A Hybrid Distributed Optimistic Concurrency Control Method for High-Performance Real-Time Transaction Processing

QIN Biao (覃 飙) and LIU YunSheng (刘云生)

College of Computer Science and Technology, Huazhong University of Science and Technology Wuhan 430074, P.R. China

E-mail: biaoqin@263.net

Received April 16, 2001; revised May 13, 2002.

Abstract The conventional lock scheme tends to suffer from a cascade of blockings, while the optimistic concurrency control (OCC) scheme may suffer from wasting resources. To overcome these problems, some researchers have proposed a combination of OCC and lock in transaction processing. Using this method, Thomasian proposed the hybrid method for conventional distributed transaction processing, and Lam proposed the DOCC-DA protocol for distributed real-time database system based on forward validation. This paper proposes a new protocol, called Hybrid Distributed Optimistic Concurrency Control Embedded in two-Phase Commit, which is based on back validation. The new protocol makes use of access invariance and runtime information which can guarantee a rerun transaction to meet its deadline and abort the fruitless run transactions as early as possible. A series of simulation experiments have been done to investigate the performance of the new protocol. The results show that its performance is consistently better than that of other protocols.

Keywords distributed real-time database, optimistic concurrency control, serializability, commit protocol

1 Introduction

In [1] Haritsa showed that, in firm or hard realtime scenarios, optimistic concurrency control (OCC) outperforms locking over a large spectrum of conditions. The main reason is that in OCC scheme conflict resolution is performed at the end of a transaction execution and restarting other conflicting transactions must result in a transaction commitment, while in lock-based protocols a transaction that restarts other conflicting transactions may eventually be aborted and discarded[2]. In OCC, a transaction T_i executes in three different phases during its lifetime. Using this property, some researchers have proposed a kind of phase-dependent control technology, such that a transaction is allowed to have multiple execution phases with different concurrency control methods in different phases. In [3], a distributed real-time OCC protocol, called DOCC-DA, was proposed. The protocol is based on forward validation. In [4], Thomasian proposed the hybrid OCC scheme, which is designed for conventional distributed database.

In this paper, a new protocol, called Hybrid Distributed Optimistic Concurrency Control Embedded in two-Phase Commit (HDOCC-E2PC), is proposed. The protocol is based on backward validation. The main contributions of this paper are the following. First, we propose the notions of fruitless run and fruitless rerun. Second, we propose the HDOCC-E2PC protocol, which can avoid the fruitless reruns and abort the fruitless run transactions as early as possible.

The rest of the paper is organized as follows. Section 2 describes a distributed real-time database system model. Section 3 discusses some advanced techniques used in the new protocol. Section 4 describes the new protocol. Section 5 shows a series of simulation experiments that compare the performance of the new protocol with that of the hybrid protocol and the DOCC-DA protocol. Section 6 summarizes the main conclusions of the study.

2 Distributed Real-Time Transaction Model

The transaction model used in this paper is firm real-time transactions. We modify the firm dead-

This work was supported by the National Natural Science Foundation of China under Grant No.60073045, National Defence Pre-Research Foundation under Grant No.00J15.3.3.JW0529 and National Doctoral Research Foundation.

line semantics in the distributed environment as follows[5].

Definition 1. *A distributed firm deadline real-time transaction is said to be committed if the master has reached the commit decision before the expiry of the deadline at its site. This definition applies irrespective of whether the cohorts have also received and recorded the commit decision before the deadline.*

Definition 2. *That a transaction's run will miss its deadline is called fruitless run.*

Definition 3. *That a transaction's rerun will miss its deadline is called fruitless rerun.*

Since conflicts occur when accessing data, a transaction may need to be restarted several times before it can be successfully committed. So we should avoid starvation locks. The buffer retention effect[6], which induces access invariance[7], incorporating S2PL can be used to avoid starvation lock. A high degree of access invariance is expected in a system with short and preplanned real-time transactions, which usually access the same set of objects in repeated executions. So a transaction can preorder the data if it has enough time to rerun when it fails.

