An Efficient behavioural analysis of Commit Protocols for Concurrency Control in Distributed Systems

Mohit Kumar [1],

Assistant Professor

Department of Computer Science and Engineering,
PSIT COE Kanpur, Uttar Pradesh, India

Arjit Gupta [2],

Mansi Srivastava [3],

Avaneesh Kumar Verma [4]

Department of Computer Science and Engineering,
PSIT COE Kanpur, Uttar Pradesh, India

ABSTRACT

This document reviews some of the commit protocols for concurrency control in distributed systems like 2-phase & 3-phase commit protocols. There were certain researches leading to the extension of 3-phase commit protocol by analyzing its disadvantages & removing them. Concurrency control has already been recognized as the primary concern with respect to distributed systems, due to various factors like blocking disadvantage in case of 2 phase commit protocol & lack of shared memory, absence of global clock in case of 3-phase commit protocol. As mentioned in previous researches, these protocol works only for transactions that accesses a single database object [1]. So in this paper we propose building the basics of each algorithm with respect to their behavior analysis and try to propose a more efficient one.

Keywords: *Concurrency Control, 2-Phase Commit Protocol, 3-Phase Commit Protocol, Modified 3-Phase commit protocol, Primary Sites, Secondary Site*

Introduction

To maintain the ACID property of a Transaction in a distributed environment there are two types of commit protocols

1. 2- phase commit protocol
2. 3- phase commit protocol

At every global transaction, each transaction is divided into many sub-transactions where the initiator acts as a co-coordinator and other agents are known as Participants. 2-phase protocol and 3 phase protocol maintain the coordination among these nodes.

1.1 2-Phase Commit Protocol

Under 2 phase protocol there are 2 phases-

Voting Phase: In the 1st phase the primary site (where transaction originates) asks the secondary sites (where transaction is sub-divided into sub-transactions) to vote either to commit or abort as per their availability. Now when the sites are idle they cast their votes and will wait for the decision from the primary site

Commit phase: On the basis of votes that the primary site received from various participants final decision is taken, whether to commit or to abort the transaction and then passed on to participants

ALGORITHM

1 Primary site initiates the transaction by sending message Phase 1 starts
2 If(want to commit)
Participants send the Ready message to the primary site Enter phase 2
Else
Participants send the Abort message to the primary site Enter phase 2
3 If(all message received & want to commit)
Send global commit to participants, logs will be written
And participants send acknowledgement to the primary site
Else
Send global abort to participants, logs will be written
And participants send acknowledgement to the primary site.

2- Phase protocol is a blocking protocol, If the coordinator fails permanently, some cohorts will never get the decision of their transactions: After a cohort has sent **agreement** message to the coordinator, it will block until a **commit** or **rollback** is received which is one the disadvantage of this protocol. [2]

1.2 3-Phase Commit Protocol

Three Phase commit protocol is a remedy of two phase protocol which removes the blocking disadvantage from the protocol. As the name suggest it consist of 3 phases

Voting Phase: In this phase, site at which the transaction originates becomes the coordinator and other sites become participants who votes either commit or abort.

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 by sending Pre_Commit to all the participants.[2]

Decision Phase: At this phase coordinator sends Gobal_commit or Global_Abort message to participants and waits for the acknowledgement from the participants to finally write the details in the log.

Recent Researches

In Modified 3 Phase Protocol, a Transaction Information Table maintains all the details related to the transaction which has 3 fields:[3]-

Transaction Number: Every transaction has its own unique identification number.

Value: This field has two values "COMPLETE" (transaction has committed) or "INCOMPLETE" (transaction aborted).

Site_ID: Each site has its own unique identification number.

Message Passing Mechanism:

1. Message 1: Coordinator send message to inconsistent site asking the site to complete the transaction number and id of inconsistent site along the message.

2. Message2: Inconsistent site send this message to the nearest site for updating information. It will send transaction number along with message.

3. Messaage3: Inconsistent site send this message to coordinator with transaction number.

ALGORITHM

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 3 ending global commit to all sites and upon receiving the acknowledgement committing the transaction.

End

Else

{

If (any one of them votes to abort)

Begin

Go to phase 3 and send global commit to all sites who have voted to commit and set the value in table TIT="INCOMPLETE" with respect to that site who have abort and set the value of the FLAG="INCONSISTENT" with respect to that site

End

}.

Now at regular interval when local clock run recover algorithm start:

Check flag status of the database object which will be in use .

If (FLAG=INCONSISTENT)

{

Begin

Coordinator checks the site id and sends message1 to that inconsistent site.

End

When inconsistent site receives the message1 (Transaction number, Site id).

It sends the message2 to nearest primary/secondary site.

If (site is not idle)

{

Inconsistent site sends the message2 at regular interval for a time which is fixed.

If (site becomes idle)

Begin

Step 1- It check the transaction no. and site ID and all sent all transaction related information to that inconsistent site.

Step 2- after receiving the information inconsistent site update database according to it and set TIT‟s value= “complete”.

Step 3-send message3 to coordinator site and coordinator will set the TIT‟s value=”complete”. Now coordinator site will check the TIT‟s value field

If (value! =“incomplete”)

{

Delete the table related to that transaction.

