The following paper was originally published in the
Proceedings of the Eleventh Systems Administration Conference (LISA '97)
San Diego, California, October 1997

For more information about USENIX Association contact:

1. Phone: 510 528-8649
2. FAX: 510 548-5738
3. Email: office@usenix.org
4. WWW URL: http://www.usenix.org

Bal – A Tool to Synchronize Document Collections Between Computers

Jürgen Christoffel – GMD

ABSTRACT

The enormous growth of computer networks and declining hardware prices allow people to have more than one computer, e.g., to use a primary computer at work, a laptop or notebook as a secondary computer for 'on the road' and maybe a third computer at home.

One non-trivial problem that users face when using more than one computer alternately is the difficulty of keeping multiple, distributed copies of their files up to date and in sync, if they can't permanently share resources between their machines, e.g., when using a notebook computer off-line for some time.

System administrators of Unix sites who need to automatically distribute files between machines have known such problems for years and have come up with various tools to solve their specific needs. But tools which help Unix administrators are not always adequate for their users too.

This paper describes *Bal*, a tool which enables Unix users to more easily keep distributed copies of their files in sync. *Bal* is written in Perl and keeps two directory trees in sync.

Introduction

Document collections are sets of documents which are in some way related. For the purpose of this paper a document collection is a set of files or directories or both which are handled by Bal as a whole.

Users who alternately use multiple computers to do their work will sooner or later face the problem of keeping their documents in sync. As long as they use computers which are connected to the same local-area network it is easy to use a client-server approach to keep files on one machine and share them via some means like NFS. But as soon as one of the computers is not always connected to the network the problems will arise because users keep distributed copies of their documents and have to synchronize those document collections between machines.

Maintaining distributed copies of such collections can be a nightmare if users can't always guarantee that they will only change one of their document collections at once before merging changes between their computers.

The Macintosh and PC user communities have for some years had the benefit of programs like PowerMerge [LEADER] or Laplink [LAPLINK] or the "Briefcases" built into Windows 95 and NT [BRIEFCASE] which keep two document collections in sync. Unix users until now have had to use whatever tools that were available to at least automate some of the work to be done to keep their directories in sync.

Bal – named for the balancing effect it has on distributed document collections – helps users and system administrators keep track of changes made to files or directories and telling them when files have changed, have been deleted, or new files have been created on either side. It can automatically copy, create, or delete files to synchronize two document collections. It warns users if files have been changed in both collections since its last run and offers help in resolving the conflicts.

Comparisons

Modern distributed file systems [TANEN] like the Andrew File System [AFS] address some of the problems but are not always easily available.

While version control software like *rcs*(1) [RCS] or *cvs*(1) [CVS] could be used in theory, these tools are inadequate for keeping home directories in sync in practice because of the overhead and restrictions imposed by them. Bal is intended to manage typical user files like mailboxes or calendar files and version control systems where not designed to handle these. Version control tools regularly keep a third copy of each document on disk and they have special commands for checking documents in or out which need to be remembered.

Tools like *rdist*(1) [RDIST] or *track*(1) [TRACK] which are adequate for specific jobs in the system administrator area don't help users (and system administrators) much because such tools have been designed for an asymmetric problem, namely to distribute master copies of files between servers and clients, which allows overwriting of files on the client.

Automatic file distribution tools like rdist or track copy files between computers using either a *push* model where a master server updates some clients periodically or a *pull* model where each client periodi-

cally queries a master server to retrieve the freshest files.

Both models are very helpful for automated software distribution where, e.g., system administrators need to periodically update system files on a cluster of machines. Tools like rdist and track are thus in widespread use on Unix machines.

The scheme used by such tools is *asymmetric* since it allows one master copy of a set of files on a server which is distributed to possibly multiple clients which are not expected to make changes to their copies. This scheme is not appropriate for users who want to keep distributed document collections in sync.

Users face a *symmetric* problem instead, because both document collections resemble different versions without a master copy or central repository. These document collections need to be merged or synchronized, i.e., documents may have to move both ways and checks have to be made to ensure that changes made in either document collection aren't lost.

Because Bal is intended to be used interactively – it needs user interaction to decide how to resolve potential conflicts – it doesn't face the security problems of the rdist family of tools.

Implementation Issues

Bal is a program to maintain identical copies of document collections on different hosts. It preserves the owner, group, mode, and modification time of files if possible.

Bal is written in Perl [PERL] and uses the high-level constructs like associative arrays provided by Perl to manage its internal data. It uses Perl's interfaces to the various DBM implementations for Unix [HASH] to persistently store information about the state of each document collection between sessions. Which DBM implementation is used is configurable.

