On the Open One-Phase Atomic Commit Protocol

LI Taoshen, SONG Qingzhen
School of Computer, Electronics and Information
Guangxi University
Nanning, China
tshli@gxu.edu.cn

Abstract—According to the computer supported collaborative design transaction under open environment, a kind of open collaborative design structure which contains four layers is given, and an open one-phase atomic commit protocol (O1PAC) is proposed. This protocol solves the problem of site autonomy by adding a participants’ list in the coordinator’s log, and decreases the cost of transaction blocked and resource wasted by permitting uncommitted data lent to other transactions. The performance comparison and experimental results show that O1PAC has less message complication and transactions. The performance comparison and experimental results show that O1PAC has less message complication and log complicity and doesn’t break the site autonomy when adapting in open cooperative design system environment.

Index Terms—computer supported collaborative design (CSCD), transaction commit protocols, one-phase commit protocol, cooperative activity, site autonomy

I. INTRODUCTION

The computer supported collaborative design (CSCD), as a branch of computer supported collaborative work(CSCW), is a effective technology in the field of integrity manufacture and has been greatly concerned by researchers recently. With the greatly development of network techniques, the CSCD environment has been upgraded from environment of the local area network(LAN) to the wide area network(WAN) and the traditional transaction management lost its predominance little by little.

Distributed database systems(DDBS) require a commit process to preserve the ACID property of transactions executed on a number of system sites, that is atomicity, consistency, isolation and durability[2]. In order to obey ACID property, transaction processing time must be short life cycle, and the coordinator must have full control of the participants. However, in the grid environment, a transaction processing may be a longer time. At the same time, because of the autonomous resources, users may not be able to lock the resources needed. Therefore, the transaction must relax the semantics of ACID[3].

Atomic commit protocol is one of important means to ensure consistency of distributed database system(DDBS). In an open distributed database system environment, a distributed transaction is a database transaction that must be synchronized among multiple participating databases which are distributed among different physical locations. Two-phase commit protocol(2PC) and three-phase commit protocol(3PC)[4] are widely employed for distributed transaction commit protocol. At present, many existing transaction commit protocols are based on 2PC, such as optimistic two phase commit protocol(O2PC)[5], prudent two phase commit protocol (P2PC)[6], reliable two phase commit protocol(R2PC)[2], presumed abort protocol (PrA) and presumed commit protocol (PrC)[7,8]. Although widely used in distributed transactional systems, 2PC protocol introduces a substantial delay in transaction processing, even in the absence of failures[9].

There is transaction blocking problem in the 2PC-based transaction commit protocol. Even thought 3PC can eliminate the blocking problem, an additional round of message transmission to achieve non-blocking property may greatly increase system’s overhead. Especially in DDBS environments, in which frequent site failures and longer message transmission times occur, neither 2PC with blocking problem nor 3PC with performance degradation problem are efficient for the commit processing[4].

More recently, the idea of One-Phase Commit (1PC) protocol suggested by Gray in [10] has been reconsidered, and variations of 1PC protocol, such as Coordinator Log [11], Implicit Yes-Vote (IYV) [12], 1-2PC protocol[13,14] and Early no Prepare protocol(ENP)[15] protocols, have been proposed. 1PC protocol reduces the cost of atomic commit by eliminating the voting phase of 2PC: two communications steps and one associated forced log writes. Although efficient, 1PC is however rarely considered in practice because of the strong assumptions it requires from the distributed transactional system. It violates site autonomy by forcing participants to externalize their local log records at the same time[16]. Violating site autonomy is a serious defect in distributed applications, especially in open-environment.

However, current commit protocols all hold update-lock to keep data consistency when they are waiting for the coordinator’s decision. But the waiting time may last for a long while because of net delay or the coordinator have collapsed, then some transactions prepared to update the locked data will be blocked. This problem becomes even more serious, even reduces entire system’ concurrency in open system because it has frequent data exchanging among different sites.

