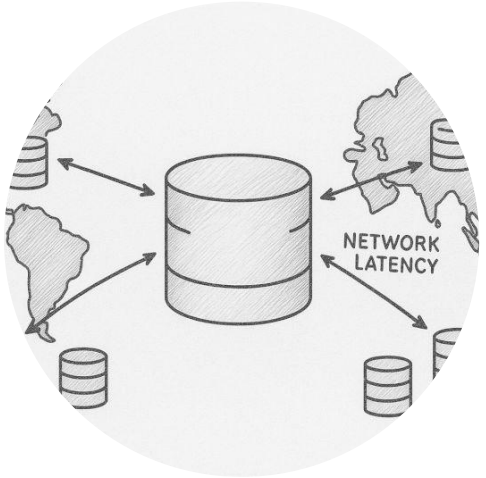
Geo-distributed OLTP: promises scale however latency defines reality



Hi!

Andrii Kosachenko

CTO, PortaOne Inc.

They sell solutions, you buy the lessons

Distributed systems mitigate, but do not eliminate, network latency.

The same is (hopefully) true for geo-distributed RDBMS.

Why bother with distributed OLTP?

Problems	Solutions
Failures, outages, disasters, catastrophes, ...	**High Availability (HA)** and **Disaster Recovery (DR)**
Performance plateau is hit, demand for elasticity, ...	**Scalability**
Data Sovereignty, Compliance, global user base, ...	**Multi-regional** setups (a.k.a. **multi-DC**, **geo-redundancy**)

Why bother with relational DBMS?

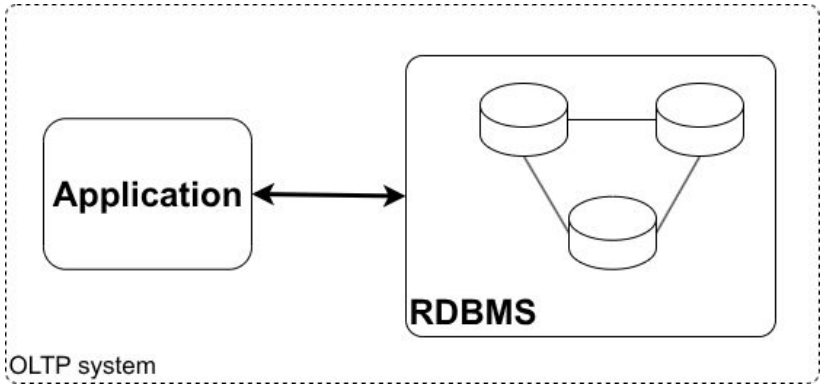
Transactions!

I.e. to prevent "we're rich!" and "we're broke!" from being true at the same time.

What is “OLTP system” exactly?

Key components

- business application(s)
- distributed RDBMS



Key characteristics

- provides domain-specific business app(s) for human-machine (H2M) or machine-machine (m2m) interactions
- high-availability: 99.999%
- relatively low latency (response time): 10-100ms
- high throughput: thousands + **small** and **concurrent** units-of-work

