通过rsync+inotify实现数据的实时备份

# rsync的优点与不足

与传统的cp、tar备份方式相比，rsync具有安全性高、备份迅速、支持增量备份等优点，通过rsync可以解决对实时性要求不高的数据备份需求，例如定期的备份文件服务器数据到远端服务器，对本地磁盘定期做数据镜像等。

随着应用系统规模的不断扩大，对数据的安全性和可靠性也提出的更好的要求，rsync在高端业务系统中也逐渐暴露出了很多不足，首先，rsync同步数据时，需要扫描所有文件后进行比对，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。而且正在发生变化的往往是其中很少的一部分，这是非常低效的方式。其次，rsync不能实时的去监测、同步数据，虽然它可以通过linux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。基于以上原因，rsync+inotify组合出现了！

# 初识inotify

Inotify 是一种强大的、细粒度的、异步的文件系统事件监控机制，linux内核从2.6.13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、移动等各种细微事件，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就是这样的一个第三方软件。在上面章节中，我们讲到，rsync可以实现触发式的文件同步，但是通过crontab守护进程方式进行触发，同步的数据和实际数据会有差异，而inotify可以监控文件系统的各种变化，当文件有任何变动时，就触发rsync同步，这样刚好解决了同步数据的实时性问题。

# 安装inotify工具inotify-tools

由于inotify特性需要Linux内核的支持，在安装inotify-tools前要先确认Linux系统内核是否达到了2.6.13以上，如果Linux内核低于2.6.13版本，就需要重新编译内核加入inotify的支持，也可以用如下方法判断，内核是否支持inotify：

|  |
| --- |
| [root@localhost webdata]# uname -r  2.6.18-164.11.1.el5PAE  [root@localhost webdata]# ll /proc/sys/fs/inotify  总计 0  -rw-r--r-- 1 root root 0 04-13 19:56 max\_queued\_events  -rw-r--r-- 1 root root 0 04-13 19:56 max\_user\_instances  -rw-r--r-- 1 root root 0 04-13 19:56 max\_user\_watches  如果有上面三项输出，表示系统已经默认支持inotify，接着就可以开始安装inotify-tools了。 |

可以到http://inotify-tools.sourceforge.net/下载相应的inotify-tools版本，然后开始编译安装：

|  |
| --- |
| #inotify-tools安装完成后，会生成inotifywait和inotifywatch两个指令，其中，inotifywait用于等待文件或文件集上的一个特定事件，它可以监控任何文件和目录设置，并且可以递归地监控整个目录树。  #inotifywatch用于收集被监控的文件系统统计数据，包括每个inotify事件发生多少次等信息。  yum install inotify-tools -y |

# inotify相关参数

inotify定义了下列的接口参数，可以用来限制inotify消耗kernel memory的大小。由于这些参数都是内存参数，因此，可以根据应用需求，实时的调节其大小。下面分别做简单介绍。

|  |
| --- |
| #表示调用inotify\_init时分配给inotify instance中可排队的event的数目的最大值，超出这个值的事件被丢弃，但会触发IN\_Q\_OVERFLOW事件。  /proc/sys/fs/inotify/max\_queued\_evnets  #表示每一个real user ID可创建的inotify instatnces的数量上限。  /proc/sys/fs/inotify/max\_user\_instances  #表示每个inotify instatnces可监控的最大目录数量。如果监控的文件数目巨大，需要根据情况，适当增加此值的大小，  /proc/sys/fs/inotify/max\_user\_watches  例如：echo 30000000 > /proc/sys/fs/inotify/max\_user\_watches |

# inotifywait相关参数

Inotifywait是一个监控等待事件，可以配合shell脚本使用它，下面介绍一下常用的一些参数：

*  -m， 即--monitor，表示始终保持事件监听状态。
*  -r， 即--recursive，表示递归查询目录。
*  -q， 即--quiet，表示打印出监控事件。
*  -e， 即--event，通过此参数可以指定要监控的事件，常见的事件有modify、delete、create、attrib等。

|  |
| --- |
| access           #文件或目录被读取。  modify           #文件或目录内容被修改。  attrib             #文件或目录属性被改变。  close           #文件或目录封闭，无论读/写模式。  open           #文件或目录被打开。  moved\_to     #文件或目录被移动至另外一个目录。  move           #文件或目录被移动另一个目录或从另一个目录移动至当前目录。  create            #文件或目录被创建在当前目录  delete           #文件或目录被删除  unmount        #文件系统被卸载 |

# rsync+inotify企业应用案例

案例描述：这是一个CMS内容发布系统，后端采用负载均衡集群部署方案，有一个负载调度节点和多个服务节点以及一个内容发布节点构成，内容发布节点负责将用户发布的数据生成静态页面，同时将静态网页传输到多个服务节点，而负载调度节点负责将用户请求根据负载算法调度到相应的服务节点，实现用户访问。用户要求在前端访问到的网页数据始终是最新的、一致的。

解决方案：为了保证用户访问到的数据一致性和实时性，必须保证多个服务节点与内容发布节点的数据始终是一致的，这就需要通过文件同步工具来实现，这里采用rsync，同时又要保证数据是实时的，这就需要inotify，即：使用inotify监视内容发布节点文件的变化，如果文件有变动，那么就启动rsync，将文件实时同步到多个服务节点。

