8-hour Online Written Exam of Advanced Computer Systems – ACS, 2021-2022

Department of Computer Science, University of Copenhagen (DIKU)
Date: January 21nd, 2022

Preamble

Solution

Disclaimer: In the following, we present solution sketches for the various questions in the exam. These solution sketches are provided only as a reference, and may lack details that we would expect in a complete answer to the exam. Moreover, some of the questions may admit more than one correct solution, but even in such cases only one solution sketch is provided for brevity. Solution sketches are colored for visibility.

Note, in addition, that the evaluation of the exam takes into account our expectations regarding solutions, the actual formulations provided, the weights of various questions, but also and most importantly the overall evaluation of the exam assignment as a whole. This evaluation is performed by both internal and external examiners and grades are finally provided by discussion and consensus. As such, it is not advised to reason about final grades based on this document.

This is your final 8-hour online written exam for Advanced Computer Systems, block 2, 2021/2022. The exam will be evaluated on the 7-point grading scale with external grading, as announced in the course description.

- Hand in a **single PDF file**, in which your exam number is written on every page.
- Your answers must be provided in English.
- Hand-ins for this exam must be individual, so **cooperation with others in preparing a solution is strictly forbidden.**
- The exam is open-book and all aids are allowed. If you use **any sources other than book or reading material for the course**, they **must be cited appropriately.**
- You may also use your computer or other devices, but you may not access online resources or communicate with any other person in preparing a solution to the exam.
- Remember to write your exam number on all pages.
- You can exclude this preamble in your hand-ins if you directly annotate this file.
- This exam has a total of 12 pages excluding this preamble, so please double-check that you have been given a complete exam set before getting started.

Expectations regarding formatted spaces

For each question in this exam, you will find formatted space where you can provide your answer. The spaces provided are designed to be large enough to provide satisfactory (i.e., concise and precise) answers to each of the questions.

Exam number: _____

Expectations regarding question importance

Questions carry indicative weights. The weights will be used during evaluation to prioritize question answers towards grading; however, recall that the exam is still evaluated as a whole. In other words, we provide weights only as an indication so that you can prioritize your time, should you need to during the exam. You cannot assume that weights will be divided equally among subquestions.

Exam number: _____

The following table summarizes the questions in this exam and their weights.

Question	Weight
Q1 RPC	6%
Q2.1 Metrics	3%
Q2.2 Experiments	4%
Q3.1 Schedules	13%
Q3.2 Cascading Aborts	3%
Q3.3 Deadlocks	4%
Q4.1 The Missing Log	8%
Q4.2 The Recovery	9%
Q5.1 Lamport and Vector Clocks	10%
Q5.2 Scalability and Reliability	10%
Q5.3 Two-Phase Commit	10%
Q6.1 Selection	5%
Q6.2 Join and Aggregate	15%

Errors and Ambiguities

If you find any errors or ambiguities in the exam text, you should clearly state your assumptions in answering the corresponding questions. Some of the questions may not have a single correct answer, so recall that ambiguities could be intentional.

Exam number: _____

1 RPC (6%)

For its nerdy users, DuRør, a Danish online video sharing platform (similar to YouTube), offers an RPC interface for features that normal users would interact with in their browser. In DuRør's context, give three example features—one for each of the three different RPC semantics that we considered in the course. Explain the meaning of the different RPC semantics and justify why your examples require the corresponding RPC semantics.

Explanations clear (see slides);

Examples: At least once: update user's email address

At most once: upload a video

Exactly once: subscribe to a paid (no ad) version incl. payment

2 Performance (8% in total)

The transactional in-memory database management system MeagreSQL implements two different techniques for concurrency control to achieve isolation of transactions: one based on locking and one on optimistic concurrency control. Your task is to design empirical experiments that would help you decide for a technique based on performance.

2.1 Metrics (3%)

Based on which metrics would you evaluate the performance of concurrency control in MeagreSQL? **Note**: it is not enough to just answer "throughput" and "latency" here. Instead you should say concretely **what** you want to measure for MeagreSQL (e.g., latency of XYZ) and **why**. Moreover, include at least one metric that is specific to the problem of concurrency control.

