Impact of Mobility on Concurrent Transactions Mixture

Ahmad Alqerem

Abstract—**This paper presents a simulation analysis of the impact of mobility on concurrent transaction processing over a mixture of mobile and fixed transactions. Our results show that the traditional concurrency protocol yield low throughput in mixture mobile and fixed environments due to the fact that traditional protocol cannot discriminate implications of restart of both types of transactions. As a result, the number of restarts can be unbounded and the ongoing mobile transactions that share the same wireless resources will be affected. We investigate adaptive adjustment the serialization order of transactions such that a conflict resolution should be give more rooms for mobile transaction. We show that by effectively capitalizing this information, the number of unnecessary restarts can be highly reduced. Based on the simulation analysis, we show that this protocol provide a significant performance gain in mixed transaction environments.**

***Index Terms*—concurrency control, mixed transaction, mobile computing, performance Evaluation.**

I. Introduction

THIS document is a template for Word (doc) versions. If you are reading a paper version of this document, so you can use it to prepare your manuscript. Concurrency control protocols proposed in the previous works [2], [3], [4] and [5] 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 the high expensive resources used by mobile transactions, it will be a serious problem for mobile transactions to suffer from restarts caused by other fixed or mobile transactions. As a result, the conflict resolution should be in favour of mobile transaction. In MDBS, transaction restart has two negative implications. First, since the number of restarts can be unbounded, resources and time spent on the restarted transactions will be wasted. Second, reserving extra resources and time to process restarted mobile transactions may seriously affect other ongoing mobile transactions that share the same wireless resources. In view of these two factors, it is very important to reduce the number of transaction restarts which negatively affect amount of resources wasted in this aspect, save the expensive resources

Ahmad Alqerem Internet Technology Department Zarqa University P.O.Box 132222 Code No. 13132 Zarqa-Jordan (Ahmad_qerm@zu.edu.jo)

of the wireless environment and improves the overall system performance. In this paper, to reduce the number of restarts, a new optimistic based concurrency control protocol, called OCC-Mix, is proposed. By deferred adjustment the serialization order of transactions, the OCC-Mix protocol can successfully resolve many conflicts between the validating transaction and the executing transactions. Consequently, the number of unnecessary restarts can be highly reduced. The characteristics of these protocols have been examined in detail by simulation. The results show that the performance of these protocols is consistently better than the pure OCC protocol over a wide range of system settings. In particular, these protocols provide a more significant performance gain in mixed transaction environments. The remainder of this paper is organized as follows: Section 2 describe the mixed system model. Section 3 discusses the differences between fixed timestamp and timestamp interval. The OCC-Mix approach is described in Section 4. In Section 5, we discuss the performance results and we conclude the study in Section 6.

II. Mixed System model.

This section is to define a mixed system model Fig. 1; it is assumed that the system consists of stationary and mobile part. Stationary units are classified as either fixed hosts, base stations BS or mobile support station MSS. Fixed host are database server connected to the existing wired network, this server is accessible by two types of users, users whom used the wired reliable network and the second type of users are not capable to connect directly to these information servers. A MSS is equipped with a wireless interface and work as a coordinator and communication interface between the mobile unit and the database server through BSC. 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. A fixed transaction is submitted directly to a database on the same host while the mobile transaction submitted by mobile device. Like in [3], 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.

