In Search of Efficient Non Blocking Atomic Commit Protocol

M. Shahza, Dr. C. Bamford, Prof. Taleb-Bendiab, Dr. Diyah Al-Jumeily
M.T. Shahzad@2006.ljmu.ac.uk, C.Bamford@ljmu.ac.uk
A.Talebendiab@ljmu.ac.uk, D.Aljumeily@ljmu.ac.uk
School of Computing and Mathematical Sciences
Liverpool John Moores University

Abstract—Commit protocols are used to ensure transaction atomicity. The two-phase commit (2PC) is a standard commit protocol used in many commercial database systems. Attempts to reduce the cost of commitment have resulted in many variants of 2PC. These are all blocking protocols. But in other protocols when the blocking issue is addressed the result is an increase in the transaction commitment cost. In this paper we propose a design for a non-blocking commit protocol which can survive coordinator failure, yet which does not increase the execution costs in comparison with 2PC, even reducing them in highly reliable environments where failure is not the issue.

I. Introduction

The two-phase commit protocol is taken as a standard despite its high cost of logging and number of messages. There were many attempts to decrease the costs of logging and communication overhead. Presumed Abort and Presumed Commit are examples with fewer log writes and lower communication overhead than 2PC. The main drawback in these protocols is their blocking nature i.e. in the event of coordinator failure, a prepared participant has to wait for the coordinator to recover and send its decision. In the emerging ad hoc and mobile computing environment with adaptive and intermittent connections, the probability of transaction failure is a lot higher than in fixed and structured networking environments [1]. 3PC is the first attempt to reduce blocking [4] but there is an extra phase called the pre commit phase which further delays commit execution. Optimistic protocol is developed on the optimistic idea that every transaction will eventually commit [3], [2]. In this protocol a waiting transaction may borrow blocked data resources from the executing transaction. Backup commit protocol attached a backup site holding a replicated copy of the protocol database but due to the extra communication between the backup site and the coordinator this method is not very effective in dealing with blocking. In section 2 we analyze the two-phase commit protocol, presumption protocols, single phase commit protocols and non-blocking protocols with their pros and cons. In section 2.5 we present our new non-blocking protocol. In the last section the conclusion is presented.

II. Commit Protocols

The cost of a commit protocol is associated with the number of communication steps and the number of log writes at the coordinator and at each participant. The Blocking or Non-blocking nature and difference in recovery procedures are other important factors that have a vital impact on the overall commit protocol performance. The imbalance between normal protocol execution cost, associated with transaction commitment and increase of cost related with non-blocking commit protocol led to a body of research and development of new low cost atomic commit protocols. An attempt is made to achieve equally good performance of protocol in the presence of failure. Despite all of its drawbacks the 2PC protocol is supported by all commercial database systems and has been standardised by ISO and X/Open.

2.1 Two Phase Commit Protocol:

Two PC is the simplest commit protocol [5]. It has two phases, Prepare and Commit. In both phases, four messages are exchanged between the coordinator and each participant in two message rounds. There are two log writes by the coordinator, one of which is forced, and each participant has to write two forced logs. Due to communication and log writes overhead and with its blocking nature 2PC is considered an expensive protocol to achieve atomicity.

2.2 Presumption Protocols:

In 2PC information about the status of the transaction i.e. committed or aborted, is explicitly stated, exchanged, and logged and there is no missing data. Presumed Commit (PrC) and Presumed Abort (PrA) use missing data to represent specific meanings. For presumed abort the coordinator does not maintain any information about an aborted transaction; similarly, for Presumed Commit, the coordinator does not maintain information for committed transactions. PrA is preferable where the number of aborted transactions is more than the number of committed transaction; PrC is preferred in systems where the number of committed
transactions is more than the number of aborted transactions, a common situation considering present system reliability. A detailed comparison between PrA and PrC is given in [3].

2.3 Non-Blocking Commit Protocols:
Three-phase protocol is the first attempt to eliminate blocking from commit protocols [4]. Although it is a non-blocking protocol, it comes with extra cost as there is one extra phase of messages to eliminate the blocking. In the pre-commit phase there are extra log writes and messages at both participants and coordinator sites which further reduce overall system performance. Optimistic protocol is another which reduces blocking instead of removing it. This protocol is developed with the optimistic idea that every transaction will eventually commit [3]. It lets the transaction borrow the prepared data for the transaction, which holds the lock of the data. Optimistic protocol improves system performance where resources are busy and transactions are likely to commit but it does not solve blocking in the case where any site fails.

