# 真正的inotify+rsync实时同步 彻底告别同步慢

**笔记本:** 剪藏
**创建时间:** 2019/5/14 13:37 **更新时间:** 2019/5/14 13:44

我们公司在用inotify+rsync做实时同步，来解决分布式集群文件一致性的问题。但当web文件越来越多(百万级数量html,jpg等小 文件)，同步就越来越慢，根本做不到实时，按照网上的调优方法都尝试过，问题根本没有解决。经过我一翻细致研究，终于把慢的核心问题研究明白，先总结一句 inotifywait响应不会有延迟，rsync也很快。大家同样有慢的烦恼，那是因为网上的inotify+rsync的教程都是坑。下面我们来分 析。

## inotifywait 单独分析

```
/usr/local/bin/inotifywait -mrq --format '%Xe %w%f' -e modify,create,delete,a
```

执行上面命令，是让inotifywait监听/data/目录，当监听到有发生modify,create,delete,attrib等事件发生时，按%Xe %w%f的格式输出。

在/data/目录touch几个文件

```
touch /data/{1..5}
```

观看inotify输出

```
ATTRIB /data/1          -- 表示发生了ATTRIB事件 路径为/data/1
ATTRIB /data/2
ATTRIB /data/3
ATTRIB /data/4
ATTRIB /data/5
```

知道上面的输出效果之后 我们应该想得到，可以用rsync获取inotifywait监控到的文件列表来做指定的文件同步，而不是每次都由rsync做全目录扫描来判断文件是否存在差异。

## 网上的inotify+rsync分析

我们来看网上的教程，我加了注释。(网上所有的教程基本都一模一样，尽管写法不一样，致命点都是一样的)

```
#!/bin/bash
/usr/bin/inotifywait -mrq --format '%w%f'-e create,close_write,delete /backup
#把发生更改的文件列表都接收到file 然后循环，但有什么鬼用呢？下面的命令都没有引用这个
do
    cd /backup && rsync -az --delete /backup/ rsync_backup@192.168.24.101::bacl
done
```

#注意看 这里的rsync 每次都是全量的同步(这就坑爹了)，而且 file列表是循环形式触发rsync ，等于有10个文件发生更改，就触发10次rsync全量同步(简直就是噩梦)，那还不如直接写个死循环的rsync全量同步得了。

#有很多人会说 日志输出那里明明只有差异文件的同步记录。其实这是rsync的功能，他本来就只会输出有差异需要同步的文件信息。不信你直接拿这句rsync来跑试试。

#这种在需要同步的源目录文件量很大的情况下，简直是不堪重负。不仅耗CPU还耗时，根本不可以做到实时同步。

## 改良方法

要做到实时，就必须要减少rsync对目录的递归扫描判断，尽可能的做到只同步inotify监控到已发生更改的文件。结合rsync的特性，所以这里要分开判断来实现一个目录的增删改查对应的操作。