Like rdist and similar tools, Bal uses a configuration file which specifies the files of the document collections to keep track of and what actions to take when synchronizing the document collections. The configuration file specifies the left and right document collections respectively in the notation used by rsh/rcp. It also allows to specify via Perl's regular expressions which files to exclude from synchronization.

To properly synchronize two document collections Bal needs to keep track of the state of each document in either collection in order to decide what to do to synchronize them. When first run it compares the two document collections and creates a reference database which contains state information for each file in both collections.

When Bal compares files for the first time, it compares the attributes of each pair of files. Attributes of interest are the *file name, device number, inode number, permissions, number of hard links, uid, gid, size* and the *modification time* of each file. For each pair of files it uses the modification dates to see which one is oldest. All checked attributes will be stored in the database for reference when Bal is run again.

On subsequent runs Bal checks the document collections against the reference database to see what has changed. After synchronizing the two document collections it stores the new state in its database. During synchronization it may encounter one of the following situations:

- a file hasn't changed in either of the document collections
- a file has changed (i.e., deleted, edited or renamed) in one document collection and needs to be updated in the other collection
- a file has been changed (i.e., edited or renamed) in both document collections

The last situation is obviously the most problematic one and Bal doesn't solve them automatically. Instead it copies conflicting files into special subdirectories and suggests various checks to help the user resolve the conflict, among them using *diff*(1) or *cmp*(1) to see if files have only been *touch*(1)ed.

When Bal is run it gathers the relevant file attributes from both the left and the right document collection and compares the attributes to those stored in the reference database. The reference database allows Bal to detect changes to the same file on both sides by comparing the file's attributes on disk with the attributes stored in the reference database.

To actually change files Bal depends on rsh/ssh and rcp/scp to transfer documents between machines and it generates the appropriate commands which it then executes.

Bal can be configured to run in batch mode where it will send a report of the synchronization process via email.

Example Session with Bal

During initialization files are normally considered different if their modification time and size disagree. But when the document collections are not in sync during initialization of the reference database, Bal can be told to use a checksum algorithm like md5 to compare pairs of files and check whether files with different timestamps but same size are actually the same. Using a checksum like md5 additionally allows to find duplicates which have different names.

Changed Files

Once the reference database exists, later runs only need to compare the attributes of each file with those stored in the database. When Bal encounters a file which has changed on one side, it generates the appropriate shell commands to bring both sides in sync again.

The following example shows a Bal run where *priv.bal* is the reference database where the attributes seen on the last run have been stored. The document

collections reside on hosts *aeppel* and *vortex* and Bal is running on *vortex*. The -v flag tells Bal to output the commands it executes; see Figure 1.

Bal copies the newer version to the other side and updates the attributes stored in the reference database for both files.

Conflicting Files

When Bal encounters a pair of files that have both changed on the left and right since its last run, it cannot solve that case automatically. Instead it explains the conflict and outputs suggestions for resolving the conflict; see Figure 2.

Bal puts the older version of the conflicting files into a special conflicts subdirectory – *.bc* in this example – and suggests an action the user might take to resolve the conflict manually. In the example above the suggestion is to run *diff*(1) to compare the two versions.

The conflicts subdirectory is always in the same subdirectory as the original document, so it is easy to see where the document came from.

The user should manually resolve the conflict, e.g., by merging the conflicting versions into one file and then can remove the conflicts subdirectory.

Renamed Files

Bal tries to detect when a file has been renamed on either side. To detect when a file has been renamed it stores the inode number of each file in its reference database. When a file is missing from either side it uses the inode number to find the new name of the file and if the other attributes in the reference database match that file it can rename the corresponding file on the other side to the new name.

If a file has been renamed differently on both sides the current implementation of Bal handles this by simply copying both files as if it where new versions of different files. One might argue that it should signal a conflict instead.

Performance

The current implementation of Bal needs about 3% of the size of the document collections to store attributes in its reference database. On a Linux machine with a Pentium Pro 200 processor it takes about 15 seconds real time to compare two directories containing about 12 Megabytes in 3,267 files. Those numbers do not take the time into account for the actual file transfers needed to bring the directories in sync again.

