Problem Set 3

Release Date: Monday, October 31, 2022

Due Date: Monday, December 5, 2022 by 11:59 pm ET

Late Due Date: Saturday, December 10, 2022 by 11:59 pm ET

Submit to Gradescope: `https://www.gradescope.com/courses/421252`

You may work in pairs on this problem set. Only one of you needs to submit on Gradescope, but the other member must be added as a group member on the submission.

The purpose of this problem set is to give you some practice with concepts related to recovery, parallel query processing, two-phase commit, and other papers we read during the second half of the course.

1 Transactions

Suppose you are running a transactional database system with two types of transactions, against a database with 100 records.

```
T1.  Rx ; read a random record x
     Ry ; read a different random record y

T2.  Rx ; read a random record x
     Wy ; update a random record y based on the value read from x
```

Here, each invocation of T1 or T2 reads a different set of records (i.e., x and y are not always the same record between invocations).

1. [1 points]:

Suppose you run many concurrent instances of T1 and T2 using rigorous two phase locking. Will there be deadlocks? Why or why not?

2. [1 points]:

Suppose you start two transactions running T2 using optimistic concurrency control with serial validation. Both transactions start before either transaction enters the validation stage. What is the probability that one of the transactions aborts?

3. [2 points]: Optional – Extra Credit

Suppose you start 100 transactions running T2 using optimistic concurrency control with serial validation. All transactions start before any transactions enter the validation stage. Estimate the number of queries that will abort.

4. [1 points]:

Suppose you start 1 transaction running T2 and 100 transactions running T1 simulataneously. If you run these transactions at READ UNCOMMITTED serialization level, in a system that uses rigorous two phase locking, is the result gauranteed to be serially equivalent? Why or why not?

2 Serializability

5. [6 points]: Below, we provide 3 interleavings of several concurrent transactions (consisting of READ and WRITE statements on objects).

For each of the interleavings indicate whether:

- The interleaving has an equivalent serial ordering. If so, indicate what the serial ordering is.
- Whether the interleaving would be permitted/could arise under strict two-phase locking.
- Whether the interleaving would be permitted/could arise under rigorous two-phase locking.

Assume in all cases that written values depend on previously read values.

```
Interleaving 1:
1 T1: READ B
2                T2: READ B
3                  T2: WRITE B
4                  T2: COMMIT
5   T1: WRITE B
6 T1: COMMIT

Interleaving 2:
1 T1: READ B
2   T2: READ C
2             T2: READ A
3                           T3: READ C
4             T2: WRITE A
5             T2: COMMIT
6                            T3: WRITE A
7                           T3: COMMIT
8 T1: WRITE A
9 T1: COMMIT

Interleaving 3:
1 T1: READ A
2              T2: READ A
3              T2: READ B
4              T2: WRITE C
5              T2: COMMIT
6 T1: READ C
7 T1: WRITE A
8 T1: COMMIT
```

3 Recovery

6. [4 points]: Suppose you are told that the following transactions are run concurrently on a database system that has just been restarted and is fully recovered, and is running Rigorous 2PL.

```
T1      T2      T3      T4      T5
--      --      --      --      --
RB
                                RE
                                RB
        RA
        RC
        WC
        COMMIT
                RC
                                WE
                        RD
WA
                        WD
                        ABORT
WA
COMMIT
                                WB
                RA
```

Suppose the system crashes after executing WD in T4. Assume that each object (e.g., A, B, etc.) occupies exactly one page of memory.

A. Show all of the records that should be in the log at the time of the crash (given your serial order), assuming that there have been no checkpoints and that you are using an ARIES-style logging and recovery protocol. Your records should include all of the relevant fields described in Section 4.1 of the ARIES paper. Also show the status of the transaction table (as described in Section 4.3 of the ARIES paper) after the analysis phase of recovery has run.

B. Suppose you have 3 pages of memory, and are using a STEAL/NO-FORCE buffer management policy as in ARIES. Given the interleaving you showed above, for each of the 5 pages used in these transactions, show one possible assignment of LSN values for those pages as they are on disk before recovery runs. You should use the value "?" if the LSN is unchanged from the prior state of the page before these transactions began. Finally, indicate which pages will be modified during the UNDO pass, and which will be modified during the REDO pass.

4 Parallel Query Processing

The standard way to execute a query in a shared-nothing parallel database is to horizontally partition all tables across all nodes in the database. Under such a setting, distributed joins can be computed by repartitioning tables over the join attributes.

Suppose these tables are partitioned across a 5 node distributed database, where each node has 10 TB of disk storage and 2 GB of RAM. Also, assume that:

- The disk can read at 500 MB/sec (for the purposes of this problem, you may ignore differences between sequential and random I/O),
- The network on each node can transmit data at 150 MB/sec, regardless of the number or rate at which other nodes are simultaneously transmitting,
- A computer cannot send over the network and read or write from its disk at the same time,
- CPU costs are negligible relative to disk or network I/O time.
- The network costs to compute the final aggregate are negligible.

For the following query and partitioning strategies of the database, estimate the total runtime of the query. Show your work by drawing or describing the query plan. All joins are in-memory hash joins or Grace hash joins. The database is a row-store. Assume projections and selections are done as early as possible.

```
SELECT COUNT(*)
FROM R JOIN S
ON R.a = S.b //S.b is a primary key, R.a is uniformly distributed
WHERE S.c > 20 //selectivity of .15
```

Here, S.b is a primary key, R contains 2 billion records, S contains 12 billion records. All fields and COUNTs are integers, and integers are 8 bytes (64 bits). Each table contains 10 integer fields, so the size of the tables are:

$$|R| = 2 \text{ billion} \times 8 \text{ bytes per field} \times 10 = 160 \text{ GB}$$

$$|S| = 12 \text{ billion} \times 8 \text{ bytes per field} \times 10 = 960 \text{ GB}$$

7. [2 points]: R is hash partitioned across all nodes on R.a and S is hash partitioned on S.b. Estimate the total query runtime, showing your work.

8. [2 points]: Now suppose R is hash partitioned across all five nodes on R.a and S is hash partitioned on S.c. Estimate the total query runtime, showing your work.

5 Locking, Redux

Consider the following transactions, with initial values of X=2 and Y=7. For writes we also indicate the values that are written (i.e., `WY ;Y=tx*2` means that a write to Y is done that sets its value to `tx*2`.)

```
T1             T2            T3
----------     ----------    --------
tx = RX        ty = RY       WY ;Y=5
WY ;Y=tx*3     WX ;X=ty+1
```

Suppose you know that the first operation that runs is RX in T1.

9. [2 points]: Assuming there are no deadlocks, list all possible values for X and Y that could result from an execution of these transactions with rigorous two-phase locking at the serialization level SERIALIZABLE?

10. [2 points]: Now suppose you run these transactions at the serialization level READ COMMITTED. List all possible values for X and Y that could result from an execution of these transactions? Assume READ COMMITTED is implemented as described in lecture 13, i.e., transactions only hold short read locks (continuing to hold write locks for the duration of the transaction).

11. [2 points]: Now suppose you run these transactions at the serialization level READ UNCOMMITTED. List all possible values for X and Y that could result from an execution of these transactions? Assume READ UNCOMMITTED is implemented as described in lecture 13, i.e., transactions don't acquire read locks at all (continuing to hold write locks for the duration of the transaction).

6 Changes Since Release

- **[11/15/2022] – Q2 Adjustment.** Question 2 was harder than intended. We have adjusted the question. The original question is now optional and worth two points of extra credit.