CONCURRENCY CONTROL STRATEGIES FOR TCMCS

CHITRA MANIKANDAN
THIAGARAJAR School of Management
Madurai, India – 625 005.

R.VARADHARAJAN
SASTRA
India

ABSTRACT

Mobile distributed computing is a newly emerging computing paradigm of the future. The broadcast-based data dissemination, in mobile distributed computing systems, poses new challenging issues on data consistency of transaction processing. Although data broadcast has been shown to be an efficient method for disseminating data items in mobile computing systems, the issue on how to ensure consistency and currency of data items provided to mobile transactions (MT), which are generated by mobile clients, has not been examined adequately. While data items are being broadcast, update transactions may change the old values into new values. Since at the time of broadcasting an item 'A', the value of 'A' is updated at the broadcasting server, without any control, mobile transactions generated by mobile clients may receive inconsistent data values. Among the concurrency control techniques providing data consistency in mobile data broadcast environments, Ordered Update First with Order (OUFO) algorithm has been shown to be an efficient and has minimal impact on server, client and the broadcast schedule. This paper addresses the issue of ensuring consistency and currency of data items required by read-write time constrained mobile transactions in data broadcast. We propose a mobile database system that uses a modified OUFO algorithm for concurrency control among time constrained read-write mobile transactions, also called as Time Constrained Mobile Computing Systems (TCMCS), in the broadcast environment. The mobile transactions are assumed to be read-write time constrained mobile transactions. The system discussed in this paper, provides site autonomy between the mobile clients and the server with minimum upstream communication, data consistency and currency that are desirable features to the scalability of applications, which are running in broadcast environments.

KEY WORDS: Data Consistency, Currency, Mobile Transactions

1. INTRODUCTION

In the recent years technical advances in the development of portable computers and the rapidly expanding cordless technology have provided portable computers with wireless connections that permit users to actively participate in distributed computing even while moving. The research in mobile computing systems has received a lot of interest in recent years. A key element in many of these mobile computing systems is the need for distributed real-time information from a database server to mobile clients [3]. The main problem on the mobile internet is the scalable dissemination of information. This problem is particularly acute with the presences of mobile users. The resulting distributed mobile environment is subject to restrictions imposed by the nature of the wireless medium, and the resulting mobility of users.

Recent research has focused on one of the most important issues: supporting efficient data retrieval and dissemination to the requests from mobile clients. The performance objective is to reduce the mean access delay for data items and minimize the response times of the requests from the mobile clients. Generally speaking, the proposed mechanisms for data dissemination to the requests from mobile clients fall into one of two main approaches: the *on-demand* approach and the *broadcast* approach [2] (also called the *pull* and *push* approach respectively).

In the *on-demand* approach, the data items required by the requests from mobile clients will be sent from the information server on request [2]. This approach is simple but does not scale well with the number of mobile client requests. In the *broadcast* approach, the information server periodically and continuously broadcasts data items to the mobile clients [2]. If there is a request waiting for a data item, it will get the data item from the "air" while it is being broadcast [6]. Under the broadcast approach, the cost for data dissemination is independent of the number of requests since a broadcast can satisfy multiple requests for the same data item, resulting in much more efficient utilization of the bandwidth. In recent years, many efficient data dissemination methods have been proposed particularly for read-only mobile transactions. Many of them are based on data broadcast or on-demand transmission. In this paper, we focus on the data dissemination problem using the broadcast approach.

The issue of how to provide consistent data items to mobile transactions in addition to minimizing data access delay has received growing interest in

recent years. Unfortunately, conventional concurrency control protocols cannot be used for mobile computing systems as the overheads for setting locks and detecting data conflicts [4] in a mobile environment could be very expensive[2]. Therefore, new concurrency control methods, which take into considerations of the data properties[4] and the characteristics of the systems, are needed.

In this paper, we study the problem of broadcasting consistent data items to ordered *read-write mobile transactions* while allowing updates to be performed concurrently at the broadcast server. It is assumed that each mobile transaction is associated with a deadline on their completion times. Hence it is called as Time Constrained Mobile Computing Systems (TCMCS). The mobile transactions should be completed within the deadline, otherwise, the usefulness of completing a transaction will be greatly affected.