ISBN: 978-988-19251-1-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

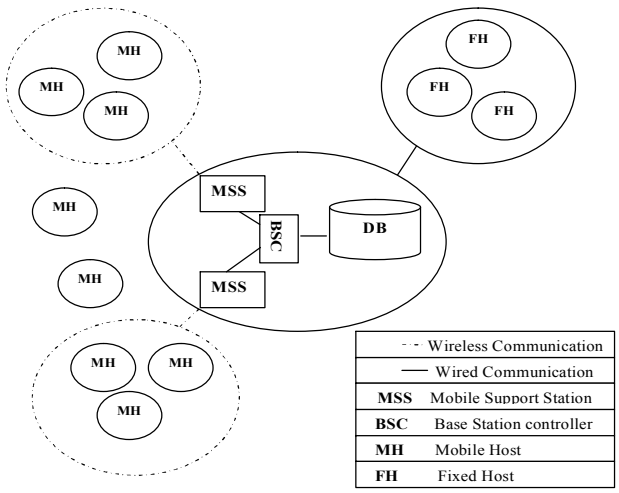

Fig 1 Mixed System Environment

III. Approach Details

Our OCC-Mix is an optimistic approach based on the dynamic adjustment of serialization order. Like OCC-TI [7], OCC-Mix uses the notion of timestamp intervals to record and represent serialization orders induced by concurrency dynamics. Timestamps are associated with both transactions and data items. Each data item has a read and a write timestamps, where the read and the write timestamps are the timestamps of last committed transactions that have read or written to the data item, respectively. For each transaction, OCC-Mix associates with each active transaction a timestamp interval expressed as a [lower bound (lb), upper bound (ub)] pair. The timestamp interval denotes the validity interval of a transaction. The timestamp intervals are also used to denote serialization order between transactions. For example, if Ti (with timestamp interval [lbi,ubi]) is serialized before Tj (with timestamp interval [lbj ,ubj]), denoted Ti → Tj, then the following relation must hold: ubi < lbj .

1) Adjustment of timestamp interval

Each transaction at the start of execution is assigned a timestamp interval of [0, ∞], i.e., the entire timestamp space. As the transaction proceeds through its lifetime in the system, its timestamp interval is adjusted to reflect serialization dependencies as they are induced. Serialization dependencies may need to be reflected while transaction is accessing data items in its read phase or by being in the conflict set of a different validating transaction.

a) Adjustment at the read phase

In this case, the timestamp interval is adjusted with regard to the read and write timestamps of the data item read or updated. In the process of adjusting, the timestamp interval may 'shut out', i.e., become empty. In that case, the transaction cannot be successfully serialized and needs to be restarted. Note that this is one of the major differences between conventional protocols and protocols based on dynamic adjustment of serialization order. In conventional OCC algorithms, restarts can only occur at validation times. In our case, however, transactions can restart at other times if a timestamp interval shut out is detected. The exact mechanics for these adjustments are shown in the procedures given in Fig. 2. In the remainder sections, we use the notation $TI(T_i)$ to denote the timestamp interval of transaction T_i and $RTS(D_i)$ and $WTS(D_i)$ to denote the read and write stamps of data item D_i ,respectively. As a transaction successfully validates, a final timestamp is assigned to it.

```
Read_Phase (Ta)
 Begin
        For each Di in RS (Ta)
          Begin
             Read (Di);
             TI (Ta) = TI (Ta) ⋂ [WTS (Di), ∞]
             If TI (Ta) = [] Then Restart (Ta);
          End
        For each Di in WS (Ta)
          Begin
             Pre-write (Di);
             TI (Ta) = TI (Ta) ⋂ [WTS (Di), ∞] ⋂ [RTS (Di), ∞]
             If TI (Ta) = [] Then Restart (Ta);
          End
 End
```

Fig 2: Adjustment of TI (T_a) at the read phase

b) Adjustment at the validation phase

In the case of being in the conflict set of a different validating transaction, the timestamp interval of the active transaction is modified to dynamically adjust the serialization order. The adjustment of the serialization order for both mobile and fixed transactions implemented with timestamp intervals creates a partial order between transactions based on conflicts and transaction type. Suppose we have a validating transaction T_v and an active transaction T_a. Let TS (T_v) be the final timestamp of the validating transaction T_v and TI (T_a) the timestamp interval of the active transaction T_a. Let TI (T_v) be the timestamp interval of the validating transaction and type (T_i) be the transaction type where type (T_i) ∈ {mobile, fixed}. We assume here that there is no blind write. So, there are two possible types of conflicts which are resolved using adjustment of serialization order between T_v and T_a: (1) *read-write* conflict which occur when the RS (T_v) ∩ WS (T_a) ≠ φ which can be resolved by forward adjustment (2) *write-read* conflict which occur when the RS (T_a) ∩ WS (T_v) ≠ φ and resolved by backward adjustment.

