Non-Blocking Commit Protocol

Shishir Kumar, Sonali Barvey

Department of CSE, Jaypee Institute of Engineering & Technology, Guna (M.P.), India

Abstract:
Despite all the drawbacks of 2-Phase Commit Protocol (2PC), like blocking, its high cost of logging and number of messages; it is supported by all commercial database systems and has been standardised by ISO and X/Open. Many other variants are proposed in this line but the blocking problem remains the same in all and alternatives are available at high cost and time also. In this paper we will propose a Non-Blocking Commit Protocol (NBCP) which survives the coordinator and participant failure and not even increases the cost of execution and time with the help of low cost main memory.

1. Introduction

Distributed database systems implements a transaction commit protocol to ensure transaction atomicity. From last few decades a variety of protocols has been proposed by researchers .To achieve their functionality these commit protocols typically require exchange of multiple messages, in multiple phases, between participating sites where distributed transaction is done. In addition to that several log record are generated, some of which has to be forced write i.e. they are flushed to disk synchronously. Due to these costs, commit processing can result in transaction execution times [13, 14, 15, 16] making choice of commit protocol is an important design decision for distributed database systems. To achieve this many different protocols are proposed. This paper is an effort to propose a new protocol in which an attempt has been made to reduce the cost of execution as well time.

2. Related Work

In the second section of this paper we have discussed the earlier defined protocols like two phase commit, presumption protocols, single phase commit protocol, optimized commit protocol and non-blocking commit protocol.
The efficiency of a commit protocol is associated with the number of communication steps, the number of log writes and its execution time at the coordinator and at each participant. The Blocking or No blocking nature and difference in recovery procedures are other important factors that have a vital impact on the overall commit protocol performance. In this paper an attempt has been made to achieve equally good performance of protocol in the presence of failure.
Two phase commit: Two PC is the simplest commit protocol [1]. 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.

Variants of 2PC: Both PA and PC seek to reduce commit process overhead by reducing acknowledge messages and forced log writes in the decision phase, while the voting phase remains the same as for 2PC. 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].

Non-Blocking commit protocol: A number of commit protocols have been designed to attack the fundamental blocking problem. Three-phase commit (3PC) [4,7,8] was among the first no blocking protocols. 3PC introduces a new "buffered phase" between the voting phase and the decision phase. In the buffered phase, a preliminary decision is reached about the result of a transaction. Cohorts can reach a global decision from this preliminary decision even in face of a subsequent master failure. However, 3PC achieves the non-blocking property at the expense of increased communication overhead by an extra round of message exchanges. Moreover, both master and cohorts must perform forced writes of additional log records in the buffered phase.

Single Phase Commit: The One-Phase Commit(1PC)protocol has been first suggested in [3] and several variations have been proposed. The *Early Prepare (EP)* protocol [2] forces each cohort to enter a prepare state after the execution of each operation. It makes cohort's vote implicitly YES [11] and this protocol exploits the Presumed Commit as well. But a coordinator

Manuscript received August 5, 2009
Manuscript revised August 20, 2009

may have to force multiple *membership* records, because the transaction membership may grow as transaction execution progresses. Above all, the main drawback comes from the fact that the log of each operation has to be written in the cohort's log disk per operation, it leads to a serious disk blocking time. Only if every sever has a stable storage so that log forces are free, EP can be considered to be used.

*Optimistic commit protoco*l [9] concentrates on reducing the lock waiting time by lending the locks the committing transactions hold. Since the lock lending is done in a controlled manner, there is no possibility of cascading aborts even if the committing transaction is aborted. This protocol has a good performance due to its reduction of the blocking arising out of locks held on prepared data.

3. Proposed Methodology

In this section a novel approach titled as Non-Blocking Commit Protocol will be presented.
Algorithm of NBCP:-

1. Every site including coordinator and participant maintains a database in its primary memory in this paper we call it as transaction database (td). Every ***td*** maintains a transaction id (Tid), primary memory id (Pm_id), transaction status (TS) , participants id (TS) and vote from each participant (V). This ***td*** is automatically deleted after completion of transaction.
2. Primary memory backup (pmb):- A pmb is maintained which holds the replicated copy of ***td*** it works simultaneously with coordinator, in case of failure or due to network delay.
3. Similarly with coordinator every participant also maintains the replicated copy of itself.

