Exercise 8 Solution

1. What is the Mean time to failure values of different RAID systems?
   - a. RAID 0 with 2 disks
   - b. RAID 2 with 2 disks
   - c. RAID 1 with 2 disks
   - d. RAID 1 with 3 disks
   - e. RAID 3 with 3 disks
   - f. RAID 4 with 3 disks
   - g. RAID 5 with 3 disks
   - h. RAID 6 with 5 disks

   Solution: Let's label the probability of one disk failure as p, and the mean time to failure of one individual disk is MTTF. P is between 0 and 1.

   RAID 0 with 2 disks and RAID 2 with 2 disks – The system fails if one of the disks fails. The probability that one of the two disks fails (disk A or disk B) is $p + p = 2p$. So, as failure probability doubles mean time to failure is halved = $\frac{1}{2} \times MTTF$.

   RAID 1 with 2 disks - The system fails if both of the disks fail at the same time. The probability that both disks fail (disk A and disk B) is $p * p = p^2$ as p is between 0 and 1, mean time to failure of the system will increase to $MTTF^2$.

   RAID 1 with 3 disks - The system fails if three of the disks fail at the same time. The probability that all disks fail (disk A and disk B and disk C) is $p * p * p = p^3$. So, the mean time to failure of the system is accordingly $MTTF^3$.

   RAID 3 with 3 disks- The system fails if 2 of the 3 disks fail at the same time. The probability that 2 disks fail is $p*p = p^2$. There are 3 different possible combinations of 2 disk failures (A, B; or A, C; or B, C), so the probability that any of the 2 disks out of these 3 disks fail is $3p^2$. The mean time to failure of the system is $\frac{1}{3} \times MTTF^2$.

   RAID 6 with 5 disks - The system fails if 3 out of the 5 disks fail at the same time. The probability that 3 disks fail is $p * p * p = p^3$. There are 10 different possible combinations of 3 disks failures out of 5 disks (A,B,C; or A,B,D; or A,B,E; A,C,D; or A,C,E; or A,D,E; or B,C,D; or B,C,E; or B,D,E; or C,D,E), so the probability that any of the 3 disks out of these 5 disks fail is $10p^3$. Mean time to failure of the system is then $\frac{1}{10} \times MTTF^3$.

2. Which of the following RAID configurations that we saw in class has the lowest disk space utilization? Your answer needs to have explanations with calculations for each case.
   - (1) RAID 0 with 2 disks
   - (2) RAID 1 with 2 disks
   - (3) RAID 3 with 3 disks

   Where does this lack of utilization of space go, i.e., where we can use such a configuration as it has some benefits gained due to the loss of space utilization?

   Solution:
   - In case 1, the space utilization is 100% because the two disks store contiguous blocks of a file in RAID 0.
   - In case 2, the space utilization is 50% because RAID 1 uses mirroring. MTTF increases so for cases where a disk can fail easily this is good. The system operates even when a disk fails.
   - In case 3, the space utilization is (3-1)/3=66.7 because RAID 3 uses one disk for storing parity data.

   Case 2 has the lowest disk space utilization with an explanation as given underlined above as a part of the answer.

3. Aries Example: After a crash, we find the following log. What will be the analysis, redo, and undo phases?

   10 T1: UPDATE P1 (OLD: YYY NEW: ZZZ)
   15 T2: UPDATE P3 (OLD: UUU NEW: VVV)
   20 BEGIN CHECKPOINT
   25 END CHECKPOINT (XACT TABLE=[[T1,10],[T2,15]]; DPT=[[P1,10],[P3,15]])
   30 T1: UPDATE P2 (OLD: WWW NEW: XXX)
   35 T1: COMMIT
   40 T2: UPDATE P1 (OLD: ZZZ NEW: TTT)
   45 T2: ABORT
   50 T2: CLR P1(ZZZ), undonextLSN=15

   Solution:

   Analysis phase:

   Scan forward through the log starting at LSN 20.
   LSN 25: Initialize XACT table with T1 (LastLSN 10) and T2 (LastLSN 15). Initialize DPT to P1 (RecLSN 10) and P3 (RecLSN 15).
   LSN 30: Add (T1, LSN 30) to XACT table. Add (P2, LSN 30) to DPT.
   LSN 35: Change T1 status to ”Commit” in XACT table. Set LastLSN=35 for T1 in XACT table.
   LSN 40: Set LastLSN=40 for T2 in XACT table.
   LSN 45: Change T2 status to ”Abort” in XACT table. Set LastLSN=45 for T2 in XACT table.
   LSN 50: Set LastLSN=50 for T2 in XACT table.