2.4 Single phase commit protocol:
There were some protocols in which logging and message cost is achieved by eliminating the entire prepare phase. There are different techniques for maintaining the logs on the sites and different recovery procedures.

2.4.1 Early Prepare Protocol:
The Early Prepare protocol totally eliminates the voting phase without assuming that participants recognize the last operation. In this protocol every operation from the coordinator to the participant is treated as the last operation. After every operation the participant goes to the prepared state and sends a positive ACK for operation execution. If the participant gets into deadlock then it sends NACK instead of going to the prepared state. The coordinator has to update its protocol database every time new sites participate in commit protocol. The number of updates for new remote sites could be high during protocol execution.

2.4.2 Coordinator Log
The Coordinator Log is derived from PrC. In this protocol the coordinator and participants logging is centralized at the Coordinator site in its stable storage space [7]. It means that with ACK of every operation each participant has to send redo and undo logs back to coordinator. This eliminates the need for participants to individually track all progress of protocol. Eliminating logging from participants comes with a large drawback. In the case of system failure the participant has to communicate with all possible coordinators for recovery to get redo and undo logs making recovery highly dependent on the coordinator and requiring extensive communication.

2.4.3 Implicit Yes Vote
Dependent recovery in coordinator log was eliminated by Implicit Yes Vote [8]. IYV implements the PrA idea by having the property of some capability of local recovery without coordinator help, but logging is still centralized in the coordinator log.

In Implicit Yes Vote both coordinator and participant maintain the redo log which helps participants to restore its previous state if a transaction aborts. After each participant force writes a commit decision log then its sends ACK which helps the coordinator to write end record in its protocol database and forget about transaction

2.4.4 Assumptions in Single Phase Commit Protocols
Strong assumptions about reliability of sites, enforcement of consistency constraints, communication medium, local autonomy and memory fault [6] are the main hurdles in the implementation of Single Phase commit protocol. These assumptions are so unrealistic that some people argue that the Single Phase commit protocol does not ensure even transaction atomicity.

Commit protocols ensure the ACID properties but when we talk about Single PC then in many protocols like CL [7] and IYV [8], participants externalize their log records to the coordinator site in which a different database management system may reside. Externalizing the log in this way is totally unacceptable in many multi database systems for data security. 2PC ensures the ACID properties at commit time as compared to single phase in which it is assumed that each participant will ensure its ACID properties at the run time of the transaction. Single PC protocols assume that all consistency constraints are checked during the transaction execution and there would be no violation of local autonomy before sending ACK of operation execution. The Coordinator acts here as a dictator that forces every participant to commit the transaction eliminating its right to abort the transaction.

In the Single Phase Commit protocols, the number of prepare phases is equal to the number of operations executed by each participant because participants are going into the prepared state after they acknowledge each operation. Single PC protocols are blocking protocols. In the case of failure of the coordinator all participants have to wait until the coordinator recovers. Blocking may occur during execution of a transaction whereas in 2PC blocking may only take place if the coordinator fails after it is in the prepared phase. Single PC protocols are more vulnerable to blocking as compared to 2PC because every participant goes in to a prepared state after acknowledging every operation. Detailed analysis and discussions of these assumptions are given in [6], [9].

2.5 New Non-Blocking Commit Protocol
Despite proposals of non-blocking commit protocols, none of them has been implemented in commercial database system. The deciding factors are the increase in communication between remote sites and the number of log writes they use. The emphasis here is to design a non-blocking mechanism with communication cost and logging cost equal to normal 2PC or ideally lower than 2PC which is currently implemented in some of the commercial databases.
2.5.1 Protocol Database:
Every site involved in the commit protocol execution maintains information about the transaction status in its protocol database. Information on the Transaction ID, Mediator ID, Transaction Status, Participant IDs and vote from each participant is maintained in the protocol database at the coordinator. The protocol database is eliminated from memory after complete execution of commit protocol.

The Mediator is included in the process holding a replicated copy of protocol database. It works in parallel with the primary coordinator. The Mediator takes control of the commit protocol if the primary coordinator fails or if it is slow to reply which may be due to writing log records or network communication issues. The replicated copy of the protocol database resides in the mediator’s main memory.

2.5.2 Commit Case:
The proposed commit protocol executes in the following steps.

1. **Forced Write**: The commit protocol initiates at the coordinator site and forced writes a transaction initiation record to its stable storage space. The transaction initiation record lets the coordinator distinguish between a committed, aborted and still active transaction.

