An Extended Three Phase Commit Protocol for Concurrency Control in Distributed Systems

Poonam Singh[1], Parul Yadav[1] Amal Shukla [2] and Sanchit Lohia[2]

[1]Amity University, Lucknow, 226070 India

[2] Institute of engineering & technology, Lucknow, 226070 India

ABSTRACT

One of the important issues of distributed system is to improve Concurrency control , I observed that Concurrency Control is very difficult task for distributed systems because of absence of global clock and lack of shared memory. To improve the concurrency control problems in my work, a new modified version of three phase commit protocol is introduced that works for the sake of concurrency control in distributed systems.

The basis of this protocol is the division of all the sites into two groups depending upon the number of queries generated and importance of the queries at these sites. The sites where more queries are generated are considered as primary sites and those having less, are considered as secondary sites. The Primary sites are given more importance while deciding whether to commit or abort a transaction. In this a modified version of three phase commit protocol is praposed that ensures if a transaction is originated from a primary site then it is bound to commit provided all other primary sites vote to commit, no matter whether secondary sites commit or not and there the advantages and disadvantages of this new version is considered.

It is to be mentioned that this protocol works only for transactions that accesses a single database object. Instances of such transactions could be debiting or crediting a bank account as in this only a single database object such as a personal bank account is accessed.

Keywords: Concurrency Control, Two-Phase Commit Protocol , Three Phase Commit Protocol, Primary Sites, Secondary Sites.

1. INTRODUCTION

There are two types of commit protocols used for concurrency control. One is the **two phase commit protocol** and other is **the three phase commit protocol** . in Two phase commit protocol has only two phases first is **voting phase** and second is **decision phase** . Two phase commit protocol has a blocking disadvantage in which either the co-ordinator or some participating site is blocked, Three phase commit protocol was introduced as a remedy to the blocking disadvantage of two phase commit protocol . It introduces an extra phase which ensures the non blocking property of this protocol but I analyesed by three phase protocol we can romove blocking problem but only caused by some sited which are not more important basis of these sites three phase commit protocol global obortion is not in the fouver of efficiency, to improve the total efficiency of distributed systems I have choosen this area and try to romove global obort problem which occure only caused by some unimportant sites.

2. PREVIOUS WORK

Three phase commit protocol is used for concurrency control in distributed systems. It is an extension of two phase commit protocol. It was introduced as a remedy to the blocking disadvantage of two phase commit protocol. This protocol has three phases—

Phase 1 (Voting Phase) : At first the site at which the transaction originates becomes the coordinator and it asks the other sites to vote to either commit or abort . The other sites send their votes . If all sites have voted to commit the transaction, it decides to commit the transaction and if even if one of the sites has voted to abort the transaction it decides to abort.

Phase two (Prepare to commit) : The coordinator tells its decision to all of the sites.If it has decided to commit then "Enter into ready to commit stage" message is sent.

Phase 3 (Decision Phase) : If the coordinator has decided to commit the transaction it sends a global_commit to all sites and waits for their acknowledgement . Only after receiving acknowledgement it decides to commit the transaction . If the coordinator has decided to abort the transaction it sends global_abort to all the sites and aborts the transaction. Only after receiving the acknowledgement it decides the fate of the transaction.

3. PROPOSED MODIFIED VERSION OF THREE PHASE COMMIT PROTOCOL

This protocol in some ways reduces the blocking disadvantage of two phase commit protocol and also overcomes in some failure mode situations. Before discussing this protocol we are going to give proposal that will be required to implement this modified version. These Proposed verison are as follows

Each site is either a "Primary Site" or "Secondary Site".

The sites where more queries are generated are considered as Primary and sites having less queries generated at them are considered as Secondary . The hardware and software vulnerability are also considered while choosing the site as the primary or secondary.

All of the Primary sites do not crash at a time. We have a flag associated with each database object having value "Consistent" or "Inconsistent".

We have a local clock at each site that runs at regular intervals.

3.1 Purpose of the Modified version of Three-phase commit protocol

This modifies version of Three-phase commit protocol ensures the commitment of a transaction originated at a primary site even if one or more of the secondary sites has voted to abort. This is unlike the Three-phase commit protocol in which if one of the sites votes to abort the whole transaction is aborted.

3.2 Alogrithm

In distributed systems the site at which transaction is originated becomes the coordinator .The transaction is then broken down into sub-transactions and then each sub-transaction is issued at different sites. So there are 4 cases to be considered

The coordinator is a primary site

If (all the sites vote to commit)
begin
then start Phase two and send "Enter into ready to commit stage" message to all primary sites and enter into Phase Three sending global_commit to all sites and upon receiving the acknowledgement commiting the transaction.
end
else
If (any of them votes to abort)
then check " if it is a primary site or not"
if (yes)
begin
then goto Phase Three and send global abort
end
else
begin
goto Phase Three and send global commit to all sites who have voted to commit and set the flag with respect to that database object as "inconsistent " at that primary site which is the coordinator and also at site which has voted to abort, sets its flag with respect to that database object as "inconsistent".
end

The coordinator is a secondary site.