}

End

Else

Begin

It sends the message2 to coordinator

If (coordinator is idle)

Begin

Repeat step 1, 2 and 3, now coordinator site will check the TIT‟s value field.

If (value! =”incomplete”)

{

Delete the table related to that transaction.

}

End

Else

{

Inconsistent site will send the message2 at regular interval until all the information will not received.

}

}

}

Proposed Algorithm

In previous research, leading to modified 3-phase algorithm there were only 3 fields, in our proposed work we add an extra field Total Number of Attempts (N).

Total Number of Attempts (N): The value of 'N' is the number of participants that have taken part in the global transaction, which prevent the algorithm from getting into inconsistent state.

The following algorithm has been modified which always check variable N and react accordingly and variable 'I', whose value get increment at every abort situation.

ALGORITHM

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 3 sending global commit to all sites and upon receiving the acknowledgement committing the transaction.

End

Else

{

If (any one of them votes to abort)

Begin

Increase the value of I by one (I = I+1)

Go to phase 3 and send global commit to all sites who have voted to commit and set the value in table TIT="INCOMPLETE" with respect to that site who have abort and set the value of the FLAG="INCONSISTENT" with respect to that site

End

}.

Now at regular interval when local clock run recover algorithm start:

Check flag status of the database object which will be in use .

If (FLAG=INCONSISTENT && (I< = N))

{

Begin

Coordinator checks the site id and sends message1 to that inconsistent site.

End

When inconsistent site receives the message1 (Transaction number, Site id). It sends the message2 to nearest primary/secondary site.

If (site is not idle)

{

Inconsistent site sends the message2 at regular interval for a time which is fixed.

If (site becomes idle)

Begin

Step 1- It check the transaction no. and site ID and all sent all transaction related information to that inconsistent site.

Step 2- after receiving the information inconsistent site update database according to it and set TIT‟s value= “complete”.

Step 3-send message3 to coordinator site and coordinator will set the TIT‟s value=”complete”. Now coordinator site will check the TIT‟s value field

If (value! =“incomplete”)

{

Delete the table related to that transaction.

}

End

Else

Begin

It sends the message2 to coordinator

If (coordinator is idle)

Begin

Repeat step 1, 2 and 3, now coordinator site will check the TIT‟s value field.

If (value! =”incomplete”)

{

Delete the table related to that transaction.

}

End

Else

{

Increase the value of I by one (I=I+1)

Inconsistent site will send the message2 at regular interval until all the information will not received.

}

}

Else

Begin

{

Primary site send Global_Abort to all, delete the table related that transaction

}

End

}

Results & Analysis

This algorithm reduces the overhead problem which is related to the 3 phase commit protocol and increases the value of variable 'I' till the maximum limit whenever a site fails to commit the transaction.

Concept of Number of Attempts: Variable N will always avoid database from moving into inconsistent state and transaction will get terminated after defined number of attempts.

Disadvantage of Timestamp mechanism: Timestamp mechanism could not be used because whenever a new transaction gets initiated time counter starts its count down and a situation may arise where system may try to complete the transaction which is not possible due to some technical issue, hence at every recovery step timestamp will be invoked and will goes on till an undefined time.

Conclusions and Future Work

It can be concluded that this protocol will not go under infinite loop and transaction will get terminated after some number of attempts and it this protocol can be used only when there is a transaction that access a single database object. In future, this protocol will be tested in real world scenario where performance of this protocol will be compare to other exiting protocols. Performance measure will be execution time and Miss Percentage is defined as the percentage of input transactions that the system is unable to complete on or before their deadlines [11]

ACKNOWLEDGMENTS

The authors are very thankful to their respected, Dr.Sanjeev Kumar Bhalla,Director, Mr.Pradeep Rai, H.O.D Computer Science Department, for providing excellent computation facilities in the campus. We are very much thankful to publishers Poonam Singh and Mohit Kumar for getting guidance from their research paper and all the technical and non-technical staffs of the PSIT College of Engineering for their assistance and co-operation.

REFERENCES

[1] Poonam Singh el at,An Extended Three Phase Commit Protocol for Concurrency Control in Distributed Systems, International Journal of Computer Applications (0975 – 8887) Volume 21– No.10, May 2011.

[2] Thomas Connolly, Carolyn Begg, "Database Systems," Addison Wesley

[3] Mohit Kumar el at, An Extension of Modified Three Phase Commit Protocol for Concurrency Control in Distributed Systems, International Journal of Research Studies in Computer Science and Engineering (IJRSCSE) Volume. 1, Issue 4, August 2014, PP 38-45 ISSN 2349-4840 (Print) & ISSN 2349-4859 (Online) www.arcjournals.org

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

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

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

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

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

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

[10] 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

[11] **Udai Shanker, Nikhil Agarwal, Shalabh Kumar Tiwari, Praphull Goel, Praveen Srivastava** Department of Computer Science and Engineering, M. M. M Engineering College, Gorakhpur, India E-mail: {udaigkp, nikhilmmmec, shalabhmmmec, goelpraphul, Praveen 047 } @gmail.com Received November 19, 2009; revised November 30, 2009; accepted December 15,2009.