显示监控到的文件名称,去掉后只显示目录路径
-e delete,create   #指定监控文件变化的类型
```

1.开启两个窗口:测试create和delete

```
inotifywait -mrq --timefmt '%d/%m/%y %H:%M' --format '%T %w%f' -e
delete,create /backup
```

2.测试close_write

```
inotifywait -mrq --timefmt '%d/%m/%y %H:%M' --format '%w%f' -e close_write
/backup
```

3.打印出事件类型

```
inotifywait -mrq --format '%w%f %e' -e delete,create /backup
```

2.5 inotify结合rsync实现实时同步-了解即可

```
#!/bin/bash

Path=/data
backup_Server=172.16.1.41
export RSYNC_PASSWORD=123456

/usr/bin/inotifywait -mrq --format '%w%f' -e create,close_write,delete /data
| while read line
do
    echo ${line}
    if [ -f ${line} ]
    then
       rsync -az ${line} rsync_backup@${backup_Server}::data
    else
       rsync -az --delete /data/ rsync_backup@${backup_Server}::data
    fi
done
```

2.6 inotify优缺点

```
inotify优点:
1.系统内核本身支持
2.监控文件系统事件变化

inotify缺点:
1.并发如果大于200个文(10-100k),同步就会由延迟
2.我们前面的脚本,每次都是全部推送一次,但是确实是增量的,也可以只同步变化的文件
3.监控到事件后,调用rsync同步是单进程(加&并发)
4.我们自己写的脚本健壮性不高
```

第3章 sersync实现同步-了解即可

1.sersync简介

```
1.sersync主要用于服务器同步,web镜像等功能.
2.基于boost1.41.0,inotify api,rsync command开发.
```

2.sersync实战

2.1 主机规划

```
nfs      172.16.1.31  rsync+inotify+sersync
backup   172.16.1.41  rsync-server
```

2.2 安装sersync

```
cd /opt/
wget http://down.whsir.com/downloads/sersync2.5.4_64bit_binary_stable_final.tar.gz
tar zxf sersync2.5.4_64bit_binary_stable_final.tar.gz
mv GNU-Linux-x86 sersync
cd sersync
cp confxml.xml confxml.xml.bak
```

2.3 配置文件修改

```
 23     <sersync>
 24         <localpath watch="/data">
#监听变化的目录
 25             <remote ip="172.16.1.41" name="data"/>    #远程rsync服务器的IP地址和模块名称
 26             <!--<remote ip="192.168.8.39" name="tongbu"/>-->
 27             <!--<remote ip="192.168.8.40" name="tongbu"/>-->
 28         </localpath>
 29         <rsync>
 30             <commonParams params="-az"/>                          #rsync传输的参数
 31             <auth start="true" users="rsync_backup"   #rsync虚拟用户名称 passwordfile="/etc/rsync.passwd"/>
#rsync密码文件路径
```

2.4 查看启动帮助说明

```
[root@nfs-31 /opt/sersync]# ./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
_____________________________________________________
```

2.6 启动服务

```
./sersync2 -rdo confxml.xml
```

2.7 测试数据是否同步

backup服务器操作：

```
cd /data
while true ;do ls |wc -l;sleep 0.1;done
```

nfs服务器操作：

```
cd /data
for i in {1..1000};do echo "${i}"; echo "${i}" > ${i}.txt;sleep 0.1;done
```

3.sersync优缺点

```
优点：
1.开源免费，国人开发

缺点：
1.同步文件性能不太好
2.安装步骤复杂，没有启动文件
```

第4章 lsyncd实现同步-推荐黑科技

1.lsyncd介绍

官方地址：

```
https://github.com/axkibe/lsyncd](https://github.com/axkibe/lsyncd
```

命令介绍：

```
Lysncd 实际上是lua语言封装了 inotify 和 rsync 工具，采用了 Linux 内核（2.6.13 及以后）里的 inotify 触发机制，然后通过rsync去差异同步，达到实时的效果。
它最令人称道的特性是，完美解决了 `inotify + rsync`海量文件同步带来的文件频繁发送文件列表的问题 —— 通过时间延迟或累计触发事件次数实现。
另外，它的配置方式很简单，lua本身就是一种配置语言，可读性非常强。lsyncd也有多种工作模式可以选择，本地目录cp，本地目录rsync，远程目录rsyncssh。
实现简单高效的本地目录同步备份（网络存储挂载也当作本地目录），一个命令搞定。
```

2.lsyncd实战

2.1 lsyncd安装

