CS542 Project Report

Three Phase Commit

Shikhar Singh, Rohit Rangan
Department of Computer Science

April 13, 2016

1 Abstract

This project involves the implementation of the Three Phase Commit protocol (3PC) and a termination protocol to deal with a single site's failure. We maintain data structures on each site; they contain the live transactions, the state of each site for a given transaction, and the status of each site (whether it is up or not). Our implementation deals with permanent failures of sites; once a site fails, we assume that it will not recover. We have defined a site as failed if we cannot connect to it (timeout). Our implementation assumes the presence of 3 sites. In order to test the commit protocol, we created a client program which generates a transaction and sends it to a random site. Multiple clients are created to highlight the concurrent commitment of transactions.

2 Methods and Procedures

2.1 Theory

2.1.1 Three Phase Commit Protocol

Three Phase Commit protocol (3PC) is a non-blocking commit protocol, which lets all nodes in a distributed system agree to commit a transaction. This protocol, for every transaction, is lead by one site i.e. the Coordinator, and a set of participants, which are the remaining sites. A transaction can be committed in the distributed database system, only after three rounds of messages, hence it is called Three Phase Commit protocol. The messages passed by coordinator and participants, and the change of states after each message is described in the state transition diagram.

2.1.2 Termination Protocol

Termination protocols are used in conjunction with non-blocking commit protocols. A termination protocol is used when a site failure occurs and the continued execution of the commit protocol impossible.The purpose of the termination protocol is to identify sites that are still operational and proceed them toward a commit decision which is consistent with both operational sites and failed sites.

The protocol

After a site-failure has been detected, the sites that are up enter the termination protocol. In the Termination protocol, The Coordinator Site, checks the state it is in at the time of failure. Based on its site, and the knowledge about the set of states the participant sites can be in, the coordinator, sends a global-commit, or a global-abort message.

Rules of making a decision: These are based on the state of the coordinator

- INITIAL - If the coordinator, is in the INITIAL STATE, a global-abort message is sent.
- WAIT - If the coordinator, is in the WAIT STATE, participants can be in INITIAL, READY or ABORT state. Therefore a global-abort message is sent.
- PRE-COMMIT -If the coordinator, is in the PRE-COMMIT STATE, participants can be in READY or PRE-COMMIT state. Therefore, the coordinator knows that all the sites must have agreed to commit, and a global-commit message is sent.

In the case where the failing site is a Coordinator itself, the termination protocol, make the operational sites, elect a new coordinator, which will then follow the same rules for termination of the transaction. For this case, there are a few additions to the set of states in which the coordinator site can be, for which the rules are :-

- READY - If the new coordinator, is in the READY STATE, participants can be in INITIAL, READY or ABORT state. Therefore a global-abort message is sent.
- PRE-COMMIT -If the new coordinator, is in the PRE-COMMIT STATE, participants can be in READY or PRE-COMMIT state. Therefore, the coordinator knows that all the sites must have agreed to commit, and a global-commit message is sent.
- ABORT -If the new coordinator, is in the ABORT, a global-abort message is sent.

2.2 Implementation

We have implemented the three phase commit protocol, and the termination protocol in case of site failure. The communication of the sites has been implemented using C++ Socket programming. In our implementation, we have 3 Sites, and a Client program which sends them the transaction.

2.2.1 Messages

All the sites and the clients communicate by send Messages to each other. For which, we have defined a structure for the message which looks like.

$$Message :< message_id >, < transaction_id >, < site_id >, < failed_site >$$

Here, message_id can be one of the following,

1. TRANSACTION - Sent by client, to one of the sites

2. START_VOTE - Sent by coordinator, to participants
3. VOTE_COMMIT - Sent by participants, to coordinator
4. VOTE_ABORT - Sent by participants, to coordinator
5. PR_COMMIT - Sent by coordinator, to participants
6. COMMIT - Sent by coordinator, to participants
7. ABORT - Sent by coordinator, to participants
8. ACK - Sent by participants, to coordinator
9. TERMINATION - Sent by the site which detects site failure
10. ERROR

transaction_id - unique identifier for the transaction, site_id - unique identifier for the site which is sending, failed_site - unique identifier for the failed site, used only with message TERMINATION.