2. SYSTEM MODEL

The mobile computing system model contains a broadcast server, many number of mobile clients, a server database, and a mobile network. The broadcast server does the update process the updated value of the data item is committed at the database of the broadcast server as shown in Figure 1. Then the broadcast process of the server broadcast the data items to a large amount of mobile clients. The broadcast server communicates with the mobile clients through low bandwidth wireless channels of the mobile network. The values of the data items in the server are changing frequently since they record real-time information in the system, e.g., weather at the locations of moving objects. The target applications are mobile information systems, which require the retrieval of real-time information for decision - making, i.e., mobile stock information systems, navigation systems and location management of moving objects. In such kind of systems, reading out -dated data items will significantly affect the usefulness of the information. For example, in a mobile auction problem, the current bid rate is 500$, the out-dated value of the bid rate is 350$, if the user is getting 350$ as bid rate value then it will be useless to the mobile client.

We assume that a well- formed concurrency control protocol, such as two phase locking, is used for concurrency control amongst the update transactions at the broadcast server. The broadcast server periodically broadcasts data items one by one, continuously, until the end of a broadcast cycle. Then, the next broadcast cycle starts immediately. Note that the proposed OUFO method can be applied to many different broadcast scheduling algorithms without any impact.

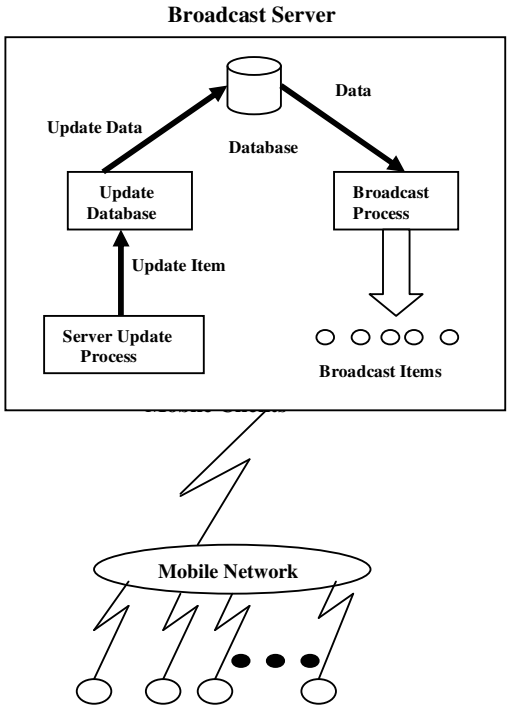

FIGURE 1. SYSTEM MODEL

MutualConsistency

Concurrency control amongst update transactions is to be done by a well-formed concurrency control protocol such as Two-Phase Locking, we have to consider the inconsistent problem of data conflicts between mobile transactions and update transactions. Our proposed protocol will concentrate on resolving this type of data conflicts to ensure that all mobile transactions will read consistent data values. In this paper, we adopt serializability as the correctness criterion of database consistency as it is widely accepted in the database community. If the serialization graph of a set of transactions is acyclic, then the schedule is serializable.

Currency

The Important characteristic of the mobile database systems is that the values of the data items are highly dynamic and the arrival rates of the updates can be very high. While data items are being broadcast from the database server to the mobile clients, updates of data items are being installed into the database. In such a dynamic environment, it is difficult to maintain a strict consistency between the external environment and the corresponding values of the data items in the database. This is especially true in the case of distributed database system. For example, in a weather monitoring system, updates are generated at the weather monitoring place database server. For most cases, stale information is much less useful to mobile clients. In order to minimize

the staleness of the information, the system has to process update transactions as soon as possible and broadcast the most up-to-second information, e.g., the latest committed value of a data item. Since providing the most updated values of data items in a mobile network can be very expensive, an alternative is to bound the degree of "staleness" of the data items within a pre-defined bound. For some data items, e.g., weather information, the data values will be totally useless if they are "older" than a certain pre-defined time bound.

3. OUFO

The *Update First with Order (UFO)* method is used for broadcasting consistent data items to mobile transactions with unordered read operations, i.e., each mobile transaction consists of a set of read operations and the operations can be executed in any order. In this paper, we propose an extension of UFO, which we call *Ordered UFO (OUFO)* for mobile transactions in which the read operations are ordered such that the read operations in a transaction have precedence relationships and their execution has to satisfy the defined precedence rules. For example, the second operation in a transaction can only start after the completion of the first operation of the transaction. The basic principle used in OUFO to ensure data consistency is that if a data conflict occurs between a broadcast transaction (BT) and an update transaction (U), the serialization order between them will always be $U \rightarrow BT$.

