Available online at http://www.journalcra.com

INTERNATIONAL JOURNAL OF CURRENT RESEARCH

International Journal of Current Research
Vol. 6, Issue, 01, pp.4754-4756, January, 2014

ISSN: 0975-833X

REVIEW ARTICLE

SYNCHRONIZATION WITH OCCASIONALLY CONNECTED MOBILE DATABASES

[1]*China Ramu S. and [2]Dr. Premchand P.

[1]Department of CSE/IT, C.B.I.T., Hyderabad, A.P.
[2]Department of CSE, Osmania University, Hyderabad, A.P.

ARTICLE INFO

Article History:
Received 05th October, 2013
Received in revised form
24th November, 2013
Accepted 19th December, 2013
Published online 31st January, 2014

Key words:
Occasionally Synchronized Database systems (OSDBs),
Relational DataBase Management Systems (RDBMS).

ABSTRACT

Considering the group of mobile computing applications with occasionally connected clients and are wish to share data, nevertheless, due to the cost of mobile communication, they only connect occasionally. Usually a constant server connection, having a collective of client data, makes easy sharing between clients by permitting the clients to download updates submitted by other clients and upload local updates. Then the server computes and retransmits these updates on a client-by-client basis. As a result, limiting scalability because the difficulty of these operations is on the order of the number of clients. In this paper, focusing on developing a group basis approach that notably improves the scalability of the resources to synchronize data. Since synchronization is a problem with occasionally connected mobile databases. A server from time to time generates update files for each client, which are downloaded and applied when convenient. Miserably, the time required to synchronize clients in this approach increases notably with increase in client population. We show that this tendency could be changed by by grouping, update operation sharing is computed only once per group, irrespective of the number of clients. As the mobile devices will have limited memory capabilities we effectively utilize the memory of mobile device to store the data without using any backend software. When the mobile database is connected to the server it will have less session time. Therefore by taking session time into consideration the connection establishment and data transfer is to be done.

INTRODUCTION

Data access is no longer restricted to the office atmosphere, so occasionally Synchronized Database systems (OSDBs) support mobile database activities by allowing clients to work on locally replicated data without a constant connection to the server. Therefore sharing data with only occasional connectivity is suited for applications that need high ease of use but not essentially strict consistency among shared data values (e.g., mostly in consumer goods, sales, and pharmaceuticals etc. The server maintains the primary database, which is partitioned into publications. By subscribing to a publication, the client makes a local replica of the published data and from time to time downloads updates for it from the server via update files. Clients carry a part of the database for their own processing and accumulate local updates while disconnected. Since the connectivity is intermittent, the clients maintain a copy of the shared data in their local database. Updates are made to this local database whenever the client connects to server. The server gathers updates that clients send to it in update files. From time to time, the server responds by generating update files for each and every client containing the relevant subsets of these updates. When convenient, clients download these files and apply the up-dates to their local replicas, and so on. (RMDE 2003; SSICDC 2005; Ziad Itani *et al.*, 2005) The clients can maintain replica of the data at the server when it connects to the server. Although practical, this type of synchronization results in performance that is un-scalable with an increasing client population. In particular, the server may become a performance bottleneck as the number of clients grows

Mobile Databases

The most important benefit of using a mobile database is offline access to data in other words, the ability to read and update data without a network connection. This helps avoid problems such as dropped connections, low bandwidth. Mobile computing gives users the chance to access data that is stored in a stationary database or some other data repository at any time and any place.

The primary limitations that have to be taken into account in case of mobile environment are:

- Resource-poorness of mobile devices compared to static computers.

***Corresponding author: China Ramu S.**
Department of CSE/IT, C.B.I.T., Hyderabad, A.P.

- Mobile connectivity is highly uneven in performance and trustworthiness. These issues are responsible for times of disconnected operations.

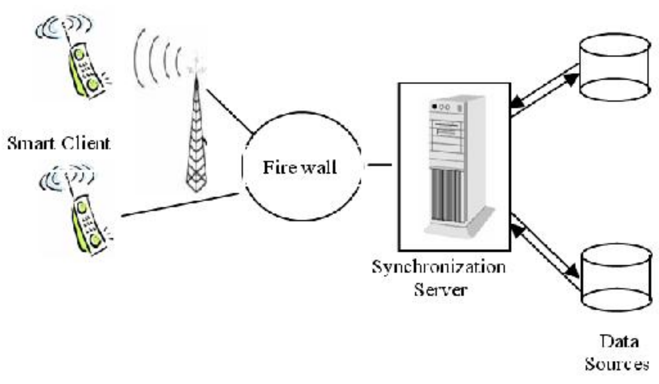


Figure 1. General architecture of Mobile Database

Synchronization Need

Synchronization is the process of making consistency. We are concerned in occasionally connected applications that synchronize with the back-end database. The following scenarios will show how occasionally connected clients are exceptionally useful in mobile computing,

- A sales agent may require to place a large order while on site with the customer, where the representative cannot connect to the server. Sales agent may need to consult details like cost inventory and record information, enter all order data, and provide estimates of delivery and concessions without connection.
- An insurance agent may require to generate a fresh insurance policy while out of the office. Agent may be required to fill all the appropriate information, estimate premium, and issue policy details with no connection to the systems in the office.