```
yum install lsyncd -y
```

2.2 创建配置文件

只监控1个目录

```
[root@nfs ~]# cat /etc/lsyncd.conf
settings {
  logfile = "/var/log/lsyncd/lsyncd.log",
  statusFile = "/var/log/lsyncd/lsyncd.status",
  inotifyMode = "CloseWrite",
  maxProcesses = 8,
}
sync {
  default.rsync,
  source = "/data",
  target = "rsync_backup@172.16.1.41::data",
  delete = true,
  exclude = { ".*" },
  delay = 1,
  rsync = {
    binary = "/usr/bin/rsync",
    archive = true,
    compress = true,
    verbose = true,
    password_file = "/etc/rsync.passwd",
    _extra = {"--bwlimit=200"}
  }
}
```

监控2个目录

```
[root@nfs ~]# cat /etc/lsyncd.conf
settings {
  logfile = "/var/log/lsyncd/lsyncd.log",
  statusFile = "/var/log/lsyncd/lsyncd.status",
  inotifyMode = "CloseWrite",
  maxProcesses = 8,
}
sync {
  default.rsync,
  source = "/data",
  target = "rsync_backup@172.16.1.41::data",
  delete = true,
  exclude = { ".*" },
  delay = 1,
  rsync = {
    binary = "/usr/bin/rsync",
    archive = true,
    compress = true,
    verbose = true,
    password_file = "/etc/rsync.passwd",
    _extra = {"--bwlimit=200"}
  }
}

sync {
  default.rsync,
```

```
  source = "/backup",
  target = "rsync_backup@172.16.1.41::backup",
  delete = true,
  exclude = { ".*" },
  delay = 1,
  rsync = {
    binary = "/usr/bin/rsync",
    archive = true,
    compress = true,
    verbose = true,
    password_file = "/etc/rsync.passwd",
    _extra = {"--bwlimit=200"}
  }
}
```

2.3 配置文件解释

```

settings {          //全局设置
 logfile = "/var/log/lsyncd/lsyncd.log",            //定义日志文件
 statusFile = "/var/log/lsyncd/lsyncd.status",      //定义状态文件
 inotifyMode = "CloseWrite",                        //指定inotify监控的事件,还可以是Modify或CloseWrite or Modify。
 maxProcesses = 8,                                  //最大进程数
 maxDelays = 10,                                    //累计到多少所监控的事件激活一次同步，即使后面的delay延迟时间还未到。
 nodaemon = true,                                   //默认不启用守护模式
 statusInterval = 10,                               //将lsyncd的状态写入上面的statusFile的间隔，默认10秒。
}

sync {      //里面是定义同步参数，一般第一个参数指定lsyncd以什么模式运行，有rsync、rsyncssh、direct三种模式。
 default.rsync,          //目录间同步，使用rsync命令。也可以达到使用ssh形式的远程rsync效果，或daemon方式连接远程rsyncd进程。
 default.rsyncssh,       //同步到远程主机目录，rsync的ssh模式，需要使用key来认证。
 default.direct,         //本地目录间同步，使用cp、rm等命令完成差异文件备份。

 source = "/data",                              //同步的源目录，即监控的目录。

 target = "rsync_backup@172.16.1.41::data",     //同步的目标模块，用于rsync模式。
 target = "rsync_backup@172.16.1.41:/data",     //同步的目标目录，可用于rsync和rsyncssh模式。
 target = "/tmp",                               //同步的本地目标目录，可用于direct和rsync模式。

 delete = true,          //让目标目录和源目录数据保持一致。
 init = true,            //当值为false，只同步进程启动以后发生改动事件的文件，原有的目录即使有差异也不会同步。
 delay = 15,             //延时15秒同步，可避免过于频繁的同步。

 excludeFrom = "",       //排除选项，后面指定排除的列表文件。
 exclude = { ".*" },     //排除匹配到的项，这里是排除的是隐藏文件。

 rsync = {
  binary = "/usr/bin/rsync",      //rsync命令的绝对路径。
```

```
  archive = true,                    //递归,即同步子目录的内容。
  compress = true,                   //传输过程中压缩文件数据,相对其他压缩工具而言，它可以获得更好的压缩率，但是需要消耗CPU资源。
  verbose = true,                    //增加在传输过程中获得的信息量，提供有关正在传输文件的信息。
  password_file = "/etc/rsync.password",          //密码文件路径
  _extra = {"--bwlimit=1000"}                     //传输限速，单位kb。
 }
}
```

2.4 启动命令

```
systemctl start lsyncd
```

2.5 检查测试

backup测试

```
cd /data
while true ;do ls |wc -l;sleep 0.1;done
```

NFS测试

```
cd /data
for i in {1..1000};do echo "${i}"; echo "${i}" > ${i}.txt;sleep 0.1;done
```

3.lsyncd优缺点

```
优点：
1.安装简单，自带启动文件
2.同步速度快，效率高

缺点：无
```