Since mobile transactions (MT) read data items from broadcast transactions, the serialization order between update transactions and mobile transactions will always be $U \rightarrow MT$. Thus the schedules will always be serializable.

3.1 Execution of update Transactions

The execution of an update transaction is divided into two phases: the execution phase and the commit phase. During the execution phase, the operations of an update transaction are executed and data conflicts with other update transactions are resolved using a conventional concurrency control protocol such as 2PL. The new values from the write operations of a transaction are written in a private workspace of the transaction instead of writing into the database immediately. When all the operations of the update transaction have been completed, it enters the commit phase in which permanent updates of the database will be performed by copying the new values from its private workspace into the database. Data conflict with the broadcast transaction will be checked in the commit phase, which is performed in a critical section. So, we can see that the update transactions adopt 2PL for resolving data conflicts with other update transactions and use an optimistic approach to detect conflicts with the broadcast transaction. There are two important advantages in dividing the execution of the update transactions into two phases. Firstly, it can significantly reduce the blocking probability and delay of the broadcast transactions. If a broadcast transaction wants to read a data item, which is already locked by an update transaction during its commit phase, the broadcast transaction will be blocked until the update transaction is committed based on the principles of 2PL. At the same time, the update transaction, which is holding the lock, may be blocked due to data conflicts with other update transactions. Due to transitive blocking, the blocking time of the broadcast transactions can be very long. Dividing the execution of an update transaction into two phases can greatly reduce the blocking probability and blocking time of broadcast transactions since data conflicts between update and broadcast transactions will occur only when the update transaction is in the commit phase, which is much shorter. Secondly, the detection of data conflicts between the update and broadcast transactions will become much easier. At the commit phase, the system will know which data items have been accessed by the update transaction. By comparing the write set of the update transaction with the read set of the broadcast transaction, the system can easily determine whether there is any data conflict between them.

3.2 Conflict Resolution between Update and Broadcast Transactions

Data conflict between an update transaction and a broadcast transaction is detected when the update transaction enters its commit phase. Re-broadcast is used to resolve the conflict by reversing the serialization order from BT -> U to U -> BT. The details of the algorithms at both the server and mobile clients are shown in the following sections.

Algorithm at the Server:

if (update transaction commit)

If $O_{BT} \cap O_U = \{ \}$

then

BT and U have no dependency

else

for each data item $d_i \in \{ O_{BT} \cap O_U \}$

re-broadcast data item di

endif

endif

where O_{BT} = set of data items of broadcast transaction BT.

O_U = set of data items of update transaction U.

By re-broadcasting the conflicting data item, the serialization order between the broadcast transaction and the update transaction is reversed. The set of data items in the broadcast transaction consists of those data items, which are broadcast in the period from to current time. Thus, the set of data items in the broadcast transaction is not fixed. After the broadcast of a data item, the data item will be included and the last data item in the broadcast transaction will be removed. And also when a *new item is added*, in the database of the broadcast server, at the time of broadcast, after the completion of the broadcast cycle, the newly added item is included in the broadcast list.

Algorithm at the mobile client

The data items requested by a MT are represented by a sequence of read operations. Processing of a MT starts from the first read operation in the sequence. Each data item received from a BT is matched with the requesting data item of the executing operation. If there is a match, the MT will read the data item and the operation will be processed. The process is repeated for the next read operation until there is no more operation in the sequence. In case a data item, which is already read, is re-broadcast while the MT is waiting for other data items, the MT will be restarted from the operation which requests that data item. It will use the re-broadcast value for the execution of the operation. There are two reasons for the restart. Firstly, it is to ensure the serializability order U -> MT. The second reason is to provide the most updated data values to the mobile transactions. The algorithm at the mobile client is shown below where it is assumed that the MT reads all its data items from BT and there is no cache at the client.

w_c = the data item required by the first operation in W_{MT}

d_c = the data item required by the first operation in L_{MT}

S_{MT} = { }

until L_{MT} exhausted

loop

read the value of the data item d_i from BT

if $d_i = d_c$

then

MT processes d_i and updates L_{MT}

if $d_i = w_c$

then

Write w_c in the client cache

Update w_c in the client cache

Write w_c at the server (like the update transaction of the server)

