Concurrency Control for Mobile Transactions

Alaa Alnaimat
Department Software Engineering
Zarqa University
Address (132222 zarqa 13132 Jordan)
a.alnaimat@zu.edu.jo

Abstract: - As technological advances are made in software and hardware, the feasibility of accessing information "any time, anywhere" is becoming a reality. In a mobile computing environment, a potentially large number of mobile and fixed users may simultaneously access shared data; . In this paper, we have addressed the problem and devised an effective OCC protocol with deferred adjustment of serialization order, called OCC-DF, for MDBS with mixed transactions. The characteristics of the OCC-DF protocol have been examined in detail by simulation. The results show that the performance of the OCC-DF protocol is consistently better than OCC-TI protocol over a wide range of system settings. In particular, the OCC-DF protocol provides a more significant performance gain in mixed transaction environments.

Key-Words — Mobile transaction, concurrency control, mobile computing, mixed database, timestamp interval.

1 Introduction
As technological advances are made in software and hardware, the feasibility of accessing information "any time, anywhere" is becoming a reality. In a mobile computing environment, a potentially large number of mobile and fixed users may simultaneously access shared data; therefore, there is a need to provide a means to allow concurrent management of transactions. Recent studies [1], [2] have suggested that the optimistic concurrency control (OCC) protocols outperform the locking-based protocols in mobile database systems (MDBS). However, the OCC protocols suffer from the problem of unnecessary transaction restarts. The problem is more intensified in mixed transactions environment where both fixed and mobile transactions exist. Mobile transactions are more vulnerable to restarts when they are in conflict with mobile transactions on data access due to limitation imposed by mobile environment. In this paper, we have addressed the problem and devised an effective OCC protocol with deferred adjustment of serialization order, called OCC-DF, for MDBS with mixed transactions. This protocol can 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. As a result, much resource can be saved and more fixed host transactions can finish their execution without affecting the execution of mobile transactions. The remainder of this paper is organized as follows. In section 2 we review the mobile database system with mixed transaction. Section 3 gives the protocol details and explanation of conflict resolution strategies. Section 4 present the experimental result and finally we conclude the paper in section 5.

2. System Model
This section is to define a MDBS model figure 1 it is assumed that the database system consists of two major components: the transaction manager (TM) and location manager .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's re their responsibility to adjust the data items and transactions based on read and write operation and transaction conflict during validation process. The scheduler maintains an access-status table to detect any possible data conflicts. All data access requests issued by an transaction are handled by the DM, which retrieves the required data item. There are two classes of transactions in the system: fixed host transaction and mobile transactions,. Each transaction consists of a sequence of read and writes operations, and ends with a commit or an abort operation.. Although the resources consumed by the fixed host are cheaper than in mobile transaction, it is desirable to have a small mean response time for the fixed host transactions.
3. Approach

In this section a validating algorithm OCC-DF is presented. OCC-TI is extended with a new final timestamp selection method and type-depended conflict resolution. We should select the final (commit) timestamp TS(Tv) in such a way that room is left for backward adjustment.

We propose a new validation algorithm where the commit timestamp is selected differently. In our validation algorithm we set TS(Tv) as the validation time if it belongs to the time interval of Tv or the Maximum value from the time interval otherwise. Additionally, the original OCC-TI is extended to adjust the serialization order based on transaction type. We have also used a deferred dynamic adjustment of serialization order. In the deferred dynamic adjustment of serialization order all adjustments of timestamp interval are done to temporal variables. The timestamp intervals of all conflicting active transactions are adjusted after the validating transaction is guaranteed to commit. If a validating transaction is aborted no adjustments are done. 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. Finally, current read timestamps and write timestamps of accessed data items are updated and changes to the database are committed. Figure 2(a)-(b) present forward and backward adjustment algorithm for dynamic adjustment of the serialization order using time stamp intervals with deferred dynamic adjustment based on transaction type.

A backward and forward adjustment algorithm creates order between conflicting transaction timestamp intervals. A final (commit) timestamp is selected from the remaining timestamp interval of the validating transaction. Therefore the final timestamps of the transactions create partial order between transactions.

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 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. Finally, current read timestamps and write timestamps of accessed data items are updated and changes to the database are committed. Figure 2(a)-(b) present forward and backward adjustment algorithm for dynamic adjustment of the serialization order using time stamp intervals with deferred dynamic adjustment based on transaction type.

![Figure 1: System Model](image)

![Figure 2(a): Forward adjustment](image)
Figure 2(b): Backward adjustment

4. Performance Evaluation
A mixed transaction MDBS is modeled and a set of experiments are carried out in order to examine the performance of our algorithm in comparison with OCC-TI protocol. We used two different type of transaction, fixed and mobile host, transactions are validated atomically, the workload in each experiment consist of a variable mix of transaction. Fraction of each transaction type is a test parameter, the database size, the arrival rate for both fixed and mobile transactions and other test parameter are summarize in table 1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrival rate of both</td>
<td>100 - 500 transaction/s</td>
</tr>
<tr>
<td>transactions</td>
<td></td>
</tr>
<tr>
<td>Fixed transaction length</td>
<td>5 - 10 uniformly</td>
</tr>
<tr>
<td>(number of operation)</td>
<td>distributed</td>
</tr>
<tr>
<td>Mobile transaction length</td>
<td>5 - 10 uniformly</td>
</tr>
<tr>
<td>(number of operation)</td>
<td>distributed</td>
</tr>
<tr>
<td>Fixed transaction writes</td>
<td>0.5</td>
</tr>
<tr>
<td>probability</td>
<td></td>
</tr>
<tr>
<td>Mobile transaction</td>
<td>0.3</td>
</tr>
<tr>
<td>writes probability</td>
<td></td>
</tr>
<tr>
<td>Database size</td>
<td>300</td>
</tr>
<tr>
<td>Number of transaction</td>
<td>10000</td>
</tr>
</tbody>
</table>

Table 1: model parameter and their baseline settings

The experiments are conducted to compare the abort ratio in our protocols with OCC-TI; the arrival rate of the transactions is varied from 100 to 500 transactions per second. In Figure 3(a)–(b) the fraction of mobile transactions is varied from 20% to 30%.

In Figure 3 shows that OCC-DF is better than OCC-TI, especially when the arrival rate is high. This is because the OCCCDF does not suffer from the unnecessary restart problem. Figures 4(a)–(b) show the abort ratio of mobile transactions. This demonstrates how the OCC-DF favors mobile transactions. OCC-DF clearly offers better chances for mobile transactions to complete their execution while conflicting with fixed host transaction. The results clearly indicate that OCC-DF meets the goal of favoring mobile transactions.

5. Conclusion
We proposed two extensions to the basic conflict resolution method used in OCC-TI. OCC-DF includes a new final timestamp selection method and conflict resolution based on the transaction type. We have demonstrated that the OCC-DF produces a correct result. Our results from experiments showed that the revised OCC-DF outperforms original OCC-TI. Additionally, the OCC-DF clearly offers better chances for mobile transactions to complete in present of fixed host transaction. The results clearly indicate that the OCC-DF meets the goal of favoring mobile transactions. Therefore the OCC-DF is a promising candidate for mixed mobile fixed host transaction management in database systems where transactions are heterogeneous.

References:
Figure 3: (a) mobile transaction fraction 20%

Figure 4: (b) mobile transaction fraction 30%