## 安装rsync与inotify-tools

inotify-tools是用来监控文件系统变化的工具，因此必须安装在内容发布节点，服务节点无需安装inotify-tools，另外需要在web1、web2、web3、webserver节点上安装rsync，由于安装非常简单，这里不在讲述。

在这个案例中，内容发布节点（即server）充当了rsync客户端的角色，而服务节点充当了rsync服务器端的角色，整个数据同步的过程，其实就是一个从客户端向服务端推送数据的过程。这点与上面我们讲述的案例刚好相反。

## 服务节点配置rsync

这里给出三个服务节点的rsync配置文件，以供参考，读者可根据实际情况自行修改。

|  |
| --- |
| #指定备份的用户，nobody为任何用户  uid=nobody  #指定备份的组，nobody为任意组  gid=nobody  #如果"use chroot"指定为true，那么rsync在传输文件以前首先chroot到path参数所指定的目录下。这样做的原因是实现额外的安全防护，但是缺点是需要以roots权限，并且不能备份指向外部的符号，连接所指向的目  录文件。默认情况下chroot值为true。  use chroot =no  #最大连接数  max connections=200  #覆盖客户端指定的超时时间  timeout=600  #端口号  port=873  #pid，lock,log文件存放位置  pid file=/var/run/rsyncd.pid  lock file=/var/run/rsyncd.lock  log file=/var/log/rsyncd.log  [upload]  #参与同步的目录  path=/mnt/images/upload/  #表示忽略一些无关的I/O错误  ignore errors  #允许读和写  read only=no  #不允许列清单  list=no  #允许某个网段进行连接  hosts allow=122.225.227.163,10.161.218.32  #认证的用户名，并非是真实存在的用户，也可以自行定义的用户  auth users=test  #密码文件存放地址  secrets file=/etc/rsyncd.password |

在服务节点rsyncd.conf文件配置完成后，依次启动rsync守护进程，接着将rsync服务加入到自启动文件中：

echo “/usr/local/bin/rsync --daemon” >>/etc/rc.local

到此为止，三个web服务节点已经配置完成。

## 配置内容发布节点

配置内容发布节点的主要工作是将生成的静态网页实时的同步到集群中服务节点，这个过程可以通过一个shell脚本来完成，脚本内容大致如下：

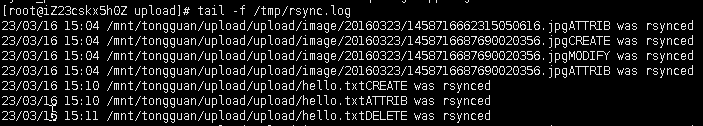
|  |
| --- |
| #!/bin/bash  . /etc/profile  host=10.162.94.58  src=/mnt/tongguan/upload/latex  dst=latex  user=test  /usr/bin/inotifywait -mrq --timefmt '%d/%m/%y %H:%M' --format '%T %w%f%e' -e modify,delete,create,attrib $src \  | while read files  do  rsync -vzrtopg --delete --progress --password-file=/etc/rsyncd.password $src $user@$host::$dst  echo "${files} was rsynced" >>/tmp/rsync.log 2>&1  done |
| #!/bin/bash  . /etc/profile  host=10.162.94.58  src=/mnt/tongguan/upload/upload  dst=upload  user=test  /usr/bin/inotifywait -mrq --timefmt '%d/%m/%y %H:%M' --format '%T %w%f%e' -e modify,delete,create,attrib $src \  | while read files  do  rsync -vzrtopg --delete --progress --password-file=/etc/rsyncd.password $src $user@$host::$dst  echo "${files} was rsynced" >>/tmp/rsync.log 2>&1  done |

脚本相关解释如下：

--timefmt：指定时间的输出格式。

--format：指定变化文件的详细信息。

这两个参数一般配合使用，通过指定输出格式，输出类似与：

这个脚本的作用就是通过inotify监控文件目录的变化，进而触发rsync进行同步操作，由于这个过程是一种主动触发操作，通过系统内核完成的，所以，比起那些遍历整个目录的扫描方式，效率要高很多。

有时会遇到这样的情况：向inotify监控的目录写入一个很大文件时，由于写入这个大文件需要一段时间，此时inotify就会持续不停的输出该文件被更新的信息， 这样就会持续不停的触发rsync去执行同步操作，占用了大量系统资源，那么针对这种情况，最理想的做法是等待文件写完后再去触发rsync同步。 在这种情况下，可以修改inotify的监控事件，

即：“-e close\_write,delete,create,attrib”。

最后，将此脚本加入系统自启动文件：

echo "/home/k12/watchLatex.sh &" >>/etc/rc.local

echo "/home/k12/watchUpload.sh &" >>/etc/rc.local

这样就完成了内容发布节点的所有配置工作。

## 测试rsync+inotify实时同步功能

所有配置完成后，可以在网页发布节点的/web/wwwroot目录下添加、删除或者修改某个文件，然后到三个服务节点对应的目录查看文件是否跟随网页发布节点的/web/wwwroot目录下文件发生变化，如果你看到三个服务节点对应的目录文件跟着内容发布节点目录文件同步变化，那么我们这个业务系统就配置成功了。