Homework 5: Spark & Query Execution

Name: Felix Wang, USC ID: 1900370012

Part 1: Spark DataFrame Queries

Please refer to the Jupyter notebook `hw5.ipynb` included in the submission.

Part 2: Query Execution

> Consider natural-joining tables R(a, b) and S(a,c). Suppose we have the following scenario.
>
> - R has 5, 000 blocks (i.e., B(R) = 5,000).
> - S has 10,000 blocks (i.e., B(S) = 10,000).
> - 102 pages available in main memory for the join.
>
> Assume the output of join is given to the next operator in the query execution plan (instead of writing to the disk) and thus the cost of writing the join output is ignored.
>
> Describe the steps for each of the following join algorithms and estimate the total number of block I/O's needed for the algorithm.
>
> For sorting and hashing-based algorithms, also indicate the size of output from each step (sorting, partitioning, merging, et).

[10 points] (Block-based) nested-loop join with R as the outer relation.

Algorithm Description:

The block-based nested-loop join algorithm proceeds as follows:

1. **Memory allocation**: Allocate $M - 2 = 100$ pages for blocks of R (outer relation), 1 page for reading S (inner relation), and 1 page for the output buffer.
2. **Outer loop**: Divide R into chunks of 100 blocks. For each chunk:
   - Read $M - 2 = 100$ blocks of R into memory
   - **Inner loop**: For each block in this chunk of R, scan the entire relation S:
     - Read each block of S one at a time
     - Join matching tuples from R and S
     - Write results to output buffer (not counted per problem statement)
3. **Iteration**: Repeat for all $\lceil B(R)/(M-2) \rceil = \lceil 5000/100 \rceil = 50$ chunks of R.

I/O Cost Calculation:

- Read R once: $B(R) = 5,000$ I/Os
- Number of R chunks: $\lceil B(R)/(M-2) \rceil = \lceil 5000/100 \rceil = 50$
- Read S once for each chunk: $50 \times B(S) = 50 \times 10,000 = 500,000$ I/Os
- **Total I/O cost** = $B(R) + \lceil B(R)/(M-2) \rceil \times B(S) = 5,000 + 500,000 = \boxed{505,000}$ block I/Os

[10 points] (Block-based) nested-loop join with S as the outer relation.

Algorithm Description:

The block-based nested-loop join algorithm with S as outer relation proceeds as follows:

1. **Memory allocation**: Allocate $M - 2 = 100$ pages for blocks of S (outer relation), 1 page for reading R (inner relation), and 1 page for the output buffer.
2. **Outer loop**: Divide S into chunks of 100 blocks. For each chunk:
   - Read $M - 2 = 100$ blocks of S into memory
   - **Inner loop**: For each block in this chunk of S, scan the entire relation R:
     - Read each block of R one at a time
     - Join matching tuples from S and R
     - Write results to output buffer (not counted per problem statement)
3. **Iteration**: Repeat for all $\lceil B(S)/(M-2) \rceil = \lceil 10000/100 \rceil = 100$ chunks of S.

I/O Cost Calculation:

- Read S once: $B(S) = 10,000$ I/Os
- Number of S chunks: $\lceil B(S)/(M-2) \rceil = \lceil 10000/100 \rceil = 100$
- Read R once for each chunk: $100 \times B(R) = 100 \times 5,000 = 500,000$ I/Os
- **Total I/O cost** = $B(S) + \lceil B(S)/(M-2) \rceil \times B(R) = 10,000 + 500,000 = \boxed{510,000}$ block I/Os

[15 points] Sort-merge join (assume only 100 pages are used for sorting and 101 pages for merging). Note that if the join cannot be completed using only a single merging pass, runs from one or both relations will need to be further merged, in order to reduce the number of runs. In this case, select the relation with a larger number of runs for further merging first if both have too many runs.

Algorithm Description:

The sort-merge join consists of two main phases: sorting both relations, then merging them for the join.

Phase 1: Sorting Relations

Step 1a - Create sorted runs for R (using 100 pages):

- Read R in chunks of 100 blocks, sort each chunk in memory, write to disk
- Number of initial runs: $\lceil B(R)/100 \rceil = \lceil 5000/100 \rceil = 50$ runs
- Each run contains 100 sorted blocks (last run may be smaller)
- I/O cost: $2 \times B(R) = 2 \times 5,000 = 10,000$ (read + write)
- **Output size: 50 sorted runs of R, totaling 5,000 blocks**

Step 1b - Merge runs of R (using 100 pages):

- Can merge up to $M - 1 = 99$ runs simultaneously (99 input buffers + 1 output buffer)
- Since $50 < 99$, all runs can be merged in a single pass
- I/O cost: $2 \times B(R) = 2 \times 5,000 = 10,000$ (read + write)
- **Output size: 1 fully sorted run of R (5,000 blocks)**

