DATA 514 Section 7 Worksheet - Jonathan Jacobs

MapReduce

Question 1

In a MapReduce framework, the Map function handles the initial processing of data. For the given SQL query operating on two relations R(a, b) and S(b, c), here's how the Map function would process the data:

1. **Input:** The Map function reads a block of either the R or S relation.

2. **Processing Logic:**

   - If the block is from relation R:
     - The function checks if R.a is less than or equal to 100. If true, it emits a key-value pair where the key is R.b and the value is a tuple containing the tag "R" and the value R.a.
     - Output format: emit(R.b, ("R", R.a))
   - If the block is from relation S:
     - It directly emits a key-value pair where the key is S.b and the value is a tuple containing the tag "S" and the value S.c.
     - Output format: emit(S.b, ("S", S.c))

3. **Output:** The outputs are key-value pairs. The key is the b value, which is common between R and S. The value is a tuple containing a label identifying whether the data came from relation R or S and the respective values from these relations.

Example Outputs:

- For an entry in R such as (a=50, b=1) and R.a <= 100, the output would be: emit(1, ("R", 50))
- For an entry in S such as (b=1, c=5), the output would be: emit(1, ("S", 5))

These outputs set the stage for the Reduce function, which will perform the grouping by R.b and calculate the maximum of S.c for each group.

Question 2

In the Reduce function for the given SQL query, the inputs are key-value pairs, where each key is a unique `R.b`, and the associated value is a list of tuples. Each tuple contains an identifier ("R" or "S") and relevant data from the relations `R` and `S`.

Computation Process:

1. For each key `R.b`:
   - **Separate**: Divide the tuples into two groups, one for "R" (relation `R`) and one for "S" (relation `S`).
   - **Filter**: For the tuples from "R," verify that they satisfy the condition `R.a <= 100` (which was already checked in the Map function).
   - **Aggregate**: For tuples from "S," collect all `S.c` values.
   - **Calculate Maximum**: Find the maximum `S.c` value among the grouped tuples.

Output:

- For each unique `R.b`, the Reduce function outputs a tuple `(R.b, cmax)`, where `cmax` is the maximum `S.c` value from relation `S` that corresponds to `R.b`.

Example: If the Reduce function receives key `1` with the list `[("R", 50), ("S", 5), ("S", 8)]`:

- It will filter the "R" tuples (only keep those passing the `R.a <= 100` check).
- For "S" tuples, it will compute the maximum `S.c` value, which is `8`.
- Thus, the Reduce output will be `(1, 8)`.

This logic is repeated for each unique `R.b` key to calculate the desired maximum `S.c` value for every group.

Parallel Processing

Question 3

The SQL query requires joining the relations D and E on the attribute A while applying a filter condition on E.C > 10. Since the data is partitioned across multiple machines using a random block distribution, a shuffle operation is necessary to bring together data from both tables based on the common key A to perform the join.

Decision on Filtering:

- **Before Shuffle**: Applying the filter E.C > 10 before shuffling reduces the data that needs to be transferred between nodes. This improves performance by minimizing network traffic and only shuffling the necessary records.
- **After Shuffle**: Filtering after the shuffle would involve moving all records across nodes, including those that don't satisfy E.C > 10, leading to inefficient data movement.

Conclusion: The filter E.C > 10 should be applied before the shuffle, as it will reduce data transfer and optimize the join process.

Question 4

In this query, the relations D and E are joined on the attribute A, with a condition applied that depends on values from both tables: E.C - D.B > 20.

Key Considerations:

- **Join Condition**: The join operation matches D.A with E.A, requiring a shuffle to ensure that matching values of A from both tables are located on the same machine.
- **Filter Condition**: The condition E.C - D.B > 20 relies on values from both tables, necessitating that data from D and E is available together before the filter is applied.

Conclusion: A shuffle is necessary before determining the condition E.C - D.B > 20 because the filter needs access to data from both tables after they are joined on A.

Question 5

To determine whether a shuffle is needed before the GROUP BY D.A and HAVING MAX(E.C) < 100 operations, the following considerations apply:

Join Condition: The query involves joining D and E on the attribute D.A = E.A. This requires a shuffle to bring together matching values of A from both relations onto the same machine.

Group By Operation: The GROUP BY D.A operation groups rows by the attribute D.A. Shuffling is required to ensure that all records with the same D.A value are collected on a single reducer for aggregation.

Having Clause: The HAVING MAX(E.C) < 100 condition checks the maximum value of E.C for each group defined by D.A. This can only be applied after aggregation, which needs to be performed after all relevant records have been grouped.

Conclusion: A shuffle is needed before the GROUP BY D.A operation to ensure that all relevant entries are grouped correctly. Similarly, shuffling is required before the HAVING MAX(E.C) < 100 operation because the aggregation must happen across all grouped records before filtering.