The adjustment of timestamp intervals (TI) in Fig. 3 iterates through the read set (RS) and write set (WS) of the validating transaction (T_v). First we check that the validating transaction has read from committed transactions. This is done by checking data item's read timestamp (RTS) and write timestamp (WST).These values are fetched when the read/write operation to the current data item is made. Then the algorithm iterates the set of active conflicting

ISBN: 978-988-19251-1-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

transactions. When access has been made to the same objects both in the validating transaction and in the active transaction, the temporal time interval of the active transaction is adjusted. Non- serializable execution is detected when the timestamp interval of an active transaction becomes empty. If the timestamp interval is empty the transaction is restarted.

```
Iterate conflicting transaction (Ta, Tv)
Begin
  For all Di ∈ (RS (Tv) ∪ WS (Tv))
    Begin
      For each Ta ∈ conflicting set of the validating transaction
        Begin
          If Di ∈ (RS (Tv) ∩ WS (Ta)) Then
          Forward adjustment (Ta, Tv);
          If Di ∈ (RS (Ta) ∩ WS (Tv)) Then
          Backward adjustment (Ta, Tv);
        End
    End
End
```

Fig 3: Iterate Readset/Writeset of Validating Transaction

(1) Forward adjustment

A read-write conflict between T_v and T_a can be resolved by adjusting the timestamp interval of the active transaction forward (i.e. $T_v \rightarrow T_a$). If the validating transaction is a mobile transaction and the active transaction is a fixed host transaction, forward adjustment is correct. If the validating transaction is a fixed host transaction and has a conflict with a mobile transaction, we should favour the mobile transaction. This is achieved by reducing the timestamp interval of the validating fixed host transaction and selecting a new final timestamp earlier in the timestamp interval see section 4. Normally, the current time or the maximum value from the timestamp interval is selected but now a different value is selected based on a predefined σ*-value*. As the σ*-value* increases, the opportunity for the active mobile transaction to commit is increases on account of the fixed host transaction. When the σ*-value* = 2 this means that the validating fixed host transaction reduces by half of its interval to the advantage of active mobile transactions. Forward adjustment can be described by procedure in Fig. 4.

```
Forward Adjustment (Ta, Tv)
  Begin
    If Tv.type = = Fixed Then
      If Ta.type = = Mobile Then
        TS(Tv)' = min(TI(Tv)) + ⌊(TS(Tv) − min(TI(Tv)))/σ⌋
        If TS(Tv)' > max ((TI (Ta)) Then
          Restart (Tv);
        Else
          TI (Ta) = TI (Ta) ∩ [TS(Tv)', ∞]
      Else Ta.type = =Fixed
        TI (Ta) = TI (Ta) ∩ [TS (Tv), ∞]
        If TI (Ta) = [] Then Restart (Ta)
    Else Tv.type = =Mobile
      TI (Ta) = TI (Ta) ∩ [TS (Tv), ∞]
      If TI (Ta) = [] Then Restart (Ta);
End
```

Fig 4: Forward adjustment

(2) Backward adjustment

A read-write conflict between T_v and T_a can be resolved by adjusting the timestamp interval of the active transaction backward, (i.e. $T_a \rightarrow T_v$). If the validating is a mobile transaction and the active conflicting is a fixed transaction, backward adjustment is correct. If the validating is a fixed transaction and the active conflicting transaction is mobile, then backward adjustment is done if the active transaction is not aborted in backward adjustment. Otherwise, the validating transaction is restarted. This is wasted execution, but it is required to ensure the execution of the mobile transaction. In backward adjustment, we cannot move the validating transaction to the future to obtain more space for the mobile transaction. We can only check if the timestamp interval of the mobile transaction would become empty. In forward ordering we can move the final timestamp backward if there is space in the timestamp interval of the validating transaction. Again we check if the timestamp interval of the mobile transaction would shut out. We have chosen to abort the validating transaction when the timestamp interval of the mobile transaction shuts out. Thus, this protocol favours the mobile transaction that uses scarce and expensive resources. Backward adjustment can be described by procedure in Fig. 5.

```
Backward Adjustment (Ta, Tv)
 Begin
   If(Tv.type = = Fixed) Then
     If(Ta.type = = Mobile) Then
       If (TS (Tv) − 1 < Min (TI (Ta)) Then
         Restart (Tv);
       Else
         TI (Ta) = TI (Ta) ∩ [0, TS (Tv)-1]
     Else Ta.type = = Fixed
       TI (Ta) = TI (Ta) ∩ [0, TS (Tv)-1]
       If TI (Ta) = [] Then Restart (Ta)
   Else Tv.type = =Mobile
     TI (Ta) = TI (Ta) ∩ [0, TS (Tv)-1]
     If TI (Ta) = [] Then Restart (Ta)
 End
```

Fig 5: Backward Adjustment

2) Final timestamp selection

In the OCC-Mix protocol we should select the final (commit) timestamp TS (T_v) in such a way that room is left for backward adjustment. In our validation algorithm we set TS (T_v) as the validation time if it belongs to the time interval of T_v or the maximum value from the time interval otherwise.