In the new protocol, data conflicts are resolved by "transaction races": the transaction which reaches the validation phase first gets to survive and other conflicting transactions have to restart. There are two reasons that the new protocol chooses not priority-based but transaction races. First, once a cohort gains all locks it required, it sends the OK message to the master and reaches the prepared state. In this situation, it has to retain all its locked data until it receives the global decision from the master — this retention is fundamentally necessary to maintain atomicity. Therefore, priority inversion is unavoidable. Second, recent study[8] has shown there appears to be little advantage to gain by incorporating priority cognizance in the conflict resolution of the OCC protocols in RTDBS (Read-Time Data-Base System).

3 Property of the New Protocol

3.1 Incorporation of S2PL in OCC

The better concurrency control scheme is to use the hybrid optimistic method[5] with OCC in the read phase, which is based on the optimistic die, and lock in the next two phases. The proposed OCC method exhibits the characteristic that the transaction will execute without blocking in the read phase and hold locks only during commit phase if its validation is successful. In the conventional OCC, the validation phase and the write phase are the integral part of the critical section, which may cause validation deadlock[3]. If the write phase can be disintegrated from the validation phase by pulling it out of the critical section, the validation deadlock can be avoided. However, this disintegration introduces another problem where the global serializablility of transaction execution cannot be guaranteed even if a transaction gets local validation at all participating sites[4].

In order to solve this problem, we adopt the sequential locking method to implement the validation scheme. So each site is assigned a unique identity (SiteID) such as $1, 2, 3, \ldots, n$. SiteIDs are in a monotonically increasing order. When a transaction enters the validation phase, its master triggers the cohort in the lowest SiteID to start its local validation first irrespective of the originating site of the master. Each cohort performs its local validation according to the sequence. Updating locks in each site follows the all-or-none principle of the real-time *Static Two-Phase Locking* (S2PL). Four types of locks are utilized in the validation scheme. The VR-lock or VW-lock has to be obtained when a data object is read or written by a transaction into its own workspace in its read phase. When a transaction enters the validation phase after it has obtained all the requested VR-locks and VW-locks, these locks will be upgraded to R-locks and W-locks respectively. The compatible matrix of these locks is shown in Table 1.

Table 1. Compatibility Lock Table

Request	VR-lock held	VW-lock held	R-lock held	W-lock held
VR-lock	OK	OK	OK	NO
VW-lock	OK	OK	NO	NO
R-lock	OK	OK	OK	NO
W-lock	OK	OK	NO	NO

3.2 Incorporation of 2PC in OCC

The validation phase and the write phase can be imbedded in the commit processing. When all database operations of a transaction have been executed, the master acts as a coordinator for commit processing and sends a prepare message, which is also used as a validation request, to the cohort in the lowest SiteID to start its local validation. Upon receiving this message, the cohort performs its local validation, where it is checked whether the local serializability is affected. If local validation is successful, the cohort upgrades all its V-locks, enters a prepared state, sends an OK vote to the master and passes the validation message to the cohort at the next site downstream. Otherwise, the cohort has to abort and sends a NO vote to the master.

The second phase of the commit protocol, which is also the write phase of the OCC scheme, starts after the master receives response message from the cohort in the last node. If its validation is successful, a commit record is logged and commit messages are sent to all its cohorts where the commit processing consists of writing a commit log record and updating the database with modified objects. Otherwise it will be restarted if there is enough time left or it will be discarded if its deadline is due. The new protocol's outline is described in Fig.1.

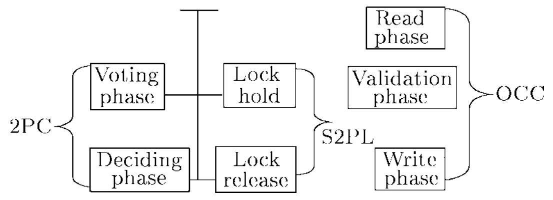

Fig.1. Outline of the HDOCC-E2PC protocol.

```
Read_phase($T_i, D_x$){
For every data object $D_x$ to be accessed Do
    If read request Then
        If $W$-lock exists or is preordered Then
            If $E(T_i) < D(T_i)$-clock Then
                Block $T_i$;
            Else
                Abort and discard $T_i$;
            Endif
        Else
            $RS(T_i) = RS(T_i) \cup D_x$;
            $RTS(D_x, T_i) = R$- timestamp$(D_x)$;
            $T_i$ VR-lock $(D_x)$;
        Endif
    Else
        If R-lock or W-lock exists or is preordered
        Then
            If $E(T_i) < D(T_i)$-clock Then
                Block $T_i$;
            Else
                Abort and discard $T_i$;
            Endif
        Else
            $WS(T_i) = WS(T_i) \cup D_x$;
            $WTS(D_x, T_i) = W$-timestamp$(D_x)$;
            $T_i$ VW-lock$(D_x)$;
        Endif
    Endif
Endfor }
```

Fig.2. The read-phase of a cohort.