2.2.2 Data Structures

All the sites maintain information about every transaction. We have used the following data structures to do so.

1. **Set of live transactions** - Every time a transaction arrives at a site for the first time, it is added in the set, and it is removed from the set only when either Committed or Aborted.
2. **Map (Transaction_id, Transaction)** - A map for all the transactions using the transaction_id as the key, is maintained, with the value being a structure to which has the following information about the transaction
   (a) Number of votes - The coordinator site, maintains the number of votes it has received in favor of the commitment of the transaction.
   (b) State of the site - This is the state of the site, corresponding to the transaction
   (c) Coordinator - Every site has this information, the site_id of the coordinator site, of the transaction.

2.2.3 The Protocols

Here are a few details about the implementation.

- The three phase commit protocol is implemented, using the Messages, each site, receives a message and then based on the rules described for 3PC in section 2.1.1.
- **Initiation of Termination protocol** - Whenever a site fails to communicate with another site, it detects a site-failure, and sends a TERMINATION message to all the other operational sites, and initiates the termination protocol. Upon receiving the TERMINATION message, the site starts the termination protocol.The termination protocol is implemented, using the Messages, each site, receives a message and then based on the rules described for termination protocol in section 2.1.2.

- **Electing a coordinator** - When a site which is the coordinator fails, the other operational sites elect a coordinator, this is done based on a predefined priority of the sites. Once all the sites get the information which site has failed. Every site checks, if it is the site with highest priority and updates the transaction information accordingly.
- **Vote Commit or Vote Abort** - The participant sites while voting, vote ABORT with a probability of $\frac{3}{20}$, and vote commit for $\frac{17}{20}$ times.

3 Data Collected

In order to test our implementation of 3PC, we run 2 client programs which generate transactions at specified intervals for 10 seconds. These tests assume no site failures and are used to test the implementation of the protocol. Each client program picks a site at random and submits the transaction, following which it sleeps for a certain interval. For every transaction, we have a coordinator (which received the transaction) and the participants.

(C_1, C_2)	N_1	N_2	N_3	N_A	N_C
(10, 10)	484	395	513	408	984
(10, 200)	268	252	326	217	629
(30, 50)	172	146	161	126	353
(200, 100)	52	50	45	39	108

Table 1: Number of transactions processed for different rates of incoming transactions.

In the above table, C_1 and C_2 refer to the interval between transactions for client 1 and client 2 respectively; N_1, N_2, and N_3 refer to the number of transactions arriving at sites 1, 2, and 3 respectively; N_A refers to the number of transactions aborted; N_C refers to the number of transactions committed.
Our implementation works well and processes 1392 transactions in around 10 seconds.

(C_1, C_2)	N_1	N_2	N_3	N_A	N_C	N_S	N_L
(10, 10)	787	715	78	257	1323	1583	3
(10, 200)	398	377	68	178	665	845	2
(30, 50)	193	231	56	93	387	482	2
(200, 100)	58	74	15	25	122	149	2

Table 2: Number of transactions processed for different rates of incoming transactions with site 2 failing.

In the above table, C_1 and C_2 refer to the interval between transactions for client 1 and client 2 respectively; N_1, N_2, and N_3 refer to the number of transactions arriving at sites 1, 2, and 3 respectively; N_A refers to the number of transactions aborted (at the sites which are alive); N_C refers to the number of transactions committed (at the sites which are alive); N_S refers to the total number of transactions generated by the client; N_L refers to the total number of transactions lost.

We simulate failures by forcibly shutting down a server at a random instant. Once this is done, the shut server will not accept any new connections. There is some loss in the transactions as incoming transactions are ignored during the termination protocol. The transactions sent during this duration are not processed. From the table, since the termination protocol is invoked once, we see that it takes around 30ms for the termination protocol to complete at a site.