CS 347
Midterm Exam Solutions – Spring 1999

Problem 1 (10 points)

(1a) What architecture (shared-X) is best if scalability is the main concern?

Shared nothing; computers do not compete for shared resources (memory, disks), so it is easy to add computers.

(1b) What architecture (shared-X) is best if high performance is the main concern?

Shared memory; frequently used data structures can be shared in memory to improve performance.

(1c) Of the horizontal partitioning techniques, which is best for point queries?

Hash partitioning.

(1d) Of the horizontal partitioning techniques, with which is it easiest to evenly distribute data?

Round robin.

(1e) With 5 possible query predicates, what is the maximum number of min-term predicates we may obtain?

2^5 or 32.

(1f) To get disjointness with derived horizontal fragmentation, what property should the join attribute have?

Join attribute should be key of owner relation.

(1g) Name two types of inter-operation parallelism.

Pipelined; independent.

(1h) What is one important disadvantage of the hill climbing query optimization technique?

It may find a local optimum, not the gobal one.

(1i) With a query separation strategy, the query is split into "simple queries." What properties do these simple queries have?

Referenced relations have a single attribute, and output has a single attribute.

(1j) Name one distributed join scheme that is useful for non-equi joins.

Asymetric (or general) fragment and replicate join.

Problem 2 (10 points)

Consider the three relations:

- $R(A, B, C, D)$, vertically fragmented into $R1(A, B, C)$ and $R2(A, D)$;
- $S(A, E, F)$, horizontally fragmented into $S1$ and $S2$ on attribute E;
- $T(E, G)$, horizontally fragmented into $T1$ and $T2$, derived from the fragmentation of S (assume E is a key of S).

(2a) Consider the query:

```
SELECT  G
FROM    S, T
WHERE   S.E = T.E AND F > 15.
```

Reduce the tree for this query (localization and simplification) using the fragmentation information given above. In simplifying, perform selections and projections as early as possible. Show the resulting tree only.

```
                    Union
                      |
                      |
                      |
            ----------------------
            |                    |
            |                    |
            |                    |
         Proj(G)              Proj(G)
            |                    |
            |                    |
            |                    |
    ------ NJ ------      ------ NJ ------
    |              |      |              |
    |              |      |              |
    |              |      |              |
 Proj(E)          T1   Proj(E)          T2
    |                     |
    |                     |
    |                     |
 Sel(F>15)             Sel(F>15)
    |                     |
    |                     |
    |                     |
    S1                    S2
```

(2b) Consider the query:

```
SELECT  B, F
FROM    R, S
WHERE   R.A = S.A AND C > 15 AND E > 20.
```

Reduce the tree for this query (localization and simplification) using the fragmentation information given above. In simplifying, perform selections and projections as early as possible. Show the resulting tree only.

```
                  Union
                    |
                    |
                    |
          ----------------------
          |                    |
          |                    |
          |                    |
      Proj(B,F)            Proj(B,F)
          |                    |
          |                    |
          |                    |
  ------ NJ ------------------ NJ ------
  |                 |                  |
  |                 |                  |
  |                 |                  |
Proj(A,F)        Proj(A,B)          Proj(A,F)
  |                 |                  |
  |                 |                  |
  |                 |                  |
Sel(E>20)         Sel(C>15)           Sel(E>20)
  |                 |                  |
  |                 |                  |
  |                 |                  |
  S1                R1                 S2
```

Problem 4 (15 points)

You are interested in comparing three extensible hashing schemes for a particular application running on a shared-nothing architecture. In this application, 3 server machines receive *transactions* from client machines. Each transaction is very simple: it requests a particular record from a large file of "customer records" that is distributed across the three servers. You are considering the following three schemes:

- *Fixed Hashing (FH).* A client uses a fixed hash function to hash the customer id of the desired record into either 0, 1 or 2, identifying one of the three servers. The client then sends a message (the transaction request) to the server, which then returns the record. Each server implements extensible hashing locally, so the number of records that can be stored at the computer can grow gracefully.

- *Centralized Extensible Hashing (CEH).* All clients send transactions to computer 0, which holds an extensible hashing directory for the *entire* file. After looking up the desired key in the directory, computer 0 sends a message to either computer 1 or 2, which fetches the record from its disks. (Computer 0 does not hold any records; computers 1 and 2 have no local directory.) Computer 1 or 2 then directly replies to the client machine.