脚本如下

```
#!/bin/bash
src=/data/                                  # 需要同步的源路径
des=data                                    # 目标服务器上 rsync --daemon 发布的名称
rsync_passwd_file=/etc/rsyncd.passwd                # rsync验证的密码文件
ip1=192.168.0.18                    # 目标服务器1
ip2=192.168.0.19                    # 目标服务器2
user=root                                   # rsync --daemon定义的验证用户名
cd ${src}
# 此方法中，由于rsync同步的特性，这里必须要先cd到源目录，inotify再监听 ./ 才能rs
/usr/local/bin/inotifywait -mrq --format  '%Xe %w%f' -e modify,create,delete
```

```
# 把监控到有发生更改的"文件路径列表"循环
do
        INO_EVENT=$(echo $file | awk '{print $1}')      # 把inotify输出切割 把
        INO_FILE=$(echo $file | awk '{print $2}')       # 把inotify输出切割 把
        echo "-------------------------------$(date)-----------------------
        echo $file
        #增加、修改、写入完成、移动进事件
        #增、改放在同一个判断，因为他们都肯定是针对文件的操作，即使是新建目录，要
        if [[ $INO_EVENT =~ 'CREATE' ]] || [[ $INO_EVENT =~ 'MODIFY' ]] || [[
# 判断事件类型
        then
                echo 'CREATE or MODIFY or CLOSE_WRITE or MOVED_TO'
                rsync -avzcR --password-file=${rsync_passwd_file} $(dirname
# INO_FILE变量代表路径哦  -c校验文件内容
                rsync -avzcR --password-file=${rsync_passwd_file} $(dirname
#仔细看 上面的rsync同步命令 源是用了$(dirname ${INO_FILE})变量 即每次只针对性的
#环境下会漏文件 现在可以在不漏文件下也有不错的速度 做到平衡)
#然后用-R参数把源的目录结构递归到目标后面 保证目录结构一致性
        fi
        #删除、移动出事件
        if [[ $INO_EVENT =~ 'DELETE' ]] || [[ $INO_EVENT =~ 'MOVED_FROM' ]]
        then
                echo 'DELETE or MOVED_FROM'
                rsync -avzR --delete --password-file=${rsync_passwd_file} $
                rsync -avzR --delete --password-file=${rsync_passwd_file} $
#看rsync命令 如果直接同步已删除的路径${INO_FILE}会报no such or directory错误 所
#并加上--delete来删除目标上有而源中没有的文件，这里不能做到指定文件删除，如果删除
#这里有更好方法的同学，欢迎交流。
        fi
        #修改属性事件 指 touch chgrp chmod chown等操作
        if [[ $INO_EVENT =~ 'ATTRIB' ]]
        then
                echo 'ATTRIB'
                if [ ! -d "$INO_FILE" ]
# 如果修改属性的是目录 则不同步，因为同步目录会发生递归扫描，等此目录下的文件发生
```

```
                then
                        rsync -avzcR --password-file=${rsync_passwd_file} $
                        rsync -avzcR --password-file=${rsync_passwd_file} $
                fi
        fi
done
```

## 每两小时做1次全量同步

因为inotify只在启动时会监控目录，他没有启动期间的文件发生更改，他是不知道的，所以这里每2个小时做1次全量同步，防止各种意外遗漏，保证目录一致。

```
crontab -e
* */2 * * * rsync -avz --password-file=/etc/rsync-client.pass /data/ root@192.168.0.18::data && rsync -avz --password-file=/etc/rsync-client.pass /data/ root@192.168.0.19::data
```

改良后我们公司这种百万级小文件也能做到实施同步了。

## 下面附上inotify的参数说明

inotify介绍-- 是一种强大的、细颗粒的、异步的文件系统监控机制，*&####&*_0_*&####&*内核从2.6.13起，加入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，就可以监控文件系统下文件的各种变化情况。

**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 |
| –excludei | 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 | 指定时间输出格式 |
| -e,–event | 后面指定删、增、改等事件 |

**inotifywait events事件说明**

| 事件名称 | 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_write | 修改真实文件内容 |
| close_nowrite | |
| close | |
| open | 文件或目录被打开 |
| moved_to | 文件或目录移动到 |
| moved_from | 文件或目录从移动 |
| 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 |
| delete_self | |
| unmount | 卸载文件系统 |

## 优化 Inotify

**# 在/proc/sys/fs/inotify目录下有三个文件，对inotify机制有一定的限制**

```
[root@web ~]# ll /proc/sys/fs/inotify/
总用量0
-rw-r--r--1 root root 09月923:36 max_queued_events
-rw-r--r--1 root root 09月923:36 max_user_instances
```

```
5 -rw-r--r--1 root root 09月923:36 max_user_watches
```

---

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

---

```
[root@web ~]# echo 50000000>/proc/sys/fs/inotify/max_user_watches -- 把他加入
[root@web ~]# echo 50000000>/proc/sys/fs/inotify/max_queued_events
```

**附录：**

**Rsync的命令格式可以为以下六种：**

```
1 rsync [OPTION]... SRC DEST
2 rsync [OPTION]... SRC [USER@]HOST:DEST
3 rsync [OPTION]... [USER@]HOST:SRC DEST
4 rsync [OPTION]... [USER@]HOST::SRC DEST
5 rsync [OPTION]... SRC [USER@]HOST::DEST
6 rsync [OPTION]... rsync://[USER@]HOST[:PORT]/SRC [DEST]
```

对应于以上六种命令格式，rsync有六种不同的工作模式：

1)拷贝本地文件。当SRC和DES路径信息都不包含有单个冒号”:”分隔符时就启动这种工作模式。如：rsync -a /data /backup

2)使用一个远程shell程序(如rsh、ssh)来实现将本地机器的内容拷贝到远程机器。当DST路径地址包含单个冒号”:”分隔符时启动该模式。如：rsync -avz *.c foo:src

