www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 03 Issue 07 July, 2014 Page No. 7070-7073

Implementation of Database Synchronization Technique between Client and Server

Naveen Malhotra[1], Anjali Chaudhary[2]

[1,2]Department Of Computer Science
Kurukshetra Institute Of Technology & Management Pehowa Road Kurukshetra,India
naveenmalhotra1367@gmail.com
anjali.cse@kitme.in

Abstract--*The objective of this paper is to provide an algorithm to solve the problem that when all clients are relying on a single server. If that database becomes unavailable due to planned server downtime or from server failures, all of the remote workers will be disconnected from their data. Data is stored on their system (user system). When the user connected to the internet data automatically sink from their client system to the server in serial order. It also works on file handling. When the system is disconnected from network all the files(images) uploaded by user,saved on client machine folder when it is again connected with the server,automatically files(images) transferred from client to server.*

Keywords—Web application, Web services, Data Synchronization, Offline mode, online mode.

I. INTRODUCTION

In computer science [1], synchronization refers to one of two distinct but related concepts: synchronization of processes, and synchronization of data. Process synchronization refers to the idea that multiple processes are to join up or handshake at a certain point, in order to reach an agreement or commit to a certain sequence of action. Data synchronization refers to the idea of keeping multiple copies of a dataset in coherence with one another, or to maintain data integrity. Process synchronization primitives are commonly used to implement data synchronization.

Thread or process synchronization

Thread synchronization or serialization, strictly defined, is the application of particular mechanisms to ensure that two concurrently-executing threads or processes do not execute specific portions of a program at the same time. If one thread has begun to execute a serialized portion of the program, any other thread trying to execute this portion must wait until the first thread finishes. Synchronization is

used to control access to state both in small-scale multiprocessing systems, in multithreaded environments, multiprocessor computers and in distributed computers consisting of thousands of units, in banking and database systems, in web servers, and so on.

Data synchronization

A distinctly different (but related) concept is that of data synchronization. This refers to the need to keep multiple copies of a set of data coherent with one another. Data synchronization is the process of establishing consistency among data from a source to target data storage and vice versa, and the continuous harmonization of the data over time.

Data synchronization technologies are designed to synchronize a single set of data between two or more devices, automatically copying changes back and forth. For example, a user's contact list on one mobile device can be synchronized with other mobile devices or computers. Data synchronization can be local synchronization where the device and computer are side-by-side and data are transferred or remote synchronization when a user is mobile and the data is synchronized over a mobile network.

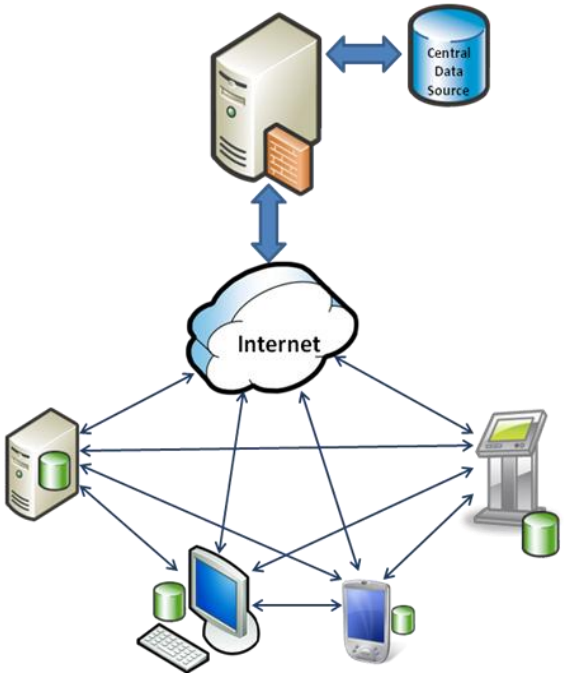

Fig. 1 Sync Framework Database Synchronization

As seen in Fig.1, a remote database is free to exchange information with any other database. This type of solution is useful when a team of people is working in remote locations and do not have access to a central database. These workers often need to share information amongst each other, but since they do not have connectivity to the central database they need to share information through some sort of peer-to-peer network.

