Vol. 5, No. 3 March 2014 ISSN 2079-8407
Journal of Emerging Trends in Computing and Information Sciences

http://www.cisjournal.org

Performance Analysis of Concurrency Control Using Locking in Mixture Transactions Environment

Ahmad al-Qerem
Asst. Prof, Department of Computer, Zarqa University, Jordan
[1] ahmad_qerm@zu.edu.jo

ABSTRACT

Concurrency control protocols proposed in the literature are mainly focused on homogeneous environments. That is, there is only one type of transactions present in the system, either fixed transactions or mobile transactions. However, it is not uncommon to have mobile applications that consist of both types of transactions. Due to high expensive resources used by mobile transactions, it will be a serious problem for these transactions to suffer from blocking caused by other fixed or mobile transactions. Since the blocking can be unbounded because of disconnection which may highly occur in such environments, reserved resources by the blocking transactions and other reserved by the waiting transactions may seriously affect other ongoing mobile transactions that share the same wireless resources. In view of this, it is very important to reduce the blocking overhead which negatively affect amount of resources wasted in this aspect. In this paper, to reduce blocking effect, a lock-based approach, called Lock-Mix is proposed. The execution of both types of transactions is divided into two phases a no blocking phase where transactions wait for locks but do not block other transactions and a blocking phase as in traditional locking. However, data accessed during the no blocking phase can lead to transaction abort. Two parameters η and μ associated with fixed and mobile transaction respectively have been used for properly balancing the blocking and abort effects. The characteristics of this approach have been examined in detail by simulation. The results show that the performance is consistently better than either the SDHP-2PL or FBOCC protocol over a wide range of system settings. In particular, this approach provides a more significant performance gain in mixture transaction environments.

Keywords: *concurrency control, mixture transactions, mobile computing, performance evaluation*

1. INTRODUCTION

The mobile computing systems are suitable for many application domains. These applications include stock trading, mobile auctions, mobile commerce applications [5]. In these applications, the consistency among data items is likely to be violated by update transactions. Thus, a concurrency control scheme is needed to preserve data currency and consistency for mobile transactions [19].

In mixture transactions environments, conventional concurrency control schemes such as two-phase locking and timestamp ordering are not suitable for the mobile transactions due to the limited bandwidth in the uplink communication channel and the limited battery power of mobile clients [7, 8, 18, 20]. There have been many research efforts reported in the literatures that focus on the concurrency control scheme for mobile data base environments [1,2,3,4,6,9]. In [13], the researcher proposed a variant of the OCC method called FBOCC (Forward and Backward OCC) that uses forward validation for update transactions and partial backward validations for read-only transactions.

However, all the concurrency control methods for mobile including FBOCC [13] and SDHP-2PL method are focused on homogeneous transactions. Existing methods perform poorly in mixture environments where the transactions types used different communication media. Furthermore, update mobile transactions will be frequently aborted and restarted in the final validation stage due to the update conflict of the same data items which frequently accessed by the fixed host transactions. This problem will increase the restart time of the update mobile transactions and waste the scars resources in mobile computing system. For SDHP-2PL schema, the basic strategy for conflict resolution is based on the HP-2PL scheme and the concept of similarity [21]

From the traditional viewpoint, various concurrency control algorithms basically differ in two aspects: the time when they detect conflicts and the way that they resolve conflicts. Locking and optimistic approach in their basic form represent the two extremes in terms of these two aspects. Locking detects conflicts as soon as they occur and resolves them using blocking. Optimistic scheme detects conflicts only at transaction commit time and resolves them using restarts. In mixture transactions environment, the way which these approaches used in resolving data conflict for such transactions mixture makes further difference among concurrency control mechanisms. The impact of these differences in concurrency control algorithms on performance has been the major theme in the performance study of concurrency control in such environments. With respect to the impact of conflict resolution methods, the effect of blocking and restart should be considered in the context of the available amount of system resources. Generally, blocking-based conflict resolution policy conserves resources, while restart based Policy wastes more

resources. Previous performance studies on conventional database systems [12,14,16]showed that locking algorithm that resolves data conflicts by blocking transactions outperforms restart-oriented algorithm in an Environment where physical resources are limited. Also, the work[17] showed that if resource utilizations are low enough so that a large amount of wasted resources can be tolerated, and there are a large number of transactions available to execute, then a restart-oriented algorithm that allows a higher degree of concurrent execution is a better choice.

In this study, we investigate the effect of blocking and restart in the context of mixture transaction environments. The timing of conflict detection and resolution also has a major impact on performance. The remainder of this paper is organized as follows:

In Section 2, we describe the mixture transactions model. Section 3 discuss the problems associated with blocking and restarts in the context of mixture transactions. The Lock-Mix approach is proposed in section 4. We give the performance evaluation of Loc-Mix protocol in Section 5. Finally, we conclude the paper in Section 6.

2. MIXTURE TRANSACTION MODEL

This section defines a mixture transactions model Fig.1; it is assumed that the database system consists of two major components: the transaction manager *(TM)* and data manager *(DM)*. TM maintains a transaction table to record the execution status of both mobile and fixed host transactions in the system. TM includes two components: scheduler and data manger. The scheduler is responsible for concurrency control. When the scheduler receives an operation, it determines whether the operation should be processed, blocked, or rejected. If an operation is rejected, the corresponding transaction will be restarted. The scheduler maintains an access-status table to detect any possible data conflicts. For example, if locking is adopted for concurrency control, the scheduler may maintain a lock table to record the locking status of any data items accessed by all executing transactions (and the collection of the blocked transactions due to lock conflicts). The same table can also be used for the optimistic methods. All data access requests issued by an operation are handled by the DM, which retrieves the required data item. Transactions involved in this system consist of a sequence of read and writes operations, and ends with a commit or an abort operation. This transaction considers an atomic process. That is, translate a database from a consistent state into another consistent state. There are two types of transactions in this environment: fixed or wired *(FT)* transactions and mobile transactions *MT*. A fixed transaction is submitted directly to a database on the same host while the mobile transaction submitted by mobile device. Like in [20], an *(MT*) is a mobile transaction which issued by a mobile host. The participation of an MH introduces dimensions inherent to mobility such as: movement, disconnections and variations on the quality of communication. TM supporting has to adapt their functionalities to deal with these dimensions. In the scope of this work we focus on systems with a client-server architecture where clients are either MH or FH interacting by accessing a shared database through invoking transactions.

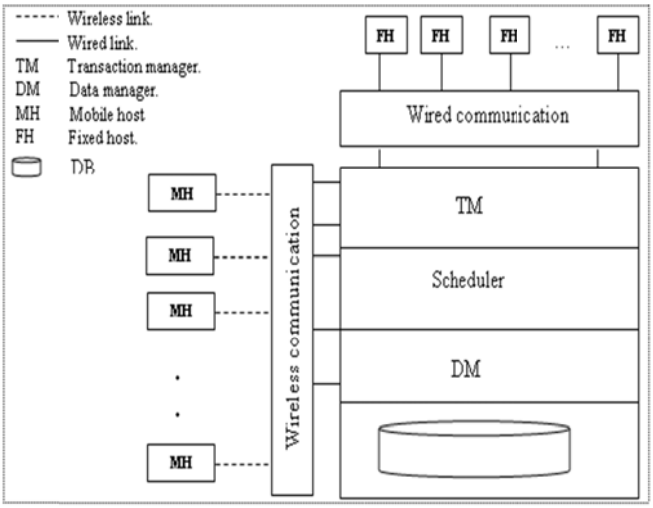

Fig 1: Mixture transactions environment

3. LOCK-MIX APPROACH

The Lock-Mix approach over transactions mixture comes up to overcome the problems found in each of locking and OCC protocols if they applied separately as the in charge concurrency control protocol. These problems are come from the nature of conflict resolution being used by these protocols.

