rsync多线程异步传输 rsync走的什么协议 转载

墨韵流香 2024-04-02 16:35:11

文章标签 rsync多线程异步传输 linux 服务器 数据同步 实时同步 Linux 文章分类 运维

阅读数 72

一、rsync 概述

rsync 是类 unix 系统下的数据镜像备份工具。一款支持快速完全备份和增量备份的工具，支持本地赋值，远程同步等，类似于 scp 命令；rsync 命令在同步文件之前要先登录目标主机进行用户身份认证，认证过后才能进行数据同步，身份认证方式取决于所使用的协议类型，rsync 一般使用两种协议进行数据同步：ssh 协议和 rsync 协议。

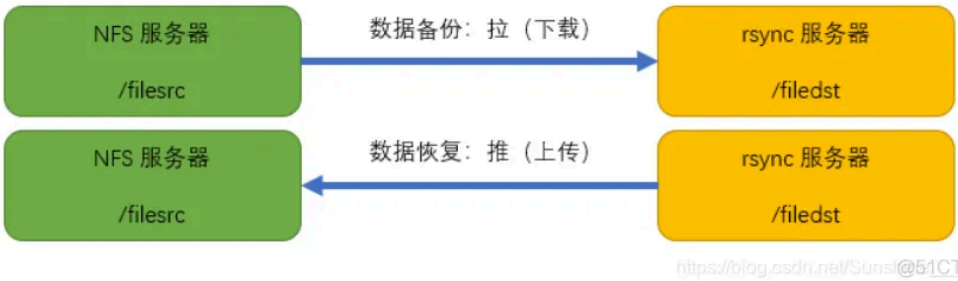
二、rsync 特性

- 能更新整个目录树和文件系统
- 有选择性的保留符号链接、硬链接、文件属性、权限、设备以及时间等
- 对于安装来说，无任何特殊权限要求
- 对于多个文件来说，文件传输效率高
- 能用 ssh 或自定义端口作为传输入口端口

三、rsync 工作原理

既然涉及到数据同步，必要的两个概念是：源地址（文件），目标地址（文件），以及以哪一方为基准，例如，想让目标主机上的文件和本地文件保持同步，则是以本地文件为同步基准，将本地文件作为源文件推送到目标主机上。
rsync 在进行数据同步之前需要先进行用户身份验证，验证方式取决于使用的连接方式：
ssh 登录验证模式： 使用 ssh 协议作为基础进行用户身份认证，然后进行数据同步
rsync 登录验证模式： 使用 rsync 协议进行用户身份认证（非系统用户），然后进行数据同步。

数据同步方式： 推送（上传）、拉取（下载）



四、rsync 实验演示

我们一般使用 rsync 来进行单向数据同步，因此我们需要确定一个基准，比如：两台服务器，一台 NFS 作为网站数据服务器（基准服务器），另外一台专门做 rsync 数据备份服务器，我们以此为基础开始我们的实验

1. **ssh 协议数据同步：将NFS 服务器数据同步备份到 rsync 服务器**

实验环境：一台 NFS 服务器，一台 rsync 服务器
在两台服务器上分别创建目录：（/filesrc、/filedst）

下行同步（下载）

```
格式：rsync -avz 服务器地址:/服务器目录/*  /本地目录
示例：rsync -avz root@192.168.88.11:/filesrc/* /filedst
    -a：归档模式，递归并保留对象属性
    -v：显示同步过程
    -z：在传输文件时进行压缩
```

上行同步（上传）

```
格式：rsync -avz /本地目录/* 服务器地址:/服务器目录
示例：rsync -avz /filedst/* root@192.168.88.11:/filesrc
注意：使用 root 用户进行实验可以，但生产环境中尽量使用单独创建的普通用户，减少权限溢出
创建用来做数据同步的用户，并给予用户对目录的相应权限，一般使用 ACL 设置权限
# useradd zhangsan
# passwd zhangsan
# serfacl -m u:zhangsan:rwx /filesrc
```

扩展：若要实现免密码数据同步，只需要做好 ssh 密钥对登录即可

2. **rsync 协议数据同步：将 NFS 服务器数据同步备份到 rsync 服务器**

实验环境：一台服务器，一台客户端

1）在两台服务器上分别创建目录（/filesrc、/filedst）

2）搭建 rsync 服务（仅需要在 NFS 服务器上搭建即可）

- 创建主配置文件

```
address=192.168.88.11                       #rsync 服务绑定IP
port 873                                    #默认服务端口 873
log file = /var/log/rsyncd.log              #日志文件位置
pid file = /var/run/rsyncd.pid              #进程号文件位置
[web]                                       #共享名：用来连接是卸载 url 上的，切记
        comment = web directory backup      #共享描述话语
        path = /filesrc                     #实际共享目录
        read only = no                      #是否仅许读取
        dont compress = *.gz *.bz2          #哪些文件类型不进行压缩
        auth users = user1                  #登录用户名（非系统用户，需要自行创建）
        secrets file = /etc/rsyncd_users.db #认证所需账户密码文件（需自行创建-同上）
```

- 创建认证所需账户密码文件

```
# vim /etc/rsyncd_users.db
user1:123456
# chmod 600 /etc/rsyncd_users.db
```

