Everything you always wanted to know about compiled and vectorized queries but were afraid to ask

Volcano style iteration was a popular approach when disk space was a restriction.However, nowadays Queries engines have two main approaches: vectorization and data-centric code generation. Both approaches focuses on making a fast system but an apple to apple comparison is not possible due to implementation specific choices by the vendor. In this paper, the two models are implemented on the same test system.
Vectorization is similar to the volcano style and is based on the pull model but the next methods pulls blocks of tuple instead of single one and can be used by SIMD. Vectorization works better for cache miss(hash joins) and a popular example is HyPer. Data-centric is based on the producer and consumer interface where it generates code for the given query. It aids cache resident workloads due to less CPU cycles and a popular example is VectorWise. Both systems are widely different due to their different design choices and in order to compare them, they implemented a compilation based relational engine(Typer) and and vectorization based engine(Tectorwise) in a single test system based on the same physical query plan.
The paper focuses on OLAP performance and uses the TPC-H benchmark. Mostly there is only moderate difference between the two approach when parallelized properly using the morsel driven parallelism. The summary describes all the performance difference observed in a succinct manner. Section 8 explores the benefits of the models on different workloads like OLTP, compilation time, profiling and so on.
Hence it can be concluded that both the models are super effective for OLAP workloads if properly parallelized and its used case can vary if the workload is changed and on other parameters described in section 8.

Comments:

I was quite surprised to see SIMD having same performance for compilation and vector. I had a bias thinking vector should be a clear winner but memory access cost is the main bottle neck here. (Amdahl's Law?)
Random intrusive thought: what if we train a machine learning model like Redshift or use a statistical model on particular data load, have both the execution engine in the system, test its performance and eventually choice the best model for a given query(like a decision tree). Or maybe have a hybrid approach by combining both? So that it can also be used for other different workloads as well.
A major bottleneck for typer is also the compiled time which was ignored in the paper. Shouldn’t that also be considered or that difference is very small for OLAP workloads?