The aim of the paper is to design a four-tier open cooperative design structure and propose a improved transaction commit protocol. The organization of this paper is as follows. Section 2 describes the a four-tier open structure of cooperative design system. Section 3 gives an open one-phase atomic commit protocol (O1PAC). Simulation experiments results and performance analysis of
O1PAC protocol are presented in Section 4 and some concluding remarks are given in Section 5.

II. OPEN STRUCTURE OF COOPERATIVE DESIGN SYSTEM

Traditional structure of cooperation design often used Client/Server(C/S) model which includes servers and clients. Its structure is simple and easy to implement basic function of collaborative design. Obviously, C/S model has frequent message transmission so it often been used in LAN environment. In modern Internet, the architecture of information system become even more complex. Because of being restricted by network speed and transmission bandwidth, some defects of C/S model are disposed progressively. Because the C/S model lacks of agility, not safe and has low efficiency in WAN, it doesn’t suit for the fast developing of network application.

With the expansion of various applications and population of network, three-tier structure containing middleware are put forward to replace the C/S structure. The client layer, the application layer and the database layer are included in this structure. The client layer only consists of GUI which accepts users‘ messages. All the part related to application and data processing place on the application layer. The database layer only preserves the data. Therefore,

Based on the character of open system and three-tier structure, we design a four-tire open structure(shows in Figure 1), which we insert another layer named web server layer into the three-tire structure. In this structure, browser shows users mutual message exchanging and saves some cache data. We put some data on browser in case of multi-version problems caused by frequency temporary design data; the web server transfers users’ order to the application server----the third layer; the most important and complicated part in this structure. The client layer only consists of GUI which accepts users‘ messages. All the part related to application and data processing place on the application layer. The database layer only preserves the data. Therefore,

Based on the character of open system and three-tier structure, we design a four-tire open structure(shows in Figure 1), which we insert another layer named web server layer into the three-tire structure. In this structure, browser shows users mutual message exchanging and saves some cache data. We put some data on browser in case of multi-version problems caused by frequency temporary design data; the web server transfers users’ order to the application server----the third layer; the most important and complicated part in this structure. The client layer only consists of GUI which accepts users‘ messages. All the part related to application and data processing place on the application layer. The database layer only preserves the data. Therefore,

Fig. 1 Open Structure of Cooperative Design system

III. AN OPEN ONE-PHASE ATOMIC COMMIT PROTOCOL

A. Descriptions of O1PAC’s principle and procedure

As mentioned above, 1PC reduces the cost of 2PC but violates site autonomy by forcing participants to externalize their local log records at the same time. In open distributed environment, violating site autonomy is a serious defect. To solve this problem, we add the list of operations submitted to each participant in the log of the coordinator, instead of the physical redo log records sent back by these participants. If the participants collapse in the course of committing, we can easily restarting the transaction processing according to the coordinator’s operations list.

In addition, to solve block and reducing resource waste, we also allow transactions that already is in the ready state lend their locked data to other transactions. In other words, we can let un-committed data to other transactions. But this method is based on the optimistic hypothesis that the lending transaction will commit eventually. On this hypothesis, two instances should be discussed respectively as follows:

(1) The creditor transaction first receives coordinator’s decision. Under this situation, transaction on one of the participants receives coordinator’s decision before it complete its transaction. Then we should choose different steps according to coordinator’s decision, if we receive “commit”, the creditor commit its transaction and leave the debtor continue its work; otherwise we receive “aborted” decision, that means the creditor should abort its transaction and the debtor also abort because it has used the un-consistent data.

(2) The debtor transaction first receives coordinator’s decision. When the debtor completes its work, the creditor hasn’t receive the coordinator’s decision yet, the debtor will be laid on the table until the creditor receives the decision. If it is “commit”, then the debtor transaction will be commit and send YES message to its coordinator; if we receives “aborted” message, both the creditor and the debtor should abort for consistency.

There is something there we should emphasize that the method of lending uncommitted date to others is not acceptable usually in case of cascading rollback. But in our optimization design, this phenomenon never could happen because we set the debtor on “shelve” not “wait” in the case of the creditor transaction didn’t complete its task and the transaction next can’t borrow anything until the creditor transaction finish its job. So there isn’t any case that the borrow chain will greater than one then the cascading rollback never happen.