2. **Prepare Message**: The coordinator then sends a prepare message to all participants and to the mediator. The structure of the prepare message is as follows.

   - Transaction ID
   - Coordinator ID
   - Mediator ID
   - Participants IDs

   ![Fig 1: Showing Prepare Message at Coordinator](image)

   After getting the prepare message from the coordinator, the mediator makes its protocol database, which it keeps in main memory. There is no forced write associated with building this protocol database which avoids disk write delays and enables the mediator to issue a commit or abort decision quicker than the coordinator.

3. **Force Write**: In reply to the prepare message if the participant decides to send a Yes vote it force writes the prepared log record to its stable storage space. We will consider the No vote in the next section.

   ![Fig 2: Remote Site Activities for Commit Case](image)

   *Vote Message*: Each participant sends a Yes vote to both the mediator and coordinator. Vote delivery to the coordinator and mediator will be done in parallel.

   At this stage it is likely that both the coordinator and mediator will get the vote approximately simultaneously. The mediator sends its decision straightaway to all participants without forced write to the log. It is likely that the mediator reaches the decision quicker than the coordinator.

4. **Decision Message**: The mediator will send its decision to all participants after collecting Yes votes from all participants without any forced write.

5. **Force Write**: With all Yes votes from the participants, the coordinator decides to commit and force writes a commit decision to its stable storage space, thereby delaying the sending of the decision to participants in comparison with the mediator.

6. **Decision Message**: The commit decision is delivered to each participant.

7. **Non-Forced log write**: Each participant writes its non-forced log write and forgets about the transaction.

There is a possibility that event 8 will follow event 5 if a participant receives the decision from the mediator before the coordinator. The mediator’s quick decision helps release locks quickly for the participants.
2.5.3 Abort Case:
There is a difference in activities for an aborted transaction between participant sites which voted Yes and those that voted No. Let’s consider both cases separately.

2.5.3.1 If Participant Decides to Abort:
If any participant decides to abort the transaction then the coordinator must abort the transaction. So we consider No from any single participant as an Abort Decision for all.

1. **Forced Write**: The first step in the abort case is the same as the commit case. Coordinator force writes a log record which contains the transaction initiation log record to its stable storage space.

2. **Prepare Message**: The coordinator sends a prepare message to the participants and the mediator which is a request to all participant to commit a executed transaction.

3. **Abort Decision**: If any participant decides to abort then its sends a No decision to the coordinator, mediator and each participant instead of sending abort vote just to coordinator as in 2PC. Bypassing the coordinator shortens the time in which prepared participants hold there data locks or resources for the transaction.

4. **Abort Decision and Forced Write**: The coordinator and mediator will receive the decision from a participant in parallel. It will force write an abort record to its stable storage space to answer all queries if any participant recovers from a failure.

2.5.3.2 Abort Case if Participant Decided To Commit:
When a participant decides to abort, the transaction, proposed commit protocol works as follows.

1. **Forced Write**: The coordinator writes the transaction Initiation record to its stable storage space which is used to track the status of commit execution.

2. **Prepare Message**: The coordinator sends a prepare message asking each participant to replay with commit vote or abort decision for executed transaction.

3. **Abort Decision From Participant**: Participant which decides to abort will not do any forced write record in its stable storage space. Prepared participants will get the abort decision only from a participant who decides to abort, not from the coordinator or mediator. The coordinator and mediator can only issue commit decision.

4. **Forced Write**: participants ready to commit writes its commit log record on stable storage space.

5. **Forced Write**: Coordinator and participant will force write abort record at same time to release resources. The decision is delivered from the aborted participant rather than from the coordinator saving one message and one log write at the coordinator.

6. **Abort Decision**: Every prepared participant sends ACK back after writing abort log record in its stable storage space. After getting Acknowledgement form each participant coordinator removes all information about the status of protocol execution form its protocol database for memory management purposes.

---

![Fig 3: Remote Site Activities for Abort Case](image)

![Fig 4: Aborting Participant Activities](image)
2.5.4 Failure Handling and Recovery.
In this section we will consider different situations of failure and see how the new commit protocol deals with failure of different sites without affecting the normal performance of the commit protocol execution.

2.5.4.1 Coordinator Failure:
In 2PC, a coordinator failure results in prepared participants keeping all data locks or resources until it recovers and sends the decision. This state is called a blocked state. In the new commit protocol the mediator works in parallel and sends the decision before the coordinator, which overcomes any coordinator failure effect. After recovering from failure the coordinator inquires of other participants about the final outcome of the transaction and then corrects its database accordingly.