4 HDOCC-E2PC Protocol

In each site, a data object table and a transaction table are maintained. The former keeps a read and a write timestamp for each data object. *R-timestamp*(D_x) denotes the largest timestamp of any transaction that has executed *read*(D_x) successfully; *W-timestamp*(D_x) denotes the largest timestamp of any transaction that has executed *write*(D_x) successfully. The latter maintains the following information for each local active transaction or cohort T_i. $RS(T_i)$ denotes the read set of T_i; $WS(T_i)$ denotes the write set of T_i; $RTS(D_x, T_i)$ and $WTS(D_x, T_i)$ denote the R-timestamp and W-timestamp of the copy of data object D_x as seen by transaction T_i respectively.

4.1 Read Phase

At any site, when the cohort of T_i wants to read or pre-write a data object D_x in its private workspace, it will first obtain the VR-lock or VW-lock. These locks will be granted according to the algorithm described in Fig.2.

4.2 Validation Phase

The master initializes the validation phase and the first phase of 2PC by sending *prepared* message to the cohort in the lowest site, which is described

```
Validation_phase_master($T_i$){
    If $T_{\text{stage2}} < D(T)$-clock Then
        Send validation and PREPARE messages
        to the lowest SiteID cohort;
    Else
        Send abort message to all of its cohorts;
    Endif }
```

Fig.3. The validation phase of the master.

```
Validation_phase_cohort($T_i$){
  Valid = true;
  Upgrade_VR_lock($T_i, D_x$);
  Upgrade_VW_lock($T_i, D_x$);
  If valid Then
    Force-write a prepare log record;
    Send an OK message to the master;
  Else
    If $E_{\text{restart}}(T_{\text{val}}) < D(T_{\text{val}})$-clock Then
    Preorder the data objects it will access;
  Write an abort record;
  Send a NO message to the master;
  Abort the cohort;
Endif; }
```

Fig.4. The validation phase of a cohort.

in Fig.3. The algorithm in Fig.4 describes that data conflicts are resolved by combining transaction races with preordering. The algorithms in Fig.5. and Fig.6, which are based on the timestamp-ordering protocol, describe how VW-lock upgrades to W-lock and how VR-lock upgrades to R-lock respectively.

```
Upgrade_VW-lock($T_i, D_x$){
  If W-lock or R-lock exists or is preordered Then
    Valid = false;
  Endif
  If $WTS(D_x, T_i) < R$-timestamp$(D_x)$ or
    $WTS(D_x, T_i) < W$-timestamp$(D_x)$ Then
    Valid = false;
  Else Upgrade VW-lock($D_x$) to W-lock($D_x$);
  Endif
}
```

Fig.5. Upgrade VW_lock.

```
Upgrade_VR-lock($T_i, D_x$){
  If W-lock exists or is preordered Then
    Valid = false;
  Endif
  If $RTS(D_x, T_i) < W$-timestamp$(D_x)$ Then
    Valid = false;
  Else
    Upgrade VR-lock($D_x$) to R-lock($D_x$);
  Endif
}
```

Fig.6. Upgrade VR_lock.