Data synchronization is enabled through specialized software that tracks data versions as they are created and utilized. The process is implemented in distributed systems where data elements are routed between several computers or systems. Each computer may modify original data versions, depending on requirements.

Data synchronization ensures that regardless of data modifications, all changes are merged with the original data source. Data synchronization is also used in data mirroring, where each data set is exactly replicated or synchronized within another device.

Examples include:

- File synchronization, such as syncing a handheld MP3 player to a desktop computer.
- Cluster file systems, which are file systems that maintain data or indexes in a coherent fashion across a whole computing cluster.
- Cache coherency, maintaining multiple copies of data in sync, across multiple caches.
- RAID (redundant array of independent disks), where data is written in a redundant fashion across multiple disks, so that the loss of any one disk does not lead to a loss of data.
- Database replication, where copies of data in a database are kept in sync, despite possible large geographical separation.
- Journaling, a technique used by many modern file systems to make sure that file metadata are updated on a disk in a coherent, consistent manner.

Data Synchronization Techniques

The increasing decentralization of information raises the need to synchronize data across numerous devices and data storage locations. Synchronization processes comprise a 'source' and 'destination' entity and, based on data appearance, is categorized into unidirectional and bidirectional synchronization.

Unidirectional synchronization

Unidirectional synchronization [2] replaces all content of the destination entity with data from the source entity.

1. Back-up synchronization methods create replica/mirror files:
   - New and updated files are copied from source to destination entity.
   - New files added to the source entity are copied to the destination entity.
   - Deleted files in the source are also deleted from the destination entity.
2. If older versions of files in the destination entity need to be kept, an archiving feature can be enabled which includes that deleted files in the source entity are not deleted from the destination entity.
3. Consolidation does not keep track of file conflicts or deletions:
   - New files added to the source are copied to the destination and new files added to destination are copied to the source.
   - Deleted files in the source are copied back from destination entities and vice versa.
   - Updated files in the source copy over older files in the destination and vice versa.

Bidirectional synchronization

Bidirectional synchronization means two-way file synchronization merging data from source and destination entity:

1. New and updated files are copied both ways.
2. New files added to the source entity are copied to the destination and vice versa.
3. Deleted files in the source are deleted from destination entities and vice versa.
4. Updated files in the source are copied over older files in the destination entity and vice versa.
5. If a file changes in both entities, the file is in conflict and needs to be reconciled manually.

For fast synchronization, files are transparently stored in a cache. Changes in the cache get automatically invalidated or updated when used with a sparse cache and entries get inserted when used with a complete cache.

II RELATED WORK DONE

1) Data Synchronization Using Cloud Storage (2012)

In this paper [3] authors (Sudha S,Brindha K,Sai Vamsy Krishna S, Gokul K and Sanath Kumar M) had explained that Cloud computing usually consists of front-end user devices and back-end cloud servers. This gives users to access a large volume of storage on the cloud. In this paper, the user can upload files from mobile or PC to the cloud storage. These files will be automatically synchronized to the user's devices when they are connected to the internet. So, user files can be viewed from anywhere, from any device. In the existing system, we need to download files manually. This paradigm provides the user to synchronize data automatically between devices. They had implemented this paradigm for windows platform.

2) Solving Problems in Software Applications through Data Synchronization in Case of Absence of the Network (2012)
In this paper [4] authors(Isak Shabani, Betim Çiço and Agni Dika) had presented an algorithm for data synchronization based on Web Services (WS), which allows software applications to work well on both configurations "Online" and "Offline", in the absence of the network. For this purpose is in use Electronic Student Management System (ESMS) at the University of Prishtina (UP) with the appropriate module. Since the use of ESMS, because of a uncertain supply of electricity, disconnecting the network and for other reasons which are not under the control of professional staff that manages the performance of this system, has interruption to the online work. In order to continue working in such conditions, are founded adequate solutions to work in offline mode and later data synchronization in normal conditions.
3) Data Synchronization for Cognitive Load Estimation in Driving Simulator-based Experiments(2012)
In this paper [5] authors(Zeljko Medenica and Andrew L. Kun) explained that analyzing the effects of driver distraction and inattention on cognitive load has become a very important issue given the substantial increase in the number of electronic devices which are finding their way into vehicles. Typically the separate equipment is used for collecting different variables sensitive to cognitive load changes. In order to be able to draw reliable conclusions it is important to possess dependable ways of synchronizing data collections between different equipment. This paper offers one low-cost solution which enables synchronizing three types of devices often used in driving research: driving simulator, eye tracker and physiological monitor.
4) A Distributed Architecture for Transactions Synchronization in Distributed Database Systems (2010)
In this paper [6] authors (Arun Kumar Yadav and Dr. Ajay Agarwal) explained that various concurrency control algorithms have been proposed for use in distributed database systems. But, the number of algorithms available for the distributed concurrency control, come into one of three basic classes: locking algorithms, Timestamp algorithms and optimistic (or certification) algorithms. In this paper, we are presenting a Distributed Transaction Processing Model and an approach for concurrency control in distributed database systems. The analysis of our approach is a decomposition of the concurrency control problem into two major sub-problems: read-write and write-write synchronization. We describe a series of synchronization techniques for solving each sub-problem and will show how to combine these techniques into algorithms for solving the entire concurrency control problem. Such algorithms are called "concurrency control methods". Our approach concentrates on the structure and correctness of concurrency control methods and also the performance of such methods up to some extent.
5) The Impact of Data Synchronization Adoption on Organizations (2009)
In this paper [7] authors (Susan G. Zucker and Shouhong Wang,) had explained that Data synchronization is required for supply chain management in the B2B e-commerce environment. This case study examined the impact of the adoption of data synchronization on three large consumer product goods organizations. The study identified process and structural inadequacies that developed as the result of the implementation, as well as how these organizations recognized benefits and future opportunities after data synchronization adoption. The findings revealed the significance of internal alignment around data cleansing and accuracy, as well as opportunities for improved external alignment from a systems perspective. The synergy created between product item management, data synchronization, and internal champions existed at all three companies. The workflow re-design, process improvements and standards development imposed on these organizations by the clean data requirement of data synchronization provided the greatest benefits from the data synchronization process.
6) Synchronization in an Embedded DBMS Environment (2006)
In this paper [8] author (Sang-Wook, Kim) had explained that the embedded DBMS is a lightweight DBMS for effective management of quite small databases contained in tiny mobile devices. Synchronization is a core function of the embedded DBMS to preserve the consistency of data replicated in the server and client databases. This paper presents a framework for synchronization in embedded DBMS environment. His first address key issues for realizing synchronization, and then propose solutions to them obtained from our development. The main issues touched here are (1) classifying conflicts, (2) identifying changes in a client database, (3) detecting conflicts, and (4) resolving conflicts. The proposed framework would help reduce the trial-and errors of embedded DBMS developers in implementing their synchronization server.

III MODEL PRESENTATION

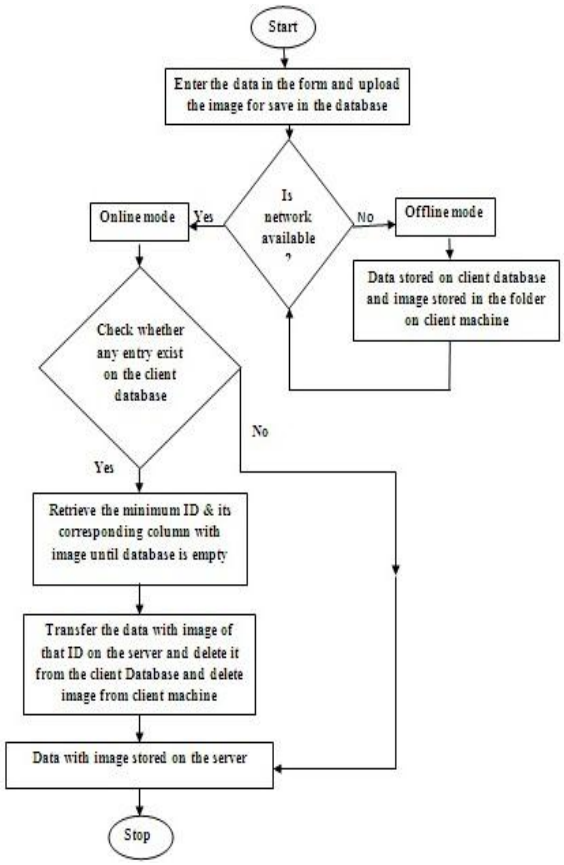

Fig. 2 Flow chart of data synchronization

As shown in Fig. 2 the steps for data synchronization algorithm are given below:-

Step 1. Enter the data in the form and upload the image to save in the database.
Step 2. Check the network connection either available or not.
Step 3. If a network connection is not available, we switch on the offline mode. Then data stored on client database and image stored in the folder on the client machine and again, it will continuously check whether the network connection is available or not.
Step 4. If a network connection is available, we switch on the online mode and it will perform the following operations:
Check whether any entry exists on the client database

1. If no entry exists on the client database, then data with images stored on the server.
2. If any entry exists on the client database, then retrieve the minimum ID & its corresponding column with the image until the database is empty.

Transfer the data with an image of that ID on the server and delete it from the client database and delete images from client machine.
Then data with images stored on the server.

IV CONCLUSION AND FUTURE SCOPE

The objective of the research is to provide an algorithm to solve the problem that when all clients are reliant on a single server. If that database becomes unavailable due to planned server downtime or from server failures, all of the remote workers will be disconnected from their data. Data is stored on their system (user system). When the user connected to the internet data automatically sink from their client system to the server in serial order.
In the future, application developers should work with serial of data sink should be in order between offline and online application. In our research when two different images from two different clients uploaded on a server then the first image replaced with another. In the future, researcher solved the problem of data synchronization with image uploading

ACKNOWLEDGEMENT

I would like to thank my parents and my friends for their support and trust.

REFERENCES

[1]en.wikipedia.org/wiki/Synchronization_(computer_science)
[2] Zach McCormick and Douglas C. Schmidt, Data Synchronization Patterns in Mobile Application Design, proceedings of the Pattern Languages of Programs (PLoP) 2012 conference, October 19-21, Tucson, Arizona.
[3] Sudha S and Brindha K "Data Synchronization Using Cloud Storage" International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 11, November 2012.
[4] Isak Shabani, Betim Çiço and Agni Dika "Solving Problems in Software Applications through Data Synchronization in Case of Absence of the Network" IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 3, January 2012.
[5] Zeljko Medenica, Andrew L. Kun "Data Synchronization for Cognitive Load Estimation in Driving Simulator-based Experiments" AutomotiveUI'12, October 17-19, 2012, Portsmouth, NH, USA
[6] Arun Kumar Yadav and Dr. Ajay Agarwal, "A Distributed Architecture for Transactions, 2010
[7] Susan G. Zucker & Shouhong Wang, "the impact of Data synchronization adoption on organizations: a Case study" Journal of Electronic Commerce in Organizations", Volume 7, Issue 3.
[8] Sang-Wook Kim, "Synchronization in an Embedded DBMS Environment" IJCSNS International Journal of Computer Science and Network Security, VOL.6 No.7A, July 2006.
[9] Thorsten Sch¨utt, Florian Schintke, Alexander Reinefeld "Efficient Synchronization of Replicated Data in Distributed Systems"
[10] Wu Jie-Ming, Yu Li-ping "Research and Design of Smart Client Data Synchronization Engine" Proceedings of the 2009 International Workshop on Information Security and Application (IWISA 2009)Qingdao, China, November 21-22, 2009.
[11] Zhen Fang, Lixin Zhang &John B. Carter "Fast Synchronization on Shared-Memory Multiprocessors:An Architectural Approach".
[12]"http://msdn.microsoft.com/enus/library/bb902827.aspx", accessed on March 2, 2014.
[13]"https://developer.appcelerator.com/question/131168/sync-local-andserver SQL-database---ant-tools-out-there-or-best-practices" accessed on March 2, 2014.