QuestDB

Community Meetup #7

2022-07-28

QuestDB Team

QuestDB

Today’s agenda

1. Update about replication
2. AMA
3. Other news

Roadmap updates

Javier Ramirez, Developer Relations at QuestDB

@supercoco9 - @QuestDB

QuestDB

Recent developments

- Replication (WIP) **<- today's focus!**
- Partitions detach/attach workflow (soon)
- CSV imports for unordered data without size limitation (soon)

What do we mean by replication

- Higher availability for reads, and higher throughput
  - Write to **one** QuestDB instance, propagate changes, and read from **many**

- Higher availability and higher throughput for both reads and writes (available only on QuestDB Cloud)
  - Write to **many** QuestDB instances, propagate changes, and read from **many**

Current model for writing table data

- Data is received and kept in memory
- Once CommitLag is reached (or maxuncommitted rows, or table idle…), the TableWriter writes data into the table files
- If Out-of-order rows are present, the operation will be costly, affecting reads
- Depending on how costly the writing operation is, ingestion performance will also be affected
- Prone to Table Busy errors

First Step: Detaching ingestion and table writes

Implementing WAL (Write Ahead Log) (Work In Progress) 🚧

- Data (and any updates) is ingested to an append-only file (multiple WAL per table)
- If we take an empty DB and apply the changes in the WAL, we should get to the current dataset
- WAL file is structured by columns and divided into segments
- Segments have sequential IDs and are tracked by a TableSequencer, who coordinates and avoids conflicts
- Periodically, changes contained in the WAL are committed by the TableWritter

WAL benefits

- Enables replication
- Decouples ingestion and writing, allowing for future optimizations
- WAL will remove ingestion slow-down in the case of Out-of-order data
- WAL will improve the TableBusy scenarios

Second step: WAL replication and read replicas

(Expected in a few months from now. Depends on WAL)

- Each WAL segment will automatically be written into Amazon S3
- On S3 notification, the TableSequencer will fetch the pending segments in the right order and will use the TableWriter to apply changes locally
- The replicated tables will achieve eventual consistency
- The unit of replication will be the table
- Initially replication using only S3. If you want to contribute by making the TableSequencer interact with segments in other Object Storages (cloud or not), do get in touch

AMA

Javier, Developer Advocate

& QuestDB Team

Questions from the community

- Why does it take a while to see data after ingesting?
- Where can I find the Grafana plugin for QuestDB?
- What timestamp format do I need to use with ILP?

We’re hiring!

- Senior Backend Engineer (Python), QuestDB Cloud
- Cloud Engineer, QuestDB Cloud

Apply at: questdb.io/careers

Thank you!

@QuestDb

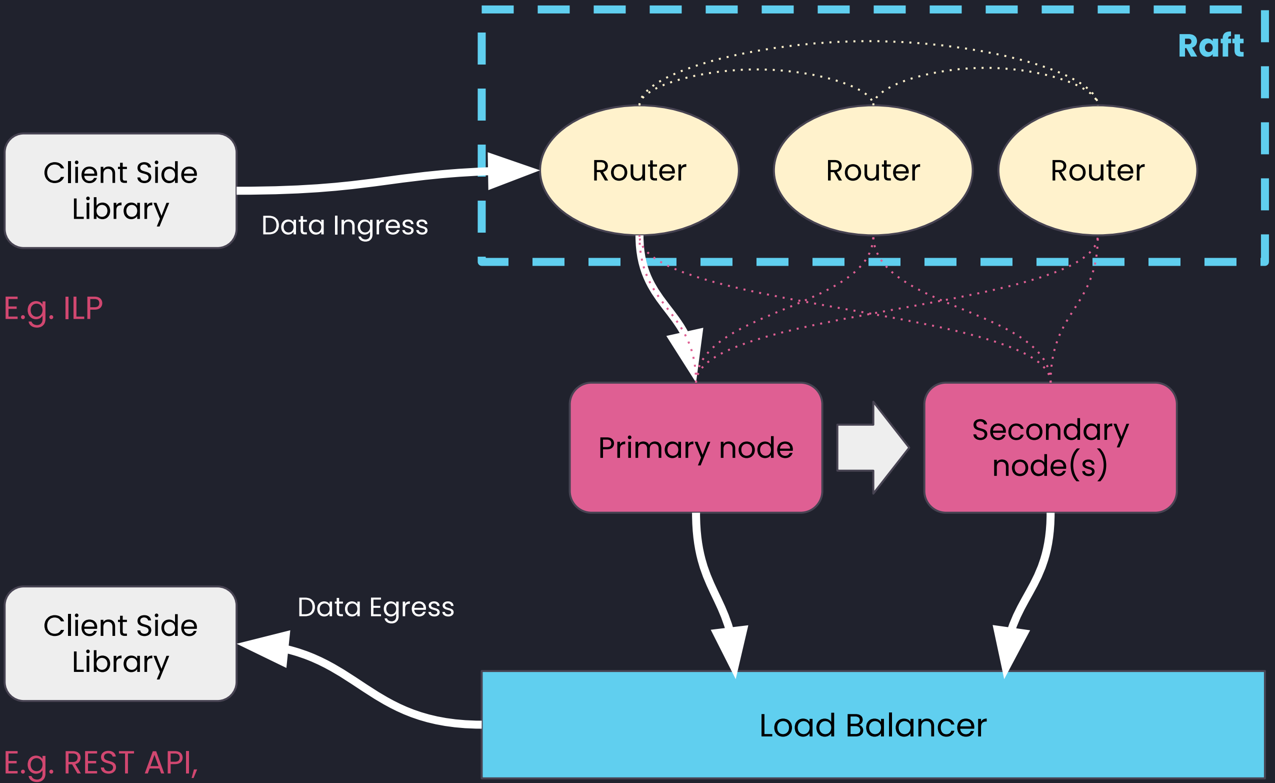
QuestDB

Join Slack: slack.questdb.io

Star us: github.com/questdb/questdb

Clustering

Diagram



E.g. ILP

E.g. REST API, Postgres wire