Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.830 Database Systems: Fall 2005

Quiz II Solutions

There are 15 questions and 11 pages in this quiz booklet. To receive credit for a question, answer it according to the instructions given. *You can receive partial credit on questions.* You have **80 minutes** to answer the questions.

Write your name on this cover sheet AND at the bottom of each page of this booklet.

Some questions may be harder than others. Attack them in the order that allows you to make the most progress. If you find a question ambiguous, be sure to write down any assumptions you make. Be neat. If we can't understand your answer, we can't give you credit!

THIS IS AN OPEN BOOK, OPEN NOTES QUIZ.
NO PHONES, NO LAPTOPS, NO PDAS, ETC.

Do not write in the boxes below

1-3 (xx/18)	4-6 (xx/33)	7-9 (xx/23)	10-13 (xx/12)	14-15 (xx/14)	Total (xx/100)

Name:

I Parallel Databases

Ben is comparing the performance of a centralized database with a parallel database. His centralized database consists of just a single machine, and the parallel database consists of a cluster of eight machines with the same hardware configuration as the machine in the centralized database. He plans to run queries over a set of stock quotes so that he can game the stock market. His database consists of three tables, a QUOTES table, listing historical stock quotes, a COMPANY table listing information about each company that is traded on the market, and an ORDERS table listing the stock purchase and sale orders he has placed.

The tables have the following schemas:

- QUOTES : (SYMBOL CHAR(4), TIME TIMESTAMP, PRICE REAL).
- COMPANY : (SYMBOL CHAR(4), NAME VARCHAR(100), ADDRESS VARCHAR(255), ...). There are a number of other fields about the financial status of each company.
- ORDERS : (SYMBOL CHAR(4), TIME TIMESTAMP, PRICE REAL, QUANTITY INTEGER). QUANTITY is negative if this is a sale order, and positive if it is a purchase.

He stores all of these tables on the centralized database.

He decides to horizontally partition these tables across all of the machines in the distributed database. He uses hash partitioning on the SYMBOL field of each of the tables. His hash partitioning algorithm does a good job of partitioning the data, so each node has about 1/8th of the tuples.

He runs three different types of queries:

A. A query to compute the most recent stock quote for the symbol 'MSFT'.

B. A query to compute the average price over the last week of each stock, ordered by price.

C. A query to compute which of his purchase orders he has lost money on (e.g., whose most recent price is less than the price he paid.)

1. [6 points]:

When Ben runs query A on the centralized database, he finds that it takes x seconds to complete. When he runs it on all 8 machines, approximately how long do you expect it will take? Justify your answer.
(Write your answer in the space below.)

Approximately x seconds. Only one of those 8 machines will contain all quotes for MSFT, so we execute the query on that machine. This is essentially the same as the centralized database case.

Name:

2. [6 points]:

When Ben runs query B on all 8 machines, he finds that the performance is only 3 times faster than on the centralized database. List one reason why he might be seeing a sub-linear speed up.

(Write you answer in the space below.)

There are several possible explanations:

- **Startup costs***: The extra cost of sending the query to all eight nodes may outweigh the benefit of parallelizing the query.*
- **Shutdown costs***: The cost of collecting results, merging sorted answers, etc. may outweigh the benefit of paralleling the query.*
- **Slow node***: One of the nodes may be slower, and we will have to wait for that slow node (this answer was penalized unless some justification was given, since we were told there is no skew and all nodes have the same hardware configuration.)*

3. [6 points]:

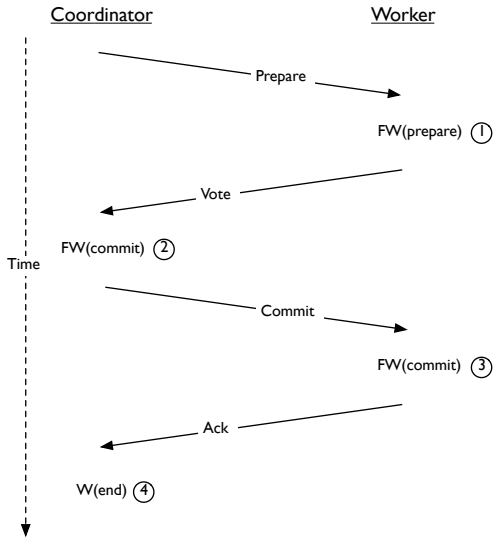
When Ben runs query C on all 8 machines, he finds that the performance is 12 times faster than on the centralized database. List one reason why he might be seeing a super-linear speed up.