A. Problems with locking approach

Two-phase locking (2PL) mechanism is accepted as the standard solution to the concurrency control problem in traditional DBMSs. 2PL depends on well-formed transactions, which do not lock again data items that have been locked earlier in the transaction, and whose execution is divided into a growing phase in which locks are only acquired and a shrinking phase in which locks are only released. During the shrinking phase, a transaction is prohibited from acquiring locks. If during the growing phase a transaction attempts to acquire a lock that has already been acquired by another transaction, it is forced to wait until the lock is released.

The performance of locking may severely degraded in our mixture system, because the significant increase in the total number of concurrent transactions results in a high lock contention level and hence a high lock conflict probability. Communication delay that leads to a longer lock holding duration makes the lock conflict probability even higher. To best explain these problems in the mixture transactions environment, consider the following example.

Example 1: Consider the interleaving of operations showed in the Fig.2, assume that transactions T1, T3 and T4 are mobile transactions and the rest of transactions are fixed.

Time						
τ_6	τ_5	τ_4	τ_3	τ_2	τ_1	Data Item
					R_1	D_1
					R_3	D_2
					R_4	D_3
		W_4	R_3	R_1	W_2	D_4
			R_3	W_6	R_5	D_5
					W_5	D_6
					W_1	D_7
					W_3	D_8
					W_4	D_9
					R_1	D_{10}
					W_3	D_{11}
R_5	W_{12}	R_9	W_8	R_{10}	W_7	D_{12}
					W_4	D_{13}

Fig 2: Operations interleaving for example1

In 2PL protocols Fig.3, transaction scheduling order is determined purely by the order in which transactions acquire locks. Once transaction T_5 locks the data item D_6, transaction T_5 is not able to unlock the data item D_6 until all T_5 operations get their own lock on their data items. Transaction T_6 is waiting for T_5 and transaction T_2 is waiting for T_6 on the same data item D_5 If we assume that $R_5(D_{12})$ is the last operation of transaction T_5, then $R_5(D_5)$ and $W_5(D_6)$ cannot free these data items until $R_5(D_{12})$ get its lock to obey the 2PL rules. Also, transactions T_1, T_3, and T_4 will be delayed until T_2 frees the data item D_4, which may occur after Transactions T_6, T_{12}, T_9, T_8, T_{10}, and T_7 have finished. This will result in a cascading delay caused by the early blocking of the 2PL protocol.

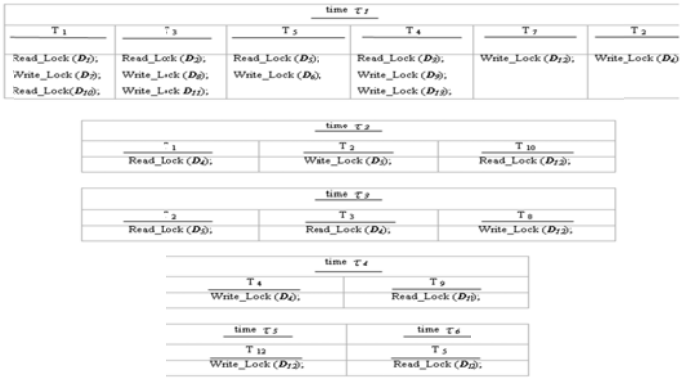

Fig 3: 2PL Interleaving Example

In mixture transactions environment, locking has been found to constrain concurrency and to add an unnecessary overhead. The following are the disadvantages of locking protocol:

- Lock maintenance represents an unnecessary overhead for read-only transactions, which do not affect the integrity of the database.
- Not permitting to release locks except at the end of the transaction execution, which although not required is always done in practice to avoid cascaded aborts, decreases concurrency.
- Most of the time it is necessary to use locking to guarantee consistency since most transaction does not overlap; locking may be necessary only in worst cases.

B. Problems with OCC approach

The goal of Optimistic Concurrency Control (OCC) is to avoid these problems of 2PL. OCC requires each transaction to consist of three phases: a read phase, a validation phase, and a write phase. During the read phase, all writes take place on local copies of the records to be written. Then, if it can be established during the validation phase that the changes the transaction made will not violate serializability with respect to all committed transactions, the local copies are made global. Only then, in the write phase, do these copies become accessible to other transactions. There are two properties of OCC that distinguish it from other approaches. First, synchronization is accomplished entirely by restarts, never blocking. Second, the decision to restart or not is made after the transaction has finished executing. To best explain how OCC protocol can poorly effect the transaction processing in our environment, consider the following example.

Example 2: consider the same transactions in *Example 1* with the interleaving shown below Fig.4.

In optimistic concurrency control Fig.4, transactions are allowed to execute unhindered until they reach their commit point, at which time they are validated. Thus, the execution of a transaction consists of three phases: read, validation, and write phase. The key component among these is the validation phase where a transaction's destiny is decided. In OCC the scheduling order is determined by the arriving order of transactions at the validation phase. If we assume that transaction T_2 reaches its validation phase first, all active transactions which conflict with T_2 should be restarted.

Fig 4: OCC Interleaving Example

4. APPROACH DETAILS

As mentioned in the previous section, a conventional 2PL scheme tends to suffer from a cascade of blocking. On the other hand, an OCC scheme may suffer from wasting resources due to transaction aborts and restarts. In OCC, transactions become more vulnerable (i.e., more likely to be involved in conflicts and be marked for abort) as they make more progress toward completion, since more data items are accessed. As a result, longer transactions would incur higher abort probabilities in OCC. In a mixture database environment the abortion of mobile transaction by a fixed host transaction will waste a valuable wireless resources to save a reliable wired resources, Furthermore, the cost of aborting a nearly completed mobile transaction by other mobile transaction is certainly higher than that of aborting a newly started mobile transaction. Our Lock-Mix approach addresses this issue by using the concept of deferred blocking [15]. The transaction execution of both types mobile and fixed is divided into a no blocking phase and a blocking phase. At the start of the execution, a fixed and mobile transaction is in the no blocking phase where it simply obtains a fixed lock synchronously for each data item access. In this phase neither fixed transaction nor mobile will block other fixed or mobile transactions. After a mobile or fixed transaction has access a predefined number of data items, they tries to enter the blocking phase to prevent other transactions from aborting them at the later stage of their execution. It will try to convert all the fixed locks on the already accessed data items into mobile locks as in the certification process of a conventional OCC. If successful, the transaction will switch to the blocking phase. We have two parameters η and μ associated with fixed and mobile transaction respectively, which used as a metrics for determent of switching point from non-blocking to blocking phase. η , μ represent predefined values which means after how many operations that get their own fixed-lock on their data items can we switch to a blocking phase. For the fixed host transaction, still have the opportunity to finish its execution at the first stage of its executions. For the mobile transaction after it decides to enter the blocking phase it will then obtain a mobile lock on each subsequent data item access and wait for the lock obtained by other mobile transactions which have already converted to the blocking phase, if held under an incompatible mode. This will prevent mobile transactions from aborting at the later stage of their life, and will also reduce the average holding time of a mobile lock, thus reducing the blocking effect caused by the fixed host transaction under the pure 2-PL protocol.