IV. Performance evaluation

We have constructed and conducted a series of simulation experiments to evaluate and compare the performance of our concurrency control approaches 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.

We have developed a simulation system, as shown in Fig. 6, to study the performance of the proposed protocols in a MDBS and to identify their performance characteristics.

ISBN: 978-988-19251-1-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

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).

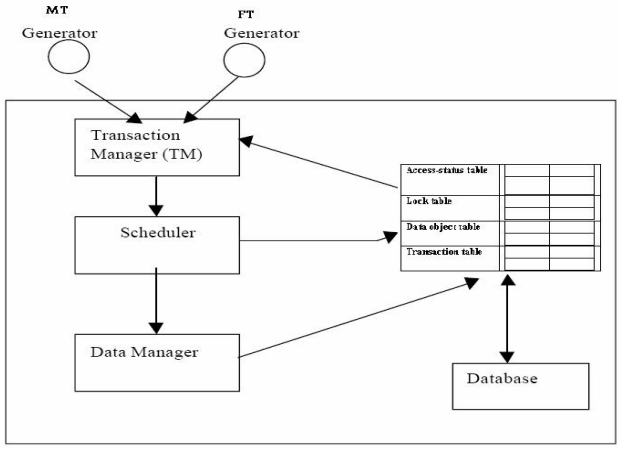

Fig 6: Simulation Model

We also did the same for the optimistic protocols, a data object table and a transaction table are maintained. The data object table keeps a read timestamp and a write timestamp for each data object in the database and the transaction table maintains the read set, the write set and the timestamp interval of each transaction.

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 [1] and [6]. 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 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.

A. Parameter Setting

Table 1 gives the parameters that control system resources. The parameters, *CPUTime* and *DiskTime* capture the CPU and disk processing times per data item. Table 1 also summarizes the key parameters that characterize system workload and transactions. The numbers of data objects accessed by a transaction are determined uniformly from the database. A data object that is read is updated with the probability, *WriteProb*.

Table 1: 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
CPUTime	2 Time unit /Operation
DiskTime	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
Transaction parameter	
Multiprogramming level	50 concurrent Transactions of both fixed and mobile
Mobile transaction Percentages	20%, 50%, 80%
Fixed host transaction length ($N_{operations}$)	3 - 15
TBA of fixed transaction	2 – 5 uniformly
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 *WriteProb*	0.5

B. Performance Metrics

In MDBS, the major performance measure that can demonstrate the effectiveness of the protocols is the restart ratio. The restart ratio measures the average number of

ISBN: 978-988-19251-1-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

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 last performance measure specifically for the OCC-Mix approach is the adjustment ratio, which is the number of serialization order adjustment made per transaction. This ratio can help to understand the effectiveness of the OCC-Mix approach in different parameter settings. The formulas of the main performance measures are listed below.

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

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.

C. Experiments and Results

We have performed three sets of experiments. In the first set, mobile transactions percentage is 20%. For the second and the third sets, the utilizations of the mobile transactions are approximately 50 percent and 80 percent, respectively. We can therefore observe whether the domination of any one kind of transaction will have any impact on the performance of the implemented approaches. All these experiments show that the OCC-Mix approach outperforms both the pure OCC protocol and the 2PL protocol for a wide range of parameter settings.

a) Impact of Mobility

Fig. 7 gives the PCR as a function of the mean mobility value when 20% of transactions present in the system are mobile transaction. Since the number of base station crossed over by a mobile transaction increase when the mobility increase, the delay between mobile transaction operations increase And the average numbers of active transactions increase, the blocking of 2PL protocol badly affect the PCR. For the OCC protocol, there is no blocking overhead. But the PCR reduction problem still exist because of mobile transaction restarts, the improvement made by OCC-Mix approach over other protocols is clear especially when the percentage of mobile and fixed transactions are equal as we can see in Fig. 8. This is because of OCC-Mix have no blocking overhead compared to the 2PL and have less number of restarts than OCC. Fig. 9 shows the PCR under all protocols when the mobile transaction are dominated, since the increase of PCR under OCC and OCC-Mix are mostly caused by the restart frequency, the difference between these protocols is reduced as we see in Fig. 9 when the percentage of mobile transaction is 80%. So the best improvement which can be gained by the OCC-Mix when the percentages of mobile and fixed transactions are equals in the system.

