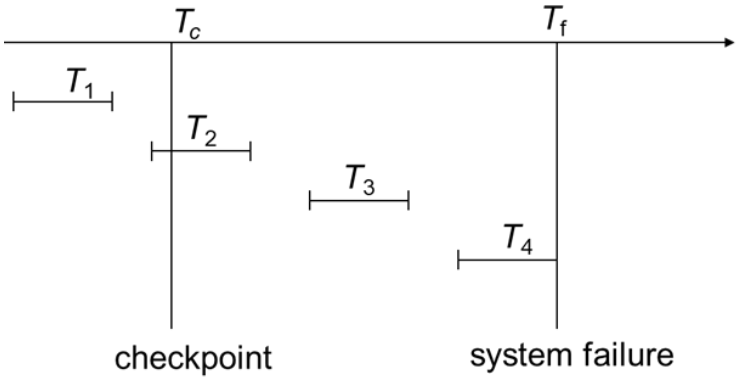
Exercise 9 Solution

1. In the following figure the first vertical line Tc denotes the point where checkpointing was done and the second on the right, Tf, is where a system crash occurs. Please discuss what would change if the checkpointing was done right at the beginning of each transaction instead of the following case in the figure.



Solution:

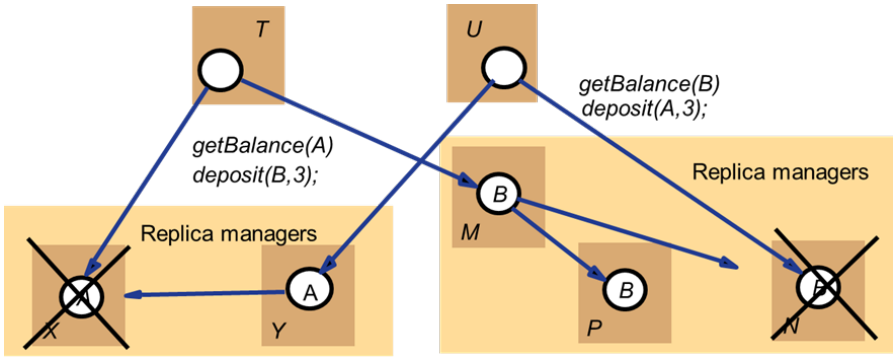
If we were to do checkpointing at the beginning of each transaction, then after a system failure there would be much less to do during recovery time. This is because unlike the case above there would be much fewer transactions to consider for redo/undo. T3 for example would not be considered if at the beginning of T4, we did a checkpoint. Thus, with more checkpointing recovery becomes fast. Having said this, this does not come for free. There would be many output operations to the disk for each checkpoint as well as entries to the log regarding checkpointing. Thus, the cost is during transaction processing there would be more overheads. One needs to consider the frequency of checkpointing carefully. There is no one good frequency and it is deployment dependent. For systems where immediate recovery is of utmost importance then frequency can be increased. For systems where transaction processing should be done fast, but recovery can take a long time, less checkpointing should be considered.

2. Discuss how timestamp strategy differs in concurrency control w.r.t its original form in distributed transactions.

Solution:

Each transaction is assigned a globally unique timestamp by the coordinator. The order of the transactions' timestamps determines which transaction has the priority to access objects at distributed servers. The timestamp is passed with each object access. This assures that if transaction X is before transaction Y based on the timestamp ordering, X will be before Y in their conflicting access to objects at all servers.

3. Given the two following transactions T and U that run on replica managers X, Y, M, P, and N, we have seen in class a simple version of the Available Copies strategy that would not work. First review the problem that would occur if X and N were to crash during execution. Then state the solution that we have discussed in class. Last but not least, discuss what would happen under the final solution, if rather than X and N becoming unavailable we have the following scenario: If Y were to become unavailable during the execution and right after U accessed A at Y, but X and N do not fail, rest of the assumptions of this scenario is the same as we discussed in class.



Solution:

For the first discussion part, please refer to the lecture. Please review them and see that with the new rule Available Copies stops dangerous executions from happening. Now for the case where Y fails instead of others. The scenario is different. We wish that: U can lock Y and is delayed at X as A is locked by T there, similar to the previous scenario. On M, P, and N either U gets the lock first and T waits, in which case there is a deadlock which will be resolved by a timer, or T locks on N as well and U also waits there and T finishes first and N later. In any case, as you see there is no problem as T and U see each other. Nevertheless, the executions above will not occur with the final solution, as seeing Y is gone U will self-terminate based on the new rule for our final solution. As you can see, the implementation of the final strategy we have seen, and in fact many strategies in DBMSs, are only approximations and in general pessimistic ones so that they guarantee proper executions but sometimes terminate transactions that would not really cause harm.

4. Assume a two-phase commit involves a coordinator and three participants, P1, P2 and P3. What would happen in the following scenarios?

   - Scenario 1: The coordinator crashed after receiving a 'yes' vote from all participants.
   - Scenario 2: P1 and P2 voted yes and P3 voted no.
   - Scenario 3: P2 crashed when it was about to send a vote message to the coordinator.

Solution:

- Scenario 1: The main fallback, in this case, is: The participants will send getDecision messages to the coordinator individually because they cannot receive further instruction from the coordinator. We assume that the coordinator eventually recovers from the crash to where it was.
- Scenario 2: The coordinator will send doAbort messages to all participants. To achieve atomicity, all the participants should commit or none will commit.
- Scenario 3: Since the coordinator cannot receive the vote from P2 within the timeout period, it will abort the transaction by sending doAbort messages to all participants.