In Lock-Mix approach, each data item can be locked in different lock modes such as the shared and exclusive modes. Shared locks on the same data item are compatible while an exclusive lock is incompatible with a shared lock or another exclusive lock on the same data item. Similarly, in the proposed scheme, the CC manager maintains a lock table where each data item can be locked in either share or exclusive mode. Furthermore, two lock types, mobile and fixed are used in different ways based on the type of requesting transaction and the order of operations inside the transaction as we can see the Fig.5. The lock type compatibility matrix is given in Table1, and the pseudo code for the concurrency control algorithm is given in Fig.6, Fig.7 and Fig.8.

TABEL 1: COMPATIBILITY MATRIX FOR LOCK-MIX PROTOCOL

Request \ older		Fixed		Mobile	
		Read	Write	Read	Write
Fixed	Read	Y	Y	Y	N
	Write	Y	Y	N	N
Mobile	Read	Y	N,S	Y	N
	Write	N,S	N,S	N	N

Fig 5: Lock request for mobile and fixed transaction

As shown in Table.1 the concept of 'superseding' denoted by S is used to resolve lock conflict between mobile lock requests and fixed lock holders. The mobile lock request supersedes incompatible fixed lock requests in the

sense that if the requested data item is currently held by fixed locks, the incompatible fixed locks are released to grant the mobile lock request and the fixed lock holders are marked for abort. Fixed lock requests are always compatible with other fixed locks. A fixed lock request in shared mode and a mobile lock held in shared mode are regarded as compatible. Otherwise, a fixed lock request is not compatible with a mobile lock, and the requester has to be put into a wait state. The compatibility matrix can be found in Table.1.

As we see from the figures (i.e. Fig.6, Fig.7 and Fig.8). that describe the Lock-Mix protocol, the execution of a fixed transaction is scheduled as in conventional 2PL protocol until the commit time. So, the possibility that a fixed transaction aborts or blocks a mobile transaction is highly reduced. Furthermore the mobile transactions blocked by a fixed host transaction executing its last operation, and a mobile transaction aborted as a result of converting a fixed host transaction from a non-blocking to blocking stage will be in their early stages. To explain why this approach is better than other 2PL and OCC on the transactions mixture considers the following example.

Fig 6: Execute mobile lock request

Fig 7: Execute fixed lock request

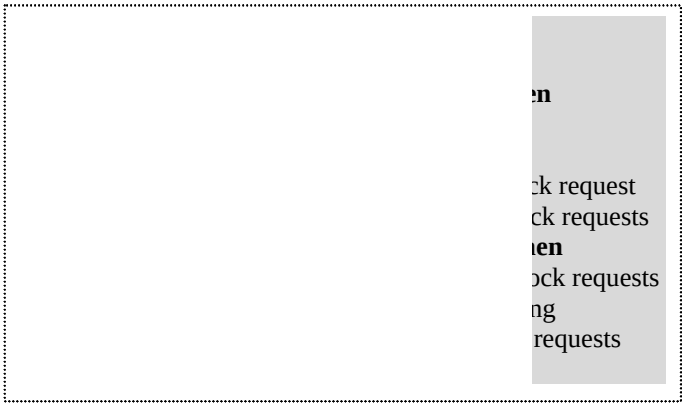

Fig 8: Execute unlock request for both mobile and fixed transaction

Example 3: Consider the same transactions in Example1 with the same operation‘s interleaving used by the 2PL example. Fig.9 below show the lock requests for each transaction under the Lock-Mix approach. The fixed lock is used to simulate the OCC protocol and the mobile lock has been used to simulate the blocking resulting from conflicts of mobile and other mobile or fixed transactions. We have two parameters η and μ associated with fixed and mobile transactions respectively, which are used as a metrics for determining the switching point from a non-blocking to a blocking phase. Assume that η = 4, μ = 6 the Lock-Mix protocol can be explained as follows:

Fig 9: Lock-Mix interleaving example

Both fixed and mobile transactions start their execution by requesting a fixed read and write lock. After the mobile transaction executes its fourth operation it will switch to blocking phase and can't be restarted by the fixed transaction or other mobile transactions. The same is true for the fixed transaction after executing its sixth operation. In Fig.9 above, transactions T_1, T_3 and T_4 are willing to lock the data item D_4, which is already locked by transaction T_2. Since R_1 (D_4) is the 4^{th} operation of T_1, it will acquire a mobile lock on D_4. Because the mobile lock requested by T_1 is supersedes the fixed lock granted to the transaction T_2 on D_4, T_2 will be marked for abort and T_1 granted a mobile lock on D_4. So, T_1 is switched into a blocking stage and cannot be marked for abort by other fixed or mobile transactions. When transaction T_3 tries to get the lock on D_4 it succeeds because it's compatible with R_1 (D_4). Transaction T_4 tries to get the mobile lock on D_4 in an exclusive mode. So, it has to wait for unlock (D_4) by transaction T_1 and T_3. Since the reaming number of operations for the transactions to switch to their blocking stage is less than transaction length, (this depends on the η , μ values) we will expect a reasonable reduction in the average blocking time for all mobile transactions. Thus, transactions T_1 and T_3 don't have to wait for transaction T_2 which would have to wait for T_5, T_6, T_7, T_9, T_8, T_{12} and T_{10} in order to unlock the data item D_4 according to the 2PL protocol. On the other hand, other transactions still have the opportunity to finish their execution without being restarted by other fixed or mobile transactions that enter their validation phase according to the pure OCC protocol.

5. PERFORMANCE EVALUATION

We have constructed and conducted a series of simulation experiments to evaluate and compare the performance of our concurrency protocols on mixed transactions environment. In the following sections, we will discuss the simulation model, the workload being used, performance metrics, and finally present our research findings in these experiments.

a. Simulation Model

We have developed a simulation system, as shown in Fig10, to study the performance of the proposed protocols against SDHP-2PL and FBOCCI in the context of mixture transactions environment to identify their performance characteristics. The simulation model consists of two major components: the transaction manager *(TM)* and data manager *(DM)*. TM maintains a transaction table to record the execution status of both mobile and fixed host transactions in the system. TM includes two components: scheduler and data manger. The scheduler is responsible for concurrency control. When the scheduler receives an operation, it determines whether the operation should be processed, blocked, or rejected. If an operation is rejected, the corresponding transaction will be restarted. In the system, different data structure has been used for different approaches. The scheduler maintains an access-status table to detect any possible data conflicts. For example, if Locking protocol is adopted for concurrency control, the scheduler maintain a lock table to record the locking status of any data items accessed by all executing transactions (and the collection of the blocked transactions due to lock conflicts). We also did the same for the optimistic protocols, a data object table and a transaction table are maintained.

