Proceedings of the 5th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases, Madrid, Spain, February 15-17, 2006 (pp322-326)

Consistent Two- phase commit in distributed database

Mushtaq Ali, Iftikhar Ahmad, Shahbaz Pervez, Dr. Nadeem Daud Potta
Computer Science Department,
CIIT, Abbottabad
Pakistan

Abstract: - Two-phase commit work well for centralized database, but in distributed database it creates problems. In some situations the two-phase commit protocol keep the database in inconsistence state and in some situation it takes too much time by using message exchanges. In this paper we present an idea for two-phase commit protocol that will always keep the distributed database in consistence state and also will take relatively small amount of time.

Key-Words: - ***2PC****: Two phase Commit,* ***ACID****: Atomicity,Consistency,Isolation, Durability,****TM:*** *Trasnaction manager,* ***DM*** ***:****Data Manager,* ***Prewrite, dm_write, centralized database, distributed database, cohorts, commit.***

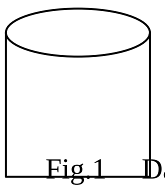
1 Introduction

A transaction has four properties that is acronym as ACID (atomicity, consistency, isolation, durability). Two of the four properties that are atomicity and durability of transactions must be maintained by the reliability protocols. [1] Atomicity requires that either all or none of the operations in the transaction is executed. Atomicity is maintained even in the face of failure. Durability requires that the effects of a successfully completed transaction are permanently recorded in the database and must not be lost because of subsequent failures. The enforcement of atomicity and durability.
require the implementation of atomic commitment protocol and distributed recovery protocols. The most popular atomic commitment protocol is two-phase commit protocol. [2] The two-phase commit protocol is used at the end of transaction execution for committing the values of effected attributes in the database. In the first phase of two-phase commit the changes are first stored in the secure area of hard disk and then in the second phase it is stored to the database from the secure portion of hard disk. This technique work well in centralized database because there is only one TM and one DM .The TM instruct the DM in the first phase to store the changes that reside in the main memory into the secure portion of the hard disk through prewrite.The database will remain in inconsistent state because the data from the secure portion of hard disk is now storing into the database. But in centralized database this incompletion of writes of a transaction does not create problem although the database goes into the inconsistent state but it cannot be accessed by other TMs because there is only one TM. When that TM become correct then its first preference will be to complete the remaining writes.
We consider the execution of the following transaction both in centralized and distributed database system

```
Transaction    T1
Begin
Read (Balance)
Read (Interest)
Balance= Balance+1000
Interest= Balance*5/100
Write (Balance)
Write (Interest)
End
```

Suppose we have a hard disk that contains a file having name account is shown below

Hard disk *Account*



Acc no	Balance	Interest
1	5000	250

Fig.1 Data before transaction execution

During the execution of the above transaction the values of balance and interest will be store in the Ram of computer as

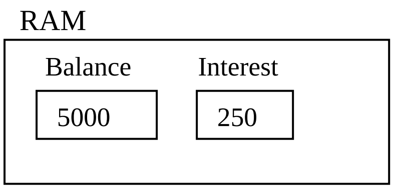


Fig .2 values in the RAM during transaction execution

Now when the last statement of the above transaction is executed that is End statement then the values of balance and interest will become is

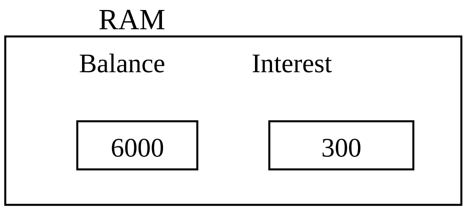


Fig 3. Values in the RAM after transaction execution

As the transaction execution is completed therefore the first phase of two- phase commit will be started .The TM will sent prewrite (Balance) and prewrite (Interest) to the DM one at a time. The DM will take the values of balance and interest from the main memory and store them in the secure portion of hard disk. So their representation on the hard disk will be as

