CARNEGIE MELLON UNIVERSITY
COMPUTER SCIENCE DEPARTMENT
15-445/645 – DATABASE SYSTEMS (FALL 2024)
PROF. ANDY PAVLO

Homework #6 (by William) – Solutions

Due: **Monday Dec 9, 2024 @ 11:59pm**

IMPORTANT:

- Enter all of your answers into **Gradescope by 11:59pm on Monday Dec 9, 2024**.
- **Plagiarism**: Homework may be discussed with other students, but all homework is to be completed **individually**.
- **You have to use this PDF for all of your answers.**

For your information:

- Graded out of **100** points; **4** questions total

Revision : 2024/12/12 21:18

Question	Points	Score
ARIES	28	
Two-Phase Commit	24	
Distributed Query Plan	18	
Miscellaneous	30	
Total:	100	

Question 1: ARIES .... [28 points]

RusTub uses ARIES recovery with fuzzy checkpoints. It also has a background thread that may arbitrarily flush a dirty bufferpool page to disk at any time.

For this question, assume objects A, B, C reside in three different pages A, B, C, respectively.

LSN	WAL Record
1	`<T1, BEGIN>`
2	`<T2, BEGIN>`
3	`<T3, BEGIN>`
4	`<T3, UPDATE, prev=3, C, 1000→2000>`
5	`<T2, UPDATE, prev=2, A, 10→20>`
6	`<T2, COMMIT, prev=5>`
7	`<T1, UPDATE, prev=1, B, 100→200>`
8	`<CHECKPOINT BEGIN>`
9	`<T1, UPDATE, prev=7, A, 20→30>`
10	`<CHECKPOINT END, ATT={T1, T2, T3}, DPT={C}>`
11	`<T3, UPDATE, prev=4, C, 2000→3000>`
12	`<T2, TXN-END>`
13	`<CHECKPOINT BEGIN>`
14	`<T1, COMMIT, prev=9>`
15	`<T4, BEGIN>`
16	`<CHECKPOINT END, ATT={T1, T3}, DPT={?}>`
17	`<T3, COMMIT, prev=11>`
18	`<T4, UPDATE, prev=15, B, 200→300>`
19	`<T4, UPDATE, prev=18, A, 30→40>`
20	`<T4, ABORT, prev=19>`
21	`<T4, CLR, prev=20, A, 40→30, undoNext=18>`

Figure 1: WAL

(a) Suppose the system crashes and, when it recovers, the WAL contains the first 10 records (up to `<CHECKPOINT END, ATT={T1, T2, T3}, DPT={C}>`). Of the object states below, which states are possibly stored on disk before recovery starts? Select all that apply.

i. **[4 points]** ☐ A=10 ■ **A=20** ■ **A=30** ☐ A=40 ☐ Cannot be determined

ii. **[4 points]** ☐ B=100 ■ **B=200** ☐ B=300 ☐ Cannot be determined

iii. **[4 points]** ■ **C=1000** ■ **C=2000** ☐ C=3000 ☐ Cannot be determined

Solution: Page A is not logged as dirty in the checkpoint record, so the value can't be 10. But LSN 9 changes page A, so it can be either 20 or 30. Page B is not logged as dirty in the checkpoint record, and there is no followup updates between LSN 8 and LSN 10, so its value should be 200. Page C is logged as dirty in the checkpoint. It's uncertain whether it hasn't been flushed at all or if it has been flushed at some stage, so its value can be 1000 or 2000. Pages on disk will not contain any updates later than the last flushed log record.

(b) **[4 points]** Select all possible values of DPT in record 16.
■ **A** ☐ B ■ **C** ☐ A, B ■ **A, C** ☐ B, C ☐ A, B, C ■ **None of them**

Solution: Dirty pages can be flushed to disks at any time arbitrarily. But LSN 10 ensures that the value stored in disk for Page B has already been updated from 100 to 200, and there is no followup updates, so B cannot appear in the DPT. Both Page A and C may be present in the DPT due to the following updates after LSN 8 (not to mention C has already been logged dirty at the last checkpoint).

Grading Note: Although not origianlly intended, "None of them" is also a valid answer since the background thread could have aggressively flushed dirty data.

(c) **[4 points]** For next 3 questions, assume that the database restarts and finds all log records up to LSN 21 in the WAL. Also assume the DPT is {C} for LSN 16. According to the lecture, which pages the analysis phase may select to be redone? Select all that apply.
■ **A** ■ **B** ■ **C** ☐ None of them

Solution: We redo all pages in the DPT and those modified after LSN 13.

(d) **[4 points]** Select all transactions that should be undone during recovery.
☐ T1 ☐ T2 ☐ T3 ■ **T4** ☐ None of them

Solution: All uncommitted or explicitly aborted transactions should be undone.

(e) **[4 points]** How many new CLR records will be appended to the WAL after the database fully recovers?
☐ 0 ■ **1** ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6

Solution: T4 will produce 1 more CLR record.