- 启动服务

```
# rsync --daemon
# netstat -antp | grep :873
```

- 设置映射用户对共享目录有权限（r）

```
# setfacl -m u:nobody:rwx /filesrc
```

注意：关闭服务可使用 kill 命令，但偶尔会造成服务被结束，但进程号配置文件不被删除的问题，若遇到此类问题可自己手动删除，再启动则正常（建议自己写一个 rsync 的服务管理脚本）

下行同步（下载）

```
格式： rsync -avz rsync://用户名@服务器地址/共享模块名 /本地目录
示例：rsync -avz rsync://user1@192.168.88.11/web /filedst
拓展：--delete：删除本地比服务器多出来的文件（源地址没有，目标地址有的删除）
    rsync -avz --delete rsync://user1@192.168.88.11/web /filedst
```

上行同步（上传）

```
格式：rsync -avz /本地目录/* rsync://用户名@服务器地址/共享模块名
示例：rsync -avz /filedst/* rsync://user1@192.168.88.11/web
```

拓展：rsync 协议的免密码可以借助一个环境变量实现

export RSYNC_PASSWORD=虚拟用户密码（客户端生成）

五、配置 rsync + inotify 实时同步

定期同步的缺点：

执行备份的时间固定，延期明显，实时性差

当同步源长期不变化时，密集的定期任务时不必要的（浪费资源）

实时同步的优点：

一旦同步源出现变化，立即启动备份，实时性好

只要同步源无变化，则不执行备份，节省资源

inotify 简介

inotify 是一个 Linux 内核特性，它监控文件系统，并且及时向专门的应用程序发出相关的事件警告，比如删除、读、写和卸载操作等。要使用 inotify ，必须具备一台带有 2.6.13 版本的内核操作系统

inotify 两个监控命令：

inotifywait：用于持续监控，实时传输结果（常用）

inotifywatch：用于短期监控，任务完成后再出结果

inotify 部署

```
# yum -y install gcc*
# tar -xf inotify-tools-3.14.tar.gz
# cd inotify-tools-3.14
# ./configure && make && make install
```

inotifywait 命令格式

```
格式：inotifywait -mrq -e 监控动作1,监控动作2 /监控目录 &
示例：inotifywait -mrq -e create,delete /filesrc &
    -m：始终保持事件监听状态
    -r：递归查询目录
    -q：只打印监控事件的信息
监控动作：modify（内容），create，attrib（权限），move，delete
```

利用 rsync + inotifywait 结合脚本实现单向实时同步

```
# vim src.sh
#!/bin/bash
a="inotifywait -mrq -e create,delete,modify /filesrc"
b="rsync -avz /filesrc/* root@192.168.88.10:/filedst"
$a | while read directory event file        #while 判断是否接收到监控记录
do
        $b
done
```

注：用户登录时要求免密码验证

实验结果验证

在服务器端创建，删除文件，查看备份端是否正常

拓展：调整 inotify 监控的文件数量

调整inotify 内核参数（/etc/sysctl.conf）

mak_queue_events	监控队列大小
mak_user_instances	最多监控实例数
max_user_watches	每个实例最多监控文件数

六、配置 unison + inotify 实现双向实时同步

rsync 在单向同步上支持的非常好，切效率高，但是在双向同步支持较差；unison则是双向同步的优秀工具，但其缺点是同步效率较低

1. **环境要求**

1) 准备好同步所需的两个目录

2) 如若用 root 来实现登录的话，生成密钥对，以便于免密码验证

3) 准备好 inotify 和 unison 的软件包

2. **安装步骤**

1) 先安装 inotify

```
# tar -xf inotify-tools-3.14.tar.gz
# cd inotify-tools-3.14
# ./configure && make && make install
```

2) 再安装 ocaml

```
# tar -xf ocaml-3.10.1.tar.gz
# cd ocaml-3.10.1
# ./configure           #忽略所有报错
# make world opt
# make install
```

3) 安装 unison

```
# tar -xf unison-2.13.16.tar.gz
# cd unison-2.13.16
# make UISTYLE=test THREADS=true STATIC=true    #已经存在 Makefile 文件，不需要 ./configure
# cp unison /usr/local/bin/ #把生成的脚本拷贝出来
```

注意：同样的操作在服务器端也做一遍

3. **配置脚本**

注：双向自动同步，监控目录和数据同步时，源目录不能使用 * 通配符传输，否则会变成死循环

filesrc 端：

```
#!/bin/bash
a="inotifywait -mrq -e create,delete,modify /filesrc"
b="/usr/local/bin/unison -batch /filesrc/ ssh://192.168.88.10//filedst/"        #-batch: 扌
$a | while read directory event file
do
        $b
done
```

filedst 端：

```
#!/bin/bash
a="inotifywait -mrq -e create,delete,modify /filedst"
b="/usr/local/bin/unison -batch /filedst/ ssh://192.168.88.11//filesrc/"        #-batch: 扌
$a | while read directory event file
do
        $b
done
```

4. **测试**

将两个脚本放入后台执行：bash src.sh &

分别在两个主机上创建文件查看是否可以实现双向实时同步（可能会有延迟）