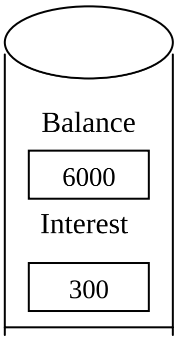


Fig 4. Values in the hard disk after prewrite

Acc_no	Balance	Interest
1	6000	300

Fig 5. Values in the file after 2PC

In the second phase if one dm_write (Balance) is received by the DM and then the TM fails then the database will remain in inconsistent state. Even though the database is in the inconsistent state it will not create any serious problem because there is only one TM and one DM, no other TM send read or write request to the DM when the TM is repaired then first it will send dm_write (Interest) to the DM and the DM will store the previously prewrite data item in the database.

But this create serious problem in distributed database [3] because there are large number of TMs and DMs if one TM fails the other TMs can instruct the DM to extract the uncommitted values from the database because other TMs works well. The structure of the distributed database is shown below.

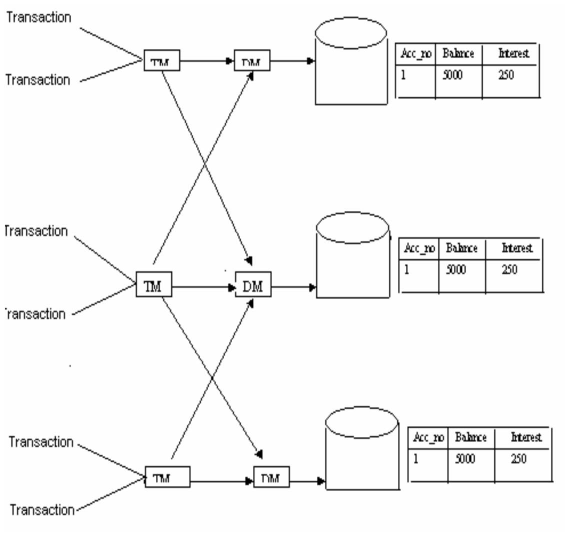


Fig 6. Structure of distributed database system

Now if one TM send prewrite (Balance) and prewrite (Interest) to all the three DMs then the DMs will store the changed values of balance and interest into the secure portion of their hard disks and so the data in their hard disks will look like as

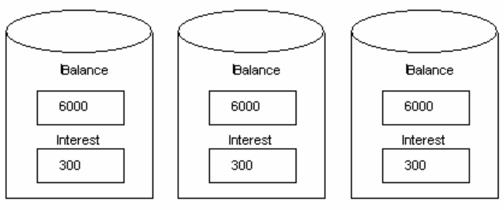


Fig.7 Values on the hard disks after prewrites in the distributed database

Now the TM will send dm_write (Balance) one by one to all the DMs then it will send dm_write (Interest) to all the DMs one by one.

Suppose if the TM send dm_write

(Balance) to all the three DMs and then fails. In this situation the database will be in an inconsistent state at three systems. Because the interest value is not changed at the three systems and any TM other than the failed TM can send a dm_read or dm_write instruction to the value of interest and thus invalid value will be accessed which is the main problem. This problem is shown as

Acc_no	Balance	Interest
1	6000	250

Acc_no	Balance	Interest
1	6000	250

Acc_no	Balance	Interest
1	6000	250

Fig.8 Values in the files when the TM fails

before dm_write (interest)

The value of interest should be 300 rather than 250 which is invalid

2. Old approaches for Two-phase commit in distributed database system:

2.1 The inconsistency in the distributed database system is handled by the 2PC protocol. The TM sends, as many prewrite instructions as there are effected data items in the RAM one for each data item to all the DMs which are responsible for storing these data items. [3] Each prewrite specify two parameters to the DM one for the address of the data item and one specify the other DMs where the copies of the same data item are stored or where the other data items that involved in the operations of transaction are stored.

In the second phase the TM must send dm_write one for each data item, for which the prewrite has already been sent.