Possible answers (all should be accompanied with a justitifaction):

Throughput as the number of transactions per minute

Average/Maximum latency for a single transaction (time from begin to end)

Ratio of aborted transactions

2.2 Experiments (4%)

Describe the experiments you would setup: How would you define/select the workloads for the experiments? What are the parameters of the workloads? How would you measure the above metrics? Would your setup differ if MeagreSQL was not an **in-memory** database? Justify your reasoning.

Exam number: _____

Possible answers (all should be accompanied with a justitifaction):

Mixture of realistic and synthetic workloads.

For the synthetic ones abstract to read/writes: parameters: number of transactions, length of transactions (number of actions), number of objects accessed, read/write ratio

Simple factor experiments

Use simple timers for transactions (take time at start/end).

For the not in-memory database, one should take into account the overhead of the disk; possibly IO should become a parameter of the workload

3 Concurrency Control (20% in total)

3.1 Schedules (13%)

We consider three transactions `T1`, `T2`, and `T3` consisting of the following actions.

```
T1: R(X) R(Y) W(Y) C        T2: R(Y) W(Y) W(Z) C        T3: W(Z) R(X) W(X) C
```

IMPORTANT: We ask you to come up with four **different** transaction schedules of `T1`, `T2`, and `T3` satisfying different constraints given below. All schedules should only involve the above read, write, and commit actions. In particular, you are **not** supposed to insert lock acquisition and release actions explicitly. For each schedule **explain** why it satisfies the constraints.

Schedules, 3.1.1: Give a serializable transaction schedule of `T1`, `T2`, and `T3` that is **not** serial.

Solution

```
T1:           R(X)         R(Y) W(Y) C
T2: R(Y) W(Y)      W(Z) C
T3:                                    W(Z) R(X) W(X) C
```

Equivalent to T2;T1;T3, not serial because interleaved. Arguing about conflict serializability is also fine. (2%)

Schedules, 3.1.2: Give a transaction schedule of `T1`, `T2`, and `T3` that is **not** conflict-serializable.

Solution

```
T1: R(X) R(Y)            W(Y) C
T2:            R(Y) W(Y)         W(Z) C
T3:                                    W(Z) R(X) W(X) C
```

The precedence graph has a cycle `T1 -RW(Y)-> T2 -RW(Y)-> T1`. (2%)

Schedules, 3.1.3: Give a conflict-serializable transaction schedule of `T1`, `T2`, and `T3` that could **not** have been generated by a scheduler using strict two-phase locking (S2PL) with lock upgrades (from shared to exclusive).

Solution

```
T1:             R(X) R(Y) W(Y) C
T2:  R(Y) W(Y)                   W(Z) C
T3:                                       W(Z) R(X) W(X) C
```

Precedence graph T2 -> T1, T1 -> T3, T2 -> T3 (no cycle). S2PL acquires locks gradually, but releasess them at once at the end. T1 cannot grab any lock for Y, because an exclusive lock is held by T2 at that point (3%)

Schedules, 3.1.4: Give transaction schedule of `T1`, `T2`, and `T3` that could have been generated by a scheduler using strict two-phase locking (S2PL) with lock upgrades (from shared to exclusive) **and** by a scheduler using conservative two-phase locking (C2PL) with lock downgrades (from exclusive to shared).

Solution

```
T1: R(X) R(Y) W(Y) C
T2:                 R(Y) W(Y) W(Z) C
T3:                                     W(Z) R(X) W(X) C
```

Serial schedules can be generated by either scheduler. (2%)

Schedules, 3.1.5: For every of the four transaction schedules you have given, argue if it could have been generated by a two-phase locking scheduler (2PL) with lock upgrades (from shared to exclusive) and lock downgrades (from exclusive to shared). Give four yes/no answers and briefly justify each of them.

Exam number: _____

1) Depends on the answer. For my answer, **yes**, works even with S2PL (show how to insert locks).

2) **No**, 2PL produces c-s schedules. But this was not proved in the lecture, so need to argue about locks.

Problem in T2 when acquiring y.

3) Depends on the answer. For my answer, **yes**, T2 can grab X(Z) and release X(Y) afterwards, before T1

starts.

4) **Yes**, any serial schedule also works with 2PL.