3)使用一个远程shell程序(如rsh、ssh)来实现将远程机器的内容拷贝到本地机器。当SRC地址路径包含单个冒号”:”分隔符时启动该模式。如：rsync -avz foo:src/bar /data

4)从远程rsync服务器中拷贝文件到本地机。当SRC路径信息包含”::”分隔符时启动该模式。如：rsync -av root@172.16.78.192::www /databack

5)从本地机器拷贝文件到远程rsync服务器中。当DST路径信息包含”::”分隔符时启动该模式。如：rsync -av /databack root@172.16.78.192::www

6)列远程机的文件列表。这类似于rsync传输，不过只要在命令中省略掉本地机信息即可。如：rsync -v rsync://172.16.78.192/www

**rsync参数的具体解释如下：**

```
-v, --verbose 详细模式输出
-q, --quiet 精简输出模式
-c, --checksum 打开校验开关，强制对文件传输进行校验
-a, --archive 归档模式，表示以递归方式传输文件，并保持所有文件属性，等于-rlptgoD
-r, --recursive 对子目录以递归模式处理
-R, --relative 使用相对路径信息
-b, --backup 创建备份，也就是对于目的已经存在有同样的文件名时，将老的文件重新命名
--backup-dir 将备份文件(如~filename)存放在在目录下。
-suffix=SUFFIX 定义备份文件前缀
-u, --update 仅仅进行更新，也就是跳过所有已经存在于DST，并且文件时间晚于要备份的
-l, --links 保留软链结
-L, --copy-links 想对待常规文件一样处理软链结
--copy-unsafe-links 仅仅拷贝指向SRC路径目录树以外的链结
--safe-links 忽略指向SRC路径目录树以外的链结
-H, --hard-links 保留硬链结
-p, --perms 保持文件权限
-o, --owner 保持文件属主信息
-g, --group 保持文件属组信息
-D, --devices 保持设备文件信息
-t, --times 保持文件时间信息
-S, --sparse 对稀疏文件进行特殊处理以节省DST的空间
-n, --dry-run现实哪些文件将被传输
-W, --whole-file 拷贝文件，不进行增量检测
-x, --one-file-system 不要跨越文件系统边界
-B, --block-size=SIZE 检验算法使用的块尺寸，默认是700字节
-e, --rsh=COMMAND 指定使用rsh、ssh方式进行数据同步
--rsync-path=PATH 指定远程服务器上的rsync命令所在路径信息
-C, --cvs-exclude 使用和CVS一样的方法自动忽略文件，用来排除那些不希望传输的文件
```

```
--existing 仅仅更新那些已经存在于DST的文件，而不备份那些新创建的文件
--delete 删除那些DST中SRC没有的文件
--delete-excluded 同样删除接收端那些被该选项指定排除的文件
--delete-after 传输结束以后再删除
--ignore-errors 及时出现IO错误也进行删除
--max-delete=NUM 最多删除NUM个文件
--partial 保留那些因故没有完全传输的文件，以是加快随后的再次传输
--force 强制删除目录，即使不为空
--numeric-ids 不将数字的用户和组ID匹配为用户名和组名
--timeout=TIME IP超时时间，单位为秒
-I, --ignore-times 不跳过那些有同样的时间和长度的文件
--size-only 当决定是否要备份文件时，仅仅察看文件大小而不考虑文件时间
--modify-window=NUM 决定文件是否时间相同时使用的时间戳窗口，默认为0
-T --temp-dir=DIR 在DIR中创建临时文件
--compare-dest=DIR 同样比较DIR中的文件来决定是否需要备份
-P 等同于 --partial
--progress 显示备份过程
-z, --compress 对备份的文件在传输时进行压缩处理
--exclude=PATTERN 指定排除不需要传输的文件模式
--include=PATTERN 指定不排除而需要传输的文件模式
--exclude-from=FILE 排除FILE中指定模式的文件
--include-from=FILE 不排除FILE指定模式匹配的文件
--version 打印版本信息
--address 绑定到特定的地址
--config=FILE 指定其他的配置文件，不使用默认的rsyncd.conf文件
--port=PORT 指定其他的rsync服务端口
--blocking-io 对远程shell使用阻塞IO
-stats 给出某些文件的传输状态
--progress 在传输时现实传输过程
--log-format=formAT 指定日志文件格式
--password-file=FILE 从FILE中得到密码
--bwlimit=KBPS 限制I/O带宽，KBytes per second
-h, --help 显示帮助信息
```