Adaptive Valid Period Based Concurrency Control without Locking in Mobile Environments

Mohammed Khaja Nizamuddin* and Syed Abdul Sattar

[1] Associate Professor, DCET, Hyderabad, India
mknizams@yahoo.com
[2] Professor, Royal Institute of Technology and Science, Chevella, India
syedabdulsattar1965@gmail.com

Abstract. In a mobile computing environment, clients can access data irrespective of their physical location. Data is shared among multiple clients and can be updated by each client independently. This leads to inconsistency of the data. Due to limitations of mobile computing environment traditional techniques cannot be used. Several concurrency control techniques are proposed in literature to prevent data inconsistency. In this paper we first analyze the existing scheme of concurrency control without locking and justify its Performance limitations. A new scheme is proposed which adaptively set the validity period of the cached data items based on the number of data items required and current load of the database server. Experimental results show performance benefits and increase in commit rate of the transactions *abstract* environment.

Keywords: Absolute Validity Interval, Valid Period, semaphore, data count, Mobile Host, Fixed Host.

1 Introduction

Concurrency control is one of the essential characteristics of transaction management to ensure consistency of database. In mobile computing environment, data management becomes a challenging issue due its limitations i.e. variable bandwidth, frequent disconnections, limited resources on mobile host etc [6]. Several valuable techniques are proposed in literature to provide concurrency control in mobile environments, however most of them are based on locking, time stamp and optimistic concurrency control.

In reference [9], a new concurrency control technique is proposed without using locks. In this scheme Absolute validity interval (AVI) value is used to achieve concurrency control [3]. This scheme has a problem of predicting new AVI value of the data item based on the update history. In this paper we justify that predicting new AVI value based on history may result in abortion of the next transaction. Several performance limitations of the scheme is also mentioned. In

* Corresponding author. Research scholar, Rayalaseema University, Kurnool, India.

N. Meghanathan et al. (Eds.): NeCoM, WiMoN, and WeST 2010, CCIS 90, pp. 349–358, 2010.

this paper we also proposed a scheme for achieving concurrency control in mobile environments by eliminating limitations of the existing scheme and increasing commit rate of the transactions.

The rest of the paper is organized as follows. Section 2 reviews existing concurrency control schemes. Section 3 describes mobile database environment. In Section 4 we discuss performance limitations of the existing lockless scheme. Section 5 explores proposed concurrency control scheme. Section 6 specifies performance metrics. Section 7 concludes the paper.

2 Related Work

In mobile database environment any mobile client can access data item irrespective of its physical location. Providing consistency of the data items is a challenging issue in case of concurrent access. Various valuable attempts are made in providing solutions for data management in mobile environments.

The conventional two phase locking protocol is not suitable as it requires clients to communicate continuously with the server to obtain locks and detect the conflicts [7]. An optimistic concurrency control technique has the problem of delayed response [11]. In [12] timeout based Commit Protocol is proposed which faces the problem of the time lag between local and global commit and more rollbacks of the transactions due to starvation.In [5] the proposed Mobile 2PC protocol preserves the 2PC principle and minimizes the impact of unreliable wireless communication.

An Optimistic Concurrency Control with Dynamic Time stamp Adjustment Protocol requires client side write operations. However because of the delay in execution of a transaction, it may never be executed [2]. In [1, 4], the conventional optimistic concurrency control algorithm in enhanced with an early termination mechanism on conflicting transactions. In [8] dynamic timer management is used for achieving concurrency control. This suffers from the problem of frequent rollbacks due to regular expiry of the timer and wastage of computation. In [10] preemptive dynamic timer adjustment strategy is proposed to enhance throughput of the system. This scheme is based on assumption of execution time and not scalable.

3 Mobile Database Environment

In mobile database environment (fig.1) the network consists of Mobile Host (MH), Fixed Host (FH) and Base station. A MH is connected with Base station through wireless links. The communication of the MH with the FH is supported by Base station. The area covered by a base station is called cell. Mobile Hosts in a particular cell cannot directly communicate with each other, instead they have to communicate through the base station. Base stations are directly connected with the power terminals, hence there is no problem of power consumption at base station. FH is connected with a wired network and usually has a large

database server, which performs all operations of a DBMS. MH may have storage capabilities and DBMS modules to perform database operations. However it has small screen size, less processing capability and variable bandwidth. Also the communication asymmetry between MH and base station makes receiving by the MH preferable than sending to base stations. A MH may not always be connected with the network. It may be disconnected or in doze mode to same battery consumption. Hence disconnections are treated as normal events and not as failures [7]. The execution of the transactions may be completely on FH or on MH or partially on both.