b) Impact of σ-value

Sigma value can highly affect the nature of interactions between fixed and mobile transactions. A larger σ-value means a little opportunity for fixed transactions to commit in their time intervals. Fig. 10 shows the fixed rollback frequency FRF for different values of σ. This measure determines the number of fixed transactions aborted because they are in conflict with other mobile transactions. As we can see in Fig. 11, the number of fixed transactions roll backed by other mobile transactions increase as the σ-value increase. A choice of σ-value =2 leads to the smallest value of *FRF*. Smaller σ-value leads to more restarts in mobile transactions as we can see in the Fig. 11. For mobile rollback frequency *MRF* which mean that the adjustments of an active mobile transaction that is in conflict with a validating fixed transaction whose timestamp interval bigger will short the timestamp intervals of other mobile and fixed transactions which may be in conflict in the future. As we shown in Fig. 10 and 11, intermediate value of σ leads to the best reduction of both fixed and mobile rollback frequency.

V. Conclusion

In this paper, a concurrency control approach called OCC-Mix has been proposed to alleviate the problem in mixed transactions environment. The idea is to avoid wasting of scars and expensive resources of mobile environment by (1) avoid unnecessary transaction restarts by dynamically adjusting the serialization order of the conflicting transactions with respect to the validating transaction and (2) make more room for mobile transaction which gives it a better chance to commit. Under the OCC-Mix approach, These done by exploiting the semantics between read and write operations in transactions in addition to the transaction type such that the serializability can be preserved without restarting the conflicting transactions of the validating transaction. Only those transactions with serious conflicts with the validating transaction have to be restarted. In case of non serious conflicts, we only need to adjust the serialization order of those conflicting transactions with respect to the validating transaction. As a result, unnecessary restarts can be removed. In addition to allowing those non serious conflicting fixed and mobile transactions to have opportunity to complete their executions, resources can be saved from being utilized by restarted transactions such that other ongoing transactions will not be affected. A series of simulation experiments has been done to investigate the performance of the OCC-Mix

ISBN: 978-988-19251-1-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

approach. It is found that the proposed protocols outperform the traditional optimistic 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.

REFERENCES

[1] Akar, M., & Mitra, U."Motion Constraint Based Handoff Protocol for Mobile Internet," IEEE Wireless Communications and Networking Conference (WCNC), March 2003] [80[(2003). Soft handoff algorithms for CDMA cellular networks. IEEE Transactions on Wireless Communications,2 (6), 1259–1274..

[2] C. Boksenbaum et al. Concurrent certifications by intervals of timestamps in distributed database systems. IEEE Trans. on Software Engineering, SE-13(4):409--419, Apr. 1987

[3] Angelo Brayner and José A. Morais F., "Increasing Mobile Transaction Concurrency in Dynamically Configurable Environments", Proceedings of the 3rd. IEEE Workshop on Mobile Distributed Computing (MDC). 2005.

[4] B. Lim, A. R. Hurson, K. M. Kavi. Concurrent Data Access in a Mobile Heterogeneous System. Proceedings of the 32nd Annual Hawaii International Conf. on System Sciences. 1999.

[5] P. Serrano-Alvarado, C. Roncancio and M. E. Adiba: A Survey of Mobile Transactions, Distributed and Parallel Databases, 16(2), 2004, pp 193-230.

[6] M. A. Viredaz, L. S.Brakmo, and W. R. Hamburgen, "Energy Management on Handheld Devices," Queue, vol. 1, no. 7, pp. 44–52, 2003.

[7] EE·Ho-Jin Choi, Byeong-Soo Jeong: "A Timestamp-Based Optimistic Concurrency Control for Handling Mobile Transactions". ICCSA (2) 2006: page796-805.