Check flag status of the database object which will be in use.
if(flag ="Inconsistent")
begin
Contact the nearest primary site to remove inconsistency and set the flag as "consistent".
and enter into voting phase .
If (all sites vote to commit)
then
begin
enter into Phase two and send prepare to commit message to all primary sites,
after that enter into phase three and send global commit.
end
else
begin
Enter into Phase two and send prepare to abort message to all primary sites and then enter into phase three and send global abort
end
end
else if (flag ="consistent")
begin
enter into voting phase .
If (all sites vote to commit)
then
begin
enter into Phase two and send prepare to commit message to all Primary Sites,
after that enter into phase three and send global commit
end
else
begin
enter into Phase two and send prepare to abort message to all primary sites and then enter into Phase Three and send global abort
end
end

A sub transaction is issued at a primary site-

If (sub-transaction is issued at a primary site)
begin
Continue with the transaction.
end

A sub transaction is issued at a secondary site-

Check the flag status of the database object that will be in use
if(flag is " inconsistent")
begin
Then contact the nearest primary site to remove consistency and set flag as "consistent" and then continue with the transaction.
End
else
begin
carry on with the transaction.
end

As we have seen above there are four cases to be considered . There is no need to check the flag_status in cases 1 and 3 when the coordinator is a primary site or when a sub-transaction is issued at a primary site respectively. This is because if the flag is set as "inconsistent" with respect to a database object at a primary site then it only implies that there exists atleast one secondary site at which that database object is incorrect (i.e, it is not consistent with the primary site) and it does not imply that the database is incorrect at that primary site. Hence there is no need to check for the flag status in case 1 and 3 . The protocol also will work correctly because whenever secondary site is the coordinator or a sub-transaction is issued at the secondary site the flag status is first checked to remove any inconsistency and only then the transaction proceeds.

3.3 Analysis of modified Three-phase commit protocol

This modified version of 3 phase commit protocol, like the original 3 phase commit protocol avoids the blocking limitation of 2 phase commit protocol. In case, the coordinator fails then one of the primary site is chosen as the new coordinator and it proceeds the transaction. The metrics for choosing the new coordinator can be anything , for example distance metrics can be chosen. In this the primary site that is nearest to the main coordinator can be chosen as the new coordinator.

3.3.1 Handling inconsistencies:

As we have seen in this protocol that whenever coordinator is a primary site and the site that voted to abort is a secondary site the protocol commits the transaction and flag is set as "inconsistent" at both sites with respect to that database object. So there is inconsistencies that have arisen. To remove these inconsistencies we have introduced the concept of local clock which runs at each primary site . This clock runs at regular intervals and checks for the database objects for which the flag is set as "Inconsistent". Then it issues a transaction to remove this consistency. Thus in regular intervals the local clock runs at each primary site to remove inconsistency. Whenever a transaction is originated at a secondary site the at first a check is done to see whether the database object that will be in use is in consistent or inconsistent state . if the database object's flag is set as "inconsistent" then a request to the nearest primary site is made to remove the inconsistency. Only after the consistency is removed then the transaction proceeds.

3.3.2 Concept of local clock

A few things are to be noted about local clock:
This clock runs at all primary sites.
This clock runs only at primary sites not at any secondary site.
The clock runs periodically i.e, at regular intervals.

Code for local clock

At regular intervals at each primary site
begin
A search is made through the database at the primary site to see if any database object's flag is set as "inconsistent".
If (any database object's flag say X is set as "inconsistent")
begin
Issue a transaction to remove inconsistency at all sites.
This transaction checks all secondary site to see if the flag with respect to database object X is "inconsistent" at that site.
for (all secondary sites)
begin
If(flag is "inconsistent with respect to database object X
begin
remove this inconsistency by making the database object X consistent with the database object X at the primary site at which the local clock is running
end
end
Set the flag as "inconsistent" at that primary site with respect to the database object X
end
end

3.3.3 Concept of flag

A flag is kept at each site with respect to each database object . A flag has two values "consistent" and "inconsistent".

Use of flag at each primary site: Local clock runs at each primary site at regular intervals .So at regular intervals the flag to check whether to see if there is inconsistency .
If there is inconsistency then a transaction is issued to remove this inconsistency.

Use of flag at secondary site: The local clock runs at each primary site periodically . Suppose at a time when local clock has not removed the inconsistency , a transaction or a subtransaction is issued at the secondary site.
This transaction at first checks the flag of the database object that will be in use.