In our scheme we assume that transactions copy all data items required from FH to local physical memory of the MH. The transactions then disconnect from the FH to decrease network load and execute transaction locally. The updates are recorded on the FH in Write through manner.

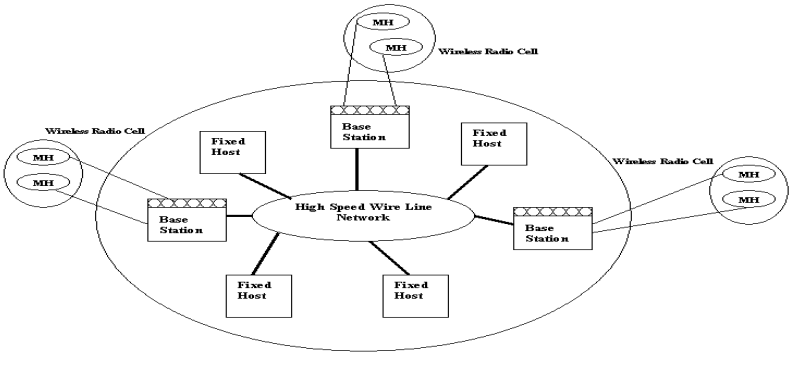

Fig. 1. Mobile Database Environment[8]

4 Performance Limitations of Existing Scheme

1. The AVI value is proposed to do cache invalidation. New value is assigned based on the last update period of the data item. This assumption leads to abortion of the next transaction if it uses this data item after slightly more period. This results is expiry of the data item and there may not be any transaction waiting for this data item. Hence predicting the new AVI value based on update history of the data item may lead to more number of transaction aborts.
2. The TLU value used by the server to save last update time of the data item is not required in maintaining concurrency control. This is included only to predict the AVI value and increasing the computation cost at the server.
3. The AVI value is set two times once at the time of copy of the data item from server and once at the time of update onto the server. Setting the new AVI at the time of update on the server is not required since there may not be any transactions waiting to acquire this data item. Also, at update time

the operations such as write the data value, log record entry etc. will already delay the commit of the transactions.

5 Proposed Concurrency Control Scheme

5.1 Proposed Scheme Features

1. We define a variable VALID PERIOD (VP) of a data item as time quantum in which a MH should use this value in its local execution. This value is deduced based on the remaining number of data items a transaction has to copy from the server to the local memory of the MH. Predicting the validity period of a data item based on total number of data items a transaction further requires to be copied from the server will give a reasonably close valid period for the current transaction than last update time, as time at which the data item is used in execution of the transactions varies. For example, transaction T1 on MH1 may copy data item X as its last data item from server and start local execution. If T1 has only 3 data items then X will be updated in next 4 time units. Hence new AVI value of X will be utmost 4.Next Transaction T2 on MH2 may copy X as its first data item which has a data count of 10.The X value is used in the local execution only after at least 9 time units. This will make X to expire without any reason.
2. We define a parameter DATA COUNT (DC) of a transaction as the remaining number of data items a transaction has to copy from the server to the local memory of the MH.
3. The VP value is a times enhanced based on the number of transactions currently present in the system. This parameter will adaptively set the VP value by considering the current network load on the Fixed Host. Since more number of concurrent transaction results in increased waiting time and data conflicts.
4. Read and Write operations of a transaction are clearly separated to ensure serializability.

The data stored on the fixed host has the following new format to control concurrent access.

Table 1. Data Item Format stored on Fixed Host

Data-Id	**Semaphore**	**VP**	**Data**

1. Data-ID denotes a unique-Id of the data item.
2. Semaphore denotes a binary variable, which has either 0 or 1 value.
3. VP denotes the Valid Period, i.e. the time period for which the data item value should be used in local execution of the MH.
4. Data denotes the current value of the data item.