MT updates W_{MT}

endif

else

if di ∈ SMT

MT repeats the processing on d_i , updates L_{MT}

Restarts its execution from the operation, which requires d_i

endif

endif

d_c = the data item requested by the next operation in L_{MT}

w_c = the data item requested by the next operation in W_{MT}

end loop

where W_{MT} = sequence of write operations in MT

L_{MT} = sequence of read operations in MT

S_{MT} = set of data items have been accessed by MT

4. IMPLEMENTATION

Here, We discuss the implementation and characteristics of our proposed algorithm. This algorithm does not require any changes in the mobile clients except in getting re-broadcast data items. The main overheads of our algorithm are:

- Division of the execution of update transactions into two phases
- Checking of the data sets of a broadcast transaction and an update transaction whenever an update transaction wants to enter the commit phase
- Re-broadcast of conflicting data items if these items are broadcast, before the start of the commit phase of the update transaction.

Dividing the execution of update transactions into two phases is trivial and should not incur additional overhead. The overhead is for checking conflicting data sets. To speed up the checking process, the data items, to be updated, may be sorted according to their Ids.

The main overhead of this is data re-broadcast. The number of re-broadcast depends on the probability of data conflict which in turn depends on the broadcast schedule, arrival rate and the update pattern of the update transactions. The algorithm at the mobile clients is simple and does not incur much additional

overhead for checking. The only additional work is to replace the old version of a
data item in case it is re-broadcast.

The main advantages of our proposed algorithm are summarized below.

Consistency: It ensures that the execution schedules Of the transactions are serializable.

Currency: It can maximize the currency of the data items read by mobile transactions by re-broadcast. In general, it can only provide the committed values at the last broadcast cycle as all the database updates are performed at the end of a broadcast cycle.

Broadcast algorithm: It can be applied with many other broadcast algorithms, which may only broadcast a subset of the data items in the database.

5. CONCLUSIONS

Data broadcast has been shown to be an efficient method for disseminating data items to transactions generated by mobile clients. Although the research in the design of broadcast algorithms has received a lot of attention in previous years, the concurrency control issue has been largely ignored. If broadcast of data items and execution of update transactions (which are important for maintaining the validity of the data items at the database) are uncontrolled, mobile transactions may observe inconsistent data values. The issue of concurrency control between update transactions and mobile transactions, which consist of ordered read-write operations ared discussed in this paper. *Ordered Update First with Order (OUFO) for read-write mobile transaction*, is not only simple, but can also maintain the schedule between update and mobile transactions to be serializable and at the same time maximize the currency of the data items observed by the mobile transactions. OUFO for read-write mobile transaction can be implemented easily in most mobile broadcast systems.

6. REFERENCES

[1]. Acharya, S., Alonso, R., Franklin, M., Zdonik, S., Broadcast Disks: Data Management for Asymmetric Communications Environments, in *Proceedings of ACM SIGMOD*, 1995.

[2] Acharya, S., Franklin, M., Zdonik, S., Balancing Push and Pull for Data Broadcast, in *Proceedings of ACM SIGMO*D, Tucson, Arizona, May 1997.

[3] Kam-Yiu Lam, Edward Chan, Hei-Wing Leung and Mei-Wai Au, Concurrency Control Strategies for Ordered Data Broadcast in Mobile Computing Systems, to appear *in Information Systems*.

[4] Bernstein, P.A., Hadzilacos, V., Goodman, N., *Concurrency Control and Recovery in Database System*, Addison-Wesley Publishing Company, 1987.

[5] Fernandez, J. and Ramamritham, K., "Adaptive Dissemination of Data in Asymmetric Communication Environment", to appear in *ACM Mobile Networks and Applications Journal*.

[6] Pitoura, E.(1998) Supporting Read-Only Transactions in Wireless Broadcasting. Proceedings of the *DEXA'98 Workshop on Mobility in Databases and Distributed Systems* , 26-28 August, p.428-433. IEEE Online Publications.

[7] Hu, Q., Lee, D.L., and Lee, W.C. (1998) A Comparison of Indexing Methods for Data Broadcast on the Air. *Proceeding 12th International Conference on Information Networking*, 21-23 Jan, p.656-659. IEEE Online Publications.