Question 2: Two-Phase Commit .... [24 points]

Consider a distributed transaction T operating under the two-phase commit protocol with the ***early acknowledgement optimization***. Let N_0 be the *coordinator* node, and N_1, N_2, N_3 be the *participant* nodes. Let the C be the client application issuing the transaction request. Assume that the client communicates directly with the coordinator node. Assume that the coordinator will treat a node that has failed to respond after ***three*** "**Phase1:PREPARE**" as dead.

The following messages have been sent:

time	message
1	C to N_0: "**REQUEST:COMMIT**"
2	N_0 to N_2: "**Phase1:PREPARE**"
3	N_0 to N_3: "**Phase1:PREPARE**"
4	N_2 to N_0: "**OK**"
5	N_0 to N_1: "**Phase1:PREPARE**"
6	N_0 to N_1: "**Phase1:PREPARE**"
7	N_3 to N_0: "**OK**"
8	N_0 to N_1: "**Phase1:PREPARE**"

Figure 2: Two-Phase Commit messages for transaction T

(a) **[6 points]** Who should send message(s) next at time 9 in Figure 2? Select *all* the possible answers.

- ☐ C
- ■ N_0
- ■ N_1
- ☐ N_2
- ☐ N_3
- ☐ It is not possible to determine

Solution: There are two possible options. If N_1 is alive, N_1 needs to respond to N_0 with a "**OK**" or "**ABORT**" message. Alternatively, N_0 will decide T_1 is dead and proceed to the "**ABORT**" procedure.

(b) **[6 points]** Assume N_1 responds "**OK**" at time 9, who does N_0 send messages to at time 10? Select *all* the possible answers.

- ■ C
- ☐ N_0
- ■ N_1
- ■ N_2
- ■ N_3
- ☐ It is not possible to determine

Solution: At the time when all participant nodes have responded with "**OK**", T is considered to be committed. N_0 must notify the participant nodes N_1, N_2, and N_3 of the decision to commit. At this time N_0 should also notify the client application since the protocol is running with the early acknowledgement optimization.

(c) **[6 points]** Suppose that N_0 decides to abort the transaction at time 9 in Figure 2. What should happen under the two-phase commit protocol in this scenario?

- ☐ N_0 resends "`Phase1:PREPARE`" to all of the participant nodes
- ☐ N_0 sends "`Phase2:COMMIT`" to all of the participant nodes
- ☐ N_0 sends "`ABORT`" to only the client
- ■ **N_0 sends "`ABORT`" to all of the participant nodes and the client**
- ☐ N_0 resends "`Phase1:PREPARE`" to N_2
- ☐ N_1 sends "`OK`" to N_0
- ☐ It is not possible to determine

Solution: The coordinator (N_0) will mark the transaction as aborted. 2PC requires that *all* participants respond with "`OK`".

(d) **[6 points]** Suppose that N_0 successfully receives all of the "`OK`" messages from the participants from the first phase. It then sends the "`Phase2:COMMIT`" message to all of the participants but N_1 and N_3 crash before they receives this message. What is the status of the transaction T when N_1 comes back on-line?

- ■ **T's status is *committed***
- ☐ T's status is *aborted*
- ☐ It is not possible to determine

Solution: Once the coordinator (N_0) gets a "`OK`" message from *all* participants, then the transaction is considered to be committed even though a node may crash during the second phase. In this example, N_1 and N_3 would have restore T when it comes back on-line.

Question 3: Distributed Query Plan . . . . [18 points]

The CMU-DB team is optimizing distributed databases for aggregations and joins in their new project, AutoOpt. This project aims to enable efficient computation of distributed joins followed by aggregations.

Given the following schema:

```
CREATE TABLE part(PRIMARY KEY p_partkey INT, p_type VARCHAR);
CREATE TABLE partsupp(ps_partkey int, ps_suppkey int, ps_cost DECIMAL,
                      PRIMARY KEY (ps_partkey, ps_suppkey));
CREATE TABLE supplier(PRIMARY KEY s_suppkey INT, s_nationkey INT);
CREATE TABLE nation(PRIMARY KEY n_nationkey INT, n_name VARCHAR);
```

AutoOpt partitions these tables across nodes based on the partition key. Consider the following query:

```
SELECT n_name, MIN(partsupp.ps_cost)
FROM part
JOIN partsupp ON part.p_partkey = partsupp.ps_partkey
JOIN supplier ON supplier.s_suppkey = partsupp.ps_suppkey
JOIN nation ON nation.n_nationkey = supplier.s_nationkey
WHERE part.p_type LIKE '%BRASS'
GROUP BY n_name;
```

You can make the following assumptions:

1. There are 4 nodes in the system.
2. The `part` table contains 20,000 rows, of which 4,000 satisfy the `p_type` predicate.
3. The `partsupp` table contains 80,000 rows.
4. The `supplier` table contains 10 rows.
5. The `nation` table contains 25 rows.

(a) **[5 points]** Which data distribution strategy minimizes the total network data transfer for the given query?

