CS W186 Introduction to Database Systems
Fall 2019 Josh Hug

DIS 8

1 Parallel Query Processing

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

   Inter-query parallelism operates between multiple queries, rather than within a single query, whereas intra-query parallelism operates within a single query (parallelism of the operators that make up the query).

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

   Advantages: because data is organized by keys, search and update operations (which require searching on the key) can be done more efficiently, since we have some sense of where the data must be (if it exists).
   Disadvantages: we must maintain the organization, which adds overhead to insertions and updates.

3. Given m machines with B page buffer each, along with N pages of data that doesn't have duplicates.

   (a) What is the number of passes needed to sort the data? Find the best case, in terms of N, B, and m.
   (number of passes to partition the data) + (number of passes to sort each partition)
   1 + ceil(1 + logB-1(N/(mB)))

   (b) What is the number of passes needed sto hash the data (once)? Find the best case, assuming that somehow the data will be uniformly distributed under the given hash function.
   (number of passes to partition the data) + (number of passes to hash each partition)
   1 + ceil(1 + logB-1(N/(mB)))

   (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?
   In general, you can't guarantee all the records for one key appears on one machine. However, since we guarantee that there are no duplicate keys, this would not be an issue. Therefore, round-robin partitioning would be useful in this specific case.

   (d) Instead of N pages, you are given R and S pages of data (one for each relation). What is the number of passes in order to perform sort merge join? Consider reading over either relation to be a pass.
   (1 pass to partition each of the two tables across machines) + (the number of passes needed to sort R) + (the number of passes to sort S) + (1 final merge sort pass, going through both tables)
   2 + (1 + logB-1(R / (mB))) + (1 + logB-1(S / (mB))) + 2

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

No, the sorting pass must complete before the merge pass can begin.