Tid	Pm_id	TS	Pid	V

Figure 1-Transaction database

Case 1 Commit:-
Coordinators action at the time of transaction initiation:-

1. *Forced Write*: The commit protocol initiates at Coordinator and forced writes a transaction initiation records to its stable storage [12].
2. Coordinator sends a prepared message to all participants' pmb and its own pmb which is a request to all participants to commit an executed transaction.
3. After getting message from Coordinators pmb, which it maintains at main memory. The pmb maintains a non-forced write because of which there is no delay in disk writing. One of the major advantages of maintaining pmb is that it takes decision quicker than coordinator as it is maintained at main memory. Another advantage is that since cost of main memory is decreasing now a day so the cost maintaining pmb is less.
4. In reply to prepare message if participant's pmb decides to send a YES vote then it quickly takes the decision and send it to Coordinator's pmb without force writing to its disk (we discuss NO vote in another section).

Collection of vote:

5. Vote message: Each participant's pmb sends a YES vote to Coordinator's pmb. Here instead of sending vote directly to Coordinator we are sending message to Coordinator's pmb because of its presence in main memory non-forced write property it reaches to the decision quickly as compared with Coordinator.

6. Decision Message: After collecting vote from every participant's pmb, Coordinator's pmb will send its decision straight away to all participant's pmb.
7. In parallel to others, coordinator also receives the message of commit from its own pmb and then it force writes the commit decision to its stable storage.
8. Non-Forced log write: Each participant writes its nonforced log write and forgets about the transaction.

All PMB's quick decision (compared to both participant and coordinator) helps in releasing the locks. It must be noticed here that all the pmb's (both coordinator and participant) are designed in such a way that they parallelly send their decision to their respective coordinator and participant.

Case 2 Abort:-
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.

When participants decide to abort: If coordinator receives any abort mess from participant's pmb then it decides to abort a transaction. i.e. it is considered as a single NO from one is abort for all.

1. Forced Write: First step will be same as it was in the commit step. Coordinator force writes a log record which contains the transaction initiation log record to its stable storage.

2. Prepare message: Coordinator sends prepare message to the entire participant's pmb and its own pmb.
3. Abort Decision: If any participant decides to abort then its pmb sends a No decision to the coordinator's pmb and each participant's pmb instead of sending abort vote just to coordinator as in 2PC.

It should be noticed that 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: Coordinator and its pmb will receive decision in parallel (as the mechanism is designed in such a way that coordinator and participant will receive messages from their pmb in parallel) . It will force write the last decision into its stable storage.

It must be taken into account that It will help out coordinator and participant at the time of recovery from failure.

When participant decides to commit: In the case, 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 its pmb. The coordinator and pmb can only issue commit decision.

It will reduce the workload of coordinator and the responsibility is distributed throughout the transaction.

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's pmb 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 (though responsibility will be fulfilled by its pmb). After getting Acknowledgement form each participant coordinator removes all information about the status of protocol execution form its protocol database for memory management purposes.

4. Failure Handling and Recovery

In this section we will consider different situations of failure and analyze the issues of dealing of proposed commit protocol with failure of different sites without affecting the normal performance of the commit protocol execution.

1. Coordinator Failure: 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 and it is the major problem with 2PC and its varients. In the new commit protocol the pmb 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. Any PMBs' Failure: Any PMBs' failure will not create any blockage situation. If it 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 or vice versa will happen.

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 (In NBCP pmb's of participant sends message earlier thus helps in overcomes effects of participant failure) 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.

5. Synchronization between Coordinator and PMB: The protocol is designed in a manner that

synchronization is not needed. The coordinator and pmb will only send a commit decision not an abort one. If any participant decided to abort then it is the responsibility of the participant's pmb to deliver the abort decision to the coordinator and its pmb and each participant's. Due to this design there is no possibility that the coordinator and participants respective pmb will end up with different decisions.