Two generators are implemented. The fixed host generator is responsible for the generation of the fixed host transactions. The second generator is for mobile transactions with specific attribute that imitate both the mobile device and the wireless environment. Submitted transaction from the mobile host is sent to a base station through a wireless link and then sends to the database via existing wired network. During the execution time, a transaction may need the user's interaction to the input data. Many studies have pointed out that the cost of communication setup is very expensive [10,11,17]. To avoid re-establishing the communication each time the transaction needs the user's interaction, we assume that the communication must be kept during the execution of the mobile transaction. Transactions are generated from both transactions generators the fixed and mobile are lined up in the

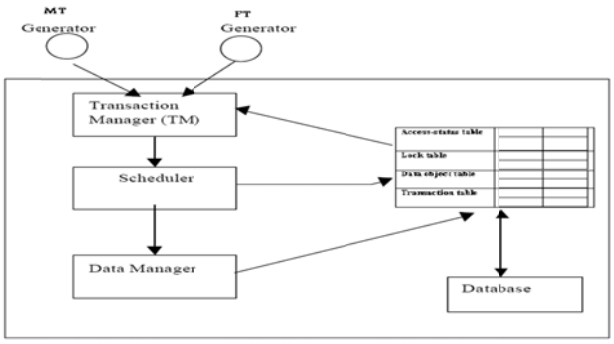

Fig 10: Simulation model

CPU queue according to the first in first out scheduling discipline. When the CPU is available, the transaction at the front of the CPU queue will be submitted to the CPU for processing. To read a data object, transactions need to line up in the disk queue for data access. The transactions will repeat these steps until all operations are processed.

When the mobile unit is within the cell of a base station, the base station will provide a communication channels available in this cell. Due to movement of a mobile user, when a mobile user enters a new cell, the new base station should provide an idle channel to mobile unit to sustain its communication. This process is called a handoff. If there is no idle channel in the new cell to provide for the user, the disconnection will occur.

The mobility of each mobile host modeled based on the number of visited base stations during transaction execution. The visited base station for each mobile host is randomly selected when the handoff occur. This will affect the number of users under each cell during the simulation which affects the bandwidth available for the users.

$$BW_t^{cell_i} = \frac{\text{BW offer by BSC}_i}{\text{Number of mobile users in cell}_i \text{ at time } t}$$

The numbers of users in each cell with random handoff imitated process are used to determine the time between arrivals of mobile transaction operations that uses by the mobile transaction generator during the simulation.

$$TBA_t^{cell_i} = \frac{1}{BW_t^{cell_i}} * TBA_{generator}$$

The power consumption is measured by monitoring the three basic metrics-energy components: (i) *transmission* power required to send a message (i.e. operation), (ii) *reception* power required to receive or listen to a message, and (iii) *idle* power required to stay in at the active state (awake) during transaction execution. Therefore, the power consumed by any mobile host during each execution is given by the equation**:**

Power consumption (MH_i) =

$$\sum_{t=t_0}^{t_f} NOpS * \beta^t + \sum_{t=t_0}^{t_f} NDR * \sigma^t + \omega * idel\_time$$

Where *NOpS* is the number of operation sending by Ti, *NDR* is the number of data items received by Ti. And β, σ are fixed cost associated of each operation and data item at the time of sending the operation and receiving the data item, respectively. ω is *idle* power required to stay in at the active state (awake) during transaction execution.

b. Parameter Setting

Table II summarizes the key parameters that characterize system workload and transactions. The number of data objects accessed by a transaction and the actual data items are determined uniformly from the database. A data object that is read is updated with the probability, *WriteProb*. The parameters, *CPU Time* and *Disk Time* capture the CPU and disk processing times per data item.

TABLE 2: SUMMARY OF WORKLOAD USED FOR BASELINE EXPERIMENT

System parameter	
Reported value per test session	means of 10 times replication of each
Database size	300 data item
CPU Time	2 Time unit /Operation
Disk Time	5 Time unit/Data item
CPU queuing discipline	First in first out
Number of cells	20
Maximum number of users per cell	100 users
Send communication cost	15 Time unit/Operation
Receive communication cost	5 Time unit/ Control message
Transaction parameter	
Mobile transaction Percentages	20%, 50%, 80%
Fixed host transaction length	3 - 15
TBA of fixed transaction operations	2 – 5 uniformly distributed
Probability of disconnection	0.1, 0.2, 0.3
Mobility values	1, 2, 3, 4, and 5 BS/Transaction
Power of the mobile host	200 – 600 ms
Mobile transaction length ($N_{operations}$)	3 – 15 operations
Fixed transaction *WriteProb*	0.5
Mobile transaction W*riteProb*	0.5

c. Performance Metrics

In Mixture transactions environment, the major performance measure that can demonstrate the effectiveness of the protocols is the PCR which reflect both blocking and restarting overheads come cross by a mobile transaction. The restart ratio measures the average number of restarts experienced by a mobile host transaction before it can commit. This measure also indicates the amount of resources spent on restarted transactions. Reducing the restart cost not only saves resources, but also helps to soothe resource and data contention. The mobile transaction rollback frequency and the fixed transaction rollback frequency help to analyze the source of transaction restarts.

The frequencies can reflect the proportion of data conflict between mobile and fixed transactions and that among mobile transactions as the utilization of mobile transactions varies. The formulas of the main performance measures are listed below.

Power consumption ratio (PCR) = $\frac{P_{initial} - P_{final}}{P_{initial}}$

Restart ratio = $\frac{N_{restart}}{N_{committed}}$

Fixed transaction rollback frequency = $\frac{N_{fixed,restart}}{N_{committed}}$

Mobile transaction rollback frequency = $\frac{N_{mobile,restart}}{N_{committed}}$

Where

$N_{restart}$: number of restarted transaction of both types.
$N_{committed}$: number of committed transactions of both types.
$N_{mobile, restart:}$ number of committed mobile transaction which caused other mobile transaction to be restarted.
$N_{fixed, restart:}$ number of committed mobile transaction which caused other fixed transaction to be restarted.

d. Experiments and Results

i. Impact of Mobility

Fig.11 and Fig.12 gives the restart ratio for mobile and fixed transactions respectively when 20% of transactions presences in the system are mobile. The restart ratio under all protocols is similar when the mobility is low. As the mobility increase the difference between SDHP-2PL and Lock-Mix become notable. Fig.12 shows the opposite behavior of these protocols for fixed transactions. From the Figures we can see the Lock-Mix improvement of decreasing mobile transaction restart while sustaining a comparable result with other protocols in term of fixed transaction restart.

