CS150A Homework 3 – Writing

School of Information Science and Technology
June 20, 2024

1 Parallel Query Processing(20 pts)

We've discussed 4 kinds of parallel join in class:

- **Parallel Hash Joins**: Use hash partitioning on both relations with the same hash function, then perform a normal hash join on each machine independently.
- **Parallel Sort Merge Join**: Use range partitioning with the same ranges on both relations, then perform sort merge join on each machine independently.
- **One-sided shuffle Join**: When one relation's data is already partitioned the way we want (hash partitioned or range partitioned on a key), just partition the other relation, then run local join (using any algorithm) at every node and union results.
- **Broadcast Join**: If one relation is small, send it to all nodes that have a partition of the other relation. Do a local join at each node (using any algorithm) and union results.

In shared nothing, the machines communicate with each other solely through the network by sending data to each other. Here, network cost is referred to the amount of data sent between machines.

Now we have a Relation R that has 10,000 pages, round-robin partitioned across 5 machines (M1, M2, M3, M4, M5). Relation S has 40 pages, all of which are only stored on M1. We want to join R and S on the condition R.col = C.col. Assume the size of each page is 1 KB.

1. Which type of join would have the lowest network cost in this scenario?
   a) Parallel Hash Joins
   b) Parallel Sort Merge Join
   c) One-sided shuffle Join
   d) Broadcast Join

   D
2. How many KB of data must be sent over the network to join R and S using this join method?

   4*40=160KB

3. Would the amount of data sent over the network change if R was hash or range partitioned among the 5 machines rather than round-robin partitioned using this join method?
   a) The network cost will not change under both of the partitioning methods.
   b) The network cost will change under both of the partitioning methods.
   c) The network cost will only change under range partitioning.
   d) The network cost will only change under hash partitioning.

   A

2 DISTRIBUTED TRANSACTION(20 PTS)

Choose the correct answers for the following questions.

1. Imagine that during a distributed transaction that uses Two-Phase Commit, the following steps take place.
   - The coordinator asks every worker to PREPARE a transaction,
   - Every worker force-writes the PREPARE message,
   - And the coordinator crashes before taking any further action.

   Which of the following statements is true?
   a) Modifications during the write are now visible to other transactions
   b) The transaction will be committed when the coordinator recovers
   c) The transaction will be aborted when the coordinator recovers
   d) The transaction will be committed if more than half of the workers report the transaction as successful

   C

2. Suppose we have a Coordinator and 4 Participants. When could the Coordinator **ABORT** a transaction?
   a) The Coordinator has written ABORT in its log
   b) The Coordinator sent PREPARE to all Participants and received 4 YES votes
   c) The Coordinator sent COMMIT to all Participants and received 2 ACK but hasn't yet heard from the other nodes
   d) A Participant has written ABORT in its log
   e) A Participant has written COMMIT in its log
   f) A Participant has voted YES

   ADF

3. Suppose we have a Coordinator and 4 Participants. When could the Coordinator **COMMIT** a transaction?

a) The Coordinator has written ABORT in its log

b) The Coordinator sent PREPARE to all Participants and received 4 YES votes

c) The Coordinator sent COMMIT to all Participants and received 2 ACK but hasn't yet heard from the other nodes

d) A Participant has written ABORT in its log

e) A Participant has written COMMIT in its log

f) A Participant has voted YES

BCEF

3 Concurrency Control(20 pts)

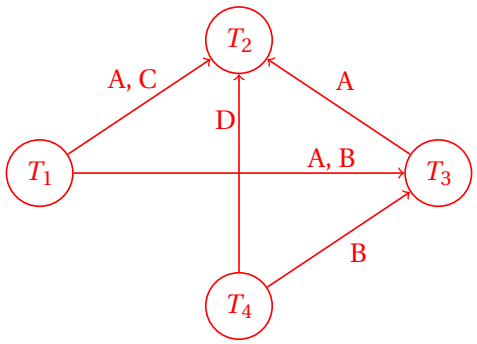
1. (6 pts)Consider the schedule given below in Table 3.1. R(·) and W(·) stand for 'Read' and 'Write', respectively.

time	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
T_1		R(B)		W(C)	W(A)					
T_2	W(E)			R(E)			W(C)	R(D)	W(A)	
T_3						R(A)	W(B)			R(B)
T_4			R(D)	R(B)		W(D)				

Table 3.1: A Schedule with 4 transactions

a) (3 pts)Draw the dependency graph of the schedule given above.

Solution:



For A, there is a W-W conflict from T_1 to T_2, a W-R conflict from T_1 to T_3, and an R-W conflict from T_3 to T_2. For B, there is an R-W conflict from T_1 to T_3 and an R-W conflict from T_4 to T_3. For C, there is a W-W conflict from T_1 to T_2. For D, there is a W-R conflict from T_4 to T_2. For E, there are no conflicts.

b) (3 pts)Is this schedule serial? Is this schedule conflict serializable? Is this schedule possible under regular 2PL? Explain your answer.

This schedule isn't serial because this schedule interleaves the actions of different transactions. This schedule is conflict-serializable because there are no cycles in its data dependency graph. This schedule is possible under 2PL because it is conflict-serializable, and 2PL is guaranteed to produce conflict serializable schedules.