(4%)

3.2 Cascading Aborts (3%)

Give a transaction schedule of three (uncommitted) transactions `Ta`, `Tb`, and `Tc` that could have been generated by conservative two-phase locking (C2PL) with lock downgrades (from exclusive to shared) and in which the abort of one transaction will cause the abort of the other two transactions. Explain your solution, in particular justifying why the caused aborts must happen. **Note**: Unlike in the previous exercise, you may choose which actions comprise `Ta`, `Tb`, and `Tc`.

Solution

```
Ta: R(X) W(X)
Tb:            R(X)
Tc:                 R(X)
```

Could include locks above. Abort of Ta causes the abort of Tb and Tc, which read dirty data written by

Ta.

3.3 Deadlocks (4%)

Consider the following transaction schedule:

```
Ta: R(X)        W(X)        W(Y)       C
Tb:       R(Y)        W(X)        W(Z)    C
```

Deadlocks, 3.3.1: We consider a strict two-phase locking scheduler (S2PL) with lock upgrades (from shared to exclusive) observing the above actions arriving in the schedule order from left to right. Will the S2PL scheduler necessarily produce a deadlock? Justify your answer.

Yes, deadlock after three executed actions. Construct wait-for graph. Tb waits for Ta to realease exclu-

sive lock for X. Ta waits for Tb to release shared lock for Y. Arguing over c-s is also fine. (2%)

Exam number: _____

Deadlocks, 3.3.2: We now consider a two-phase locking scheduler (2PL) with lock upgrades (from shared to exclusive) and lock downgrades (from exclusive to shared) observing the above actions arriving in the schedule order from left to right. Will the 2PL scheduler necessarily produce a deadlock? Justify your answer.

Yes, deadlock after three executed actions. Same as before, but now need to consider potential lock releases. Yet, Ta can't release the lock for X before acquiring the lock for Y, which in turn is held by Tb and can't be released before acquiring the lock for X. (2%)

4 Recovery (17% in total)

We consider a transactional database that uses the ARIES algorithm as its recovery mechanism. At the beginning of time, there are no transactions active in the system and no dirty pages. A checkpoint is taken. After that, three transactions, T1, T2, and T3, enter the system and perform various operations, **including at least one page update per transaction**. Eventually, a crash happens, the system restarts and proceeds with the recovery. As the result of the analysis phase, the algorithm arrives at the following transaction table (sorted by lastLSN) and dirty page table (sorted by recLSN):

XACT_ID	status	lastLSN
T1	running	10
T3	committed	7

PID	recLSN
P42	3
P65	8
P(66 + <EN>)	4 + (<EN> modulo 3)

Here, <EN> refers to your personal Exam Number. For example, if your Exam Number is 154 then the last line of the dirty page table reads "P220 | 5" (because 4 + (154 modulo 3) is 4 + 1 = 5).

4.1 The Missing Log (8%)

Complete the below log, in the usual format, to be a log at the moment of the crash, which could have led to the shown situation. Assume that the log sequence numbers are the (consecutive) natural numbers starting with 1. Note that T1 is required to be active at LSN 3.

```
LSN     PREV_LSN     XACT_ID     TYPE         PAGE_ID   UNDONEXTLSN
---     --------     -------     ----         -------   -----------
1       -            -           begin CKPT   -         -
2       -            -           end CKPT     -         -
3       -            T1          ...
...
```

Solution

```
LSN     PREV_LSN     XACT_ID     TYPE         PAGE_ID   UNDONEXTLSN
---     --------     -------     ----         -------   -----------
1       -            -           begin CKPT   -         -
2       -            -           end CKPT     -         -
3       -            T1          update       P42       -
4       -            T2          update       P42       -
5       -            T3          update       P220      -
6       4            T2          update       P220      -
```

Exam number: _____

7	5	T3	commit	-	-
8	6	T2	update	P65	-
9	8	T2	commit	-	-
10	3	T1	update	P65	-
11	9	T2	end	-	-

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX CRASH XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

OK to either commit or abort (and undo) T2; the end record for T2 must be there. But the end log record T3 should not be there.