5.2 Proposed Client (MH) Algorithm

1. MH Connects to the fixed host. Checks the availability of all the data items required by comparing its semaphore value to zero.
2. MH then encounters any one of the two cases.
   Case(A):If semaphore=0 is true, then MH submits data count (DC)(number of data items required) value to the FH, copies the data item format and decrements DC.
   Case(B): If semaphore=0 is false, then wait in the queue till the data item is available.
3. Repeat step (2) for all data items until DC=0.
4. Start execution of the transaction locally on the MH.
   Case(A): for read operation of the data item directly execute step (5).
   Case(B): for write operation of the data item check the following equation
   $Currentaccesstime \geq timestamp + validperiod$ ——————————-(1)
   (Timestamp is the time at which the data item is read by the mobile host)
   i) if equation (1) if true, it means that the period for which the data item is given for update is exceeded. Hence the current value of the data item may be invalid. Increment DC and execute from step (1).
   ii) if equation (1) is false, it means that the data item is still valid. Execute step (6).

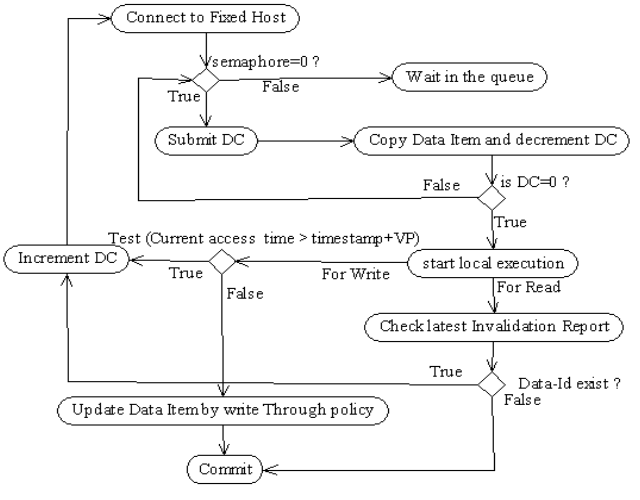

Fig. 2. Activity Diagram of MH

5. Search the Data-Id of the data item in the latest invalidation report.
   Case(A): If Data-Id exist, it means that the value of the data item is updated on the server and the current value is invalid. Increment DC and execute from step (1).

Case(B): If Data-Id does not exist, it means that the value of the data item is still valid.

6. Write data item locally and to maintain consistency, update the value of the data item on the server by write through policy. Write through policy is, whenever the data item is updated in the local memory at the same time it should also be updated on the server by changing the semaphore value to zero (=0).
7. If all data items are updated on the Fixed Host then transaction on Mobile Host commits.

5.3 Proposed Server (FH) Algorithm

1. Create the data item format for each data item by initializing semaphore, Data-Id and Data value. Set valid period zero (VP=0).
2. Start timer.
3. Wait for data item request from MH.
4. If request arrives check semaphore value of the data item.
   Case(A): If Semaphore=1, add this transaction in waiting queue.
   Case (B): If Semaphore=0, allow the MH to read the data item. If the MH wants to perform a write operation then set the semaphore value to '1' and assign new valid period ($VP = \alpha * DC$).
5. Wait for data item update or expiry of valid period.
6. Case(A): If valid period expired before update, then set the semaphore value and valid period of the data item to zero. Invoke the next transaction waiting in the queue. Execute from step (4)
   Case(B): If update request of a data item comes, then generate invalidation report by including Data-Id of the data item updated and send only to those

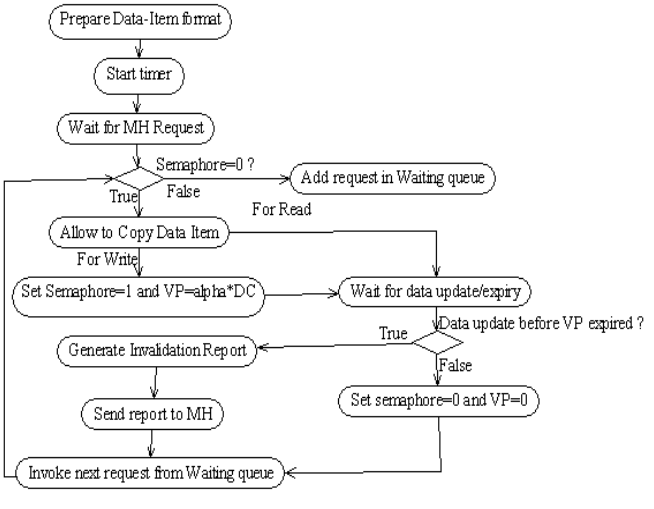

Fig. 3. Activity Diagram of FH Server

MH which has a copy of this data item. Set the semaphore value and valid period of the data item to zero. Invoke the next transaction waiting in the queue. Execute from step (4).