2. (6 pts)Consider the schedule given below in Table 3.2. S(·) and X(·) stand for 'shared lock' and 'exclusive lock', respectively. T_1, T_2, T_3, and T_4 are four transactions. LM stands for 'lock manager'. Transactions will never release a granted lock

time	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
T_1	S(B)						S(A)	
T_2		S(D)			X(B)			
T_3			X(A)	X(D)				S(C)
T_4						X(C)		
LM	g							

Table 3.2: Lock requests of 4 transactions

a) (3 pts)For the lock requests in Table 3.2, determine which lock will be granted or blocked by the lock manager. Please write 'g' in the LM row to indicate the lock is granted and 'b' to indicate the lock is blocked, or the transaction has already been blocked by a former lock request. For example, in the table, the first lock (S(A) at time t_1) is marked as granted. Write your answer below:

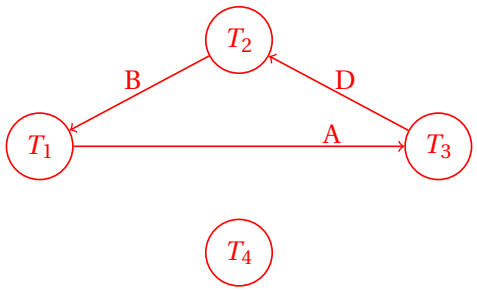
time	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
LM	g							

Solution:

time	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
LM	g	g	g	b	b	g	b	b

b) (3 pts)Draw the waits-for graph for the lock requests in Table 3.2 and explain whether there is a deadlock.

Solution:



Cycle ($T_3 \leftarrow T_2 \leftarrow T_1 \leftarrow T_3$) exists and schedule deadlocks.

c) (4 pts)To prevent deadlock, we use the lock manager (LM) that adopts the Wait-Die policy. We assume that in terms of priority: $T_1 > T_2 > T_3 > T_4$. Here, $T_1 > T_2$ because T_1 is older than T_2 (i.e., older transactions have higher priority). Please mark 'g' for grant, 'b' for block (or the transaction is already blocked), 'a' for abort, and '-' if the transaction has already died. Write your answer below:

Solution:

time	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
LM	g							

time	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
LM	g	g	g	a	a	g	g	-

d) (4 pts)Now we use the lock manager (LM) that adopts the Wound-Wait policy. We assume that in terms of priority: $T_1 > T_2 > T_3 > T_4$. Here, $T_1 > T_2$ because T_1 is older than T_2 (i.e., older transactions have higher priority). Please mark 'g' for grant, 'b' for block (or the transaction is already blocked), 'a' for abort, and '-' if the transaction has already died. Write your answer below:

time	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
LM	g							

Solution:

time	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
LM	g	g	g	b	b	g	a	-

4 RECOVERY

Your database server has just crashed due to a power outage. You boot it up, find the following log and checkpoint information on disk, and begin the recovery process. Assume we use a STEAL/NO FORCE recovery policy.

LSN	Record	prevLSN
30	update: T3 writes P5	null
40	update: T4 writes P1	null
50	update: T4 writes P5	40
60	update: T2 writes P5	null
70	update: T1 writes P2	null
80	begin_checkpoint	-
90	update: T1 writes P3	70
100	end_checkpoint	-
110	update: T2 writes P3	60
120	T2 commit	110
130	update: T4 writes P1	50
140	T2 end	120
150	T4 abort	130
160	update: T5 writes P2	null
180	CLR: undo T4 LSN 130	150

Transaction Table			Dirty Page Table	
Transaction	lastLSN	Status	PageID	recLSN
T1	70	Running	P5	50
T2	60	Running	P1	40
T3	30	Running		
T4	50	Running		

1. The log record at LSN 60 says that transaction 2 updated page 5. Was this update to page 5 successfully written to disk? The log record at LSN 70 says that transaction 1 updated page 2. Was this update to page 2 successfully written to disk?

   The update at LSN 60 may have been written to disk. The log entry was flushed before the write itself. It was not yet flushed at the time of the checkpoint, but may have been flushed later. The update at LSN 70 was flushed to disk. We know this because it's not in the dirty page table at the time of the checkpoint.

2. At the end of the analysis phase, what transactions will be in the transaction table, and what pages will be in the dirty page table?

   Solution: Note that P1 and P5 were already in the dirty page table during the checkpoint, and their recLSN's do not change during analysis. Any transactions that were running (T1, T3, and T5) after the log was processed need to be aborted. This involves writing an abort log record, changing the status to aborting, and updating the lastLSN.

3. At which LSN in the log should redo begin? Which log records will be redone (list their LSNs)? All other log records will be skipped.

   Redo should begin at LSN 40, the smallest of the recLSNs in the dirty page table. The following log records should be redone: 40, 50, 60, 90, 110, 130, 160, 180. 30 is skipped because it precedes

Transaction Table		
Transaction	**lastLSN**	**Status**
T1	190	Aborting
T3	200	Aborting
T4	180	Aborting
T5	210	Aborting

Dirty Page Table	
PageID	**recLSN**
P1	40
P2	160
P3	90
P5	50

Figure 4.1: answer of q2

LSN 40. 70 is skipped because P2.recLSN = 160 > 70. Entries that are not updates are skipped. The CLR record is not skipped, nor is the LSN that it undoes.