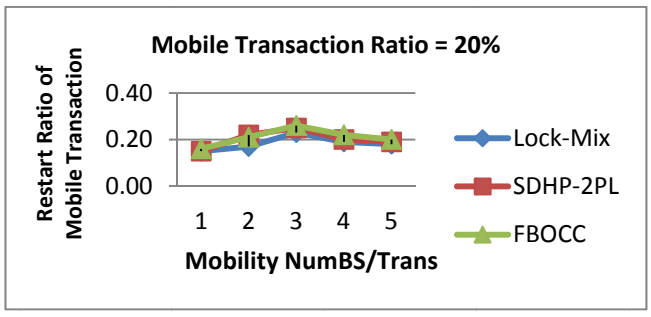

Fig 11: Restart ratio for mobile transaction (MT = 20 %)

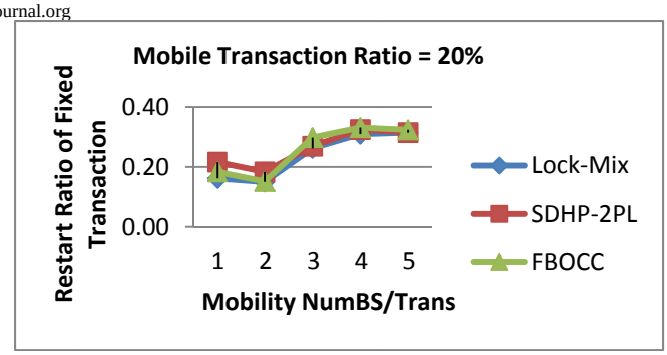

Fig 12: Restart ratio for fixed transaction (MT = 20 %)

Fig.13, Fig.14 gives the restart ratio when the utilization of mobile transactions is 50 percent. Note that the restart ratio become notable at low mobility value. It is due to the increment of mobile transactions. Similar to Figure12, the restart ratio increases as the mobility increases until it is reaches the peak for the Lock-Mix approach when the mobility is 4. That is, there is an increasing delay between the mobile transaction operations as the number of mobile transactions increases. When the mobility is greater than 4, the number of simultaneously active transactions increases in the system and this increase the blocking effect of Lock-Mix approach. Even though, the Lock-Mix approach performing better than other protocols because of its flexible blocking nature regarding mobile transaction. When the mobility > 4, restart ratio is decreasing under all protocols. This is due to increasing the number of disconnected mobile client which is consequently decrease the data contention and the number of mobile transactions in the system and this leads Lock-Mix approach to be slightly better than FBOCC and SDHP-2PL when the mobility is greater than 4.

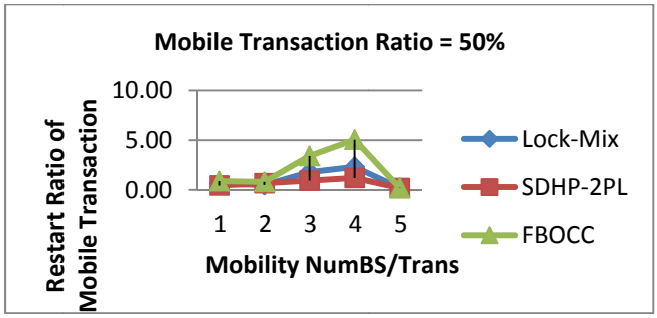

Fig 13: Restart ratio for mobile transaction (MT = 50 %)

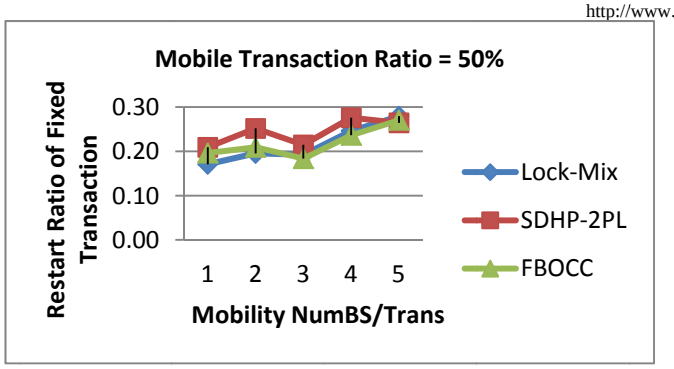

Fig 14: Restart ratio for fixed transaction (MT = 50 %)

Fig.15 gives the restart ratio when the system utilization of mobile transactions is 80 percent. In this experiment, due to the domination of mobile transaction the improvement made by the FBOCC and Lock-Mix approaches is less than in the Fig.13 where the utilization of mobile and fixed transactions are equally likely in the system. This domination of mobile transaction leads to the situation where most of data conflicts are among mobile transaction them self. For the FBOCC and Lock-Mix approaches they reach the peak when the mobility is also 4. As the mobility increase, the disconnection of mobile host increase and the restart ratio decrease as a side effect of this situation. This also can be deduce from the Fig.14 and Fig.16 for restart rate of fixed host transaction where the Lock-Mix is getting better in this experiment as the possibility for the restarted mobile transaction caused by committed mobile transaction increase. In particular, such a reasonable reduction of the mobile restart ratio saves much wireless resource from processing unnecessarily restarted mobile transactions such that other mobile transactions have more opportunity to complete their execution while moving.

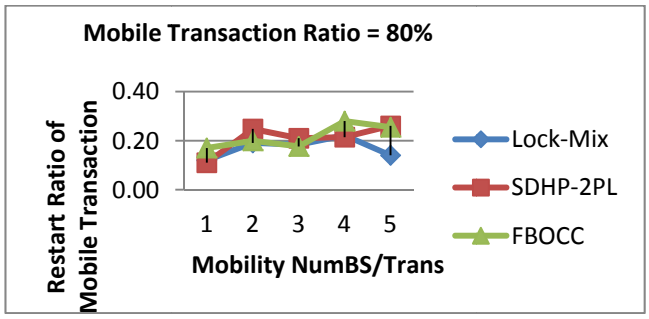

Fig 15: Restart ratio for mobile transaction (MT = 80 %)

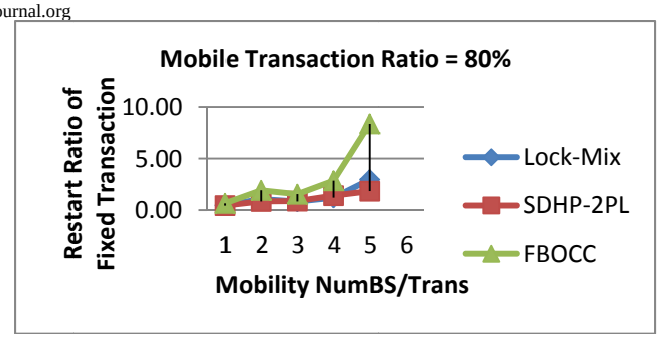

Fig 16: Restart ratio for fixed transaction (MT = 80 %)

ii. Impact of Data Contention

Different degree of data contention may affect the system performance when these protocols are applied. In these experiments, we simulate the transaction execution at different proportion of read and write operations in a mobile transaction. Fig.17 shows the restart rate for mobile transactions as the write probability varies when 20% transactions present in the system are mobile. It can be seen that there is no difference between the performances of the protocols at both ends of the write probability. When all operations are read, there is no data conflict and no adjustment is required. It makes no difference which protocol is employed. On the other hand, when all operations are writing, the performance of the Lock-Mix approach is also same as that of the other two protocols because it is impossible to adjust the serialization order between the conflicting transactions since all data conflicts are serious. That is, the Lock-Mix approach resembles the other two protocols when all operations are either read or write. In other words, the Lock-Mix approach functions more effectively when transactions have a mix of both read and write operations. Fig.18 and Fig.19 give the restart rate when the mobile transactions percentages are 50% percent and 80%, respectively. Since there are both read and write operations in mobile transactions, when there is write-read conflict between a mobile transaction and a fixed transaction, the fixed transaction restart can be avoided by backward adjusting the fixed transaction. On the other hand, when all operations in fixed transactions are write ones, the situation becomes similar to that in Fig.17. All data conflicts are serious and no adjustment can be made. For the Lock mix protocol, when all transactions are read only transaction, it has the similar performance like other protocols. But, the restart ratio is highly increase under these protocol when the write probability of transaction operations increase, there is a significant difference in the performance comparing to other protocols as we can see from all Figures.

