1 Parallel Query Processing

1. What is the difference between inter- and intra- query parallelism?

2. What are the advantages and disadvantages of organizing data by keys?

3. Assume for parts (a) and (b) that we have m=3 machines with B=5 buffer pages each, along with N=63 pages of data that don't contain duplicates.

   (a) In the best case, what is the number of passes needed to sort the data?

   (b) What is the number of passes needed to hash the data (once)? Find the best case, assuming that somehow the data will be uniformly distributed under the given hash function.

   (c) If you don't have a hash function that can uniformly partition the data, would round-robin partitioning be useful here? Why or why not?

(d) Assume that relation R has R pages of data, and relation S has S pages of data. If we have m machines with B buffer pages each, what is the number of passes in order to perform sort merge join (in terms of R, S, m, and B)?
Consider reading over either relation to be a pass.

(e) Can you use pipeline parallelism to implement this join?

4. All of the data for a relation with N pages starts on one machine, and we would like to partition the data onto M machines. How much data (in KB) would be sent over the network to partition the data through each of the following: range, hash, and round-robin partitioning?

   Assume that the size of each page is S (in KB). Also, assume we use uniform hash functions and are able to construct ranges that have the same number of values in them.

5. Relation R has 10,000 pages, round-robin partitioned across 4 machines (M1, M2, M3, M4). Relation S has 10 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.

   (a) What type of join would be best in this scenario, and why?

   (b) How many KB of data must be sent over the network to join R and S?

   (c) Would the amount of data sent over the network change if R was hash partitioned among the 4 machines rather than round-robin partitioned? What about range partitioned?