If the TM fail before sending all of the dm_write then the DMs who did not receive dm_write for a data item will examine all the DMs who are responsible for storing the desired data item if any of them has received the dm_write for the desired data item then the DM who did not receive the dm_write will automatically store the previously prewrite data items in the database otherwise it will not store the previously prewrite data items in the database.

This approach works well if there is one data item replicated or not on which the transaction perform operation.

But this approach create problem when the transaction perform operation on more than on data items and these data items are replicated or not.

We apply this approach by using the transaction T1.and Fig 5 above.

Suppose the TM send
prewrite(Balance,DM1,DM2,DM3)
prewrite(Balance,DM2,DM1,DM3)
prewrite(Balance,DM3,DM1,DM2) and
prewite (Interest,DM1,DM2,DM3)

to the three DMs one at a time.

They store their values on the secure portion of the hard disk. As shown below.

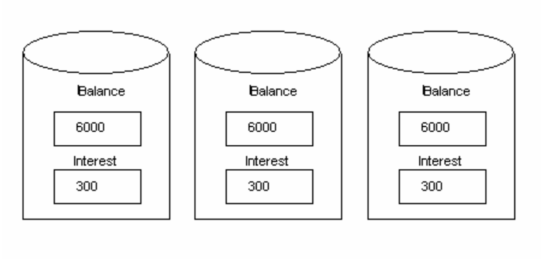


Fig.9 Values on the hard disks after prewrites in the distributed database

Now suppose the TM send
dm_write (Balance, DM1, DM2, DM3)
dm_write(Balance,DM2,DM1,DM3)
dm_write(Balance,DM3,DM1,DM2) to the three DMs and then fail.

In this situation the three DMs will store the value of Balance in the database as shown below

Acc_no	Balance	Interest
1	6000	250

Acc_no	Balance	Interest
1	6000	250

Acc_no	Balance	Interest
1	6000	250

Fig.10 Values in the files when the TM fails before dm_write (interest)

and then each DM will examine the other two DMs to find out that which DM has received the dm_write (Interest). Neither of them has received the dm_write (Interest) and hence the database will go into the inconsistent state. The TM other than the one, which is failed, can access the invalid database.

2.2 The Protocol Messages to commit a distributed transaction PrN requires two messages from TM to cohort and two messages from cohort to TM four messages in all The protocol involves the following steps [2];

a) A TM notices all cohorts that the transaction is to be terminated via the PREPARE message.

b) Each cohort then sends a vote message either a COMMIT-VOTE or an ABORT-VOTE on the outcome of the transaction. A cohort responding with a COMMIT-VOTE is now prepared.

c) The TM commits the transaction if all cohorts send COMMIT-VOTEs If any cohort sends an ABORTVOTE or the TM times out waiting for a vote the TM aborts the transaction. The TM sends the outcome message i.e. COMMIT or ABORT to all cohorts.

d) The cohort terminates the transaction according to its outcome either committed or aborted the cohort then ACKs the outcome message.

This approach causes a problem.
The TM reserve specific amount of time for the commit-vote or abort-vote message arrived from each cohort. If a cohort send commit-vote but it reach to the TM late means the time period of that cohort is out then the TM will make presumption that whether to send commit or abort to their cohorts if it send commit to all of the cohorts involved in the commitment then it is ok. The commitment will be carried out successfully.
But if the TM sends abort to all of their cohorts then the commitment will not be performed by the cohorts although all of the cohorts are ready to commit.
Similarly when the abort-vote from one the cohort reaches late to the TM then the TM will have to make presumption.
The presumption may or may not be correct. If the presumption is made correct then its ok otherwise it can create problem.

3. Consistent two-phase commit Protocol in distributed Database.

When a transaction change the values of data items in a database then the TM should send one prewrite and one dm_write for each of the DM on which those data items are replicated.

1). In the first phase of the two phase commit protocol the TM should send one prewrite instruction to each of the DM for the entire transaction on which the values of the data items are replicated that are changed by the transaction.
The prewite should contain two parameters. The first parameter should specify the addresses of all the data items that are reside in the main memory and whose values are to be stored in the database and the second parameter should specify the numbers of DMs who are responsible for storing the values of the data items effected by the operations of a single transaction in the database.