If(flag is "inconsistent")
begin
Contact the nearest primary site to remove the inconsistency and set the flag as "consistent" at both sites.
end
else
begin
Carry on with the transaction.
End

3.4 Advantages of modified 3 phase commit protocol

This modified version of 3 phase commit proves to be very useful in the cases where we perform some transaction on single database object in distributed systems. This protocol ensures the commitment of transaction originated at a primary site even when some secondary site have voted to abort . This is unlike the original 3 phase commit protocol in which if one of the sites votes to abort then the whole transaction is aborted . So this protocol ensures that if the transaction has originated from a primary site and all of the primary sites have voted to commit then the transaction will commit even if one or more of the secondary sites have voted to abort.

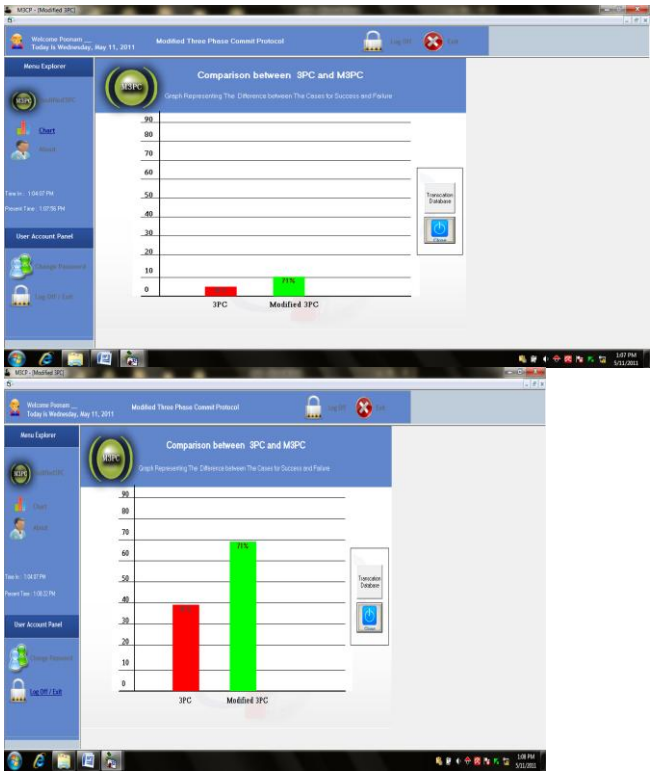
Another advantage of this protocol is that this protocol is non-blocking protocol. If the coordinator fails then one of the primary site is chosen as the new coordinator and the transaction proceeds.

3.5 Limitations of modified 3 phase commit protocols

There are few disadvantages of this protocol. We list some of them below

This protocol can only be applied to global transactions that access the single database object.
This protocol introduces some overhead, as a flag is needed to be maintained with respect to each database object. Also a local clock is needed at each primary site whose purpose is to remove inconsistency at regular intervals.

