Evaluating the Vertical and Horizontal Read and Write Scalability of MobilityDB and CrateDB

Bachelor's Thesis Exposé

Eryk Karol Kściuczyk

Spatio-temporal databases are increasing in popularity due to increased production of such data by numerous IoT devices. As the need for such databases increases, developers try to reach for familiar and open-source solutions. MobilityDB, built on top of the spatial extension PostGIS, extends PostgreSQL's capabilities regarding spatio-temporal aspects. Previous research has shown that MobilityDB can also scale horizontally using another extension, Citus. As a comparison, CrateDB is a natively distributable SQL database designed for scalability and high performance.

While the runtimes of specific queries of BerlinMOD benchmark have been evaluated on 4 and 28 node clusters of MobilityDB, its horizontal and vertical scalability still remains underexplored. Previous research mostly focuses on designing queries for single node systems, and often does not compare systems between one another. Another option, CrateDB, also has not been fully evaluated for its usability in spatiotemporal use cases. Moreover, benchmarking with a wider range of cluster sizes is necessary to establish the scalability pattern. Additionally, a comparison between MobilityDB+Citus and CrateDB in terms of horizontal scalability has not been conducted, which could provide valuable insights into their respective strengths and weaknesses.

In this thesis we implement a spatiotemporal benchmark that evaluates the scalability of relational databases both horizontally and vertically. We build a data generation and benchmarking tool to stress the respective systems (MobilityDB and CrateDB). The benchmark setup can be seen on figure 1.

We will compare the performance of the distributed MobilityDB and CrateDB on different hardware configurations using Google Cloud Platform. Additionally, we will evaluate several scenarios, including vertically scaling a single node and scaling out by distributing both solutions across several instances. We use our benchmark results to evaluate the systems for their scalability and also compare their performance against each other.

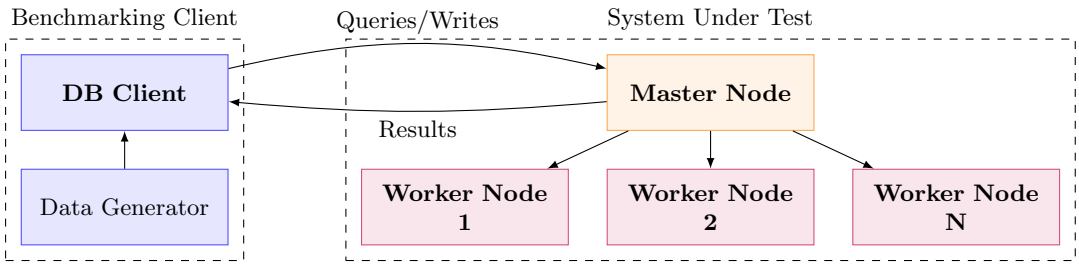

Figure 1: Benchmarking client will write generated spatiotemporal data and execute the workload against System Under Test (MobilityDB/CrateDB cluster). The cluster consists of a single master node and multiple worker nodes

We measure metrics such as query latency and write throughput using confidence intervals, while also visualizing the distribution using Empirical Cumulative Distribution Function (ECDF) plots, to reveal performance patterns and tail latencies. We repeat each experiment multiple times to ensure the validity of our results.