Project Proposal: Parallel Query Processing in PostgreSQL

Kai Franz

https://kai-franz.github.io/15418-project/

Summary

I am going to implement a simple vectorized database execution engine. This will allow the database management system (DBMS) to leverage instruction-level parallelism and SIMD instructions to perform query scans.

Background

DBMSes are used to efficiently process large amounts of data. Database queries, which are executed by DBMSes, have a high potential for parallelization because they require applying a computation to billions or trillions of tuples. In this proposal, I will focus on one essential primitive that is used in every database query: the filter operator. This operator reads through a table or other data source, checking if each tuple satisfies a given condition. Because performing this check on each tuple is an independent operation, this operation can be executed in parallel. PostgreSQL is an open-source DBMS that exploits this parallelism by using several worker processes to execute a scan; however, PostgreSQL does not exploit instruction-level parallelism or SIMD instructions. My goal is to implement these two forms of parallelism for the scan operator in PostgreSQL.

Challenges

The workload that I will focus on is online analytics processing (OLAP). These queries are read-only and involve scanning large amounts of data.

The workload for the scan operator is well-suited for parallelism due to the independent nature of each predicate evaluation. The main challenge is the high memory-to-computation ratio: the scan operator reads through an entire table, performing a small amount of work for each tuple before moving on to the next one. There is no locality in this operator. Therefore memory bandwidth may become a bottleneck.

Resources

- I am building off the PostgreSQL 15.0 code base (https://github.com/postgres/postgres/tree/REL_15_STABLE).

- I will be using the MonetDB/X100 paper as reference (https://www.cidrdb.org/cidr2005/papers/P19.pdf).

Goals

- **75%:** Implement a vectorized executor that exploits instruction-level parallelism.
- **100%:** In addition to the previous goal, add support for predicate evaluation.
- **125%:** Use SIMD intrinsics to accelerate predicate evaluation.

The demo that I plan to show will be a speedup graph.

Platform Choice

I have chosen PostgreSQL (which is written in C) as the platform of choice for this system because it is open-source and I am familiar with the codebase from research. In addition, PostgreSQL has potential to improve its performance on OLAP workloads.

Schedule

- **11/07:** Do background reading on the linked paper. Construct microbenchmarks to test the performance of PostgreSQL.
- **11/14:** Begin implementing a simple vectorized scan operator in PostgreSQL.
- **11/21:** Finish vectorized scan implementation.
- **11/28:** Add support for predicate evaluation
- **Milestone:** Have a working system that shows a speedup over the existing system on microbenchmarks
- **12/5:** Implement SIMD operators for integer comparison
- **12/12:** Run benchmark workloads (TPC-H/TPC-DS). Optimize any bottlenecks.
- **Final Report:** Have a working vectorized execution engine that supports predicate evaluation.