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 open source and familiar 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] It does it by sharding data across multiple nodes while retaining the rich features of PostgreSQL. While query run times of specific queries have been evaluated[2], the vertical and horizontal scalability of the platform has not been fully explored and measured.

MobilityDB's distributed capabilities, enabled by integration with Citus, claim to enhance scalability. However, the platform's ability to efficiently scale read and write operations under real-world workloads has not been rigorously evaluated. Current research has primarily focused on optimizing read query performance, such as complex spatial and spatio-temporal queries. However, write throughput—critical for handling high-ingest workloads typical of IoT applications—remains underexplored, especially in scenarios involving concurrent writes across distributed nodes. Questions remain about latency, throughput, and scalability in vertical and horizontal configurations when handling realistic spatio-temporal datasets.

In this thesis we aim to benchmark the vertical and horizontal scalability regarding both read and write operations with generated spatio-temporal data to measure metrics like request latency and write throughput. We plan to deploy MobilityDB on single-node and multi-node setups. Additionally, we will analyze how query options affect impact on the performance in different configurations. Afterwards, we want to compare scalability trade-offs between vertical and horizontal approaches.

Benchmarking data will be generated using a custom data generation tool specifically developed for this study. The tool will simulate the movement and locations of birds, chosen due to their seasonal migration patterns, localized movements within standard habitats, varying flight heights and times. Additionally, modeling different bird species will enable simulation of diverse flight altitudes, fly times and migration times, providing a robust dataset for spatio-temporal analysis. We will compare the performance of MobilityDB combined with Citus on different hardware configurations using Google Compute Engine service on Google Cloud Platform. We will evaluate several scenarios, including vertically scaling a single node and scaling out by distributing MobilityDB across several instances. While testing horizontal scaling, we will benchmark different hardware configurations of nodes, to determine the point at which hardware limitations become a bottleneck for performance, especially we will focus on the coordinator node.

For evaluation of our results, we will employ multiple statistical and visualization approaches to ensure comprehensive analysis of the system's performance characteristics. 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. For write throughput analysis, we will present time series plots showing sustained throughput over extended periods, accompanied by statistical summaries including median, 95th, and 99th percentiles. 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

---

[1] Bakli, Sakr and Zimanyi, "MobilityDB: A Mobility Database Based on PostgreSQL and PostGIS," ACM Transactions on Database Systems (TODS), Volume 45, Issue 4, Article No. 19, Pages 1-42, December 6, 2020

[2] Bakli, Sakr, and Zimanyi, "Distributed Mobility Data Management in MobilityDB," Proceedings of the 2020 21st IEEE International Conference on Mobile Data Management (MDM), June 2020
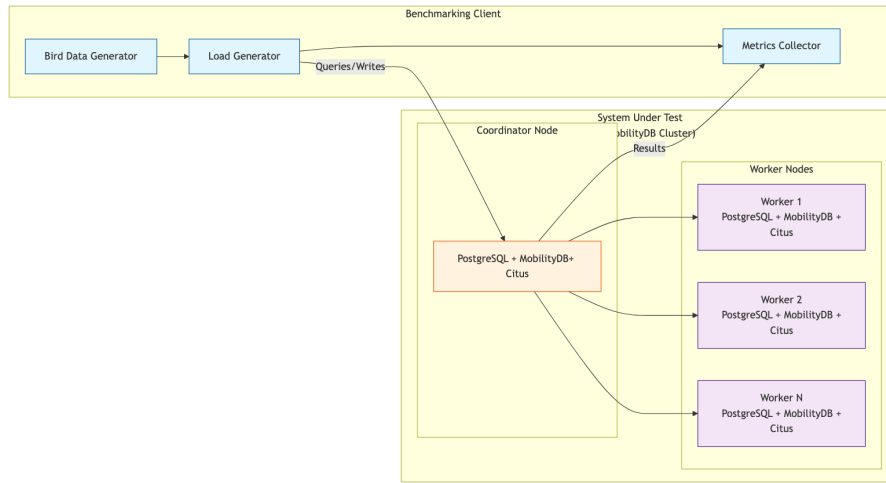
Figure 1: Architecture of the benchmarking setup. The benchmarking client generates synthetic bird movement data and executes the workload against the distributed MobilityDB cluster. The System Under Test consists of a coordinator node that distributes queries across multiple worker nodes, each running PostgreSQL with MobilityDB and Citus extensions. Metrics are collected to analyze system performance and scalability.

times with sufficient duration (minimum 30 minutes per run) to ensure statistical reliability and to capture any performance degradation or variance over time.