Assignment 5
Due date: Jun 3, 2022

Consider a basic version of 2PC protocol wherein the coordinator sends a prepare message to cohorts which respond with either a “yes” or a “no” vote. On receipt of their votes, the coordinator makes a decision to commit or abort the transaction and transmits the decision to the cohorts. Cohorts on receiving the decision, send an ack to the coordinator, which, in turn, on receiving all the ack can safely forget about the transaction (i.e., delete the entry of the transaction from its protocol database which it maintains to track the status of the commit processing of the transaction).

Question 1: Explain what log records are written by the coordinator and by the cohorts during the execution of the basic 2PC and which of those log records are forced writes to disk?

Question 2: Explain what timeout action, if any, is taken by the coordinator in case it times out waiting for an ack from the cohort for the message containing its decision (i.e., commit/abort). (i.e.,., can the coordinator simply wait without taking any action).

Question 3: 2PC is a blocking protocol. Explain what does blocking mean? Also, clearly explain under what circumstances is 2PC blocking?

Question 4: Consider the paxos algorithm. Show how removing each one of the conditions below can lead to violating safety (provide a scenario for each one):

1. Remove the leader election phase (A node can become a leader without going through the leader election phase.)
2. A leader can propose any value it likes even if it received a previously accepted value in one of the prepared messages it collected in the leader election phase.
3. In the leader election and replication phases, instead of a majority, a node only collects 2 votes regardless of the total number of nodes.
4. A node responds to a propose message even if its ballot number is lower than the highest ballot number it has ever seen.
5. A leader uses a ballot number in the replication phase that is different than the ballot number it used to become a leader.
6. Two nodes may use the same ballot numbers (ballot numbers are not unique.)

Question 5 This question examines if atomicity of transactions can be maintained without sites participating in a 2PC protocol. Let us consider an integrated heterogeneous distributed database system consisting of three independent database management systems (DBMSs) -- DBMS1, DBMS2, and DBMS3. We refer to a DBMS as `cooperative' if its interface supports begin, prepare, commit, and abort commands. On the other hand, an `uncooperative' DBMS does not support a prepare primitive. An uncooperative DBMS, on receipt of a commit request of a transaction attempts to commit the transaction. If it cannot commit the transaction (say due to integrity constraint violation), it returns an abort status. Else, it acknowledges that it has successfully committed the transaction.

Can atomicity of distributed transactions be guaranteed in an environment in which some DBMSs may be `uncooperative'. Specifically, state if atomicity of transactions be preserved in the following two scenarios. If your answer to either (1) or (2) is yes (that is, atomicity can be preserved), specify the protocol that can be used to guarantee atomicity in that scenario. Do not worry about handling failures or specifying timeout actions in your protocol(s). That is, assume that all messages sent by one site to another reach successfully and no site has to timeout waiting for a message from another. Furthermore, no process or processor failures occur (a database management system may, however, refuse to commit a transaction if committing the transaction results in the violation of its integrity). If, on the other hand, your answer is no (that is, atomicity cannot be preserved), explain why not.

1. DBMS1 is `uncooperative', but DBMS2 and DBMS3 are `cooperative'.

2. DBMS1 and DBMS2 are `uncooperative', but DBMS3 is `cooperative'.

Question 6: Suppose that 2PC with Presumed Abort is used as the commit protocol. Explain how the system recovers from failure and deals with a particular transaction T in each of the following cases:

A. A subordinate site for T fails before receiving a prepare message.
B. A subordinate site for T fails after receiving a prepare message but before making a decision.
C. A subordinate site for T fails after receiving a prepare lnessage and force-writing an abort log record but before responding to the prepare message.
D. A subordinate site for T fails after receiving a prepare message and force-writing a prepare log record but before responding to the prepare lnessage.
E. A subordinate site for T fails after receiving a prepare message, force-writing an abort log record, and sending a no vote.
F. The coordinator site for T fails before sending a prepare message.
G. The coordinator site for T fails after sending a prepare message but before collecting all votes.
H. The coordinator site for T fails after writing an abort log record but before sending any further messages to its subordinates.
I. The coordinator site for T fails after writing a commit log record but before sending any further messages to its subordinates.
J. The coordinator site for T fails after writing an end log record. Is it possible for the recovery process to receive an inquiry about the status of T from a subordinate?

Question 7: Paper summary. Read the paper on "Flexible Paxos: Quorum Intersection Revisited" by Howard et. al.,. Summarize the key ideas in the paper. Answer the following questions:

1. What problem in Paxos are they trying to solve? Why is it important?
2. What is the insight in paxos correctness that they observed for their solution??
3. What is interesting or new about the work?
4. Describe their approach?
5. What practical impact would the modifications that they proposed have on real systems?
6. What are the strengths and weaknesses of the paper? Are there any improvements that can be made to the paper?