(Write your answer in the space below.)

When the query is distributed, each node processes only about 1/8th of the data, which may allow the entirety of the database to fit into the aggregate memory of the cluster. In this case, a non-external join can be used, which is much faster than one of the external join algorithms.

II Two-phase commit

The two phase commit protocol between a coordinator and a worker site can be summarized by the following diagram:



Name:

In this figure, the lines represent messages between the sites, and W(...) and FW(...) represent unforced and forced log writes, respectively.

In this problem, you will analyze why each log write is needed and what would go wrong if it were missing or were not forced. You should assume that all votes are "yes" votes and that the basic variety of two phase commit is being used (i.e., no presumed abort or presumed commit). Also assume that the worker or coordinator can crash at any time, and that the correct behavior of the system is to **commit** if a crash happens after the commit point and to **abort** if the crash happens before the commit point.

4. [20 points]:

- Suppose that the force-write at (1) were an unforced log write. Describe, in one or two sentences, what could go wrong:

  (Write you answer in the space below.)

  If the worker crashes after sending its vote without writing the log record to disk (which contains the information that it is prepared and voted yes), it will abort the transaction on recovery (since it will assume it crashed before preparing.) This is a problem because the coordinator will assume that it can commit because the worker sent a yes vote.

- Suppose that the force-write at (2) did not happen. Describe, in one or two sentences, what could go wrong:

  (Write you answer in the space below.)

  The coordinator may send out a commit message to a subset of the workers and then crash. When it recovers, it will abort the transaction (because it has no commit record for it). When workers poll for the state of the transaction, the coordinator will reply 'abort', which is inconsistent with the fact that the coordinator has already told some workers to commit.

- Suppose that the force-write at (3) were an unforced log write. Describe, in one or two sentences, what could go wrong:

  (Write you answer in the space below.)

  The worker may send the acknowledgment and then crash; since the commit record is not forced, on recovery it will ask the coordinator for the status of the transaction. In the meantime, the coordinator may have heard an from all sites and written the end record and forgotten about the transaction; hence, the coordinator will tell the worker to 'abort', which is incorrect because all other sites committed the transaction.

- Suppose that the unforced write at (4) did not happen. Describe, in one or two sentences, what could go wrong:

  (Write you answer in the space below.)

  If the coordinator crashes and there is no end record in the log, the coordinator will resend commit messages to all of this workers. This is not a problem, as they will have already committed and will hence do nothing and simply send an ack; hence, this only leads to wasted communication when the coordinator crashes.

Name:

III Replication

The goal of the voting-based replication protocol described in the paper by Davidson et al., is to *ensure that each read of an object o returns the most recently written value of o*, even when o is stored at multiple sites and even when only some of those sites are read or written on each access. In particular, an update of an object o replicated on n sites must be sent to a quorum of at least w of those sites, and reads must be done from a quorum of at least r sites. If a quorum of nodes is not accessible, updates or reads must block. Since not all copies of an object are updated by a given write, each object must have a version number stored with it, and readers must return the most recent version that they read. The paper lists the following two constraints on r and w:

A. $r + w > n$

B. $w > n/2$

5. [3 points]: Condition A ensures that any read of o goes to at least one site that has also seen the most recent write of o. What is the purpose of Condition B? What could go wrong if writes were sent to fewer than $n/2$ nodes? Limit your answer to one or two sentences.

(Write your answer in the space below)

Otherwise we might end up with two network partitions of size w, both convinced they have the latest update.

6. [10 points]:

Dana Bass claims that if she stores the time that each object was last modified with each copy of the object, and if clocks between different writers are perfectly synchronized and no two writes happen at exactly the same time, she can eliminate one of the above constraints. Indicate below which constraint is not needed, or circle "both are needed" if you think Dana is wrong. Explain your answer in one or two sentences.

(Circle one.)

A is not needed B is not needed Both are needed

(Write your explanation below.)

In the previous scenario we needed a w larger than $n/2$ to ensure that a write would fail if two different (parallel) writes attempted to use the same version number. However, if we use timestamps and no two writes happen at the same time, condition A is sufficient to ensure that we will read from at least one server with a copy of the object with the latest timestamp.