Based on above ideas and open collaborative design structure as showed in Figure 1, we propose an open one-phase atomic commit protocol(O1PAC). The procedure of O1PAC Protocol is described as follows: which contains four layers is given, and

Step 1: When the coordinator assigns a task to one participant, it also adds participant’s information to the participant_list (which saved in the coordinator’s log);

Step 2: The participant receives its task and turns to “active” state;
Step 3: When a participant finish its job, it sends ACK message to the coordinator along with its results, then it comes to “ready to commit” state. If a participant can’t do its job well, it will send NACK message. After this period, the participant must release its read-lock no matter what message it sends.

Step 4: If the coordinator receives any NACK or abort message, it aborts the whole transaction and force-write an abort record and then submits an abort message to all participants. If the coordinator receives one ACK message, it waits for other participants’ messages until all participants give ACK answer. If all received messages are ACK message, then coordinator commits the coordinator’s transaction, writes a commit record and notifies all participants.

Step 5: A participants receives the coordinator’s commit (or abort) message, it commits (or abort) its own transaction and ends its task. The participant needs send acknowledgement if the message is positive.

Step 6: The coordinator delete participant’s information form the participant_list after if receives the participant’s acknowledgement, and if all participant’s info has been delete, the coordinator writes an end record and ends whole transaction.

We allow transactions that already in the ready state lend their locked data to other transactions for the sake of solving block and reducing resource waste. This characteristic can be insured by our protocol too. After step 3, if any transaction wants to visiting the locked data it will.

If the debtor transaction finishes its job before step 5, it should be laid aside until the creditor receives global decision. The debtor will commit its transaction in case of the creditor commit its transaction, otherwise the debtor will be aborted consequently. And if the debtor accomplished its job after step 5 when the creditor has received global decision, the debtor’s transaction will be committed in case of the creditor’s committing, if the creditor aborted, the debtor has to be aborted because it used inconsistent data.

B. O1PAC’s state diagram

State diagram is a kind of directed graph which pictures the coordinator and the participant’s state transformation process. The node denotes the coordinator and the participant’s state, the line means that a state turns to another under the condition described by the line’s notation.

Participant:

![Fig. 2 The participant’s state diagram](image)

Coordinator:

![Fig. 3 The coordinator’s state diagram](image)

The participant’s state diagram is picture 2. When the participant is “start”, it will turn to “execute” if it receives a task from the coordinator. Then it sends ACK to the coordinator (The transaction result also be appended with the ACK message) when the participant finishes the job, sends NACK if the job can’t be finished. “Ready to commit” transaction can commits itself when it receives the coordinator’s “Admit to commit” message, and if the participant committed it send acknowledgement to the coordinator and become “complete”.

The coordinator’s state diagram is Figure 3. The coordinator start with “start” state, and turn to “wait” after it distributed all tasks. The “wait” coordinator will abort all transaction if it receives any “abort” message form the participants. If the message is ACK, the coordinator should wait until all participants’ messages are ACK, then the coordinator commits global transaction and become “commit” state. After receiving all acknowledgements form all participants, the coordinator becomes “complete”, then, the whole transaction completed.

C. Algorithm’s description of O1PAC

Combined with O1PAC’s processing strategies for the node failure and transmission failure, algorithm of O1PAC is described as follows:

Coordinator:

```
Begin
  initialize state of coordinator as “Begin”;
  set participants list Parti_list as empty;
  allocate tasks(sub-transaction) to participant;
  While not task allocation completed Do
    add address of participant to Parti_list;
    Wait();
  If receives any “Abort” message or NACK message then
    force-write an “Abort” record in log;
    send “Abort” message to all participants;
    end Transaction;
  Else
    force-write an “Commit” record in log;
    send “Commit” message to all participants;
    If receives “ACK” message of all participants then
      delete all participants from Parti_list;
      end Transaction;
  End;
End;
```