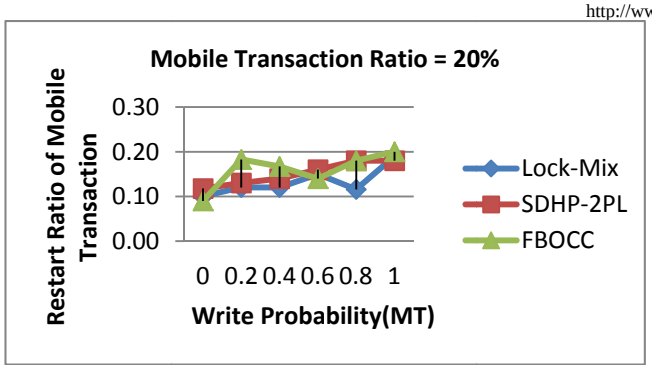

Fig 17: Restart ratio for mobile transaction (MT = 20 %)

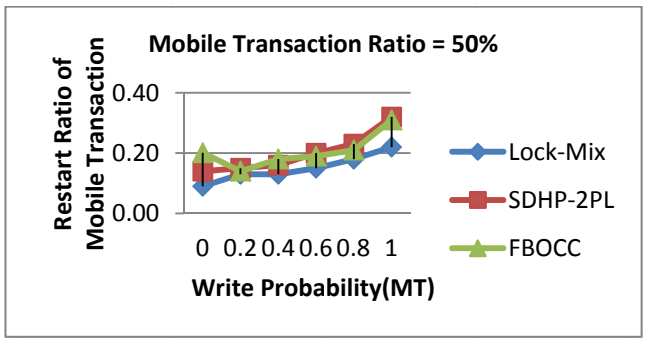

Fig 18: Restart ratio for mobile transaction (MT = 50 %)

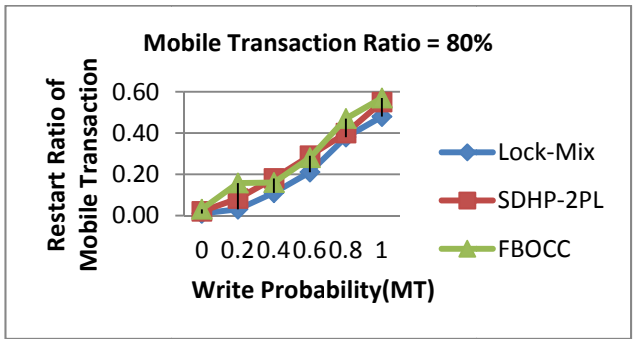

Fig 12: Restart ratio for mobile transaction (MT = 80 %)

iii. Impact of Workload

To have a deeper understanding of the data conflict between fixed and mobile transactions, two performance measures, called fixed transaction rollback frequency and mobile transaction rollback frequency, are collected. The frequencies indicate the amount of data conflict between fixed and mobile transactions and among mobile transactions, respectively. The latter frequency also represents the fraction of committed mobile transactions which have roll-backed other mobile transactions.

Fig.20 and Fig.21 gives the mobile transaction rollback frequency when 20% of transactions present in the system are fixed. Most of data conflicts are among fixed transactions. As the workload increases, data conflict intensifies and it is more likely for a transaction to rollback others in order to be committed. When the system begins to saturate, frequency decreases as the number of committed transactions decreases. For a transaction being restarted, there must be one other transaction to roll it back.

Fig.22 and Fig.23 give the frequencies when the utilization of fixed transactions is 50 percent. In Fig.22, it can be observed that the amount of data conflicts between mobile and fixed transactions increases as the number of fixed transactions increases until the system begins to saturate. In these two figures, it can also be observed, for the Lock-Mix approach, that it is more likely for a mobile transaction to rollback a fixed transaction than for a fixed transaction to rollback another fixed transaction, though the utilization of mobile transactions is equal to that of fixed transactions. Since the Lock-Mix approach makes more room for mobile transactions, any fixed transaction that has data conflicts with a mobile transaction and whose timestamp interval shuts out, will be roll backed by the mobile transaction when it commits. A further increase of the utilization of mobile transactions exacerbates the situation.

Fig.24 and Fig.25 give the frequencies when the utilization of fixed transactions is 80 percent. When the workload is low, under the Lock-Mix approach data conflict between mobile and fixed transactions increases. As the workload increases, data conflicts among fixed transactions increase. On the whole, we can observe that the Lock-Mix approach can effectively help to reduce the number of restarts, whether they are due to data conflicts between mobile and fixed transactions or among mobile transactions. Moreover, even the Lock-Mix approach negatively affects the number of committed fixed transactions, as the restart ratio of fixed transaction increases when the percentage of mobile transaction present in the system increases. This ratio is acceptable since the wasted resources are concentrated in the fixed part of the network which can be tolerated especially when the performance gain in the scarce wireless resources is high as is shown by the restart ratio of mobile transactions.

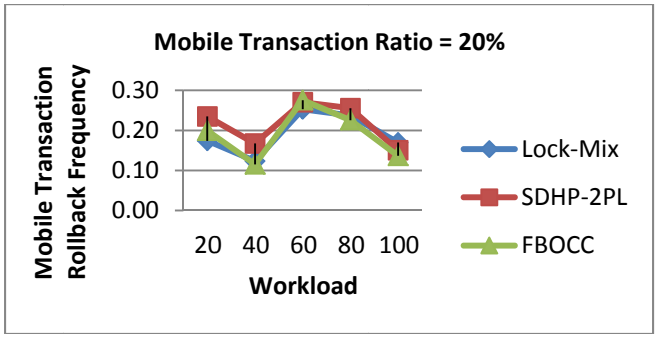

Fig 20: Mobile transaction rollback frequency (MT = 20%)

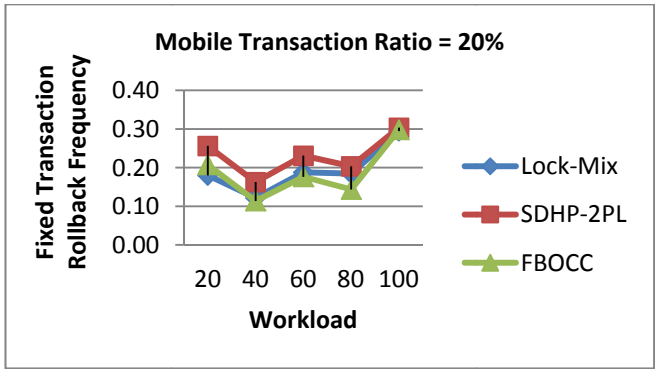

Fig 21: Fixed transaction rollback frequency (MT = 20%)

