TimescaleDB: Re-engineering PostgreSQL as a Time-series Database

Time-series data is now everywhere—IoT, user event streams, system monitoring, finance, adtech, industrial control, transportation, and logistics—and increasingly used to power core applications. It also creates a number of technical challenges: to ingest high volumes of structured data; to ask complex, performant queries for both recent and historical time intervals; to perform specialized time-centric analysis and data management. And this data doesn't exist in isolation, entries must often be joined against other relational data to ask key business questions (e.g., tracking a shipping container is much more powerful once combined with information about its goods).

Many developers working with time-series data today turn to polyglot solutions: a NoSQL database to store their time-series data (for scale), and a relational database for associated metadata and key business data. This leads to engineering complexity, operational challenges, and even referential integrity concerns.

In this talk, I describe why these operational headaches are unnecessary and how we re-engineered PostgreSQL as a time-series database in order to simplify time-series application development. In particular, the nature of time-series workloads—appending data about recent events—presents different demands than transactional (OLTP) workloads. By taking advantage of these differences, we can improve insert rates by 20x over vanilla Postgres and achieve much faster queries, even while offering full SQL (including JOINs).

TimescaleDB achieves this by storing data on an individual server in a manner more common to distributed systems: heavily partitioning (sharding) data into chunks to ensure that hot chunks corresponding to recent time records are maintained in memory. This right-sized chunking is performed automatically, and the database can even adapt its chunk sizes based on observed resource demands. Yet it hides this behind a “hypertable” that can be inserted into or queried like a single table: even at 100B+ rows over 10K+ chunks. While this adds a few additional milliseconds for query planning, it enables TimescaleDB to avoid the performance cliff that Postgres experiences at larger table sizes (10s of millions of rows).

TimescaleDB is packaged as a Postgres extension, released under the Apache 2 license.