4.2 The Recovery (9%)

Based on your log, complete the recovery procedure. Show:

1. the sets of winner and loser transactions;
2. the values for the LSNs where the redo phase starts and where the undo phase ends;
3. the set of log records that may cause pages to be rewritten during the redo phase;
4. the set of log records undone during the undo phase;
5. the contents of the log after the recovery procedure completes.

For each of the items above, briefly justify your answer.

Solution

1. Winner T3 (and T2). Loser T1. (1 %)
2. Redo starts at 3 (min. recLSN in DPT), undo ends at 3 (oldest LSN belonging to a loser) (2 %)
3. Redo 3, 4, 5, 6, 8, 10 (all pages may have been in memory at the moment of the crash). (1 %)
4. Undo 10, 3 (all belong to loser transactions) (1%)
5.

LSN	PREV_LSN	XACT_ID	TYPE	PAGE_ID	UNDONEXTLSN
1	-	-	begin CKPT	-	-
2	-	-	end CKPT	-	-
3	-	T1	update	P42	-
4	-	T2	update	P42	-
5	-	T3	update	P220	-
6	4	T2	update	P220	-
7	5	T3	commit	-	-
8	6	T2	update	P65	-
9	8	T2	commit	-	-
10	3	T1	update	P65	-
11	9	T2	end	-	-

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX CRASH XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

12	7	T3	end	-	-
13	10	T1	abort	-	-
14	13	T1	CLR	P65	3
15	14	T1	CLR	P42	-
16	15	T1	end	-	-

(4%)
1 can be solved without having solved the log exercise
For others, correctness relative to the solution of previous exercise.

Exam number: _____

5 Reliability and Distributed Transactions (30% in total)

A government decides to make a national social media platform, FacePub, to provide an alternative to platforms controlled by BigTech companies in other countries. FacePub allows users to form groups such that all members in the group will receive the messages sent by any other member of the group.

Figure 1 shows a scenario where Alice, Bob, and Emily form a group. Bob sends the message m_1 to FacePub (event e_1), which is sent to Emily and Alice (events e_3 and e_5). After receiving Bob's message (event e_4), Emily sends a reply m_2 (event e_6), which is then sent to Alice and Bob (events e_8 and e_{11}).

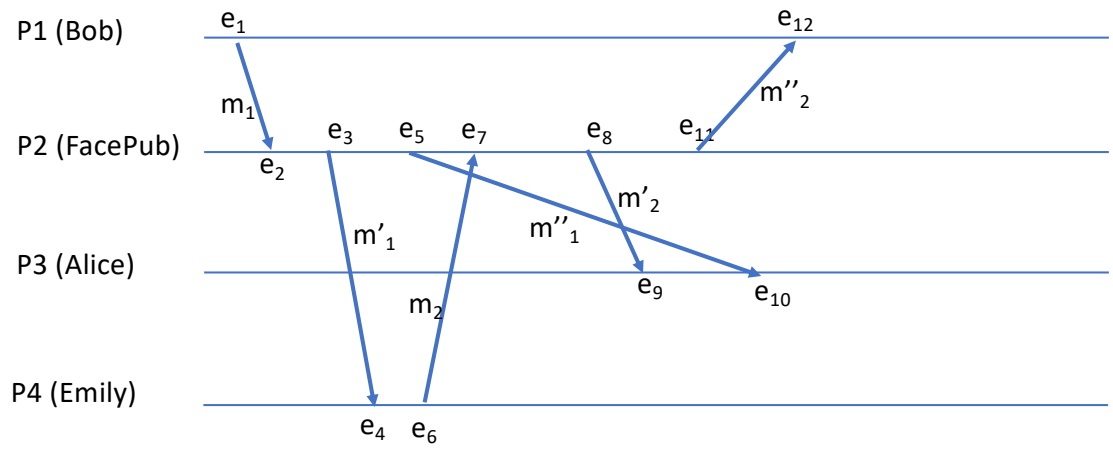

Figure 1: FacePub Scenario

5.1 Lamport and Vector Clocks (10%)

Based on this scenario and the messages exchanges shown in Figure 1, answer the following questions.