6. Missing Vote from any participant: If the coordinator and pmb 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' pmb 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.

7. Due to presence of Primary Memory Backup (pmb) which is available with both coordinator and participant the NBCP becomes more reliable in terms of response time, failure frequency, missing vote and steps for the forced write.

5. Performance Comparison

The performance of NBCP has been compared with other coordinator on the basis of time and number of log writes needed by the coordinator and participants and communication between the coordinator and participants in order to complete the transaction commitment.
The number of log writes and communication costs have been compared for different 2PC variants and the new purposed commit protocol (NBCP).

Result on the basis of cost comparison between different commit protocols is as mentioned below-

TABLE1: Cost Comparison among different Commit Protocols.

2 PC Variants	Transaction Commitment					
	Coordinator			Participant		
	R	F	P	R	F	Q
PrN	2	1	2	2	2	2
PrA	2	1	2	2	2	2
PrC	2	2	2	2	1	1
New Commit	1	1	2	2	1	1
	Transaction Abortion					
PrN	2	1	2	2	2	2
PrA	0	0	2	2	1	1
PrC	2	1	2	2	2	2
New Commit	2	1	2	2	1	1

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)

1. 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.

2. 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.

3. If the ***pmb*** 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 *pmb* eliminates the need to wait for the coordinator's second log write before the decision.

4. It is done after completion of the transaction commitment and is only for recovery in the case when the coordinator and its pmb fail at one time.

5. 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.

6. 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 due to presence of pmb at participant side it costs less then less than PrN and PrC.

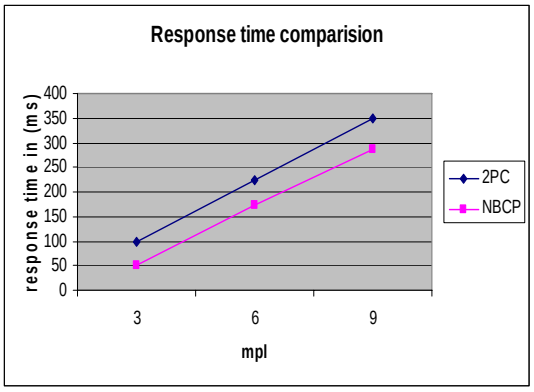

Figure 2- Performance Non-blocking commit protocol in terms of Response time

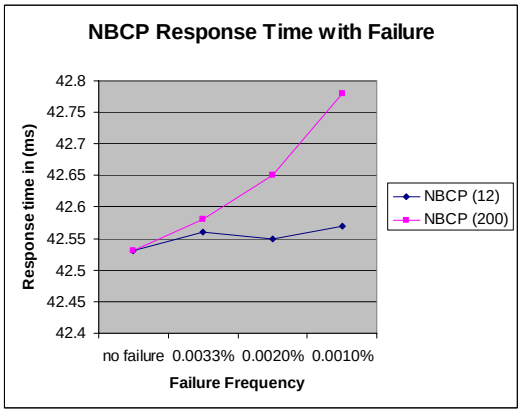

Figure 3- Performance Non-blocking commit protocol in terms of failure

We have compared the performance of regular 2 phase commit (2PC) protocol and the proposed non-blocking commit protocol (NBCP).

Simulation result with response time:-

1. The performance metric for the comparison is the average response time, throughput, log disk utilization and CPU utilization.

2. Figure 2 shows the response time of transactions under 2PC and NBCP with varying MPL, respectively. We can see from the figure that the proposed NBCP protocol reduces the response time dramatically compared with 2PC. If we look at the composition of response time, we can see that the long commit delay has a huge effect on the operation time since the commit delay directly affects the lock holding time.

3. Even though all the operations are performed in the memory, transactions running 2PC protocol show long operation time compared with the NBCP running transactions.

Simulation result with failure frequency:-

4. Figure 3 shows the performance of NBCP with failures. We have tested three cases, one failure every 5,000 transactions, every 10,000 transactions.

