

TRANSACTION CLASH MANAGEMENT IN A DISCONNECTABLE COMPUTER AND NETWORK

FIELD OF THE INVENTION

The present invention relates to the detection and resolution of inconsistent updates performed on disconnected computers, and more particularly, to clash handling during transaction synchronization when disconnectable computers are reconnected.

TECHNICAL BACKGROUND OF THE INVENTION

“Disconnectable” computers are connected to one another only sporadically or at intervals. Familiar examples include “mobile-link” portable computers which are connectable to a computer network by a wireless links and separate server computers in a wide-area network (WAN) or other network. Disconnectable computers can be operated either while connected to one another or while disconnected. During disconnected operation, each computer has its own copy of selected files (or other structures) that may be needed by a user. Use of the selected items may be either direct, as with a document to be edited, or indirect, as with icon files to be displayed in a user interface.

Unfortunately, certain operations performed on the selected item copies may not be compatible or consistent with one another. For instance, one user may modify a file on one computer and another user may delete the “same” file from the other computer. A “synchronization” process may be performed after the computers are reconnected. At a minimum, synchronization attempts to propagate operations performed on one computer to the other computer so that copies of items are consistent with one another.

During synchronization, some disconnectable computers also attempt to detect inconsistencies and to automatically resolve them. These attempts have met with limited success.

For instance, the Coda File System (“Coda”) is a client-server system that provides limited support for disconnectable operation. To prepare for disconnection, a user may hoard data in a client cache by providing a prioritized list of files. On disconnection, two copies of each cached file exist: the original stored on the server, and a duplicate stored in the disconnected client’s cache. The user may alter the duplicate file, making it inconsistent with the server copy. Upon reconnection, this inconsistency may be detected by comparing timestamps.

However, the inconsistency is detected only if an attempt is made to access one of the copies of the file. The Coda system also assumes that the version stored in the client’s cache is the correct version, so situations in which both the original and the duplicate were altered are not properly handled. Moreover, Coda is specifically tailored, not merely to file systems, but to a particular file system (a descendant of the Andrew File System). Coda provides no solution to the more general problem of detecting and resolving inconsistencies in a distributed database that can include objects other than file and directory descriptors.

Various approaches to distributed database replication attempt to ensure consistency between widely separated replicas that collectively form the database. Examples include, without limitation, the replication subsystem in Lotus Notes and the partition synchronization subsystem in Novell NetWare® 4.1 (LOTUS NOTES is a trademark of International Business Machines, Inc. and NETWARE is a registered trademark of Novell, Inc.).

However, some of these approaches to replication are not transactional. Non-transactional approaches may allow partially completed update operations to create inconsistent internal states in network nodes. Non-transactional approaches may also require a synchronization time period that depends directly on the total number of files, directories, or other objects in the replica. This seriously degrades the performance of such approaches when the network connection used for synchronization is relatively slow, as many modem or WAN links are.

Moreover, in some conventional approaches potentially conflicting changes to a given set of data are handled by simply applying the most recent change and discarding the others. In other conventional systems, users must resolve conflicts with little or no assistance from the system. This can be both tedious and error-prone.

Thus, it would be an advancement in the art to provide a system and method for detecting and handling inconsistent changes to copied items when two disconnectable computers are reconnected.

It would also be an advancement to provide such a system and method which are not limited to file system operations but can instead be extended to support a variety of database objects.

Such a system and method are disclosed and claimed herein.

BRIEF SUMMARY OF THE INVENTION

The present invention provides a system and method for handling clashes during the synchronization of operations performed on first and second disconnected computers. Each disconnected computer contains a replica of a distributed database. In one embodiment, the first computer is a mobile client computer and the second computer is a central server computer; in another embodiment, each computer is a server on a network.

Synchronization of the database replicas is performed after the computers are reconnected. Synchronization includes a “merging out” step, a “merging in” step, and one or more clash handling steps. During the merging out step, operations performed on the first computer are transmitted to the second computer and applied to the second replica. During the merging in step, operations performed on the second computer are transmitted to the first computer and applied to the first replica.

Some of the clash handling steps detect transient or persistent clashes, while other steps recover from at least some of those clashes. Persistent clashes may occur in the form of unique key clashes, incompatible manipulation clashes, file content clashes, permission clashes, or clashes between the distributed database and an external structure. Recovery may involve insertion of an update before or after a clashing update, alteration of the order in which updates occur, consolidation of two updates into one update, and/or creation of a recovery item.

In one embodiment of the present invention, operations performed on each computer are represented by entries in an update log kept on each computer. During recovery, the log may be accessed to help regress persistently clashing updates from the computer’s replica. Clash detection, recovery, and regression are performed recursively after an update regression to handle clashes caused by the update regression. Log management steps permit the identification of clashing updates in a log, the removal of a clash condition by use of a repairing update, and compression of the update log.