Database System Course
Assignment 3: Hash Join Optimization

May 13, 2025

1 Introduction

In this assignment, you will optimize hash join operations for a given join order. The main goal is to improve the execution speed of join operations in an in-memory database system. You will work with the provided benchmark and implement optimization techniques to enhance performance.

2 Assignment Overview

In this assignment, you are required to:

1. Implement optimization techniques for hash joins with a fixed join order.
2. Evaluate your optimization's impact on performance across the provided test cases.
3. Provide a detailed analysis of your implementation, explaining why certain optimizations worked or did not work.
4. The test dataset can be downloaded here. The testing framework and data loading interfaces are provided in the `run_job.cpp` file.

3 Task: Hash Join Optimization

Your primary task is to optimize hash join execution for a given join order to achieve better performance. We strongly recommend implementing the Robust Predicate Transfer (RPT) algorithm as described in the recent paper.

However, since we are working with an in-memory database, the additional overhead of creating Bloom filters (as used in RPT) might not always lead to substantial performance improvements. A 20% improvement would be considered a good achievement. Therefore, you are encouraged to explore additional or alternative optimization techniques.

3.1 Potential Optimization Approaches

You may consider the following optimization techniques:

1. Join implementation optimizations (Reference 2)
2. Implementation of RPT algorithm with appropriate Bloom filter configurations (Reference 3)
3. Hash partitioning strategies (Reference 4)

If you determine that approaches other than RPT provide better results for this specific in-memory scenario, you may focus on those optimizations instead. What's most important is achieving better performance and being able to explain your approach.

4 Implementation Requirements

Your implementation should:

1. Work correctly with the provided benchmark framework
2. Improve join performance compared to the baseline implementation
3. Be well-documented with comments explaining your optimization strategies

5 Report Requirements

You must submit a detailed report that:

1. Follows academic paper standards and structure
2. Explains your optimization approach in detail
3. Provides a performance breakdown analysis for each implemented technique
4. Discusses why certain optimizations were effective or ineffective
5. Includes performance measurements and comparisons with the baseline
6. Analyzes any trade-offs in your approach

The report should be comprehensive enough that another student could implement your approach based solely on your explanation.

6 Evaluation

Your assignment will be evaluated based on:

1. Performance improvement (10 points, 1 point per test case)
2. Quality and thoroughness of your report (10 points)

The total score for this assignment is 20 points.

7 Submission Guidelines

Submit your work as a zip file containing:

1. Your optimized implementation code
2. A detailed report in PDF format explaining your optimizations
3. Any additional scripts or tools you created for testing/evaluation

8 Tips

1. Consider the trade-offs between different optimization strategies
2. Remember that the goal is performance improvement with a clear understanding of why your optimizations work

9 References

1. Junyi Zhao, Kai Su, Yifei Yang, Xiangyao Yu, Paraschos Koutris, and Huanchen Zhang. ”Debunking the Myth of Join Ordering: Toward Robust SQL Analytics.” (2024).

2. Yiming Qiao and Huanchen Zhang. ”Data Chunk Compaction in Vectorized Execution.” (2024).

3. Harald Lang, Thomas Neumann, Alfons Kemper, Peter Boncz. ”Performance-Optimal Filtering: Bloom Overtakes Cuckoo at High Throughput.” (2019).

4. Maximilian Bandle, Jana Giceva, Thomas Neumann. ”To Partition, or Not to Partition, That is the Join Question in a Real System.” (2021).

Good luck with your implementation!