   Redo phase:

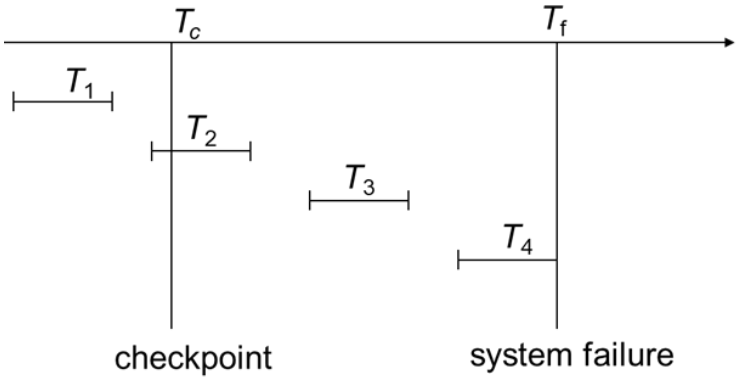
   Scan forward through the log starting at LSN 10.
   LSN 10: Read page P1, check PageLSN stored in the page. If PageLSN<10, redo LSN 10 (set value to ZZZ) and set the page’s PageLSN=10.
   LSN 15: Read page P3, check PageLSN stored in the page. If PageLSN<15, redo LSN 15 (set value to VVV) and set the page’s PageLSN=15.
   LSN 30: Read page P2, check PageLSN stored in the page. If PageLSN<30, redo LSN 30 (set value to XXX) and set the page’s PageLSN=30.
   LSN 40: Read page P1 if it has been flushed, check PageLSN stored in the page. It will be 10. Redo LSN 40 (set value to TTT) and set the page’s PageLSN=40.
   LSN 50: Read page P1 if it has been flushed, check PageLSN stored in the page. It will be 40. Redo LSN 45 (set value to ZZZ) and set the page’s PageLSN=50.

   Undo phase:

   T2 must be undone. Put LSN 50 in ToUndo.
   LSN 50: Put LSN 15 in ToUndo
   LSN 15: Undo LSN 15 - write a CLR for P3 with ”set P3=UUU” and undonextLSN=NULL. Write UUU into P3.

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:

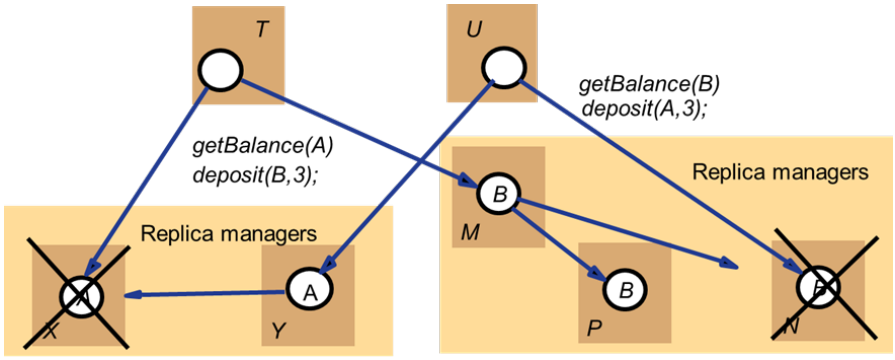
If we were to do checkpointing at the beginning of each transaction, then after a system failure there would be much less to do during recovery time. This is because unlike the case above there would be much fewer transactions to consider for redo/undo. T3 for example would not be considered if at the 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.