Evaluating the Vertical and Horizontal Read and Write Scalability of MobilityDB

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 the MobilityDB extension can be partially used in conjunction with Citus, a PosgreSQL extension, which provides horizontal scalability options.[1] While specific queries runtimes of BerlinMOD benchmark have been evaluated on 4 and 28 node clusters[1], horizontal and vertical scalability of the platform has not yet been fully explored, especially write performance. BerlinMOD queries are not designated for distributed MOD and different set of queries is required to further assess its performance. Moreover, testing with a wider range of cluster sizes is necessary to establish the scalability pattern.

In this thesis we aim to benchmark the vertical and horizontal scalability using generated spatio-temporal data to measure metrics like request latency, query runtime, and read/write throughput under different data sizes. Benchmarking data will be generated using a custom data generation tool specifically developed for this study. The tool will simulate the movement of birds, chosen due to their seasonal migration patterns, localized movements within standard habitats, varying flight heights and times. The testing setup can be seen on figure 1. We will compare the performance of MobilityDB combined with Citus on different hardware configurations using Google Compute Engine service on Google Cloud Platform. Additionally, we will evaluate several scenarios, including vertically scaling a single node and scaling out by distributing MobilityDB across several Afterwards, we want to compare scalability trade-offs between those two approaches.

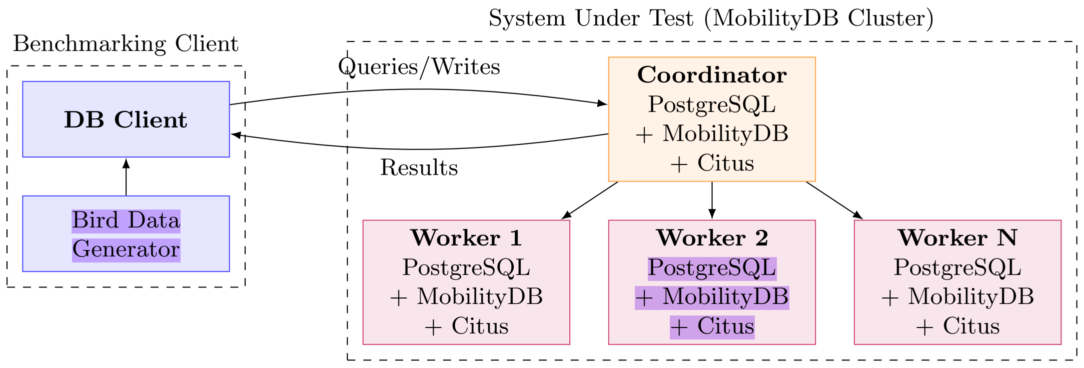

Figure 1: Benchmarking client will write generated bird movement data and execute the workload against System Under Test (MobilityDB cluster). The cluster consists of a coordinator and multiple workers, each running PostgreSQL with MobilityDB and Citus extensions.

Query latencies and throughput measurements will be analyzed using confidence intervals (95% CI) to account for performance variability in distributed systems. To visualize the distribution of latencies across different scaling configurations, we will utilize Empirical Cumulative Distribution Function (ECDF) plots, to reveal performance patterns and tail latencies. In order to validate the statistical significance of performance differences between vertical and horizontal scaling approaches, we will conduct paired t-tests where appropriate. Each benchmark configuration will be run multiple times to ensure statistical reliability.

[1] M. Bakli, M. Sakr, and E. Zimanyi, "Distributed moving object data management in MobilityDB," in Proc. 8th ACM SIGSPATIAL Int. Workshop on Analytics for Big Geospatial Data (BigSpatial '19), Chicago, IL, USA, 2019, pp. 1-10, doi: 10.1145/3356999.3365467.