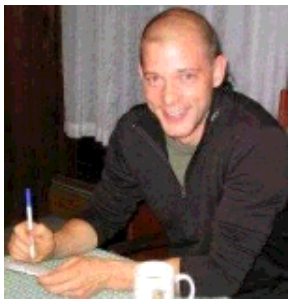
LinuxFocus article number 326
http://linuxfocus.org

Rsync: The Best Backup System Ever



by Brian Hone
<bhone(at)eink.com>

About the author:

Brian Hone is a system administrator and software developer at E Ink corporation. In his spare time he surfs in very cold water and dangles off rock faces.

Abstract:

Backup is one of the hardest and most neglected parts of system administration. It is also one of the most important. It is the last defense against hardware failures, security breaches, and the biggest threat of all: end users. While there are many backup systems out there costing many thousands of dollars, which archive to expensive tape drives using buggy proprietary software, there is a better way: Rsync and a cheap disk array.

The Problem

I can give you a long list of reasons why backup is a system administrator's nightmare. If you're a system administrator, though, I probably don't need to. Some of those reasons are: expensive hardware which is broken more often that it is operational, expensive software which is a management nightmare, and long hours spent restoring multiple versions of files. To make matters worse, there is usually very little corporate priority placed on backups, until that inevitable day when they're needed. If you've done backup/restore, odds are you've had this conversation:

User: "I lost a file. I need you to get it back right away."
SysAdmin: "Ok, what's it called?"
User: "I don't know, I think it had an 'e' in the name."
SysAdmin: "Ok, what directory was it in?"
User: "I don't know, it could be in one of these three..."
SysAdmin: "*Sigh* Do you know what date you last used the file?"
User: "Well....I think it was a thursday in either February or April. What's the problem? I thought you

people had a backup system to take care of this kind of thing."

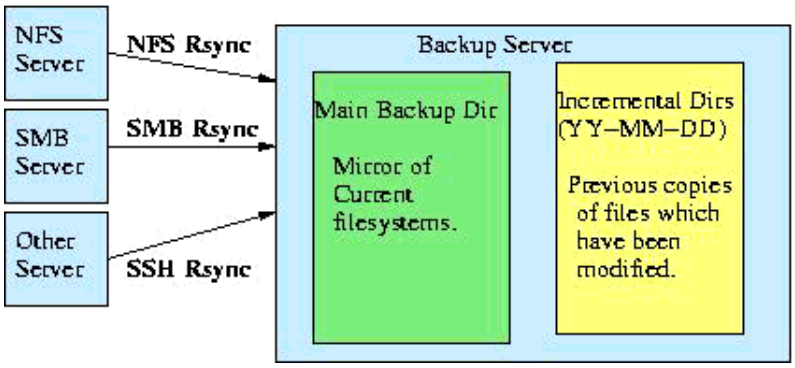
The Rsync Alternative

Rsync is a powerful implementation of a beautiful little algorithm. Its primary power is the ability to effeciently mirror a filesystem. Using rsync, it is easy to set up a system which will keep an up to date copy of a filesystem using a flexible array of network protocols, such as nfs, smb or ssh. The second feature of rsync which this backup system exploits is its ability to archive old copies of files which have been changed or deleted. There are far too many features of rsync to consider in this article. I strongly recommend that you read up on it at rsync.samba.org.

The System

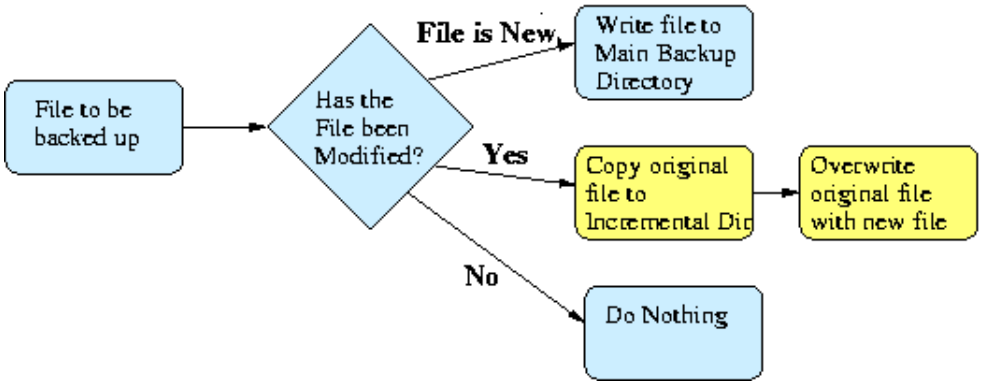
In brief, this system uses a cheap Linux box with a lot of cheap disk and a small shell script which calls rsync. **[Fig 1]** When doing a backup, we tell rsync to create a directory named 'YY-DD-MM' as a place to store incremental changes. Next, rsync examines the servers we backup for changes. If a file has changed, it copies the old version to the incremental directory, and then overwrites the file in the main backup directory. **[Fig 2]**

Figure 1: Structure of an Rsync Backup Server



In general, a day's changes tend to be between a small percentage of the total filesystem. I find the typical average size to be between .5% and 1%. Therefore, with a set of backup disks which is twice the size of our fileservers, you can keep 50-100 days of incremental backups on hard drive. When the disk becomes full, just swap in a new set of disks, and move the old ones offsite. In practice, it is possible to keep over six months of incrementals on disk. In fact, if you can find space somewhere, you can copy your incrementals to another server before rotating the disks. In this way, you can keep an arbitrarily large number of incrementals on disk.

Figure 2: Backup of a file using Rsync



The Advantages: Disaster Recover and File Restoration Made Easy

Go back to the imaginary conversation above. Now, instead of a cumbersome tape-based system, imagine having six months of incremental backups happily waiting for you on your Linux box. Using your favorite combination of locate/find/grep, you can find all occurances of files owned by our imaginary user, which contain an 'e' and are timestamped on a thursday in February or April, and dump them into a directory in the user's home directory. The problem of figuring out which version is the correct one has just become my favorite kind of problem: someone else's.

Next, imagine our favorite scenario - complete failure. Lets say you have a big nfs/samba server which you lose. Well, if you've backed up your samba configs, you can bring your backup server up as a read-only replacement in minutes. Let's see you try that with tape.

How Rsync/Hard Drive Backup Stacks up Against Tape

	Tape Backup	Rsync
Cost	Very High	Low
Full Backup	Fast	Fast
Incremental Backup	Fast	Fast
Full Restore	Very Slow, probably multiple tapes	Fast - it's all on disk
File Restore	Slow, maybe multiple tapes, often hard to find correct version.	Very Fast - it's all on disk and you have the full power of UN*X search tools like find, grep and locate
Complete Failure	Only option is full restore	Can be turned on as a fileserver in a pinch.

The Tools

There are a lot of ways to set this up. All the tools here are open-source, included in standard distributions, and very flexible. Here, we describe one possible setup, but it is far from the only way.

- **The Server:** I use RedHat Linux. Any distribution should work, as should any UN*X. (I've even set this up with Mac OS X) One caveat: a lot of RAM helps.
- **Disk:** The easiest way we've found of building a big cheap set of disk is a PCI firewire card connected to a bunch of cheap IDE disks in external firewire cases. Setting up Linux to use these as one big RAID partition is fairly painless.
- **The Software:** Rsync is a great tool. It is sort of a jacknife of filesystem mirroring. If you don't know about it, check it out at rsync.samba.org.
- **Connecting to Fileservers:** Rsync is very flexible. We use nfs and smbfs. You can also use rsync's own network protocol by running an rsync daemon on the fileserver. You can also tell rsync to use ssh for securely backing up remote sites. See the resources below for information of setting up these connections.

Scripting It

The basic form of this script came from the rsync website. There is really only one command:

rsync --force --ignore-errors --delete --delete-excluded --exclude-from=exclude_file --backup --backup-dir=‘date +%Y-%m-%d‘ -av

The key options here are:

- *--backup*: create backups of files before overwriting them
- *--backup-dir=‘date +%Y-%m-%d‘*: create a backup directory for those backups which will look like this: 2002-08-15
- *-av*: archive mode and verbose mode.

The following script can be run every night using Linux's built in cron facility. To start the script at 11pm each night, use the command "crontab -e", and then type the following:

0 23 * * * /path/to/your/script

The Script

Here's my shell script to tie it all together. Again, there are a lot of ways of doing this. This is just one implementation.

```
#!/bin/sh

############################################################
```

```
# Script to do incremental rsync backups
# Adapted from script found on the rsync.samba.org
# Brian Hone 3/24/2002
# This script is freely distributed under the GPL
############################################################

###################################
# Configure These Options
###################################

###################################
# mail address for status updates
#  - This is used to email you a status report
###################################
MAILADDR=your_mail_address_here

###################################
# HOSTNAME
#  - This is also used for reporting
###################################
HOSTNAME=your_hostname_here

###################################
# directory to backup
# - This is the path to the directory you want to archive
###################################
BACKUPDIR=directory_you_want_to_backup

###################################
# excludes file - contains one wildcard pattern per line of files to exclude
#  - This is a rsync exclude file.  See the rsync man page and/or the
#    example_exclude_file
###################################
EXCLUDES=example_exclude_file

###################################
# root directory to for backup stuff
###################################
ARCHIVEROOT=directory_to_backup_to

########################################
# From here on out, you probably don't  #
#   want to change anything unless you  #
#   know what you're doing.             #
########################################

# directory which holds our current datastore
CURRENT=main

# directory which we save incremental changes to
INCREMENTDIR=`date +%Y-%m-%d`

# options to pass to rsync
OPTIONS="--force --ignore-errors --delete --delete-excluded \
 --exclude-from=$EXCLUDES --backup --backup-dir=$ARCHIVEROOT/$INCREMENTDIR -av"

export PATH=$PATH:/bin:/usr/bin:/usr/local/bin

# make sure our backup tree exists
install -d $ARCHIVEROOT/$CURRENT
```

```
# our actual rsyncing function
do_rsync()
{
   rsync $OPTIONS $BACKUPDIR $ARCHIVEROOT/$CURRENT
}

# our post rsync accounting function
do_accounting()
{
   echo "Backup Accounting for Day $INCREMENTDIR on $HOSTNAME:">/tmp/rsync_script_tm
   echo >> /tmp/rsync_script_tmpfile
   echo "################################################">>/tmp/rsync_script_tmpfil
   du -s $ARCHIVEROOT/* >> /tmp/rsync_script_tmpfile
   echo "Mail $MAILADDR -s $HOSTNAME Backup Report < /tmp/rsync_script_tmpfile"
   Mail $MAILADDR -s $HOSTNAME Backup Report < /tmp/rsync_script_tmpfile
   echo "rm /tmp/rsync_script_tmpfile"
   rm /tmp/rsync_script_tmpfile
}

# some error handling and/or run our backup and accounting
if [ -f $EXCLUDES ]; then
        if [ -d $BACKUPDIR ]; then
                # now the actual transfer
                do_rsync && do_accounting
        else
                echo "cant find $BACKUPDIR"; exit
        fi
        else
                echo "cant find $EXCLUDES"; exit
fi
```

Resources

- Rsync: http://rsync.samba.org
- NFS: http://nfs.sourceforge.net/nfs-howto
- SMBFS: http://samba.org
- Linux RAID: http://linas.org/linux/raid.html

2005-01-14, generated by lfparser_pdf version 2.51