2). In the second phase the TM should send one dm_write containing the addresses of all of the data item whose values are stored in the secure portion of the hard disks during fist phase to each of the DMs which are involved in the commitment of the intended transaction.

Now we apply this latest two phase commit protocol for the commitment of the above transaction.
The TM will send prewrite ((Balance, Interest), DM1, DM2, DM3), prewrite ((Balance, Interest), DM2, DM1, DM3), prewite ((Balance, Interest), DM3, DM1, DM2) to the three DMs above one by one. The first DM number specify the DM to which the prewrite is to transmited and the second and third DM numbers are send to the DM1 in order to aware it that these two DMs (DM2, DM3) are also involved in the storage of the same data items.
All of the three DMs will take values of the specified data items from the main memory and will store them in the secure

portion of the hard disk. Here our first phase of two-phase commit is finished.

Now in the second phase the TM will send

dm_write((Balance,Interest),DM1)
dm_write((Balance,Interest),DM2)
dm_write((Balance,Interest),DM3)

One by one to the above three DMs.

Now suppose that one dm_write((Balance,Interest),DM1) is received by the DM1 and then the TM fail.

Now what will be do in this situation. As the other DMs have not received the dm_write instruction and they know that what other DMs are involved in the storage of the same data items, so they will examine DM1 that whether it has received the dm_write. As the DM1 already received that dm_write, so the DM2 and DM3 will automatically store their previously stored values in the database.

So it has been proved that by using this protocol the database will never become inconsistent no doubt what ever the transaction will be.

4. Conclusions

The two phase commit protocol for centralized database is explained in the [3]. While the explanation of the two phase commit for distributed database having name "A New Presumed Commit Optimization for two phase commit" is available in[2].the two phase commit protocol that is designed for the centralized database does not work correctly for the distributed database system.

In this paper we present a two phase commit protocol that will work accurately and efficiently for the distributed database system.

References

[1]. M. Tamer Ozsu and Patrick Valduriez, "Distributed and Parallel Database Systems",Department of Computing Science , University of Alberta Edmonton,Canada T6G 2H1 NRIA, Rocquencourt 78153 LE Chesnay Cedex France.

[2]. Butler Lampson and David Lomet "A New Presumed Commit Optimization for Two Phase Commit", Digital Equipment Corporation Cambridge Research Lab February 10,1993.

[3]. PHILIP A. BERNSTEIN AND NATHAN GOODMAN, "Concurrency Control in Distributed Database Systems" Computer Corporation of America, Cambridge, Massachusetts 02139.

[4]. John McDermotta and Sushil Jajodiab, "ORANGE LOCKING: CHANNEL-FREE DATABASE CONCURRENCY CONTROL VIA LOCKING", Naval Research Laboratory, Washington, DC 20375, USA,Department of Information Systems and Systems Engineering, George Mason University, Fairfax, VA 22030, USA.

[5].David Lomet, "Consistent Timestamping for"Transactions in Distributed Systems" Digital Equipment Corporation Cambridge Research Lab, September 7,1990.

[6]. Michael j.carrey and miron livny "distributed concurrency performance" A study of algorithms,distribution,and Replication" Computer science Department University of Wisconsin Madison,wi 53706.

[7]. Bharat Bhargava, "Concurrency control in database system" Fellow, IEEE

[8]. Stefin blott and henry F.Korth,"an almost serial -protocol for transaction execution in main memory database system" Bill laboratories.

[9]. Gjermund Hanssen "Concurrency control in distributed Geographical database systems" Department of Mapping Sciences, Agricultural University of Norway, PO Box 5034, N-1432 °As, Norway?? gjermund.hanssen@itek.norut.no

[10]. M. Tamer Özsu "DISTRIBUTED DATABASE SYSTEMS" University of Waterloo Department of Computer Science Waterloo, Ontario Canada N2L 3G1 {tozsu@db.uwaterloo.ca}