Offline rsync

(March 2007)

The problem statement

Picture this:
A lowly developer splits his working time between home and the company premises. He uses the following routine: at the end of the working day at home/company, he compresses the source tree and picks it up on his USB stick. At company/home, he decompresses the package and hacks on.

This procedure is starting to show some issues, however. For one, the data barely fit in the USB stick, even compressed. He is contemplating a bigger USB stick (maybe one of the new USB hard drives), but even if he uses that, the problem is that the procedure takes too long: the repository snapshot he works on is huge, and compressing it and taking it with him simply takes too long.

Of course, if this developer had a fast Internet access at home, he could use rsync; that is, provided that the company allows remote access at all - but does it?

And, naturally, the changes he makes daily couldn't *possibly* be *that* big - why should he take a whole source tree every time?

He defends himself, saying that he must be absolutely certain that everything is updated, and that he couldn't just take *patch* files for his work, since other parts of the tree are updated by others. What else could he do?

Well, there is something...

Hello, rdiff

Obviously, rsync can't be used unless both the work machine and the home machine are connected on a network. The rsync algorithm, however, is quite ingenious. You can read all about it here. The amazing thing is, that **simultaneous access to both "files to synchronize" is not necessary**. The algorithm takes a "snapshot" of file A stored in machine 1, in terms of block-based CRCs. It then sends these CRCs over the network to machine 2, where they are used to compute the differences with file B. Finally, the differences are sent in the opposite direction, to update file A.
What if we didn't use the network to send the CRCs and the differences?
What if we just stored them, inside, say, a USB stick?

That's exactly the idea that rdiff implements. You can download the source code and compile it for yourself here (as part of librsync), or get it pre-compiled for your platform (under Debian, `apt-get install rdiff` is enough).

Here is a simple example of its usage, to update a single file:

```
(At home)
/home/jdoe/code$ rdiff signature SrcFileA.c SrcFileA.c.signature
(The CRCs of the blocks in file SrcFileA.c are now stored
 in SrcFileA.c.signature. You take the .signature file to work)

(At work)
/home/JohnDoe/code$ rdiff delta SrcFileA.c.signature \
        SrcFileA.c SrcFileA.c.delta
(Using the CRCs calculated at home, find the differences of the
 work file and store them in SrcFileA.c.delta)

(Back home)
/home/jdoe/code$ rdiff patch SrcFileA.c SrcFileA.c.delta \
        SrcFileA.c.new
(update SrcFileA.c to SrcFileA.c.new using the deltas)
```

Right!
It works, but... do we really have to go through all this, just to synchronize a single file? The source tree has thousands of files, inside a huge tree!

Automation scripts

Naturally, no-one in their right mind would ever attempt to go through this procedure manually for a large tree of files. The lowly developer I described above is actually me, so at some point (during a very boring series of presentations I had to attend) I wrote 3 perl scripts to automate the process.
The results are depicted in the "screenshot" below:

```
(At home)
/home/jdoe/code$ snapOldData.pl SrcTree/
/home/jdoe/code$ ls -lF
drwxr-xr-x     7 jdoe coder 0 Mar 24 13:07 SrcTree/
-rw-r--r-- 12345 jdoe coder 0 Mar 24 13:07 SrcTree.signs.tar.bz2
/home/jdoe/code$ cp SrcTree.signs.tar.bz2 /mnt/usbstick/

(At work)
/home/JohnDoe/code$ cp /mnt/usbstick/SrcTree.signs.tar.bz2 .
/home/JohnDoe/code$ createDeltas.pl work/ SrcTree.signs.tar.bz2
/home/JohnDoe/code$ ls -lF
drwxr-xr-x     7 jdoe coder 0 Mar 24 13:07 work/
-rw-r--r-- 12345 jdoe coder 0 Mar 24 13:07 work.deltas.tar.bz2
-rw-r--r-- 12345 jdoe coder 0 Mar 24 13:07 SrcTree.signs.tar.bz2
/home/JohnDoe/code$ rm /mnt/usbstick/SrcTree.signs.tar.bz2
/home/JohnDoe/code$ cp work.deltas.tar.bz2 /mnt/usbstick/

(Back home)
/home/jdoe/code$ cp /mnt/usbstick/work.deltas.tar.bz2 .
/home/jdoe/code$ applyDeltas.pl work.deltas.tar.bz2 SrcTree/
```

The scripts will also update the home work tree with all the new files that have appeared, and they will keep a backup of the files changed (with the extension .rdiff_oldVersion) so that you can revert to your old versions if you so wish.
You can download them here:

- snapOldData
- createDeltas
- applyDeltas

I hope you will find them useful...
Some stats from a Pentium D at 2.8 GHz, running a Debian Linux VMWARE image:

The source tree weighs in at 350MB.
(At home) Size of signature file: 3.9MB
(At work) Time it takes to generate deltas: 5min
(At work) Size of generated "deltas" file: 6.5MB
(Back home) Time it takes to apply the deltas: 6min

(*Looks like I won't be replacing my 256MB USB stick any time soon.*)