Name:

IV Online Query Processing

7. [9 points]:

The CONTROL paper presents techniques to improve the responsiveness of query processing. However, CONTROL techniques also tend to slow down the overall processing rate of queries. List three techniques proposed in the paper that promote interactivity but increase the overall completion time of queries:

(Write your answer in the spaces below.)

A. Technique 1: *Index stride prioritizes certain values but decreases the performance of a disk scan as it adds random I/O.*

B. Technique 2: *Outputting partially completed aggregates incurs more communication cost between the database and application.*

C. Technique 3: *Using a ripple join or other online join algorithm slows down performance (as discussed in the next problem.)*

V Symmetric Hash Join

The symmetric hash join algorithm (and variants of it like the ripple join algorithms presented in the Eddies paper) are often used in streaming and adaptive databases to allow a database system to process tuples from either input relation without requiring access to the entirety of the other relation. In the next few questions, you will analyze the performance of this algorithm relative to a traditional hash join.

8. [6 points]:

Suppose you are using a symmetric hash join on two stored relations, R and S, on disk. The algorithm maintains two hash tables, $H(R)$ and $H(S)$ over R and S respectively. The implementation reads a tuple t from S, inserts it into $H(S)$, and then probes into $H(R)$ to find any tuples with which t joins. It then reads a tuple from R, inserts it into $H(R)$ and uses it to probe into $H(S)$. It alternates between the two inputs in this manner until they have both been consumed. You should assume that the algorithm reads a relatively large run of data at a time from each of the base relations (so the the base relations are scanned sequentially off of disk.) For the time being, assume that $H(R)$ and $H(S)$ fit into RAM.

How do you expect the performance of this approach to compare to a simple (in memory) hash join? Will it be substantially more expensive? Why or why not?

(Write your answer in the space below)

Every tuple that we read for either relation requires us to do both a hash table build and probe (as opposed to just a build or probe.) We are also required to keep two hash tables in memory, rather than just one.

Name:

9. [8 points]:

Now, suppose that $H(S)$ and $H(R)$ no longer fit into RAM. Can we use one of the algorithms from the Shapiro paper (grace, hybrid, or simple) while still preserving the non-blocking nature of the symmetric hash join? If we can, which do you expect would perform best? If we cannot, then why not? Limit your answer to one or two sentences.

(Write your answer in the space below.)

No, we cannot use one of these algorithms because they are blocking. They all depend on the ability to scan all of one relation, extract a particular partition of it into a memory-sized chunk, and write the rest out to disk before processing the other relation. A few of you proposed clever alterations to this scheme that do work (and we generally gave you credit for such schemes.)

VI Sensor Network Aggregation

We read several papers about sensor network databases. Ben Bitdiddle is trying to design an efficient method for computing aggregates in a sensor network based on the scheme that is briefly sketched on page 50 of the "Query Processing in Sensor Networks" article.

Ben's sensor network has the topology show in Figure 1, with the dashed lines representing nodes that can hear each other, and the heavy lines representing the routing tree that is used to move data towards the root of the network (node 1). We say that a node is at *depth* d if it is d hops from the root of the network; we denote the depth of the tree as $tree\_depth$.

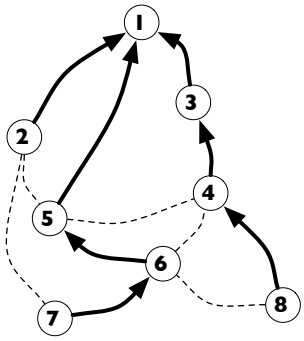


Figure 1: Ben's sensor network topology

For the next three questions, assume the network loses no messages during transmission.

Name:

10. [2 points]:

If each message sent by the sensors in the network can contain exactly one sensor value, how many messages would be transmitted if we want to collect one reading per sensor at the root of the network?

(Write your answer in the box.)

13

(Provide a one sentence explanation.)

One message must be sent over each link.

11. [2 points]:

If each message sent by the sensors in the network can contain up to 3 sensor values, what is the minimum number of messages that would be transmitted if we wanted to collect one reading per sensor at the root of the network?

(Write your answer in the box.)

7

(Provide a one sentence explanation.)

For example, 4 adds its reading to the message it received from 8. Similarly, 3 adds its reading to the message it received from 4. Hence, after just 3 total transmissions, the root node receives a reading from nodes 3, 4, and 8. The same logic applies to the other branches of the tree.