6 Performance Metrics

Same concurrent execution scenario of the transactions as presented in [9] is taken to show performance enhancement. The table. 2 depicts execution steps of the transactions. X,Y,Z are the data items. Each Mobile Host (MH) requires two data items. Copy denotes the operation of copying the data item from Fixed Host (FH) to MH. Read, Write, commit are the operations of normal transaction execution.

Table 3. represents the above Mobile Host transactions execution schedule by the existing scheme [9].The Current Access Time depicts the time of the each logical operation of transaction execution. RW (A) represents copying the data item into the private memory and performing the write operation on A. R (A) represents copying the data item into the private memory for performing read

Table 2. Execution Steps of transactions

MH1	**MH2**	**MH3**
Copy(X)	Copy(Y)	Copy(Z)
Copy(Y)	Copy(Z)	Copy(X)
Write(X)	Read(Y)	Write(Z)
Write(Y)	Write(Z)	Write(X)
Commit	Commit	Commit

MH=Mobile Host

Table 3. Transaction Execution on MH [9]

CT	**MH1**	**MH2**	**MH3**	**AVI**			**Timestamp**		
				X	**Y**	**Z**	**MH1**	**MH2**	**MH3**
11	RW(X)	R(Y)	RW(Z)	5	6	4	X(11)	Y(11)	Z(11)
12	RW(Y)	WAIT(Z)	WAIT(X)	5	6	4	Y(12)	–	–
13	WRITE(X)	—	—	3	6	4	–	–	–
14	WRITE(Y)	Inv.Report	RW(X)	3	5	4	–	–	X(14)
15	COMMIT	RW(Z)	ABORT	2	5	7	–	Z(15)	–
16	—	R(Y)	—	-	5	7	–	Y(16)	–
17	—	WRITE(Z)	—	-	-	5	–	–	–
18	—	COMMIT	—	-	-	-	–	–	–

CT=Current Access Time , AVI=Absolute Validity Interval

only operation. WAIT (A) specifies that the mobile host is waiting to acquire the data item A. WRITE (A) denotes write through policy for updating the data item A. A (T) denotes the time T at which the data item A is copied by the Mobile Host.

In the above schedule MH3 aborts due to wrong prediction of AVI of data item Z, Since by the time it acquires Z after waiting for it, transaction MH2 acquires it violating Serializability. Hence MH3 aborts execution.

Table 4. represents Mobile Host transactions execution schedule with the proposed scheme. For the present scenario $\alpha = 3$.MH3 now commits due to appropriate assignment of the VP (valid period) value of the data item Z. With a cost of 1 time unit the system is able to give full commit rate, though some waiting time of the transaction may increase. The results of the scheme over increased number of transactions given more commit rate. The number of parameters required to update by the server is also minimized. The server updates the values of the data item format only when some transaction uses it.

Table 4. Transaction Execution on MH in proposed scheme

CT	MH1	MH2	MH3	VP			DC			Timestamp		
				X	Y	Z	MH1	MH2	MH3	MH1	MH2	MH3
10	–	–	–	0	0	0	2	2	2	0	0	0
11	RW(X)	R(Y)	RW(Z)	6	0	6	1	1	1	X(11)	Y(11)	Z(11)
12	RW(Y)	WAIT(Z)	WAIT(X)	6	3	6	0	1	1	Y(12)	–	–
13	WRITE(X)	–	–	0	3	6	0	1	1	–	–	–
14	WRITE(Y)	Inv.Report	RW(X)	3	0	6	0	1	0	–	–	X(14)
15	COMMIT	–	WRITE(Z)	3	0	0	0	1	0	–	–	–
16	–	RW(Z)	WRITE(X)	0	0	3	0	0	0	–	Z(16)	–
17	–	R(Y)	COMMIT	0	0	3	0	0	0	–	Y(17)	–
18	–	WRITE(Z)	–	0	0	0	0	0	0	–	–	–
19	–	COMMIT	–	0	0	0	0	0	0	–	–	–

DC=DATA COUNT VP=VALID PERIOD $VP = (\alpha * DC)$ $\alpha = 3$

Table 5. represents data item format values stored on the server while concurrent execution of the transactions.