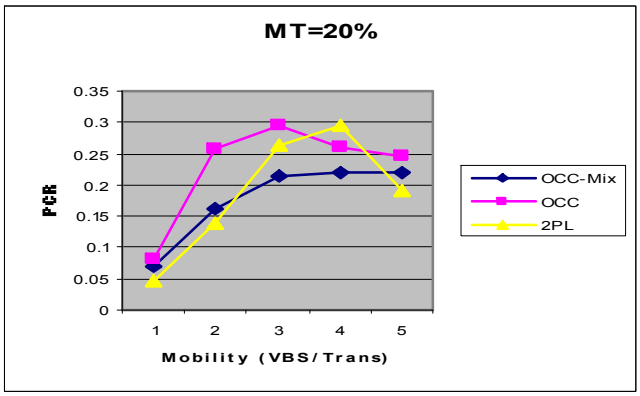

Fig 7: Power consumption rate (MT = 20%)

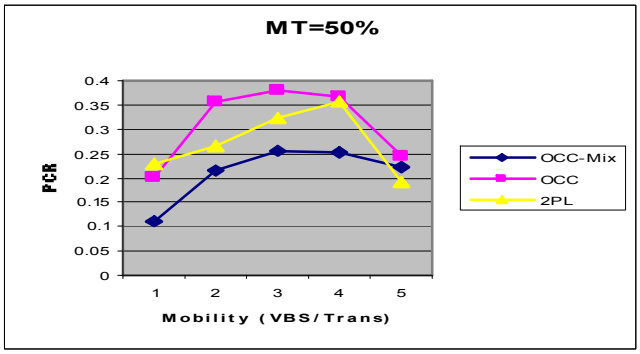

Fig 8: Power consumption rate (MT = 50%)

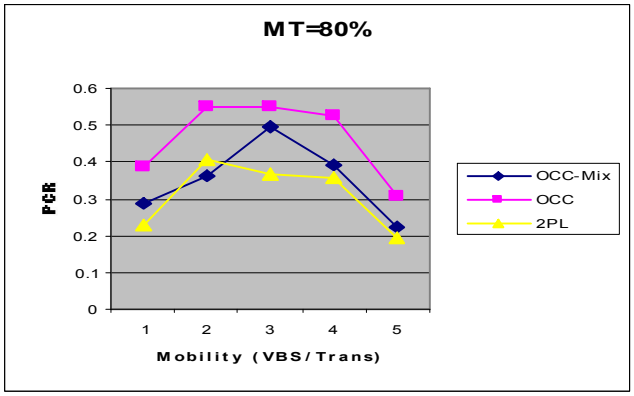

Fig 9: Power consumption rate (MT = 80%)

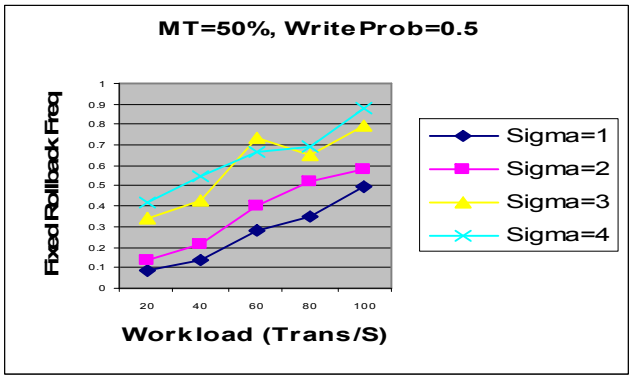

Fig 10: Fixed rollback Frequency

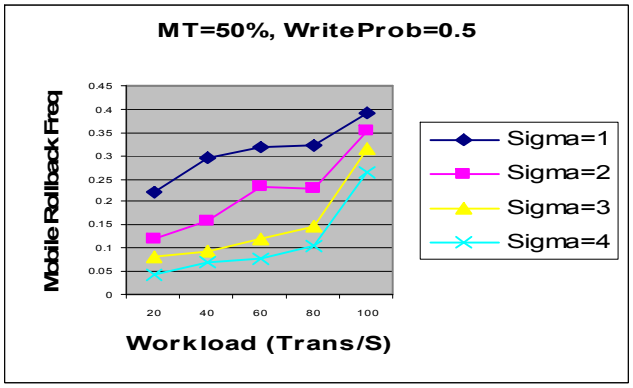

Fig. 11: Mobile rollback Frequency

ISBN: 978-988-19251-1-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)