Name:

Ben proposes the following scheme for computing an average aggregate over the temperature in the sensor network during the current *epoch*. He divides the epoch into $tree\_depth$ equal-duration time slots, numbered $1 \ldots tree\_depth$, where time slot 1 is at the beginning of the epoch and time slot $tree\_depth$ is at the end. Each time slot is long enough for several messages to be transmitted, and nodes are assumed to be time synchronized such that they agree when each time slot begins and ends.

Algorithm 1 Ben's aggregation algorithm

1: At the beginning of the epoch, each node acquires a temperature reading with value t and creates a *partial state record* (PSR), $newP$, consisting of a $< sum, count >$ pair, initialized with $sum = v$ and $count = 1$.

2: If the node is at depth d, it transmits $newP$ to its parent during a timeslot = $tree\_depth - d$. In this way, a node n's children (who are at depth $n.d + 1$, by definition) will transmit their PSRs to n before n transmits to its parent.

3: Each node merges $newP$ with the k PSRs, $P_1 \ldots P_k$, it receives from children before transmitting its own PSR, as follows:

$$newP.sum = newP.sum + \sum_{i=1}^{k} P_i.sum$$

$$newP.count = newP.count + \sum_{i=1}^{k} P_i.count$$

4: The root can compute the average over the whole network at the end of the epoch as $newP.sum/newP.count$.

12. [3 points]:

If one $< sum, count >$ pair can fit into a network message, how many messages would be transmitted if we want to compute the average temperature over all sensors at the root of the network using Ben's protocol?

(Write your answer in the box.)

7

(Provide a one sentence explanation.)

For the same reason as above; each parent node aggregates all of the readings from its children before sending up, so this is as though an arbitrary number of messages can be fit in each transmission.

Ben observes that many of the PSR records are lost during transmission, because of the high network loss rates in his sensor network. He notes that 20% of all of the individual (link-level) transmissions are lost and that losses are uniform (e.g., each link has a 20% loss probability.)

Name:

13. [5 points]:

Estimate the total fraction of the sensor readings (with no added retransmissions) that are aggregated at the root of the network when the protocol described in Algorithm 1 is used in the lossy network shown in Figure 1.

(Write your answer in the box.)

4.7/7

(Show your work in the space below.)

There are two ways to think about this, both yielding the same answer.

A. *Consider node 4. Every iteration the expected number of readings it receives from 8 is .8. Node 4 then adds its own reading, yielding 1.8 readings. This message then has a .8 probability of reaching node 3, i.e., node 3 receives an expected 1.8 * .8 = 1.44 readings. Doing this for the full network yields: 2*((((.8+1)*.8)+1)*.8) + .8 = 4.7 expected readings at the root. So a message reaches the root with probability 4.7 / 7 = .67.*

B. *Consider node 8, at depth 3. The expected fraction of its readings that reach the root is* $.8^3$*. Since there are two nodes at depth three, two nodes at depth two, and three nodes at depth one, we expect that* $2 * .8^3 + 2 * .8^2 + 3 * .8$ *= 4.7 messages will arrive.*

We also accepted 5.7 / 8 = .71 as an answer (if the root also produces a sensor reading which always arrives).

To help address the problem that loss introduces, Ben observes that the network topology contains many links that are not being used, and proposes that, rather than sending their readings to just one parent, each node sends its PSR to several parents, which continue to run the protocol described in Algorithm 1, except that they also send their results up to multiple parents.

14. [6 points]:

Dana points out that though this scheme may result in more messages being successfully received at the root of the network, it may not result in the average values computed at the root being any closer to the true average over all nodes. Provide one reason why this might be the case.

(Write your answer in the space below.)

Messages may be duplicated and counted twice at the root, which could cause answers to be arbitrarily skewed.

Name:

15. [8 points]:

Dana suggests that Ben try a scheme similar to the one proposed in the CONTROL paper: use the central limit theorem to estimate the mean of the network by treating the PSR records that do arrive as a sample of the sensor readings from the whole network. In particular, she claims that, if the PSR computed at the root has $sum = s$ and $count = n'$, then the true mean lies in a Gaussian around s/n' (with a variance that depends on the variance of the underlying distribution of data.) Is she correct? Why or why not?

(Write your answer in the space below.)

To apply the central limit theorem, a random sample is required. The data received at the root of the network is not randomly sampled, since nodes nearer to the root of the tree have a much higher probability of getting their data through than nodes nearer to the leaves.

End of Quiz II Solutions

Name: