6.830 Problem Set 3

Assigned: 10/28
Due: 11/30

The purpose of this problem set is to give you some practice with concepts related to query optimization and concurrency control and recovery.

1 Parallel Query Processing

Recall that 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. Recall also that hashing can be used to repartition one or both tables as needed to compute joins in a distributed fashion. Suppose you are trying to compute the following query:

```
SELECT *
FROM R,S
WHERE R.a > v1
AND S.b > v2
AND R.c = S.d
```

You are given the following:

- Both tables are 5,000 MB,
- The disk can read at 50 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 40 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,
- The selectivity of both selection predicates is 0.2,
- Each tuple in R joins with exactly one tuple in S,
- Each machine in your distributed database has 500 MB of memory.

Question 1: Suppose both tables are stored on a single node. Describe the best query plan for executing this query on that node.

Question 2: Ignoring CPU costs, estimate the time to answer this query on a single node.

Question 3: Now, suppose that the tables are hash-partitioned on R.a and S.b across a 4 node distributed database. Describe the best distributed query plan for executing this query.

Question 4: Ignoring CPU costs, estimate the time to answer this query on the distributed database.

Question 5: Now, suppose that the tables are hash-partitioned on R.c and S.d across a 4 node distributed database. Describe the best distributed query plan for executing this query.

Question 6: Ignoring CPU costs, estimate the time to answer this query on the distributed database.

2 Locking

Below are 3 simplified tables from the Wikipedia database.

```
page:     (id int auto_increment, title char(100), latest int)
revision: (id int auto_increment, comment char(100), pageid int, textid int)
text:     (id int auto_increment, content char(5000))
```

`page.latest` references `revision.id`, `revision.pageid` references `page.id`, and `revision.textid` references `text.id`. Each `id` field starts at 0 and are ordered contiguously.

There are clustered B+Trees on `page.id`, `revision.id`, and `text.id`, as well as a secondary B+Tree index on page.title.

There are 10M pages, 200M revisions, and each revision has one text tuple (200M text tuples). Integers occupy 4 bytes. Data pages hold 8,000 bytes, and each index page holds 500 pointers and key values.

You can assume that each page has the same number of revisions.

For each of the following queries:

- Indicate the query plan that the database would most likely use to answer the query.
- Estimate the number of read (S) and write (X) locks that would be acquired by the execution of your plan assuming the use of page level locking. Assume that locks must be acquired on both index and data pages before they are read or written.
- Estimate the the number of read and write locks that would be required by your plan assuming row locking? Suppose that locks must be acquired for each distinct value read from an index (e.g., if the actor with id "10" is read from the database, a single index lock is acquired.)

Question 7:

a. `SELECT * FROM pages WHERE id > 5,000,000`

b.
```
INSERT INTO revision
VALUES (default, 'some comment', 1000, 10000)
```

c.
```
UPDATE revision
SET comment = 'new comment'
WHERE pageid =25000
```

Question 8:

Describe a workload where using table level locking will out perform page or row level locking. Please explicitly write down an example of the types of queries you have in mind and provide a quick explaination of your reasoning.

3 ARIES

Question 9:

Suppose an ARIES-based database system (using page-level logging) runs the following transaction over a single-table database of gradstud with the schema <studid int, name char(100), area char(100), officeid int>. Assume that the table is stored in a heap file on disk but that records were inserted in officeid order, such that records in the file are sorted by officeid (but the database system does not maintain this sorted order as additional updates happen.) Each disk page is 2000 bytes. Due to the recession, the students are packed like sardines. Students 0-4 are in office 1, students 5-9 are in office 2, and students 10-14 are in office 3. To make room for a particularly bright new student, existing students need to be shuffled around... transactionally.

```
BEGIN TRANSACTION
SELECT name
  FROM gradstud
  WHERE officeid = 1;
UPDATE gradstud
  SET officeid = 2
  WHERE studid = 1;
ABORT;

BEGIN TRANSACTION
UPDATE gradstud
  SET officeid = 2
  WHERE officeid = 3;
INSERT INTO gradstud
  VALUES (15, 'Billy', 'Quizboy', 3)
COMMIT
```

If these are the only transactions that have ever run on the system, what will the contents of the log look like after they complete? Show the log records, indicating their type, the transaction they apply to, and any additional data needed to describe the state of the records. You don't need to write out the contents of the before and after images in detail, but you should describe what they consist of.

The ARIES paper mentions that CLRs allow it to avoid performing an UNDO more than once if there is a crash during recovery: "... ARIES will rollback only those actions that had not already been undone. This is possible since history is repeated for such transactions and since the last CLR written for each transaction points (directly or indirectly) to the next non-CLR record that is to be undone." (p. 112).

Question 10: You might think that it would be enough to have each CLR record just point to the next record that is to be undone – that, is, to omit the redo information in the PageID and Data fields of the log record (p. 113), so that a CLR contained only the fields LSN, Type, TransID, and UndoNxtLSN. Explain why this would be a bad idea.

Question 11:

Suppose an ARIES-based database crashes. When it comes back online, you find the log and disk pages look as follows (here, the notation `UPDATE T1, X` means that transaction T1 updated page X.) Assume that update data pages are only flushed to disk when a `CHECKPOINT` is taken.

```
 0 BEGIN  T1
 1 BEGIN  T2
 2 UPDATE T1 A
 3 UPDATE T2 B
 4 UPDATE T1 C
 5 CHECKPOINT
 6 ABORT  T1
 7 CLR    LSN=4
 8 BEGIN  T3
 9 COMMIT T2
10 UPDATE T3 B
11 CHECKPOINT
12 UPDATE T3 D
   *crash*
```

a. What transactions were running at the time of the crash?

b. What transactions will the system UNDO?

c. What data pages will the system modify during the REDO pass of recovery?

d. What data pages will the system modify during the UNDO pass of recovery?

4 Two phase commit

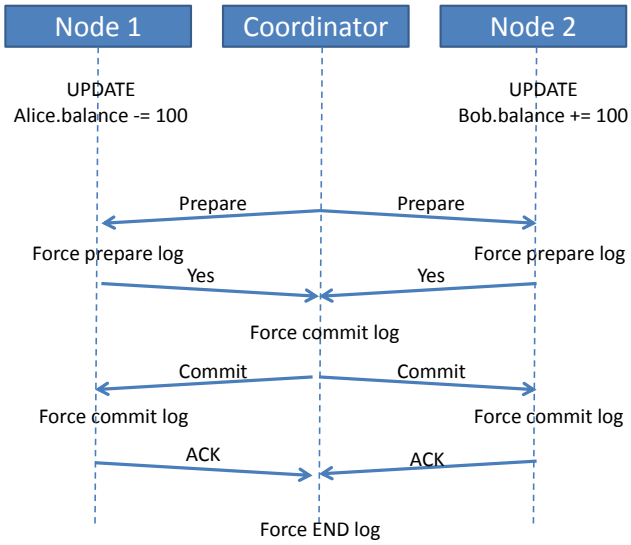

Figure 1: Diagram of a 2PC protocol that successfully commits

Two-phase commit is a protocol that ensures atomic commits in the face of *temporary* system failures. Figure 1 shows a diagram of a balance transfer of $100 from Alice to Bob followed by a normal execution that successfully commits. The following questions will insert failures or slightly change the protocol and ask you to reason through the consequences.

Question 12: Ben BitDiddle forgets to flush the COMMIT log record on the Coordinator node before sending the COMMIT messages. Suppose the Coordinator crashes after sending the messages and Node 1 crashes prior to recieving the commit message. Describe why the state may be inconsistent when both machines are brought back up.

Question 13: Suppose Node 1 writes a PREPARE log (that is buffered in memory) but does not flush the log record to persistant storage. Describe a failure that will lead to inconsistent state and report Alice and Bob's resulting account balances.

Question 14: Suppose Node 1 writes a COMMIT log (that is buffered in memory) but does not flush the log record to persistant storage. Describe a failure that will lead to inconsistent state and report Alice and Bob's resulting account balances.

Question 15: Suppose Node 1, Node 2, and the Coordinator are geographically far apart and network roundtrips are slow. How does that impact the performance of this protocol?

Question 16: Recall that 2PC requires 2 log writes and 2 network messages from each participant. Describe a workload for which 2PC may not be the best choice. Describe a workload where this overhead is acceptable.

5 Column Stores

Ben BitDiddle has a large database where each table consists of hundreds of Integer columns. Ben reads the C-Store paper during 6.814 and is convinced that moving his data into a column-oriented database will dramatically reduce the amount of storage space his data uses, and also speed up his queries. Short on time, he installs a traditional row-store database (e.g., PostgreSQL) and stores each column as a separate table.

Question 17: Contrary to his expectations, the storage size of his data in his new database has actually *grown*! Why may this be the case?

Question 18: In general, what types of `SELECT` queries would perform faster on row-oriented databases than on column oriented databases?

6.830 / 6.814 Database Systems

Fall 2010