Step 2a - Create sorted runs for S (using 100 pages):

- Read S in chunks of 100 blocks, sort each chunk in memory, write to disk
- Number of initial runs: $\lceil B(S)/100 \rceil = \lceil 10000/100 \rceil = 100$ runs
- I/O cost: $2 \times B(S) = 2 \times 10,000 = 20,000$ (read + write)
- **Output size: 100 sorted runs of S, totaling 10,000 blocks**

Step 2b - Merge runs of S (multiple passes needed):

- First merge pass (using 100 pages):
  - Can merge up to 99 runs at once
  - Since $100 > 99$, process in groups: first 99 runs → 1 merged run, last 1 run stays separate

- I/O cost: $2 \times B(S) = 2 \times 10,000 = 20,000$ (read + write)
- **Output size: 2 sorted runs of S (totaling 10,000 blocks)**

- Second merge pass (using 100 pages):
  - Merge the remaining 2 runs into 1 final sorted run
  - I/O cost: $2 \times B(S) = 2 \times 10,000 = 20,000$ (read + write)
  - **Output size: 1 fully sorted run of S (10,000 blocks)**

Phase 2: Final Merge Join

Step 3 - Merge join sorted R and S (using 101 pages):

- Both R and S are now fully sorted (1 run each)
- Use 1 buffer for R, 1 buffer for S, 1 buffer for output
- Perform merge join by scanning both relations once in sorted order
- I/O cost: $B(R) + B(S) = 5,000 + 10,000 = 15,000$ (read only, output not written to disk)

Total I/O Cost Calculation:

- Sorting R: $10,000 + 10,000 = 20,000$
- Sorting S: $20,000 + 20,000 + 20,000 = 60,000$
- Final merge join: $15,000$
- **Total I/O cost** = $20,000 + 60,000 + 15,000 = \boxed{95,000}$ block I/Os

[15 points] Partitioned-hash join (assuming 101 pages are used in partitioning the relations).

Algorithm Description:

The partitioned-hash join consists of two main phases: partitioning both relations using a hash function, then probing each partition pair.

Phase 1: Partitioning Relations

Step 1a - Partition R (using 101 pages):

- Use hash function h_1 to partition R into $n = 100$ partitions
- Memory allocation: 1 input buffer + 100 output buffers (one per partition)
- Read each block of R, apply h_1 to the join attribute, write to appropriate partition
- Average partition size: $B(R)/n = 5,000/100 = 50$ blocks per partition
- I/O cost: $2 \times B(R) = 2 \times 5,000 = 10,000$ (read + write)
- **Output size: 100 partitions $R_0, R_1, \ldots, R_{99}$, each approximately 50 blocks**

Step 1b - Partition S (using 101 pages):

- Use the same hash function h_1 to partition S into $n = 100$ partitions
- Memory allocation: 1 input buffer + 100 output buffers
- Read each block of S, apply h_1 to the join attribute, write to appropriate partition
- Average partition size: $B(S)/n = 10,000/100 = 100$ blocks per partition
- I/O cost: $2 \times B(S) = 2 \times 10,000 = 20,000$ (read + write)
- **Output size: 100 partitions $S_0, S_1, \ldots, S_{99}$, each approximately 100 blocks**

Phase 2: Probing (Join) Each Partition Pair

Step 2 - Join partition pairs (using 102 pages):

For each partition $i = 0, 1, \ldots, 99$:

1. **Build phase**:

- Read partition R_i into memory (approximately 50 blocks)
- Build an in-memory hash table using hash function h_2 (different from h_1)
- Memory required: 50 blocks for R_i + 1 block for S_i + 1 output buffer = 52 pages
- Verification: $52 < 102$ ✓ (fits comfortably in memory)

2. **Probe phase**:
   - Read partition S_i block by block
   - For each tuple in S_i, probe the hash table built from R_i
   - Output matching tuples to next operator (not written to disk)
3. **I/O for partition** i: Read R_i + Read S_i = $50 + 100 = 150$ blocks

Total Probing I/O:

- Sum over all partitions: $\sum_{i=0}^{99}(|R_i| + |S_i|) = B(R) + B(S) = 5,000 + 10,000 = 15,000$ I/Os

Total I/O Cost Calculation:

- Partitioning R: $10,000$
- Partitioning S: $20,000$
- Probing all partitions: $15,000$
- **Total I/O cost** = $10,000 + 20,000 + 15,000 = \boxed{45,000}$ block I/Os

Total I/O Cost:

- Partitioning phase: 10,000 + 20,000 = 30,000
- Probing phase: 15,000
- **Total Cost = 30,000 + 15,000 = 45,000 block I/Os**