Participant:

```
Begin
  initialize state of coordinator as “Begin”;
  receives allocated task(sub-transaction);
  While not sub-transaction ended Do
    If can not complete transaction then
      send NACK message to Coordinator;
    Else
      execute the task;
      end Transaction;
    End;
End;
```
If receives "Transaction end" command then
force-write an "Abort" record in log;
send "Abort" message to Coordinator;
end self-transaction;
Else continue to execute sun-transaction;
End while
// Transaction Completed
force-write an "Commit" record in log;
send "Commit" message to Coordinator;
end self-transaction;
If receives "Commit" message then
send "ACK" message to Coordinator;

IV. PERFORMANCE ANALYSIS AND EXPERIMENTATION

A. O1PAC’s performance analysis

For comparing O1PAC to other commit protocols, such as B2PC, PrC, PrA and IYV, we use two parameters: message complication and log complication. Message complication is the message transmitted time between the coordinator and the participants during a commit process, and log complication means the force write time in this period. In analysis below, we pretend that there only exists one coordinator and the participants are n in all experiments. Then, the analysis results about the message complications and log complications are:

<table>
<thead>
<tr>
<th>Commit Protocol</th>
<th>Message Complication</th>
<th>Log Complication</th>
</tr>
</thead>
<tbody>
<tr>
<td>B2PC</td>
<td>4n</td>
<td>2n+1</td>
</tr>
<tr>
<td>PrC</td>
<td>3n</td>
<td>n+2</td>
</tr>
<tr>
<td>PrA</td>
<td>4n</td>
<td>2n+1</td>
</tr>
<tr>
<td>IYV</td>
<td>2n</td>
<td>n+1</td>
</tr>
<tr>
<td>O1PAC</td>
<td>3n</td>
<td>n+1</td>
</tr>
</tbody>
</table>

From the data in table 1, we can see that O1PAC’s message complication is little bigger than IYV but smaller than B2PC during commit process, and the log complication is the best among all commit protocols. When the transaction is abort, O1PAC’s message complication also between IYV’s and B2PC’s and the log complication equals IYV’s log complication. These results can be explained. We add a participant-list in the coordinator’s log record of IYV so that the site autonomy got guaranteed, then we should delete some information in this log record after the participant’s transaction has been ended. So, O1PAC’s extra message transmission is worthy. In a word, O1PAC’s cost is small than most of commit protocols and it solves the problem encountered in open system. O1PAC is suit for adapting in open cooperative design system environment.

B. Experimental results

Based on the collaborative design structure as showed in Figure 1 and O1PAC’s algorithm, we have implemented an experimental system in Microsoft Visual Studio.net platform. Web server and application server of this system all adopt Windows Server, application layer uses simple browser, database is MS SQL Server 2000. The control procedures of transaction commit are packed in Web Services, and then issued on network. Thus, user in any platform and any place can access it by using any language.

Figure 4 gives the comparison of transaction abort rate in t=600ms and t=1500ms. When runtime of transaction is longer (i.e., long transaction), its abort rate of transaction is higher than shorter transactions, but still keeps in a lower level. This is because the increase of transaction execution time will lead to the increase of system resources and data amount, so that more transactions are aborted or rolled back. This experimental result shows that O1PAC is effective and feasible.

<table>
<thead>
<tr>
<th>Test times</th>
<th>0</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abort rate (%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t=600ms</td>
<td>0.05</td>
<td>0.1</td>
<td>0.2</td>
<td>0.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t=1500ms</td>
<td>0.03</td>
<td>0.07</td>
<td>0.11</td>
<td>0.15</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

V. CONCLUSIONS

An efficient CSCD system needs a practical and effective mechanism to coordinate the accesses to the cooperative objects during the execution of cooperative activities. The traditional transaction commit protocols are not efficient to provide highly concurrency for cooperative design transaction. In this paper, we designed a four-tier open cooperative design structure for open cooperative design system is designed, proposed an improved open one-phase atomic commit protocol (O1PAC) is proposed and analyzed O1PAC’s performance. As a result, O1PAC has less message complication and log complication and doesn’t break the site autonomy when adapting in open cooperative design system environment.

REFERENCES