4.3 Write Phase

In this phase, the master performs the global conflict detection based on each cohort's message. The write phase algorithms are described in Fig.7,

```
Write_phase1_master($T_i$){
  If NO message is received or $D(T) <$ clock Then
    Force-write an Abort log record;
    Send Abort message to the cohorts;
  Else
    Force-write a Commit message;
    Send Commit message to all the cohorts;
  Endif }
```

Fig.7. Write phase1_master.

```
Write_phase_cohort($T_i$){
  If Abort message is received Then
    If $E_{\text{restart}}(T_{\text{val}}) < D(T_{\text{val}})$-clock Then
      Preorder the data it will access;
  Endif
      Force-write an Abort log record;
      Send an Ack message to the master;
  Else
    Force-write a Commit log record;
    Send an Ack message to the master;
  Endif}
```

Fig.8. Write phase_cohort.

```
Write_phase2_master($T_i$){
  After receiving Ack from all cohorts;
  Write an End log;
  If the transaction has already committed Then
    For each $D_x$ in $RS(T_i)$ Do
      Update $R$-timestamp$(D_x)$;
    For each $D_x$ in $WS(T_i)$ Do
      Update $W$-timestamp$(D_x)$;
    Release all its locks;
    Forget the transaction;
  Else
    If $E_{\text{restart}}(T_{\text{val}}) \leqslant D(T_{\text{val}})$-clock Then
        Restart $T_{\text{val}}$;
    Else
        Release all its locks;
        Discard $T$;
    Endif
  Endif}
```

Fig.9. Write phase2_master.

Fig.8 and Fig.9. From these algorithms, we can find the write phase is also the second phase of the 2PC.

4.4 Correctness

In this section, we give an argument on the correctness of the new protocol.

Theorem 1. *The sequential locking method can ensure local serializability.*

Proof. In the sequential locking method, updating locks in the local site is based on S2PL. A cohort has to update all of its required locks before finishing the local validation. If any one of its requesting locks is being used by another transaction, its validation fails. Then all seized locks have to be released and preordered. So the scheme is deadlock free in the local site because failing cohort cannot hold any locks. □

Theorem 2. *The sequential locking method can ensure global serializability when each site is locally serializable.*

Proof. In the sequential locking method, SiteIDs are sorted in a monotonically increasing order. We prove there cannot be a circular wait by assuming that a circular wait exists. Let the set of transactions involved in the circular wait be $\{T_0 \to T_1 \to \cdots \to T_n \to T_0\}$, where T_i is waiting for upgrading locks on data items located in SiteID$_i$, which are being locked by transaction T_{i+1}. Then since transaction T_{i+1} is holding locks on data items located in SiteID$_i$, while it is requesting locks on data items located in SiteID$_{i+1}$, we have already assumed SiteID$_i <$ SiteID$_{i+1}$. So this condition means SiteID$_0 <$ SiteID$_1 < \cdots <$ SiteID$_n <$ SiteID$_0$. By transitivity, SiteID$_0 <$ SiteID$_0$ is impossible. Therefore, there

Table 2. Baseline Values for the Model Parameters

N_{site}	Number of sites	4
SiteID	Site IDs	0, 1, 2, 3
AR	Arrival rate	4 transactions/s
T_{com}	Communication delay	100 ms (constant)
SF	Slack factor	1—4 (uniform distribution)
P_{write}	Write operation probability	0.0–1.0
PageCPU	CPU page processing time	5ms
PageDisk	Disk page processing time	20ms
DBSize	Database size	200 data objects/site
N_{oper}	Number of operations in a transaction	3–20 uniformly distributed

can be no circular wait. So the scheme is free of deadlock in the viewpoint of global transaction and the global serialization order is thus given by the validation order. □

Corollary 1. *The HDOCC-E2PC protocol can ensure global serializability.*

Theorem 3. *The HDOCC-E2PC protocol is the 2PC protocol with time cognizance.*

Proof. The new protocol is an extension of the 2PC protocol. It takes a transaction deadline into consideration when validating and making decision for commitment. But the basic property of 2PC does not change. So the HDOCC-E2PC protocol is the 2PC protocol with time cognizance. □

Corollary 2. *The HDOCC-E2PC protocol can ensure atomic commitment.*

5 Performance Evaluation

5.1 System Model

The model consists of a database that is distributed, in a non-replicated manner, over 4 sites connected by a network. We choose the sequential transaction model. Each transaction is assigned with a deadline based on the application requirements. The transaction's expected deadline is defined as follows: $D(T) = T_{ar} + E(T) * SF$, where T_{ar} is the arrival time of the transaction, SF is a random variable uniformly distributed between two bounds, and $E(T)$ is a transaction's process time.

In our model, all cohorts inherit their master's priority. Messages also retain their sending transaction's priority[5]. The transaction priority assignment used in all of the experiments described here is the widely-used Earliest Deadline First (EDF) policy.

The major performance measure is the miss rate, which is defined as: $MissPercent = T_{\text{miss}} / T_{\text{total}}$, where $T_{\text{miss}} =$ number of transactions missing their deadlines, $T_{\text{total}} =$ total number of transactions processed. Another measure is the restart ratio, which denotes the number of transaction restarts per transaction completed. The baseline settings of the values for the parameters are shown in Table 2.

5.2 Simulation Results

Using the model described above, we conducted an extensive set of simulation experiments comparing the performance of the new protocol with that of the DOCC-DA and the hybrid OCC protocol. In this section, we present the results of a representative set of experiments.

5.2.1 Impact of Arrival Rate

In the following four experiments, the write probability is fixed at 0.5. Fig.10 shows the impact of arrival rates on the performance of the three protocols. The hybrid protocol is a high-performance traditional transaction processing protocol, which does not take transaction's priority into consideration. The higher priority transactions have the greater probability to miss their deadlines. So it has worse performance than the other two protocols. In the DOCC-DA protocol, the transactions have little chance to meet their deadlines if they restart. Mostly the transaction restarts only waste system resources. The new protocol makes use of access invariance and runtime information, which can guarantee a rerun transaction to meet its deadline and abort the fruitless run transactions as early as possible. So it outperforms the DOCC-DA protocol.

In Fig.11, we can find that the restart ratios of the new protocol is lower than those of the other two protocols. This is because the new protocol has the property of avoiding the fruitless rerun. The DOCC-DA protocol combines DASO with Thomasian write rule to avoid unnecessary restart. But it cannot avoid the fruitless rerun. So it has higher restart ratio than the new protocol. In the hybrid protocol, the conflict resolution is restart-based. So its restart ratio is the highest.

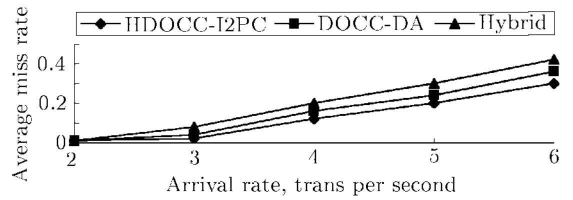

Fig.10. Trans miss rate.

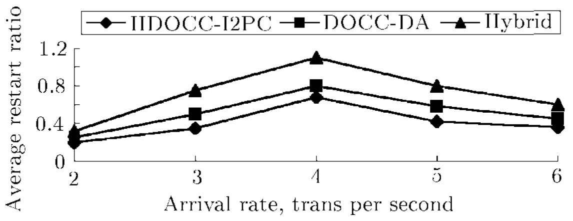

Fig.11. Trans restart ratio.

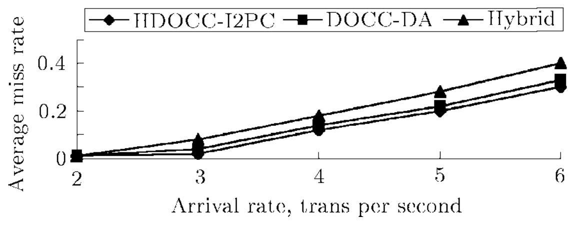

Fig.12. Trans miss rate with modified protocol.

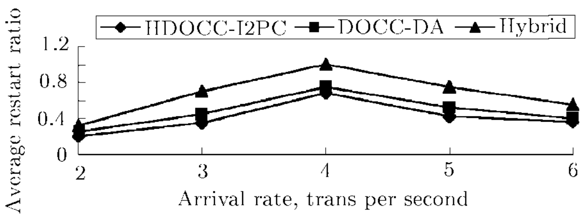

Fig.13. Trans restart ratio with modified protocol.

We can modify the hybrid method and the DOCC-DA protocol by integrating the property of discarding the fruitless run transactions earlier. Their performances have some improvement as shown in Fig.12 and Fig.13. But their performance is still worse than the new protocol's. This is because the hybrid method does not schedule transactions to acquire CPU according to their priority but FCFS. And DOCC-DA protocol has the following two faults. First, the system has no property of access invariance and the rerun may be fruitless. Second, the validation packet is very large and the communication delays caused by message exchanges constitute substantial overheads to the response time of a distributed transaction.

5.2.2 *Impact of Different Write Probabilities*

The OCC protocol performance is mainly influenced by two factors. One is the write probability; the other is the transactions' arrival rate. The mentioned above is mainly referred to the transactions' arrival rate. Now we will focus on the write probability. From Fig.14, we can find the average miss rate increases with the write probability and the new protocol outperforms the DOCC-DA protocol at all write probabilities. The performance gains by avoiding the fruitless reruns and discarding the fruitless run transactions as early as possible.

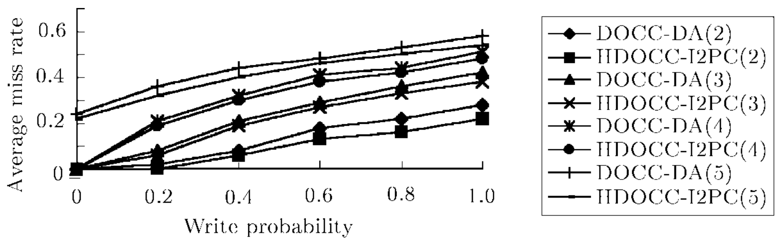

Fig.14. Trans miss rates with different write probabilities.

6 Conclusions

In OCC, some researchers have proposed a kind of phase-dependent control technology, so that a transaction is allowed to have multiple execution phases with different concurrency control methods in different phases. Furthermore, satisfying the time constraint of distributed real-time transactions is to avoid the fruitless reruns and abort the fruitless run transactions as early as possible. In this paper, a new protocol, called HDOCC-E2PC, is proposed. It has the above properties. Simulation experiments have been done to investigate the performance of the new protocol compared with those of the DOCC-DA protocol and the hybrid protocol. From the simulation experiments, we can find that the miss rate of the new protocol is lower than those of the other two protocols for a wide range of workload parameters and the restart ratio of the new protocol is also lower than those of the other two protocols. So the new protocol has a better performance.

References

[1] Haritsa J R, Carey M J, Livny M. Dynamic real-time optimistic concurrency control. In *Proc. 11th Real-Time Systems Symposium*, Florida, 1990, pp.94–103.

[2] Abbott R, Garcia-Molina H. Scheduling real-time transactions: A performance evaluation. *ACM Transactions on Database Systems*, 1992, 17(3): 513–560.

[3] Lam K W, Lee V C S, Hung S L. Transaction scheduling in distributed real-time systems. *The International Journal of Time-Critical Computing Systems*, 2000, 19: 169–193.

[4] Thomasian A. Distributed optimistic concurrency control methods for high-performance transaction processing. *IEEE Transaction on Knowledge Data Engineering*, 1998, 10(1): 173–189.

[5] Haritsa J, Ramamritham K, Gupta R. The PROMPT real-time commit protocol. *IEEE Transactions on Parallel and Distributed Systems*, 2000, 1(2): 160–181.

[6] Yu P S, Dias D M. Analysis of hybrid concurrency control schemes for a high data contention environment. *IEEE Trans. Software Engineering*, 1992, 18(2): 118–129.

[7] Franaszek P A, Robinson J T, Thomasian A. Concurrency control for high contention environments. *ACM Trans. Database Systems,* June, 1992, 17(2): 304–345.

[8] Datta A, Son S H, Kumar V. Is a bird in the hand worth more than two in the bush? Limitations of priority cognizance in conflict resolution for firm real-time database systems. *IEEE Transactions on Computers,* 2000, 49(5): 482–502.

QIN Biao, born in 1972, is a Ph.D. candidate in Huazhong University of Science and Technology. His main research interests include distributed database, real-time database, active database and electronic commerce.

LIU YunSheng, born in 1940, is a professor and a Ph.D. supervisor in Huazhong University of Science and Technology. His main research interests are advanced databases, including real-time database, active database, main memory database, and mobile database, and their integration; database and information system development; real-time data engineering; and software methodology and engineering technology.