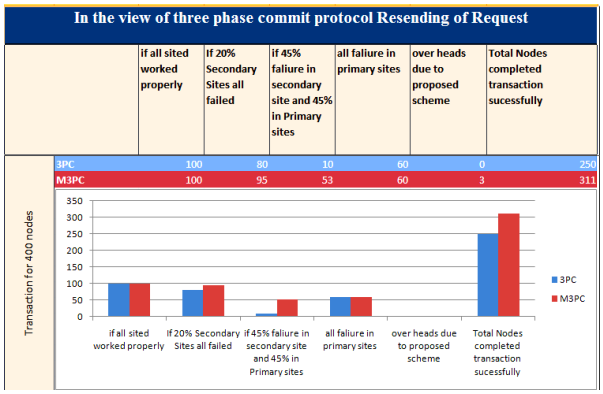
4. RESULTS

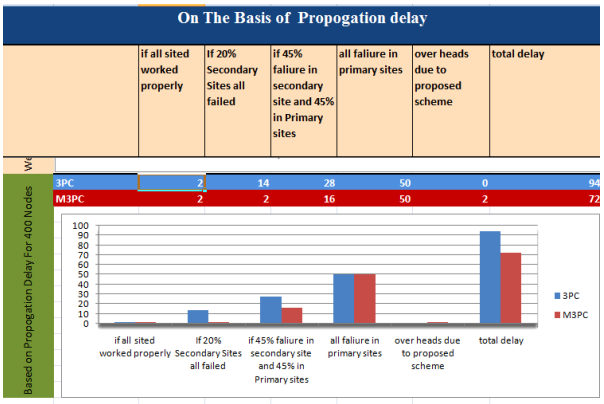


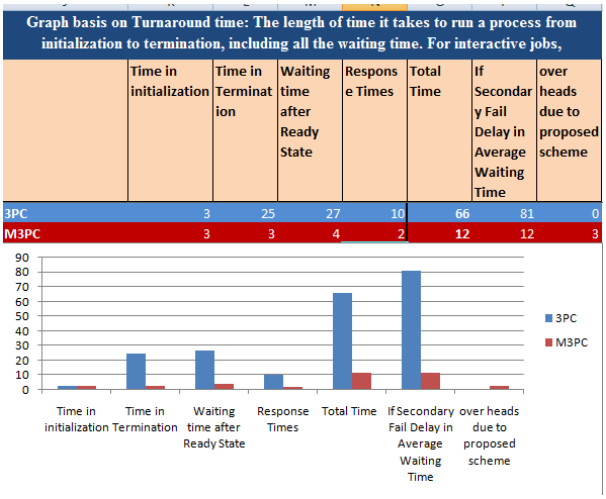
After some transaction we can check performance 3PC and and Modified 3PC aur protocol is better perform then 3PC for single database objects

Some other comparisons on the basis of:

- **Total Efficiency**
- **Propagation Delay**
- **Tarn Around Time**
- **Resending of Request**







5. CONCLUSION

It can be concluded that the modified version can be used only when there is a transaction that accesses a single database object and ensures commitment of the some transactions that would have otherwise failed in Three phase commit protocol so it definitely reduces the probability of a transaction abortion and improve the overall performance of distributed systems.

Remarks and Discussion: as per rule of concurrency control in any transaction ACID properties must be maintain in proposed version of three phase commit protocol it is not necessary if secondary site fails because distributed environment is a large collection of computers only cause of some unnecessary sites fail total transaction failure is not in the favor of efficiency so proposed modification is highly recommended.

Recommendations for Future Work: I have simulate my work in stand alone system to check the authenticity of the proposed protocol in future I will simulate my work in a totally distributed environment and correction in protocol at network level is recommended.

6. ACKNOWLEDGMENT

I like thank to my students Krishna kumar, Vikrant Sachan, Rakesh Kumar, Ashish Baghel, Mohd. Saifulla, Amrit Kumar, Vikas Mathur ,Saikat Chakma ,Satendra Kumar for their help.

7. REFERENCES

[1] Tanenbaum Andrew S. 2006, Distributed Systems: Principles and Paradigms, 2/E Maarten Van Steen.2006.

[2] Bernstein Dr. Philip. 2001 Concurrency Control, Database Hall of Fame (WS2001)

[3] Krishna Reddy P., Bhalla Subhash, TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 15, NO. 3, MAY/JUNE 2003,

[4] Lectures on distributed systems, Distributed Deadlock, Paul Krzyzanowski, Philip Bernstein, Eric Newcomer, Principles of Transaction Processing (for the Systems Professional), Morgan Kaufmann Publishers, January 1997

[5] Philip Bernstein, Vassos Hadzilacos, Nathan Goodman Concurrency Control and Recovery in Database Systems,Addison-Wesley, 1987

[6] Commit Protocols CS60002: Distributed Systems Distributed Systems Pallab Dasgupta Dept. of Computer Sc. & Engg Indian Institute of Technology Kharagpur.

[7] Shanker Udai, Agarwal Nikhil ACTIVE-A Real Time Commit Protocol ªScheduling Real-Time Transactions: A Performance Evaluation,º Proc. 14th Int'l Conf. Very Large Databases, Aug. 1988.

[8] Agrawal R., Carey M., and Livny M., ªConcurrency Control Performance Modeling: Alternatives and Implications,º ACM Trans. Database Systems, vol. 12, no. 4, Dec. 1987.

[9] Bernstein P., Hadzilacos V., and Goodman N., Concurrency Control and Recovery in Database Systems. Addison-Wesley, 1987.

[10] A. Bestavros and S. Braoudakis, ªTimeliness Via Speculation for Real-Time Databases,º Proc. 15th Real-Time Systems Symp., Dec.1994.

[11] Bhargava B., ed. Concurrency and Reliability in Distributed Database Systems. Van Nostrand Reinhold, 1987.

[12] Carey M. and Livny M. , ªDistributed Concurrency Control Performance: A Study of Algorithms, Distribution, and Replication,º Proc. 14th Int'l Conf. Very Large Databases, Aug. 1988.

[13] Chrysantis P., Samaras G., and Al-Houmaily Y., ªRecovery and Performance of Atomic Commit Processing in Distributed Database Systems,º Recovery Mechanisms in Database Systems. V.Kumar and M. Hsu, eds. Prentice Hall, 1998.

[14] Skeen, D., Stonebraker,M.: A formal model of crash recovery in a distributed system. Concurrency control and reliability in distributed systems, 295–317 (1987).