[8] Hameed, S. and Vaidya, N.H. (1997) Efficient Algorithms for Scheduling Single and Multiple Channel Data Broadcast. *Technical Report* 97-002, Department of Computer Science, Texas, A&M University, Feb.

[9] Haug, T. Overview of GSM: Philosophy and Results. *International Journal of Wireless Information Networks*, vol. 1, no. 1, p.7-16, (1994).

[10] Imielinski, T. and Badrinath, B.R. (1994) Mobile Wireless Computing: Challenges in Data Management. Communications of the ACM, vol. 37, no. 10, Oct, p.18-28. Association for Computing Machinery.

[11] Kayan, E. and O. Ulusoy, An Evaluation of Real-Time Transaction Management Issues in Mobile Database Systems. *The Computer Journal,* vol.42, no.6, (1999).

[12] Leong, H.V. and Si, A. (1995) Data broadcasting strategies over multiple unreliable wireless channels. Proceedings of the *4th International Conference on Information and Knowledge Management*, Baltimore, MD USA, 28 Nov-2 Dec, p.96-104.

[13] Leong, H.V. and Si, A ,. Database Caching over the Air-Storage. *The Computer Journal* , Volume 40, number 7, p.401-415, (1997).

[14] Ozsoyoglu G. and Snodgrass, R. (1995) Temporal and Real-Time Databases: A Survey., *IEEE Transactions on Knowledge and Data Engineering* vol. 7, no. 4, p.513-532.

[15] Pitoura, E. and Bhargava, B. *Technical Report* Dealing with Mobility: Issues and Research Challenges., Purdue University, Nov (1993).

[16] Pitoura, E. and Bhargava, B. (1995) Maintaining Consistency of Data in Mobile Distributed Environments. Proceedings of the *15th International Conference on*

Distributed Computing Systems, 30 May-2 Jun, p.404-413. IEEE Online Publications.

[17] Pitoura, E.(1998) Supporting Read-Only Transactions in Wireless Broadcasting. Proceedings of the *DEXA'98 Workshop on Mobility in Databases and Distributed Systems* , 26-28 August, p.428-433. IEEE Online Publications.

[18] Pitoura, E. & Chrysanthis, P.K. (1999) Scalable Processing of Read-Only Transactions in Broadcast Push. Proceedings of the *19th IEEE International Conference on Distributed Computing System*, 31 May-4 Jun, p.432-439. IEEE Online Publications.

[19] Pitoura E. and Chrysanthis, P.K. (1999) Exploiting Versions for Handling Updates in Broadcast Disks. *Proceedings of International Conference on Very Large Data Base*, Edinburgh, UK, 7-10 Sep, p. 114- 125.

[20] K. Ramamritham, P. Deolasee, A. Katkar, A. Panchbudhe, and Prashant Shenoy (2000) Dissemination of Dynamic Data on the Internet. Proceedings of *International Workshop on Databases in Networked Information Systems*, Un iversity of Aizu, Japan, December, p. 173-187.

[21] Jayavel Shanmugasundaram, Arvind Nithrakashyap, Rajendran Sivasankaran and Krithi Ramamritham (1999) Efficient Concurrency Control for Broadcast Environments. Proceedings of *ACM SIGMOD International Conference on Management of Data*, Philadelphia, Penn., 1-3 June, p. 85-96.

[22] Srinivasan, R., Liang C., Ramamritham, K. (1998) Maintaining Temporal Coherency of Virtual Data Warehouses. *Proceedings of Real Time Systems Symposium*, Madrid, Spain, 2-4 Dec, p.60-70.

[23] Ulusoy, O. (1998) Real-Time Data Management for Mobile Computing. *Proceedings of International Workshop on Issues and Applications of Database Technology* (IADT'98), Berlin, Germany, p. 233- 240.

[24] Wolfson, O, Sistla, A.P., Chamberlain, S. and Ye sha, Y, *Distributed and Parallel Databases*. Updating and Querying Databases that Track Mobile Units., vol. 7, p.257-287, (1999).

[25] Xuan, P., O. Gonzalez, J. Fernandez & Ramamritham, K. (1997) Broadcast on Demand: Efficient and Timely Dissemination of Data in Mobile Environments. Proceedings of *3rd IEEE Real-Time Technology Application Symposium*, 9-11 Jun, p. 38 – 48.