Lamport and Vector Clocks, 5.1.1: Write the Lamport logical clock of each event, assuming that each process starts with a clock at 0.

e_1: 1, e_2: 2, e_3: 3, e_4: 4, e_5: 4, e_6: 5, e_7: 6, e_8: 7, e_9: 8, e_{10}: 9, e_{11}: 8, e_{12}: 9

Lamport and Vector Clocks, 5.1.2: Write the vector clock of each event, assuming that each process starts with a clock at $(0,0,0,0)$

e_1: (1,0,0,0), e_2: (1,1,0,0), e_3: (1,2,0,0), e_4: (1,2,0,1), e_5: (1,3,0,0), e_6: (1,2,0,2), e_7: (1,4,0,2), e_8: (1,5,0,2), e_9: (1,5,1,2), e_{10}: (1,5,2,2), e_{11}: (1,6,0,2), e_{12}: (2,6,0,2)

Exam number: _____

Lamport and Vector Clocks, 5.1.3: Alice receives Emily's message before Bob's message. This is confusing for Alice, since Emily replied to Bob. Can Alice tell, using Lamport logical clock or vector clock, that the messages are not received in the same order as the messages appear in Emily's chat? If yes, how? If not, why? Briefly justify your reasoning.

Lamport logical clock: no; know that m1 was sent before m2, but this is not enough to know in which order they were received

Vector clock: yes; FacePub's clock is attached to the messages

5.2 Scalability and Reliability (10%)

Scalability and Reliability, 5.2.1: The government expects FacePub to be accessed by about 2 million users. Which kind of replication would you choose to scale the system to be fast and work for that many users?

open question; different answers possible. Natural choice: P2P, could argue that actions are independend and conflicts are rare

Scalability and Reliability, 5.2.2: Explain if the kind of replication you choose in the previous question also provides better reliability in case of failure of the FacePub service. If yes, how? If not, suggest another replication scheme that provides better reliability.

P2P good reliability. Synchronous replication could be even better. E.g., if we can't tolerate message loss.

Exam number: _____

5.3 Two-Phase Commit (10%)

The government decides to implement another version of the FacePub system, where Two-Phase Commit (2PC) is used to carry out the distribution of messages to all group members. Suppose the processes are running on separate machines, and the transaction coordinator runs on the machine of the FacePub service.

Two-Phase Commit, 5.3.1: Briefly describe the procedure of 2PC to commit for handling the message received from Bob. Write down the coordinator's log records regarding the 2PC protocol in this case. For each log record, specify its type, such as "prepare", and the information necessary to tolerate node failures.

Bob starts the protocol. Transaction is prepared by each other participant.

If all say yes, then log commit and send "follow" message to all participants.

If ack received by all participants, log end.

Log: prepare -> commit -> end

Two-Phase Commit, 5.3.2: Describe a scenario (using 2PC) where Emily fails just after having received message m_1' from Figure 1.

Transaction blocked until Emily returns

Coordinator may abort

If Emily recovers and still has the message, she may proceed in the usual 2PC manner (respond to coordinator)

If Emily recovers, but looses the message, Emily will only know about the transaction if the prepare is resend by the coordinator

Exam number: _____

6 Data Processing (20% in total)

TicketHub is a company offering online ticket selling services for events. The data analyst at TicketHub is required to analyze some sales statistics. The following tables are provided to the data analyst:

- $\mathtt{Tickets}(\underline{\mathtt{tID}}, \mathtt{cID})$, which stores the ID of the ticket (tID) and the ID of the customer (cID) who purchased the ticket.
- $\mathtt{Customers}(\underline{\mathtt{cID}}, \mathtt{name}, \mathtt{age}, \mathtt{city})$, which stores the information of each customer, including ID, name, age and city of residence.

The $\mathtt{Tickets}$ table has $120,000$ pages, and $\mathtt{Customers}$ has $5,000$ pages.

6.1 Selection (5%)

The first task is to select all the records in $\mathtt{Customers}$, where $\mathtt{age}$ is greater than 50 and $\mathtt{city}$ is equal to "Copenhagen".

