Evaluating High-Energy Physics Data Queries Using DuckDB

Nahid Jahanian
nahid.jahanian@gmail.com

December 8, 2024

Introduction

DuckDB is a lightweight, in-process analytical database management system (DBMS) designed for online analytical processing (OLAP) workloads. Inspired by the VLDB paper, *"Evaluating Query Languages and Systems for High-Energy Physics Data"*, this report explores the execution of high-energy physics queries using DuckDB. Its ease of setup, minimal hardware requirements, and performance for OLAP workloads position it as a strong candidate for local data analysis. The source codes and scripts used for this evaluation are hosted in my GitHub repository:https://github.com/Nahidnj/DuckDB.git.

Hardware and Software Environment

The benchmarks were performed on the following hardware and software configuration:

- **Machine:** MacBook Air (2022, Model Identifier: Mac14,2)
- **Processor:** Apple M2 Chip with 8 Cores (4 Performance + 4 Efficiency)
- **Memory:** 8 GB Unified RAM
- **Operating System:** macOS Ventura 13.0.1

This setup represents a lightweight, portable environment suitable for running embedded databases like DuckDB. Unlike the high-performance clusters used in the VLDB paper for distributed systems like Presto or BigQuery, this setup emphasizes ease of use and local execution capabilities.

Data Preparation

The original VLDB paper used datasets with sizes of approximately **17 GB** and **2 TB** to evaluate query performance across different systems. Since running such large datasets locally is impractical, I downloaded smaller versions of the datasets. The datasets I used are:

- `Run2012B_SingleMu_1000.parquet` (1,000 events),

- `Run2012B_SingleMu_4000.parquet` (4,000 events).

These smaller datasets allowed me to run benchmarks efficiently while maintaining compatibility with the high-energy physics queries described in the paper.

Results

The execution times (in seconds) and result counts for each query on the two datasets are summarized in Table 1. A visualization of execution times is shown in Figure 1.

Table 1: Query Performance on DuckDB

Query ID	Dataset	Execution Time (s)	Result Count
Q1	1000 Events	0.0048	1,000
Q2	1000 Events	0.0018	1,699
Q3	1000 Events	0.0031	680
Q4	1000 Events	0.0026	1,000
Q5	1000 Events	0.0351	100,000
Q6a	1000 Events	0.0118	1,000
Q6b	1000 Events	0.0028	1
Q7	1000 Events	0.0605	1,000
Q8	1000 Events	0.1742	200,000
Q1	4000 Events	0.0027	4,000
Q2	4000 Events	0.0057	6,387
Q3	4000 Events	0.0096	2,523
Q4	4000 Events	0.0076	4,000
Q5	4000 Events	0.0312	100,000
Q6a	4000 Events	0.0795	4,000
Q6b	4000 Events	0.0065	1
Q7	4000 Events	0.6577	4,000
Q8	4000 Events	0.1287	200,000

Discussion of Results

Execution Times

Execution times for queries on the smaller dataset (`Run2012B_SingleMu_1000.parquet`) were significantly faster, as expected. Computationally intensive queries like Q7 and Q8 showed notable increases on the larger dataset.

Result Counts

Result counts scaled linearly with dataset size for most queries. Queries involving aggregations (Q5, Q8) consistently returned the same number of results.

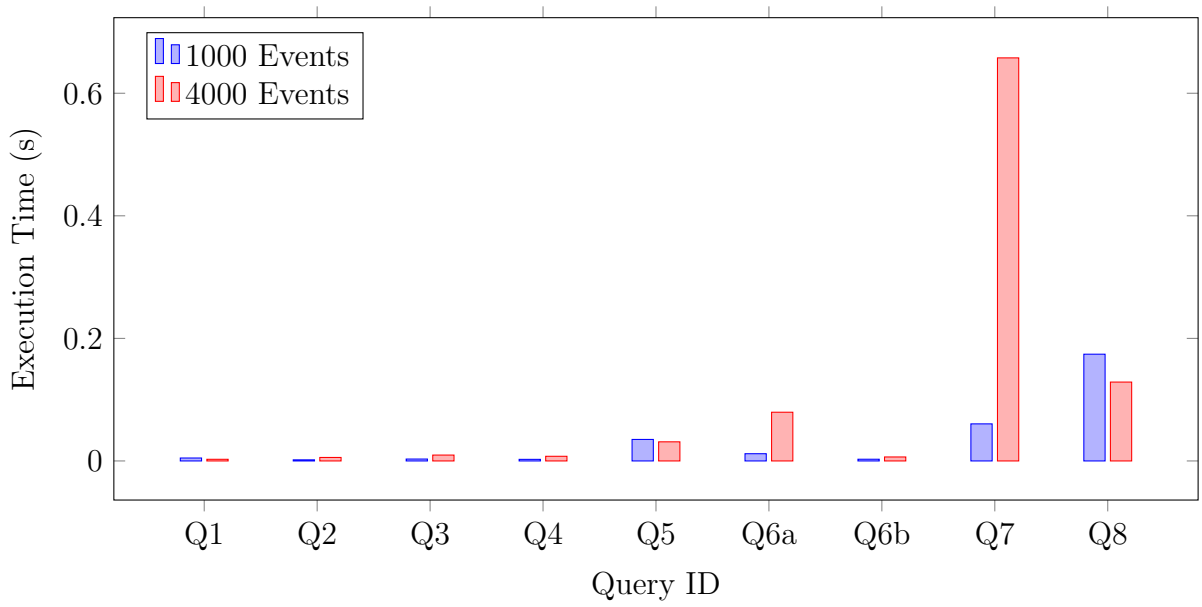

Figure 1: Execution Times for Queries on DuckDB

Conclusion

DuckDB demonstrated excellent performance for executing high-energy physics queries on small datasets, showcasing its capabilities for local analytical workloads. Its lightweight nature and strong OLAP optimizations make it a versatile tool for data analysis. However, its scalability may be limited for very large datasets compared to distributed database systems.