- *Distributed Extensible Hashing (DEH).* All servers hold a copy of the extensible hashing directory for the *entire* file. A client picks a server at random and sends its transaction. The server looks up the key in its copy of the directory, and then forwards the request to the appropriate server. (In some cases, actually a third of the cases since there are 3 servers, the desired record is on the local disk, so no message is sent.) The server that finds the record sends it directly to the client that requested it.

Each server can execute 6×10^6 instructions per second. Assume that

- A directory lookup takes 1000 instructions (same cost for either a local or global directory);

- Fetching a record from disk takes 1000 instructions (we are focusing on CPU costs only for this simple problem);

- The *combined* cost for receiving and sending a message is 1000 instructions (this is the *total* cost for both operations).

Ignore all other processing costs. (These numbers are not realisitc, but simplify the problem!)

(4a) What is the maximum number of transactions per second that the three servers collectively can process using the FH scheme? Briefly explain your answer.

Each trasnaction involves a mesage receive, a directory lookup, a record fetch, and a message send, or 3000 instructions. If T is the total number of transactions per second, $T/3$ will come to one server, which will execute $(T/3)3000$ i/s. The maximum value of T occurs when this quantity equals the processor speed, so $T = (6 \times 10^6 \times 3)/3000 = 6Ki/s$.

(4b) What is the maximum number of transactions per second that the three servers collectively can process using the CEH scheme? Briefly explain your answer.

If T is the total transactions per second, server 0 must process all of them, for each doing a receive, lookup, send. Thus, server 0 executes $T \times 2000$ i/s, and T cannot exceed $(6 \times 10^6)/2000$ t/s, or 3000 t/s.

Servers 1 and 2 each process $T/2$ t/s, each incurring a receive, a record fetch, and a send, or 2000 instructions. The number of instructions per second performed at each of these servers, $(T \times 2000)/2$, must be less than 6×10^6. Thus, T cannot exceed $(6 \times 10^6 \times 2)/2000$ or 6000 t/s.

The critical limit is hence at server 0 (it is the bottleneck), so the maximum trasnaction rate is 3000 t/s.

(4c) What is the maximum number of transactions per second that the three servers collectively can process using the DEH scheme? Briefly explain your answer.

A particular server gets $T/3$ of the total t/s directly. Of these, 1/3 are fully processed locally, involving a receive, a lookup, a record fetch, and a send. The remaining 2/3 incur only a receive, a lookup, and a send.

Of the $2T/3$ t/s that do not go to the particular server, 1/3 are sent later for a record lookup. These incur a receive, a record fetch, and a send.

Thus, the total work at a server is $T[(1/3)(1/3)3000 + (1/3)(2/3)2000 + (2/3)(1/3)2000]$. This load must again be less than the 6×10^6 available i/s. Solving for T we get 4900 t/s.

[A few of you said that the maximum rate occured if clients happened to be lucky and sent *all* transactions to the server that had their desired record. In this case, each server gets $T/3$ transactions, each involving a receive, a lookup, a record fetch, and a send (3000 instructions). Thus, T can be as large as 6000 t/s. This answer was also counted as correct.]

(4d) After running your system for a year, you wish to increase the processing capacity by adding 2 servers, for a total of 5 servers. Because the system must run continuously, you *cannot take time to reorganize* the records. (That is, if a record was originaly placed on one computer, then it cannot be moved to another computer at a later time.) How many transactions per second can you run with 5 servers under each scheme? Briefly explain.

To simplify the analysis, assume that records are evenly distributed across all the computers that may hold records. (This will not be true right after you add the computers, but will hopefully be true after a while.) Note that in some of the schemes the new computers will not be able to hold records, because the scheme is not flexible enough. You have to figure out if the new computers can hold records!

For FH, we cannot change the hash function (this would cause records to move), so we cannot use the extra machines. Thus, the maximum rate is still 6000 t/s.

For CEH, server 0 is still the bottlneck, so we are still limited by its 3000 t/s.

For DEH, we can use the extra machines. The work done at each server is now $T[(1/5)(1/5)3000 + (1/5)(4/5)2000 + (4/5)(1/5)2000]$. This leads to a maximum value of T of 7890 t/s.