Selection, 6.1.1: Suppose there is a clustered B+ tree index on $\mathtt{Custerms.age}$, and a non-clustered B+ tree index on $\mathtt{Customers.city}$. Both indices are indirect indices, i.e. the data entries in the leaf nodes of the B+ Trees contain pointers to the actual data records. There are $150,000$ records having $\mathtt{age} > 50$, $100,000$ records having $\mathtt{city} = \text{"}\mathtt{Copenhagen}\text{"}$, and $60,000$ records meeting both conditions. Suppose each page can contain 100 records, and both indices are already fully loaded into the main memory. What are the possible approaches to answer the selection query? What is the worst-case I/O cost of using each of these approaches? Briefly explain your answer?

1) Sequential scan: $5,000$ I/O.

2) Using the index on $\mathtt{age}$, $1,500$ I/O

3) Using the index on $\mathtt{city}$, $100,000$ I/O

4) Using both the indices on $\mathtt{age}$ and $\mathtt{city}$. Read the data entries from both indices that meet the two conditions respectively. For each data entry from the $\mathtt{age}$ index, if it is not contained in the data entries of the $\mathtt{city}$ index, remove it. Finally, we get the pointers to the $60,000$ records that meet both conditions. Since the records are sorted on $\mathtt{age}$, and we do not change the order of the data entries from the $\mathtt{age}$ index, the worst-case I/O of reading the result records is $1,500$.

Exam number: _____

6.2 Join and Aggregate (15%)

Another task of the data analyst is to produce sales statistics: "for each customer, return the total number of tickets purchased by them". The schema of the results should be $(\underline{\text{cID}}, name, age, city, nTickets)$. Suppose the database server used by the data analyst has 200 buffer pages. Answer the following questions.

Join and Aggregate, 6.2.1: The first step is joining the tables Tickets and Customers on cID. Consider Grace Hash Join and Sort Merge Join. Which one of them is the most efficient to perform this step? Briefly explain your answer.

Grace hash join is the most efficient. $5000 > 200 > \sqrt{5000}$, so it needs 2 passes. I/O= $3 * (120,000 + 5,000) = 375,000$. SMJ cannot be completed in 2 passes, because it requires $B > \lceil \sqrt{5000} + \sqrt{120000} \rceil = 346$. But we have only $B = 200$.

Join and Aggregate, 6.2.2: To optimize the Sort Merge Join in the previous question, you should consider the use of Tournament Sort, and the optimization of doing the join during the final merging pass with sorting. Furthermore, you should consider further opportunities to avoid the unnecessary merging to reduce the I/O cost of Sort Merge Join. What is the expected I/O cost for the most efficient Sort Merge Join algorithm for joining Tickets and Customers. Briefly explain your answer.

The 0th pass of C produces 13 runs, and the 0th pass of T produces 300 runs. We then merge (only) 115 runs from T, after which we have 300-115+1 = 186 runs. Now we have 13 runs from C and 186 runs from T. We can use 199 buffer pages as the input buffer to load one page from each of these runs, and do the merge join. The total cost would be 3*5,000 + 2*120,000 + 2*115/300*120,000 + 185/300*120,000 = 421,000

Join and Aggregate, 6.2.3: The next step is grouping the records of the results from the previous step based on the values of cID, calculating the count of each group of records. Note that final results should be sorted based on cID. Suppose the previous step is carried out by using Grace Hash Join, and the join results have 360,000 pages. How would you carry out this step? What is its I/O cost? Briefly explain your answer.

Sort the data on cID. The data can be sorted in 3 passes. Generate the aggregate while merging the sorted runs in the final pass. IO=$5 * 360,000$. A similar optimization as in the previous can be done to reduce the unnecessary merges. The hash-based algorithm can generate the groupby aggregate with the same I/O. But it cannot generate sorted results, hence subsequent sorting is needed, which makes it more expensive.

Join and Aggregate, 6.2.4: Reconsider the scenario in the previous question. Suppose Sort Merge Join is used instead of Grace Hash Join in the previous step. What should be the algorithm to perform the grouping and calculate the aggregate? What is its I/O cost? By considering both this and the previous step together, which join algorithm would you choose for the previous step? Briefly explain your answer.

Since the join results have already been sorted by cID, we can simply scan the join results and calculate the aggregate. The I/O cost would be $360,000$. By considering both steps together, SMJ should be chosen.