The following Figure 2 shows basic communication flow to synchronize between SyncML client and server. First client sends initialization message to the server and server sends ok message to the client. Next client sends modifications and mapping of id’s to the server, it receives modifications sync engine will do the modifications and mapping at the server and sends ok message to the client

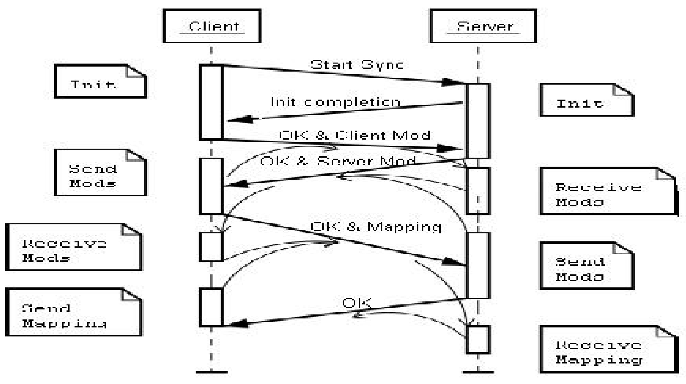


Figure 2. Communication flow between Client and Server

Grouping Problem

The precise combination in use by the server very much effects the overall effectiveness of the system. So to establish the preferred grouping, we have to know the information applicable to all clients. We wish to design groups that increase the performance of the system. The creation of a grouping speaks about to breakup design

To carry out data-centric as a group, we require:

- a set of clusters, C_G, and
- A mapping of clients to groups, f (i).

Assume that $X = X_1, X_2, \dots, X_N$ set of parts describing the data associated with group j, and assume that $f(i) \rightarrow Y$ where f(i) is a function determining the groups related by means of client. Clients subscribe the usual OSDB systems which permit supervisors to describe vertical and parallel divisions of tables of a RDBMS. Sybase shows this procedure as using data qualifications like the where clause in a SQL select command, building fine-granularity replication possible (Gorelik and Wang 1994)

BACKGROUND WORK (Server Side Design)

Data Group design

The server maintains the primary database, which is partitioned into publications. Each update file is defined by a data group. A datagroup is a set of publications, and thereby determines the contents of an update file. Specifically, each update is compared to each publication in each datagroup. If an update modifies a datagroup's publication, then the update is written to the datagroup's update file. In general, there is no constraint to the number of contents of datagroups. By subscribing to a datagroup, the client will download the update files based on it. In commercial OSDB systems, a datagroup is created for each client. The publications in each datagroup are exactly those that the client desires, so each client downloads the minimal necessary updates. We call this datagroup design as client-centric design.

The problems with the client centric approach are

- As the client population raises the memory required is also increases,
- Under Client centric approach, a similar type of unnecessary duplication of effort occurs in the network because the same data, contained in different update files.

Instead of client centric approach go for data centric approach, in which the datagroup is generated according to the data. The clients will subscribe to the datagroup according to their requirement.

Grouping Estimate Algorithm

Grouping Estimate Algorithm is accountable for identifying the set of shared clusters derived through increasing on both sides of tree until sharing falls less than entry level.

```
Step 1. Generate a graph, G (V,E), where V =1, … , M and E= {(i, j) | (i, j) ∈ V) and i ≠ j. Also for each edge, (i, j), allot the weight $W_k$
Step 2. Locate an edge, (i, j), with Max(W(i,j
Step 3. Combine nodes i and j in G .
    • Construct a new node, N, where $C_N$= $C_i$ ∩ $C_j$, and remove (i, j) from E.
    • ∀ edge occurrence to node i or j, let h be the other endpoint of the edge (h ∉ {i, j}). ∀ h, create a new edge,(h, N), with weight W(h, N).
    • Eliminate nodes i and j, add N to V, and delete all edges from E with i or j as an endpoint.
Step 4. If E = empty, go to Step 6, or else, find an edge, e, occurrence to N with highest weight.
Step 5. If W (h, N) > threshold then go to Step3
Step 6. Let N be the set of nodes combined in the source. Create a cluster with fragments.
Step 7. ∀clients, i, which subscribe to several fragment in Cluster $C_f$ , set f(i) by combining with $C_f$ (i). Where f(i) is function determining the group related by means of client
Step 8. Eliminate the fragments in $C_f$ from all clients.
Step 9. ∀i, ∍ $C_i$ ≠ empty, repeat Step1 thro Step9.
```

Experimental Work

Our experiments are designed to show the relative performance in terms of processing of server, storage, and transmission costs with respect to data-centric grouping to synchronize the data. The trial purpose we required J2ME Wireless Toolkit to achieve the goal of synchronization with mobile clients to the server, for which we have used java enabled server and client. The J2ME Wireless Toolkit is a wide-ranging set of tools for developing mobile applications with which, we developed a mobile client which is connected to the server. At the server end we have used MySQL as backend to store the data. First we have developed an application, which is going to store the data in a table format at the client side without using any backend software. By using Record Store method in J2ME. The mobile devices will have less session time, within that session time the data must be updated. Number of mobile devices will connect to the server by using ID mapping.

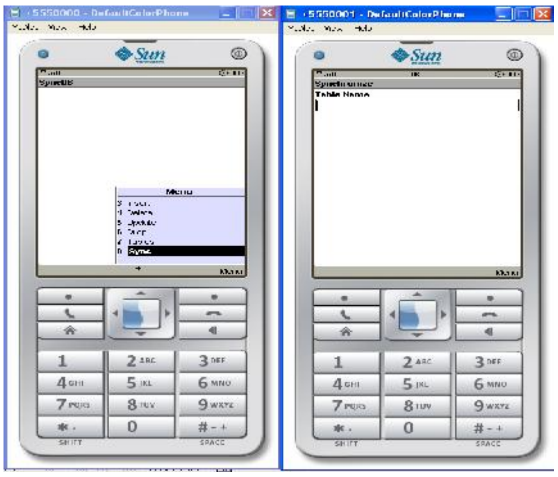
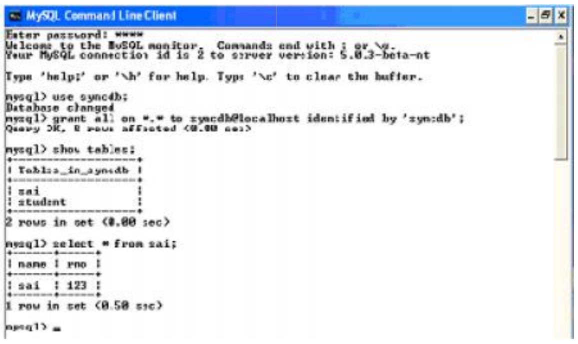


Figure 3. To synchronize the data with the server



```
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2 to server version: 5.0.3-beta-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use syncdb;
Database changed
mysql> grant all on *.* to syncdb@localhost identified by 'syncdb';
Query OK, 0 rows affected (0.00 sec)

mysql> show tables;
+------------------+
| Tables_in_syncdb |
+------------------+
| sai              |
| student          |
+------------------+
2 rows in set (0.00 sec)

mysql> select * from sai;
+------+-----+
| name | rno |
+------+-----+
| sai  | 123 |
+------+-----+
1 row in set (0.50 sec)

mysql> _
```

Figure 4. To synchronize the data with the server (contd.)

Conclusion

As the client population increases synchronization is the major problem with occasionally connected mobile databases, this could be handled with data-centric approach. in data-centric approach we proposed grouping model of data, which will reduce greatly cost of resources such as server processing, bandwidth, and storage space. We proposed an estimation algorithm, which is accountable for identifying the set of shared clusters derived through increasing on both sides of tree. The data is replicated to the server from the client and vice versa, without having any duplicate data at the server as well as at client. Here we propose mobile device as a mobile database which will have limited memory capability, so that without using any backend database in the mobile device for storing the data.

Acknowledgement

The first author expresses his gratitude to Dr. Y Rama Devi, Professor, Dept. of CSE, CBIT for her support and also take the opportunity to express thanks to Management of CBIT, and Dr. B. Chenna Kesava. Rao, Principal, CBIT for their encouragement and cooperation.

REFERENCES

Efficient Pull Based Replication and synchronization for Mobile Databases: Ziad Itani, Hassan Diab, Hassan Artail, 2005 IEEE.

Fault-Tolerant Master–Slave Synchronization for Lur'e Systems Using Time-Delay Feedback Control Maiying Zhong and Qing-Long Han.

Gorelik A., Y. 1994. Wang Sybase replication server. Proceedings of ACM, International conference.

Replication in Mobile Database Environments: A Client-Oriented Approach Christoph Gollmick, International Workshop on Database and Expert Systems Applications, 2003 IEEE.

Scalable Synchronization of Intermittently Connected Database Clients: Wai Gen Yee and Ophir Frieder, 2005 ACM.

****