Title: Reconciliation in Files EveryWhere File System

Name: Marcos Bento, U. Nova de Lisboa
Additional authors: Nuno Preguiça (UNL), Carlos Baquero (U. Minho), J. Legatheaux Martins (UNL)
E-mail: marcosbento@gmail.com
Student: yes

In recent years, mobile computing environments have been changing with the increasing use of different types of portable devices with large amounts of storage, ranging from mobile phones to laptops, and from MP3 players and digital cameras to portable storage devices, such as flash-disks. These devices can act as sophisticated, large-capacity storage devices either attached to a computer or directly connected to a network. These devices allow users to always carry with them (partial) replicas of their private data and of data they are sharing with other users. Thus, the characteristics of this new environment differs from the assumption taken in older mobile data management systems.

The Files Everywhere system (FEW) is a distributed file system that manages files stored in computing devices and portable storage devices. Users can group related files in containers and share their containers with other users. Users can create copies of containers in multiple storage devices. Additionally, temporary copies of recently used files are created in portable storage devices. This approach, combined with an optimistic read any, write any model of data access, provides high data availability. Replicas are synchronized using a peer-to-peer epidemic model.

The system integrates a new reconciliation mechanism that executes in each replica independently.Unlike reconciliation mechanisms used traditionally in distributed file systems, our approach relies completely on the propagation of updates as semantic-rich operations. To this end, as we intend to allow applications to continue using the file system interface, when a user closes a file that he has modified, the system infers the semantic-rich operations comparing the original (as when the user has opened the file) and the final states of the file.

Reconciliation is performed using operational transformation (OT) algorithms [1]. These algorithms have been developed originally for synchronous groupware and allow replicas to converge independently by executing, in each replica by a different order, a specially transformed version of all operations. These techniques allow replicas to always reflect all known updates and guarantee eventual replica convergence without the use of undo-redo techniques. These properties are particularly interesting in our peer-to-peer synchronization model, as they allow each replica to immediately integrate updates received from any other replicas.

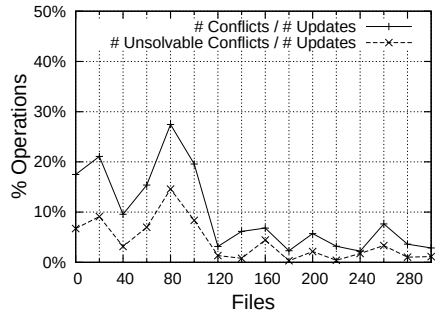

Figure 1. Gnuplot Project CVS log statistics (conflicts include all concurrent updates).

Besides adapting the OT algorithm for an asynchronous synchronization model, we have created a new set of transformations for text files. As in reconciliation solutions based on RCS (e.g. CVS), our solution also created multiple versions when conflicts arise. However, unlike those systems, it treats versions as first-class citizens, allowing file access to files with versions and allowing versions to evolve without creating additional versions. This is important in a system that uses peer-to-peer synchronization, as a user may want to delay file conflict resolution to a later time but he may still want to be able to observe the changes produced by other users or even to change other parts of the file (or even his version for a part that is in conflict).

In our poster, we will present the developed reconciliation techniques. As motivation for the need of a sophisticated reconciliation solution, we will also present some results from a CVS usage study (with results obtained from `sourceforge.net` that show that in environment where users share files, conflicts are much more frequent that reported in previous studies in distributed file system environments. Figure 1 exemplifies some of the results obtained – in this case, for files with more than 25 updates (the top 100 files), 10% to 30% of updates correspond to concurrent updates and 5% to 15% updates correspond to unsolved conflicts.

References

[1] C. Sun and C. A. Ellis. Operational transformation in real-time group editors: Issues, algorithms, and achievements. In *Proc. CSCW'98*, 1998.