Future Work

Bal is designed for interactive use and thus is a good candidate for a window-based user interface. We plan to provide an Emacs major mode to interface with Bal and a graphical user interface based on Perl/Tk in the future.

```
vortex[788]: bal -v -f priv.bal
rcp -p lesetips aeppel:priv/lesetips
rcp -p aeppel:priv/lisa-11/register-info lisa-11/register-info
rcp -p aeppel:priv/prep.consult prep.consult
rcp -p aeppel:priv/quotes quotes
rcp -p quotes.news aeppel:priv/quotes.news
rcp -p aeppel:priv/quotes~ quotes~
rcp -p aeppel:priv/signatures signatures
rcp -p aeppel:priv/xyzzy-9707.tar.gz xyzzy-9707.tar.gz
rcp -p aeppel:priv/spooky spooky
rcp -p aeppel:priv/todo todo
```

Figure 1: Example of Bal output

```
vortex[798]: bal -v -f home.bal
rcp -p .zshenv aeppel:.zshenv
rcp -p aeppel:.emacs .emacs
rcp -p Notes/8-97 and aeppel:Notes/8-97
# conflict: mtime mismatch for .netscape/bookmarks.html
#     Sun Aug  3 14:20:04 1997  .netscape/bookmarks.html
#     Wed Jul  2 19:36:58 1997  aeppel:.netscape/bookmarks.html
mkdir .netscape/.bc
rcp -p aeppel:.netscape/bookmarks.html .netscape/.bc/bookmarks.html
rcp -p .netscape/bookmarks.html aeppel:.netscape/bookmarks.html
# action: diff .netscape/bookmarks.html .netscape/.bc/bookmarks.html
```

Figure 2: Call for manual conflict resolution.

To improve the utilization of low-bandwidth connections Bal could be augmented to transfer only the modified parts of changed files. The rsync algorithm [RSYNC] could be applied to update files over low-bandwidth high-latency bi-directional communications links but that would force Bal to use it's own transport protocol instead of simply relying on rsh/ssh to do the work.

Additionally Bal could be augmented to recognize when files have been compressed and instead of transferring the compressed file, compress the file on the other side too.

A future version of Bal will be capable of synchronizing document collections even if one of them is contained inside a tar file. This will make it possible to use Bal to synchronize document collections via tapes without unpacking whole tar files.

Conclusion

Bal is especially suited to the needs of users who use computers as peers and not as clients and server. Because Bal allows transparent compression during transfers it is well suited for users who use one Unix system at work and another one at home and connect via low bandwidth dial-in connections and want better utilization of the low bandwidth connection. The author uses it regularly to synchronize his Linux workstations at home and at his office over a dial-in 64k ISDN connection.

Availability

Bal has been in beta test at the authors site since May '97. It will be made available for anonymous ftp over the Internet from ftp.gmd.de and on the Comprehensive Perl Archive Network (CPAN) once it has left the beta test phase.

Author Information

Jürgen Christoffel studied computer science at the University of Bonn, Germany. He has been working with Unix systems since 1984 and joined GMD as a system administrator in 1988. He has been a Perl user since 1989 and Perl has saved him from C since then. In his spare time he teaches courses in Perl and Emacs. His email address is christoffel@gmd.de .

References

[AFS] Andrew File System (AFS) Homepage at Transarc, http://www.transarc.com/afs/transarc.com/public/www/Public/ProdServ/Product/AFS/ .

[BRIEFCASE] *Windows 95 Resource Kit*, Microsoft Press, 1995.

[CPAN] The Comprehensive Perl Archive Network is available online at http://www.perl.org/CPAN/ .

[CVS] Brian Berliner, "CVS II: Parallelizing Software Development," *Proc. USENIX Winter 1990 Conf.*, January 22-26, 1990, Washington, D.C., pp 341-352.

[HASH] Margo Seltzer and Ozan Yigit, "A New Hashing Unix", *Proc. Usenix 1991 Winter Conf.*, January 21-25, 1991, Dallas, TX, pp 173-184.

[LAPLINK] http://www.travelingsoftware.com/products .

[LEADER] Leader Technologies, *PowerMerge User Guide*, 1992-1994, Leader Technologies, Newport Beach, CA.

[PERL] Larry Wall, Tom Christiansen und Randal Schwartz, *Programming Perl, 2nd edition*, O'Reilly and Associates, 1996.

[RCS] Don Bolinger and Tan Bronson, *Applying RCS and SCCS: From Source Control to Project Control*, O'Reilly, 1995.

[RDIST] Michael A. Cooper, "Overhauling Rdist for the '90s," *Proc. Usenix LISA VI*, October 19-23, 1992, Long Beach, CA, pp 175-188.

[RSYNC] Andrew Tridgell and Paul Mackerras, "The rsync algorithm," Australian National University, TR-CS-96-05.

[TANEN] Andrew S. Tanenbaum, Chapter 5 "Distributed File Systems," *Distributed Operating Systems*, Prentice-Hall, 1995.

[TRACK] Bjorn Satdeva and Paul M. Moriarty, "Fdist: A Domain Based File Distribution System for a Heterogeneous Environment," *Proc. Usenix LISA V*, September 30 - October 3, 1991, San Diego, CA, pp 109-126.