5. We have also used two different log sizes, 12 bytes and 200 bytes. When the log size is small, failures virtual has no effect on the performance. Even with frequent failures and large log size, we can see that the performance degradation is less than 1% of the response time.

The rapidly growing main memory size and the falling price have made the main memory database a reality In new NBCP we compare the results on the basis of cost and time NBCP shows that it is better 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 responsibly 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. Similarly at another hand when we tried to compare the cost on the basis of time we have seen results that this NBCP is reducing time at the time of simulation.

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 both on the basis of time and cost. 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:

[1] B. Lampson,"Atomic Transactions. Distributed Systems:Architecture and implementation-An Advanced Course",Lacture Notes in Computer Science, vol 105, pp 246-265, Springer-Verlag, 1981.
[2] J. W. Stamos and F. Cristian. A low-cost atomic commit protocol. In 9th IEEE Symp. on Reliable Distributed Systems (SRDS'90), pages 66–75, 1990.
[3] J. N. Gray. Notes on database operating systems. Operating Systems: an Advanced Course, 60:397–405, 1991.
[4] D. Skeen, "Nonblocking commit protocols " in Proceedings of the 1981 ACM SIGMOD international conference on Management of data Ann Arbor, Michigan 1981, pp. 133-142 .
[5] J. R. Haritsa, K. Ramamritham, and A. R. Gupta, "The PROMPT Real-Time Commit Protocol " IEEE Trans. Parallel Distrib. Syst., vol. 11, pp. 160-181 2000.
[6] P. K. Chrysanthis, Y. J. Al-Houmaily, and S. P. Levitan, "An Argument in Favour of Presumed Commit Protocol " in Proceedings of the Thirteenth International Conference on Data Engineering 1997, pp. 255-265.
[7] D. Skeen. Crash Recovery in a Distributed Database Systems. PhD thesis, Department of Electrical Engineering and Computer Science, University of California at Berkeley, 1982.
[8] D. Skeen. A quorum-based commit protocol. In Proc. Of Berkeley Workshop, pages 69–80, 1982.
[9] B. Lampson and D. Lomet. A new presumed commit optimization for two phase commit. In 19th International Conference on Very Large Data Bases, Doublin, Ireland, 1993, 1993.
[10] Commit processing in distributed on-line and real-time transaction processing system thesis by Jayant Harista IISC Banglore.
[11] Y. J. Al-Houmaily and P. K. Chrysanthis, "The Implicit-Yes Vote Commit Protocol with Delegation of Commitment," in Proceedings of 9th International Conference on Parallel and Distributed Computing Systems, 1996, pp. 804-810 .
[12] J. W. Stamos and F. Cristian, "Coordinator Log Transaction Execution Protocol," Distributed and Parallel Databases, vol. 1, pp. 383-408, 1993.
[13] Revisiting commit protocols in Distributed Database System by Ramesh Gupta Jayant Harista, Krithi Ramaritham. Proceedings of 1997 ACM Sigmond International Conference on Management of Data, Tucson, Arizona, USA, May 1997
[14] Commit Processing in Distributed Real-Time Databases Ramesh Gupta Jayant Harista, Krithi Ramaritham Procedings of 17th IEEE Real Time Systems Symposium, Washington, DC, USA 1996 pages 220-229
[15] M.OZU and P. Valdurize : Principles of Distributed Systems : Princeton Hall 1991
[16] B. Bhargava, (editor) Concurrency and Reliability in Distributed Database systems. Van NostrendReinhold 1987.

Authors Profile :

Dr. Shishir Kumar is currently working as Associate Professor & Head is Dept. of Computer Science & Engineering, Jaypee Institute of Engineering & Technology, Guna, India. His qualification is Ph.D. (Computer Science). He is having around 12 years of teaching & research experience.

Sonali Barvey is student of M. Tech. (CSE) from Jaypee Institute of Engineering & Technology, Guna, [A constituent center of Jaypee University of Information Technology, Wakanaghat]. She is having around 4 years of teaching experience. Her area of interest is Distributed Databases.