2.5.4.2 Mediator Failure:
Mediator failure will not create any blockage situation. If the mediator fails before sending a commit decision, in this situation participants have to wait for the coordinator decision. It will delay decision delivery by one forced write which is done at the coordinator.

2.5.4.3 Participant Failure:
A prepared participant after recovery from failure can enquire of either the coordinator or any of the participants, courtesy of the detail prepare message which include IDs of all participants. From the reply to its enquiry, the recovering participant updates its protocol database and forgets about the transaction after releasing all locks pertaining to transaction.

2.5.4.4 Synchronization between Coordinator and Mediator:
The protocol is designed in a manner that synchronization is not needed. The coordinator and mediator will only send a commit decision not an abort one. If any participant decided to abort then it is the responsibility of the participant to deliver the abort decision to the coordinator, mediator and each participant. Due to this design there is no possibility that the coordinator and mediator will end up with different decisions.

2.5.4.5 Missing Vote form any participant:
If the coordinator and mediator are waiting for a response from a participant which might be lost in the network due to communication failure, they wait for the time stamp to expire and then they will send a Waiting message to all participants again, indicating that response from one or more participants is still not yet received. Participants vote again after receiving a Waiting message just like a second prepare request. It’s up to the participant to keep its prepared state or send abort decision to give resources to waiting transaction.

2.6 Performance Comparison:
The cost associated with any commit protocol is represented by the number of log writes needed by the coordinator and participants and communication between the coordinator and participants in order to complete the transaction commitment.

In this section we will compare the number of log writes and communication costs between different 2PC variants and the new purposed commit protocol.

<table>
<thead>
<tr>
<th>2 PC Variants</th>
<th>Transaction Commitment</th>
<th>Transaction Abortion</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Coordinator</td>
<td>Participant</td>
</tr>
<tr>
<td></td>
<td>r</td>
<td>f</td>
</tr>
<tr>
<td>PrN</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>PrA</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>PrC</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>NewCommit</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Where
r = total number of log writes.
f = number of forced log records
p = number of messages from coordinator (or mediator) to participant
q = number of messages from each participant to coordinator (or mediator)

In the new commit protocol we are considering the number of log writes and messages which are used to commit a transaction. This does not include logging at the coordinator which is after the transaction commits and after every participant releases their locks.

For a committed transaction, the logging cost at the coordinator site is less than 2PC and all of its variants. There is only one forced log write which is the transaction initiation record at start of the transaction. If the mediator fails before decision delivery then the participants have to wait for the coordinator’s second log write and then delivery of its commit decision. The presence of the mediator eliminates the need to wait for the coordinator’s second log write before the decision. It is done after completion of the transaction commitment and is only for recovery in the case when the coordinator and mediator fail at one time.

At the participant side, the cost of logging and communication is less then PrN (i.e. 2PC with no presumption) and PrA and equal to PrC. For the abort case, the cost at the coordinator is the same as PrN and PrC. PrA has lower cost in aborting a transaction at the coordinator site. At participants, the new commit protocol has the same cost of aborting a transaction as PrA but less than PrN and PrC.

This comparison shows that the new commit protocol is more efficient than all other popular mentioned protocols. All these other protocols are blocking protocols, which mean that in the...
event of failure of the coordinator every participant has to wait until it recovers. This delay could be for a considerable time. Blocked participants cannot release resources until they get the decision from the coordinator. The new protocol is the best option for systems where the failure rate is high. It simply distributes the responsibility of the coordinator to manage the transaction to give it more reliability and speed. There is an argument that present reliable systems do not fail often but even in highly reliable systems this protocol would be a good choice as it is the cheapest option to commit a transaction. So if we consider a database system which is very reliable but resources are in heavy demand because of its usage then the new protocol can give a shorter time for commitment, resulting in early release of resources which improves system performance.

III. Conclusion
Commit protocols are used to provide reliable methods to enforce ACID properties in database transactions. Current data applications demand much lower execution time and enhanced reliability even in the event of failure and concurrency. There has been a lot of work done on commit protocols and recently there is renewed interest in searching for efficient and reliable commit protocols which can fulfill the needs of present mobile computing and real time computing. In this paper we critically analyzed two phase commit protocols and its variants. We presented a new commit protocol which is non-blocking and can give even better performance in reliable systems where failure rate is not very high.

References: