Microsoft

Azure Cosmos DB for PostgreSQL

Distributed SQL service built on open-source Postgres & Citus

Marco Slot

Principal Software Engineer at Microsoft

Azure Cosmos DB for PostgreSQL

Distributed PostgreSQL for modern cloud-native applications

True PostgreSQL

Not a fork

Latest PostgreSQL version within 1 week

50+ PostgreSQL extensions

Scale out from 1 to 1000s of cores

Distributed query execution

Online scaling with zero downtime

Local and Global replication of data

Built for relational workloads

Transactions

Primary/Foreign Keys, Joins, and Constraints

Custom types, stored procedures

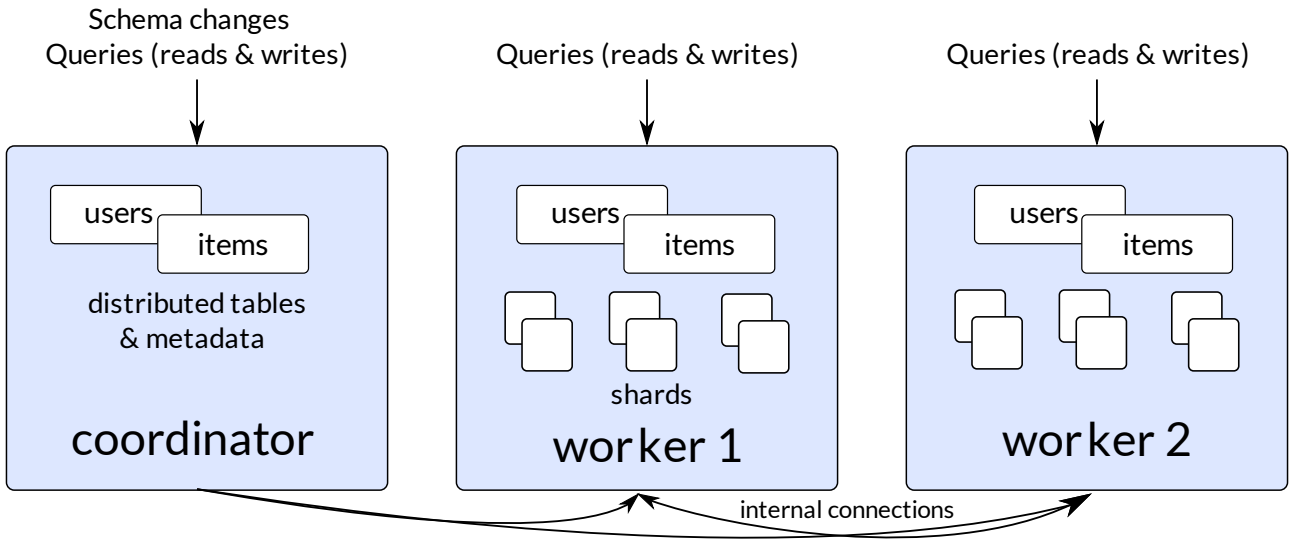
Cosmos DB managed service platform

Global Distribution – Seamless Elasticity – High Availability – Point in time recovery – Azure integrations

Available Open Source as the Citus extension to PostgreSQL

Citus: Distributed PostgreSQL as an Extension

Citus is a PostgreSQL extension that uses planner, executor, and utility command hooks to transparently distribute and replicate PostgreSQL tables across a shared-nothing PostgreSQL cluster.



Fully open source:
https://github.com/citusdata/citus

SIGMOD '21:
"Citus: Distributed PostgreSQL for Data-Intensive Applications"

Citus Features & Gaps

Most PostgreSQL features just work on Citus tables

- Joins
- Transaction blocks
- Subqueries & CTEs
- Sequences
- Expression indexes
- Partial indexes
- Custom types
- Prepared statements
- Stored procedures
- Time-partitioning
- …

Distributed database superpowers with PostgreSQL-level efficiency

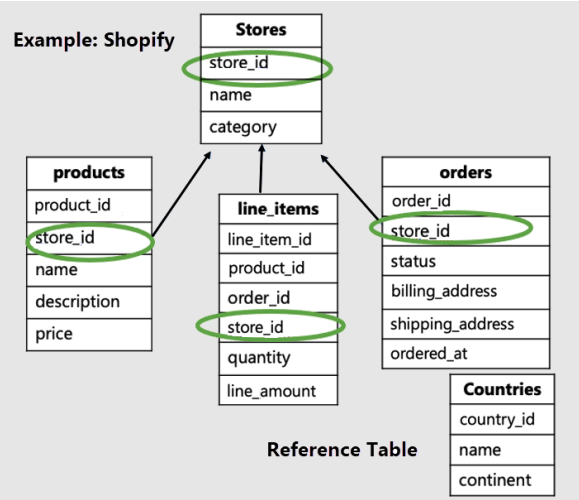
- Distributed & reference tables
- Co-location
- Scale OLTP throughput
- Fast co-located joins, foreign keys, ..
- Parallel, distributed queries
- Transactional ETL (INSERT..SELECT)
- Fast data loading (COPY)
- Online rebalancing
- Stored procedure call routing
- Columnar compression
- …

Some gaps remain

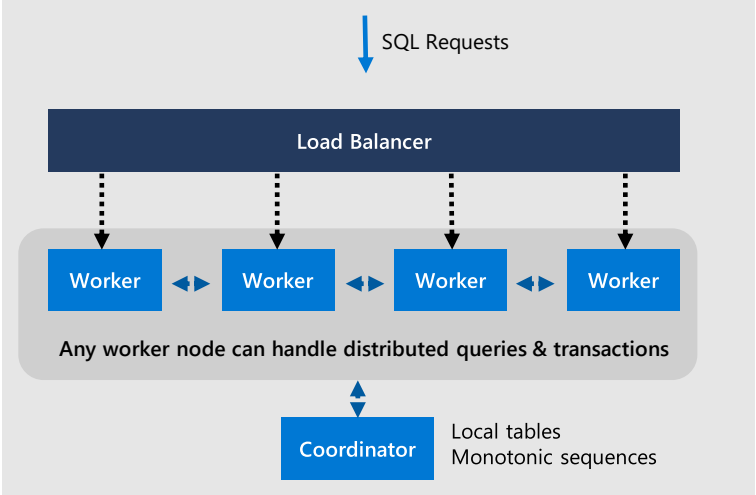
- Schema-level sharding
- DDL from any node
- Automatic shard splits
- Non-co-located foreign keys, triggers
- Unique constraints on non-dist. column
- Cross-node snapshot isolation
- Geo-partitioning
- Database-level sharding
- Non-co-located correlated subqueries
- Vectorized execution
- …

Common workload patterns

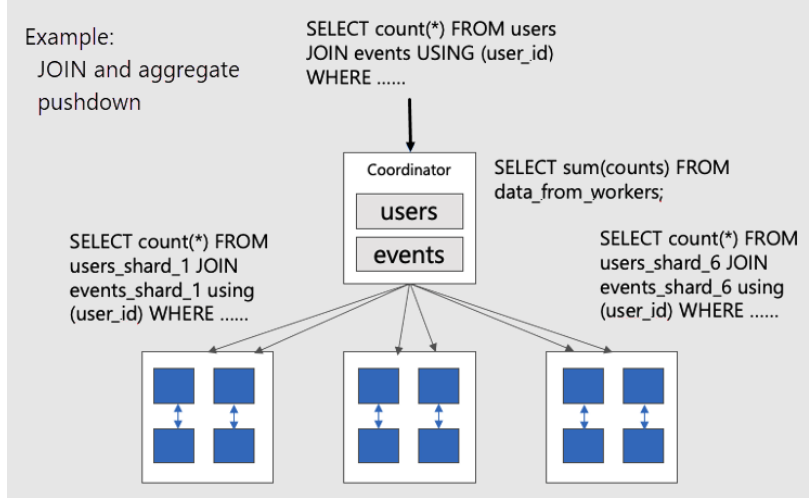
Multi-tenant OLTP (e.g. Software-as-a-service)



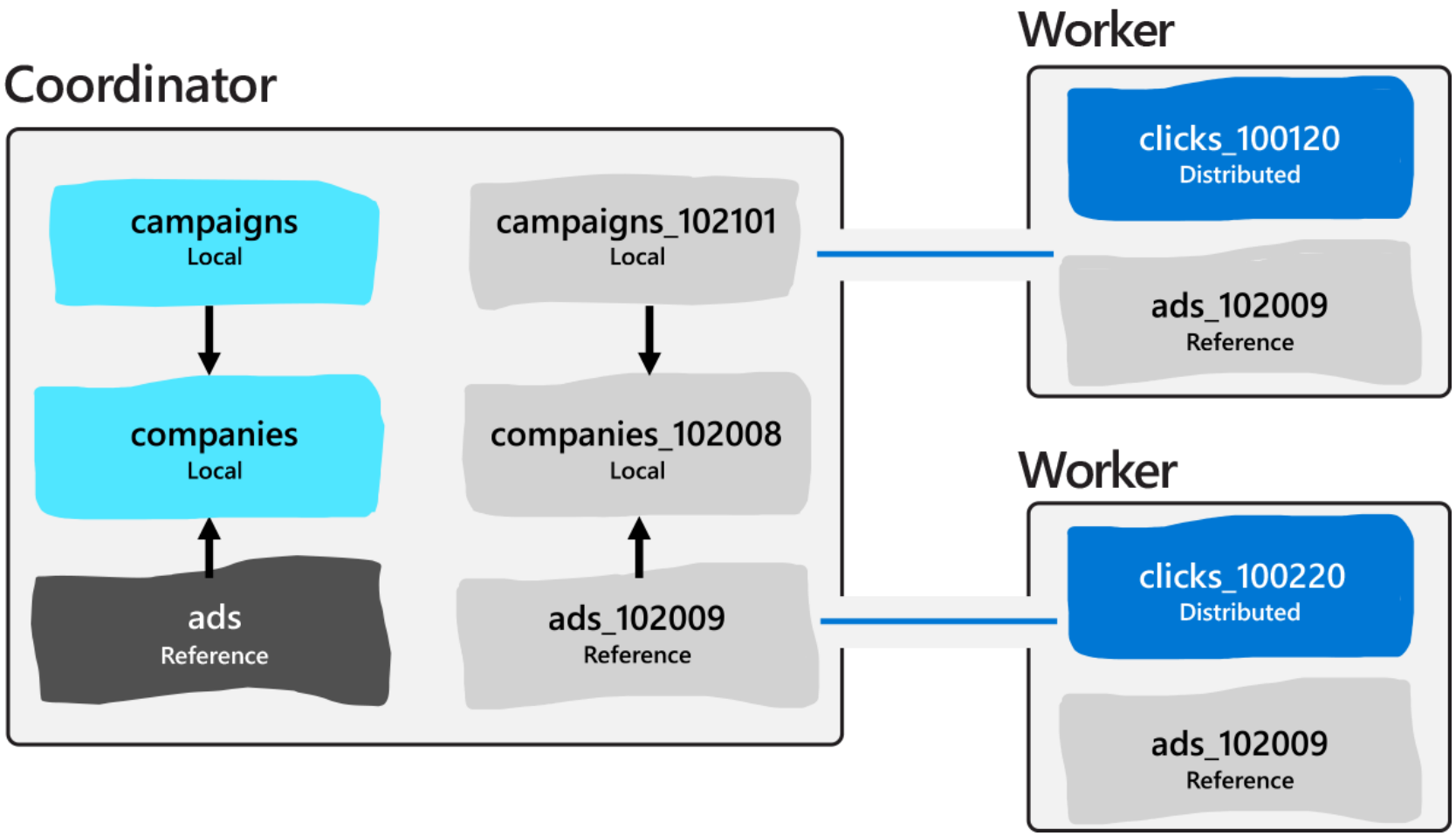
High throughput CRUD (e.g. IoT)



Real-time analytics (e.g. customer dashboards)



Hybrid local-distributed databases



Lessons learned in 10 years of Citus development

Relational database workloads are highly latency-sensitive due to the need to evaluate relationships, interactive protocols, ORMs

→ Pack related data together using co-location, reference tables, local tables, …

At scale, efficiency is too important to trade efficiency for scale

→ Lean on existing RDBMS functionality to inherit price-performance characteristics

Distributed PostgreSQL only makes sense for specific workload patterns

→ Target multi-tenant (SaaS), real-time analytics (IoT, time series), CRUD, or hybrid.

…

Lessons learned in 10 years of Citus development

Scalability is not (just) about transaction throughput

→ Real workloads are complex. Infrequent O(N) operations often dominate at scale.

PostgreSQL development never stops

→ Contribute to PostgreSQL, build extensions, do not fork

Developing a complex mission-critical distributed database in which all features are related is hard

→ Do small, independently useful projects with long-term goals in mind

Thank you!

We will at some point be hiring again ☺
(in Amsterdam, Istanbul, Redmond, San Francisco, or remote)

marco.slot@microsoft.com

Cosmos DB for PostgreSQL: https://aka.ms/AzureCosmosDBPGblog
Citus on GitHub: https://github.com/citusdata/citus