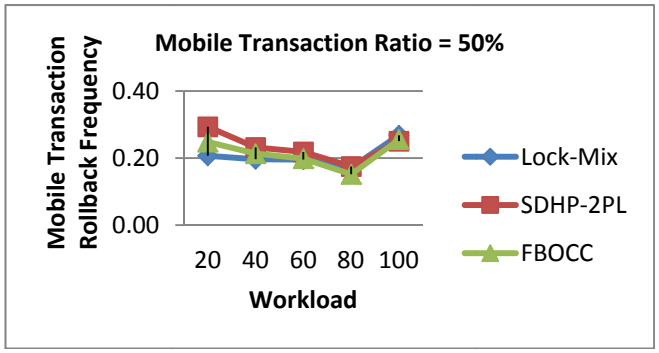

Fig 22: Mobile transactions rollback frequency (FT = 50%)

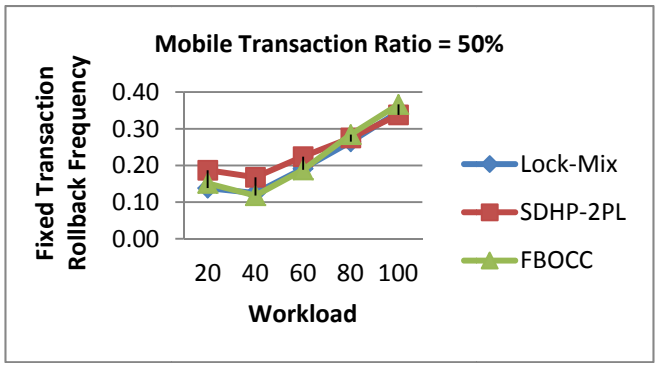

Fig 23: Fixed transactions rollback frequency (FT = 50%)

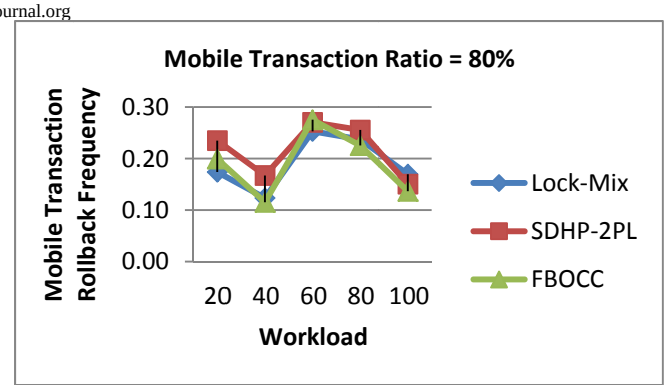

Fig 24: Mobile transactions rollback frequency (FT = 80%)

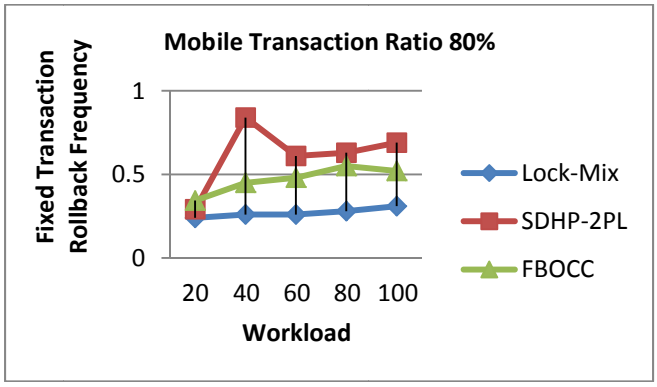

Fig 25: Fixed transactions rollback frequency (FT = 80%)

iv. Impact of μ -value and η -value

As μ -value decrease, switching to a blocking stage for fixed transaction occurred early. This will increase the restart effect caused by the fixed transactions on other mobile transactions especially when μ -value become less than η -value. Smaller values of both μ -value and η -value will increase the effect of blocking result from the interactions between mobile and other mobile or fixed transactions and vies versa. This can be seen by comparing Fig.26 and Fig.27 which show the PCR of the mobile transaction under different workload for different values of μ -value and η -value. We chose the PCR as an indication for mobile transaction because it's mainly affected by restarting and blocking overhead resulting from other mobile or fixed transactions. So, PCR can reflect such effects for both μ -value and η -value. On the other hand, reductions in PCR for mobile transactions also have an inverse trend in term of the restart rate and blocking overhead on the fixed transactions. As the Figures indicates, a choice of μ -value =0.5* η -value leads to the best value of PCR with a reasonable restarted fixed transactions. Larger values of η -value leads to more restarts in mobile transactions where smaller values of η -value increase the

effect of blocking and increase the PCR for mobile transactions.

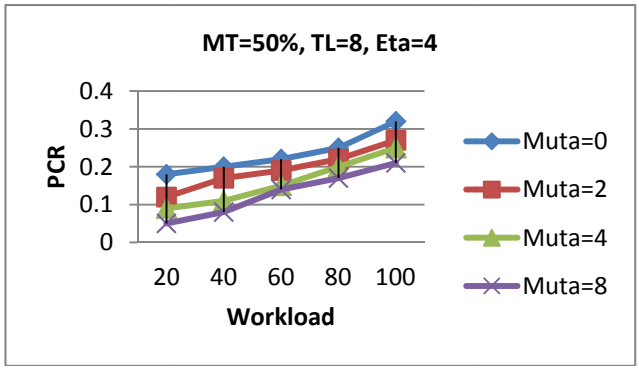

Fig 26: PCR at different μ -value

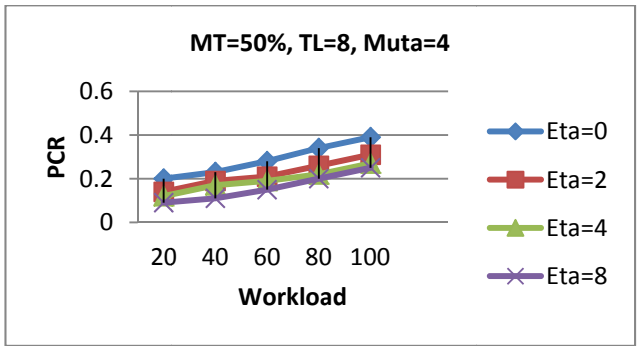

Fig 27: PCR at different η -value

6. CONCLUSIONS

Recent studies on concurrency control in mobile database system reported that the optimistic approach outperformed locking protocols in reducing the resource consumption in the wireless environments. However, most of the works based on the optimistic approach experienced the problem of unnecessary transaction restarts. This problem is detrimental to the wireless resources in mobile computing environment because transaction restarts can significantly increase the system workload and intensify resource and data contention. In mixture transactions environment, the presence of fixed transactions exacerbates the problem where the fixed and mobile transactions are equally likely to be restarted. in such environment different type of transactions use resources with different characteristics while they shared the same data access, the mobile transaction use a very scars wireless resources where the fixed transaction use the reliable fixed host resources. Since the resources used by the mobile transaction are more expensive, reducing the number of mobile transaction restarts is very important in these environments. In order to save resources while sustains a reasonable performance for fixed transaction execution, the mobile transaction abortion caused by a fixed transaction and unnecessary restart caused by other mobile transaction should be avoided. Therefore, when there are data conflicts between mobile and fixed transactions, mobile transactions are given more room than fixed transaction. As a result, mobile transactions are less likely to be aborted in mixture transaction environment.