6.1 Advantages of Proposed Scheme

1. Valid Period (VP) value of a data item is decided based on remaining number of data items a transaction has to copy from Fixed Host. This ensures sufficient time Period in which a transaction use this data item and successfully complete its execution. Hence commit rate of the system increases.
2. Parameter 'α' enhances the VP of a data item based on number of transactions active in the system. Considering current work load of the Fixed Host while assigning new VP of a data item appropriates data sharing period of the concurrent transactions.

Table 5. FH data item status while execution

CT	VP			Semaphore		
	X	Y	Z	X	Y	Z
10	0	0	0	0	0	0
11	6	0	6	1	0	1
12	6	3	6	1	1	1
13	0	3	6	0	1	1
14	3	0	3	1	0	1
15	3	0	0	1	0	0
16	0	0	3	0	0	1
17	0	0	3	0	0	1
18	0	0	0	0	0	0

3. The Value of VP a data item is set to zero after update, since when no transaction requests for that data item, the value will not be invalid without update. It is set only when a transaction request the data item for Write.

7 Conclusion

In this paper we have shown the limitations of the existing concurrency control scheme. We also justified that predicting the AVI value based on update history is not desirable. The proposed concurrency control scheme improves the drawbacks of the existing scheme by adaptively selecting validity period of the data items based on required number of data items and current load of Fixed Host server. It also increases the overall commit rate of the system.

References

1. Anand, Y., Wen, C.H., Chih, F.W.: Improving Concurrency Control in Mobile Databases. In: Lee, Y., Li, J., Whang, K.-Y., Lee, D. (eds.) DASFAA 2004. LNCS, vol. 2973, pp. 642–655. Springer, Heidelberg (2004)
2. Ho, J.C., Byeong, S.J.: A Timestamp- Based Optimistic Concurrency Control for Handling Mobile Transactions. In: Gavrilova, M.L., Gervasi, O., Kumar, V., Tan, C.J.K., Taniar, D., Laganá, A., Mun, Y., Choo, H. (eds.) ICCSA 2006. LNCS, vol. 3981, pp. 796–805. Springer, Heidelberg (2006)
3. Joe, C.H.Y., Chan, E., Lam, K.Y., Leung, H.W.: An Adaptive AVI-based Cache Invalidation Scheme for Mobile Computing Systems. In: 11th International Workshop on Database and Expert Systems Applications (DEXA 2000), p. 155. IEEE computer society, USA (2000)
4. Minsoo, L., Sumi, H.: HiCoMo:High Commit Mobile Transactions. In: Distributed and Parallel Databases, vol. 11, pp. 73–92. Kluwer Academic Publishers, Dordrecht (2002)

5. Nadia, N., Anne, D., Habiba, D.: A Two-Phase Commit Protocol for Mobile Wireless Environment. In: 16th Australasian Database Conference, Australia, pp. 135–143 (2005)
6. Patricia, S.A., Claudia, R., Michel, A.: A Survey of Mobile Transactions. In: Distributed and Parallel databases, vol. 16, pp. 193–230. Kluwer Academic Publishers, Dordrecht (2004)
7. Salman, A.M., Lakshmi, R.: An Algorithmic approach for achieving Concurrency in Mobile Environment. In: INDIACom, India, pp. 209–211 (2007)
8. Salman, A.M., Lakshmi, R.: Single Lock Manager Approach for Achieving Concurrency in Mobile Environments. In: Aluru, S., Parashar, M., Badrinath, R., Prasanna, V.K. (eds.) HiPC 2007. LNCS, vol. 4873, pp. 650–660. Springer, Heidelberg (2007)
9. Salman, A.M., Nizamuddin, M.K.: Concurrency Control Without Locking in Mobile Environments. In: 1st International Conference on Emerging Trends in Engineering and Technology, pp. 1336–1339. IEEE Computer society, USA (2008)
10. Salman, A.M., Lakshmi, R.: Concurrency Control Strategy to Reduce Frequent Rollbacks in Mobile Environments. In: International Conference on Computational Science and Engineering, pp. 709–714. IEEE Computer society, USA (2009)
11. Victor, C.S., Kwok, W.L., Son, S.H.: Concurrency Control Using Timestamp Ordering in Broadcast Environments. The Computer Journal 45(4), 410–422 (2002)
12. Kumar, V., Prabhu, N., Dunham, M.H., Seydim, A.Y.: TCOT- A Timeout based Mobile Transaction Commitment Protocol. IEEE Transactions on Computers 51(10), 1212–1218 (2002)