- ☐ Partition all tables randomly without any specific range or replication strategy.
- ■ **Replicate `partsupp`, `supplier`, and `nation` tables across all nodes, and partition `part` table by `p_partkey`.**
- ■ **Replicate `supplier` and `nation` table across all nodes, and partition `part` and `partsupp` tables by `p_partkey` and `ps_partkey` respectively.**
- ☐ Partition all tables by their primary key.

Solution: This strategy minimizes cross-node data transfer by localizing joins involving `part` and `partsupp` to individual nodes and avoids the need to transfer `supplier` and `nation` data across nodes because it is replicated everywhere. In addition, this strategy avoids the need to transfer the entire `partsupp` table everywhere.

Grading Note: Question originally intended to account for replication costs (which would make (b) worse than (c) due to needing to move `partsupp` to all nodes. Due to ambiguity, both (b) and (c) are correct.

(b) **[5 points]** Assuming the selected strategy from question (a) is implemented, what is the estimated total data transferred over the network for the **join** operation? Assume that *only* the central node can perform aggregations.

- ☐ Less than 5,000 rows
- ☐ Between 5,001 to 25,000 rows
- ■ **Between 25,001 to 100,000 rows**
- ☐ More than 100,000 rows

Solution: Given that `supplier` and `nation` data are replicated across all nodes and joins are localized, the data transfer will be the result of all local joins. Worst case scenario is the max cardinality of the join output: 40,000 rows.

(c) **[5 points]** If AutoOpt introduces a feature that allows each node to compute a partial aggregation before sending it to the central node for final aggregation, what is the new estimated total network data transfer?

- ■ **Less than 5,000 rows**
- ☐ Between 5,001 to 25,000 rows
- ☐ Between 25,001 to 100,000 rows
- ☐ More than 100,000 rows

Solution: Here, the only data transfer will be the result of local aggregations needing to be combined at a central node for final computation. Since we have 4 nodes and 25 rows in `nation`, this translates into an upper bound of 100 row transfers.

(d) **[3 points]** What are the primary drawbacks of implementing a feature that allows for intermediate aggregation results to be computed on each node before sending these results to a central node for final aggregation? Consider the impact on system resources. Select all that apply.

- ☐ It significantly increases the amount of data transferred over the network.
- ☐ It *requires* more data to be shuffled between nodes.
- ■ **It increases the computational load on each node.**
- ■ **It increases the memory usage on each node due to the storage of intermediate results.**
- ☐ It decreases the overall system performance.

Question 3 continues. . .

Solution: Performing intermediate aggregations locally increases each node's computational load and memory usage, as they must handle and store aggregation results. This method reduces network traffic but demands more computational and memory resources.

First option is incorrect since the goal is to decrease data transfer. The second option is incorrect since it does not necessarily require more data to be shuffled between nodes. The last option is too general; local aggregations often enhance performance by easing network demands.

Question 4: Miscellaneous .... [30 points]

(a) **[3 points]** A distributed DBMS can immediately commit a transaction under network partitioning without any loss of data consistency.

□ True

■ **False**

Solution: False. This is a violation of the CAP theorem.

(b) **[3 points]** ARIES employs two passes of the log during the recovery process to handle both redo and undo operations.

□ True

■ **False**

Solution: ARIES is a three phase algorithm and requires at most three passes over the log.

(c) **[3 points]** The CAP theorem implies that a distributed system cannot simultaneously guarantee consistency, availability, and partition tolerance.

■ **True**

□ False

Solution: By definition.

(d) **[3 points]** In ARIES, only transactions that commit will have an associated "TXN-END" record in the log.

□ True

■ **False**

Solution: Transactions will commit when the COMMIT log is written, TXN-END is an optimization in the ARIES protocol that indicates it can be removed from the transaction table.

(e) **[3 points]** In the context of distributed DBMS, data replication increases availability but can lead to challenges in maintaining data consistency across nodes.

■ **True**

□ False

Solution: CAP limitations.

(f) **[3 points]** Both PAXOS and Two-Phase Commit protocols can be used to implement distributed transactions.

Question 4 continues. . .

■ **True**
□ False

Solution: 2PC is a degenerate case of Paxos, which is a consensus protocol for distributed transactions.

(g) **[3 points]** In reference to recovery algorithms that use a write-ahead log (WAL). Under NO-STEAL + FORCE policy, a DBMS will have to undo the changes of an aborted transaction during recovery.

□ True
■ **False**

Solution: The FORCE policy guarantees that all changes from committed transactions are immediately written to disk, ensuring durability, while it is the NO-STEAL policy that prevents changes by uncommitted transactions from being written to disk, avoiding the need for undo operations on disk for those transactions.

(h) **[3 points]** Fuzzy checkpoints need to block the execution of all transactions while a consistent snapshot is written to disk.

□ True
■ **False**

Solution: See motivation of checkpointing.

(i) **[3 points]** With consistent hashing, if a node fails, then only a subset and not all keys will be reshuffled among the remaining nodes.

■ **True**
□ False

Solution: See motivation of consistent hashing.

(j) **[3 points]** In a system with strong consistency requirements, it is best for the DBMS to implement active-passive replication with synchronous replication and continuous log streaming.

■ **True**
□ False

Solution: Synchronous replication can ensure strong consistency.