In this study, concurrency control protocol, called Lock-Mix, is proposed to alleviate the problem in mixture transactions environment. In the Lock-Mix approach, the transaction execution can be divided into a non blocking phase where transactions wait for locks but do not block other transactions and a blocking phase as in conventional locking. Data accessed during the non blocking phase can lead to transaction aborts as in the FBOCC scheme. Since transactions in the proposed scheme explicitly wait for locks during the non blocking phase, the abort probability is reduced by avoiding accessing data items currently under validation. Furthermore, except for deadlocks, no data accesses during the blocking phase can lead to an abort of a transaction. By properly choosing the switching value η and μ , the protocol can strike a balance between the effects of transaction abort and lock wait. We show that this approach can lead to better performance at different parameters settings.

A series of simulation experiments has been done to investigate the performance of the Lock-Mix approach. And the results are reported in section IV. It is found that the proposed protocols outperform the FBOCC and SDHP-2PL concurrency protocol for a wide range of workload parameters. The first order improvement can be observed in decreasing the number of mobile transaction restarts that leads to a significant saving of resources. The second order improvement can be observed in the reduction of power consumption rate that is crucial to the mobile computing environments.

ACKNOWLEDGMENT

This research is funded by the Deanship of Research and Graduate Studies in Zarqa Private University /Jordan"

REFERENCES

[1] K. Ali and b. Ahmad," A Concurrency Control Method Based on Commitment Ordering in Mobile Databases", International Journal of Database Management Systems (IJDMS) vol.3, no.4, November 2011.

[2] Minsoo Lee, Sumi Helal, "HiCoMo: High Commit Mobile Transactions", Distributed and Parallel Databases, Vol.11, pp.73 -92, 2002, Kluwer Academic Publishers.

[3] Mohammed Khaja Nizamuddin, Syed Abdul Sattar, "Adaptive Valid Period Based Concurrency Control In: Recent Without Locking in Mobile Environments" Trends in Networks and Communications, Springer CCIS, Vol.90, Part 2, pp 349-358, 2010, Springer Heidelberg.

[4] Nadia Nouali, Anne Doucet, Habiba Drias, "A Two-Phase Commit Protocol for Mobile Wireless the Australasian Database Environment", Vol. 39, 16 Conference, 2005.

[5] Patricia Serrano-Alvarado, Claudia Roncancio, Michel Adiba, "A Survey of Mobile Transactions", Distributed and Parallel databases, 16,193-230, 2004, Kluwer Academic Publishers.

[6] D. L. R. Salman Abdul Moiz, "A Real Time Optimistic Strategy to achieve Concurrency control in Mobile Environments using ondemand multicasting," International Journal of Wireless & Mobile Networks (IJWMN), vol. 2, pp. 172-185, 2010.

[7] S. Madria, M. Baseer, V. Kumar, and S. Bhowmick, "A transaction model and multiversion concurrency control for mobile database systems," Distributed An Efficient Concurrency Control Technique for Mobile Database Environment Year 013 2 Cand Parallel Databases, vol. 22, pp. 165-196,2007.

[8] S. K. Madria, M. Baseer, and S. S. Bhowmick, "A Multi-version Transaction Model to Improve Data Availability in Mobile Computing," in International Conferences DOA, CoopIS and ODBASE 2002, pp. 322-338.

[9] C. Sung Ho, L. JongMin, H. Chong-Sun, and L. WonGyu, "Hybrid concurrency control for mobile computing," in High Performance Computing on the Information Superhighway, 1997. HPC Asia '97, 1997, pp. 478-483.

[10] A. C. Carmine Cesarano, Vincenzo Moscato, Antonio Picariello, Antonio d'Acierno, "A Hybrid Approach for Improving Concurrency of Frequently Disconnecting Transactions," International Journal of Computer Science and Network Security (IJCSNS), vol. 7, pp. 205-215, 2007.

[11] A. Chianese, A. d'Acierno, V. Moscato, and A. Picariello, "Pre-serialization of long running transactions to improve concurrency in mobile environments," in Data Engineering Workshop, 2008. ICDEW 2008. IEEE 24th International Conference on, 2008, pp. 129-136.

[12] C. S. Lee, K.-W. Lam, and S. H. Son, "Concurrency Control Using Timestamp Ordering in Broadcast Environments," The Computer Journal, vol. 45, pp. 410-42 2, January 2002.

[13] C. S. Lee, K. Wa Lam, and T.-W. Kuo, "Efficient validation of mobile transactions in wireless environments," Journal of Systems and Software, vol. 69, pp. 183-193, 2004.

[14] Salman Abdul Moiz ,Dr. Lakshmi Rajamani, "Concurrency Control Strategy to Reduce Frequent Rollbacks in Mobile Environments," CSE, vol.2, pp.709-714, 2009 International Conference on Computational Science and Engineering, 2009.

[15] P.S. Yu, M. Daniel, D.M. Dias, Performance analysis of concurrency control using locking with deferred blocking, IEEE Transactions on Software Engineering 19 (10) (1993) 982–996.

[16] Vasileva, S. Algorithm for Primary Copy Locking with Timestamp Ordering. // V International Conference on Information Systems and GRID Technologies, organized by University of Sofia "St. Kliment Ohridski" and BulAIS - Bulgarian Chapter of AIS, 27--28 may 2011, Sofia, Bulgaria, 2011, pp. 236—24

[17] Ho-Jin Choi, Byeong-Soo Jeong, "A TimestampBased Optimistic Concurrency Control for Handling Mobile Transactions", ICCSA 2006, LNCS 3981, pp. 796 -805, 2006.

[18] Ahmad alQerem, " Impact of mobility on concurrent transactions mixture" Proceeding of the international conference on Computers, digital communications and computing Pages 116- 123 CDCC'11 USA ©2011 ISBN: 978-1-61804-030-5, 2011

[19] Q. Lu, M. Satyanaraynan, Improving data consistency in mobile computing using isolation only transactions, in: Proceedings of the Fifth Workshop on Hot Topics in Operating Systems, Orcas Island, Washington, May, 1995.

[20] Ahmad al-Qerem, "Framework for Transaction Execution Strategies in Mobile Data Base Systems," International Journal of Electronics and Electrical Engineering, Vol. 1, No. 1, pp. 15-18, March 2013. doi: 10.12720/ijeee.1.1.15-18

[21] Kam-yiu Lam , Tei-wei Kuo , Gary C. K. Law , Wai-hung Tsang A Similarity-Based Protocol for Concurrency Control in Mobile Distributed Real-Time Database Systems * Parallel and Distributed

Processing Lecture Notes in Computer Science Volume 1586, 1999, pp 329-338.

AUTHOR'S PROFILE

Ahmad Alqerem obtaining a BSc in 1997 from JUST University and a Masters in computer science from Jordan University in 2002. PhD in mobile computing at Loughborough University, UK in 2008. He is interested in concurrency control for mobile computing environments, particularly transaction processing. He has published several papers in various areas of computer science. After that he was appointed a head of internet technology Depts. Zarka University