What OLTP system does and how

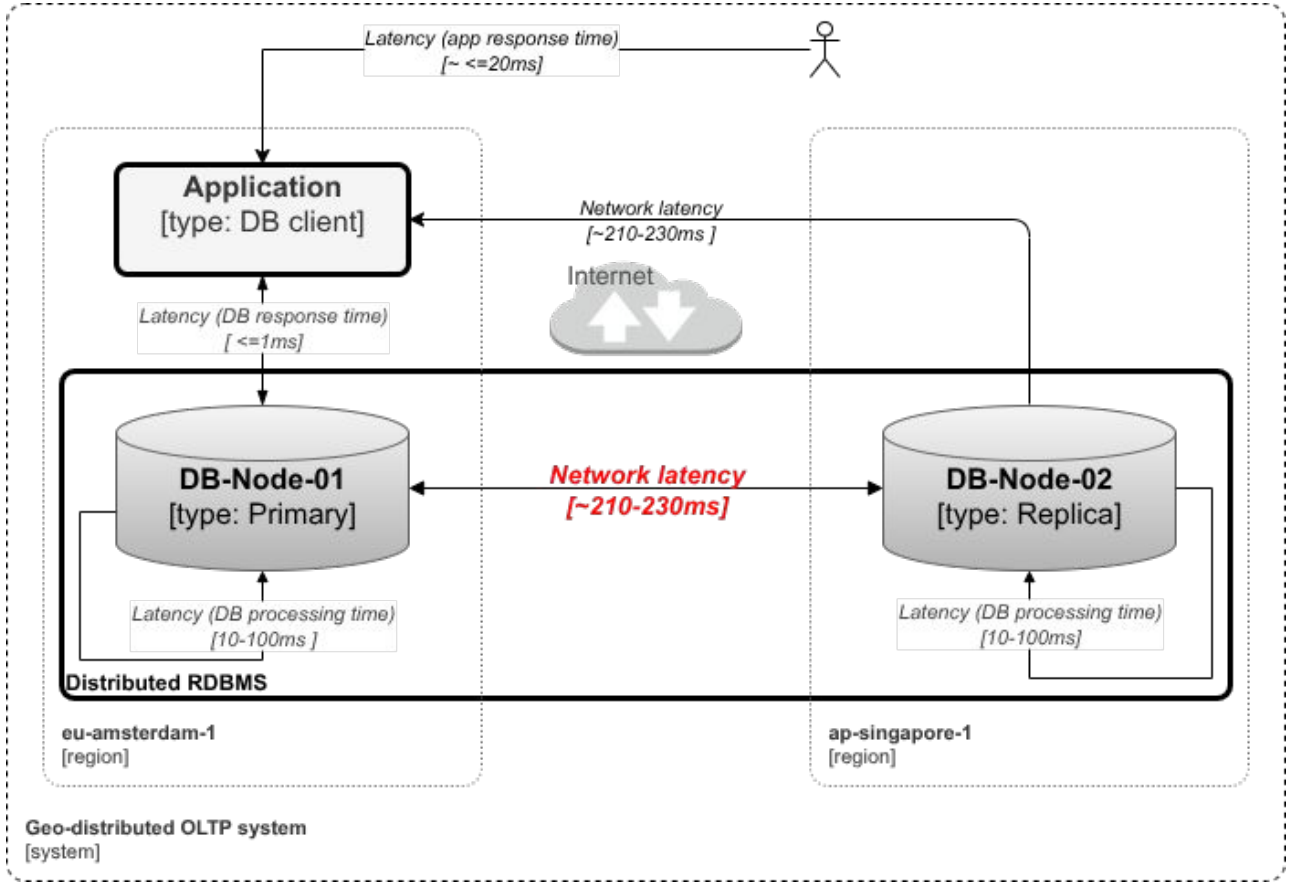
What

- **Business app**: handles **app-units-of-work** that happen whilst many humans and(or) machines interact at rapid rate
- **RDBMS**: provides persistent data storage and handles **db-units-of-work** (transactions)

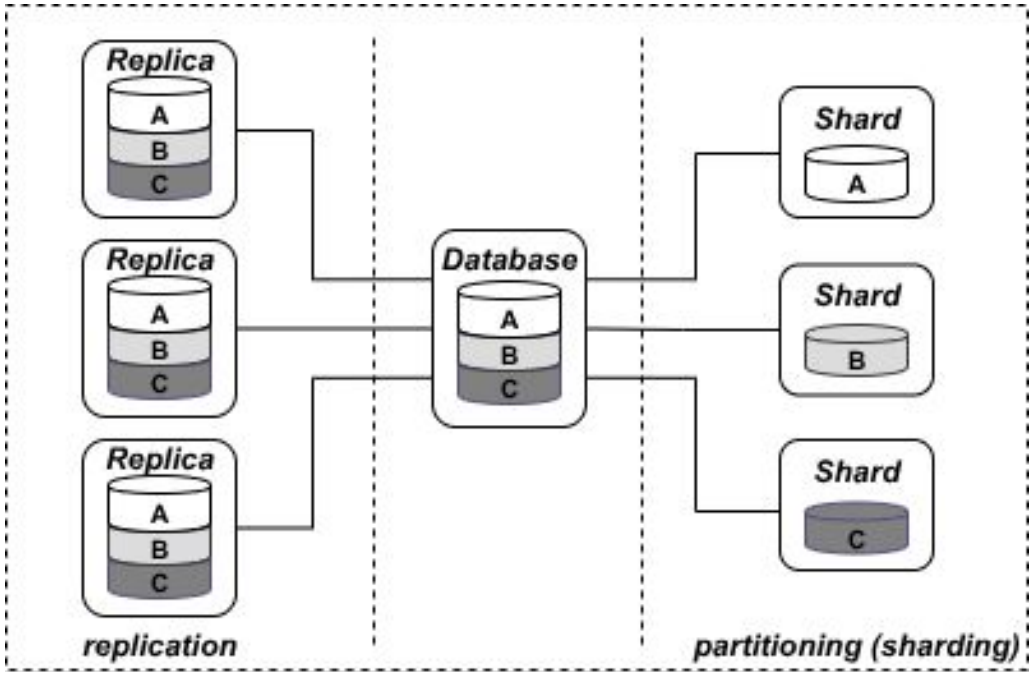
How

- single **app-unit-of-work** is assembled from one (or multiple) **db-units-of-work** and is processed (hopefully **fast enough**)
- RDBMS availability is achieved via replicating and partitioning data over multiple DB nodes (possibly across multiple geographic DCs)

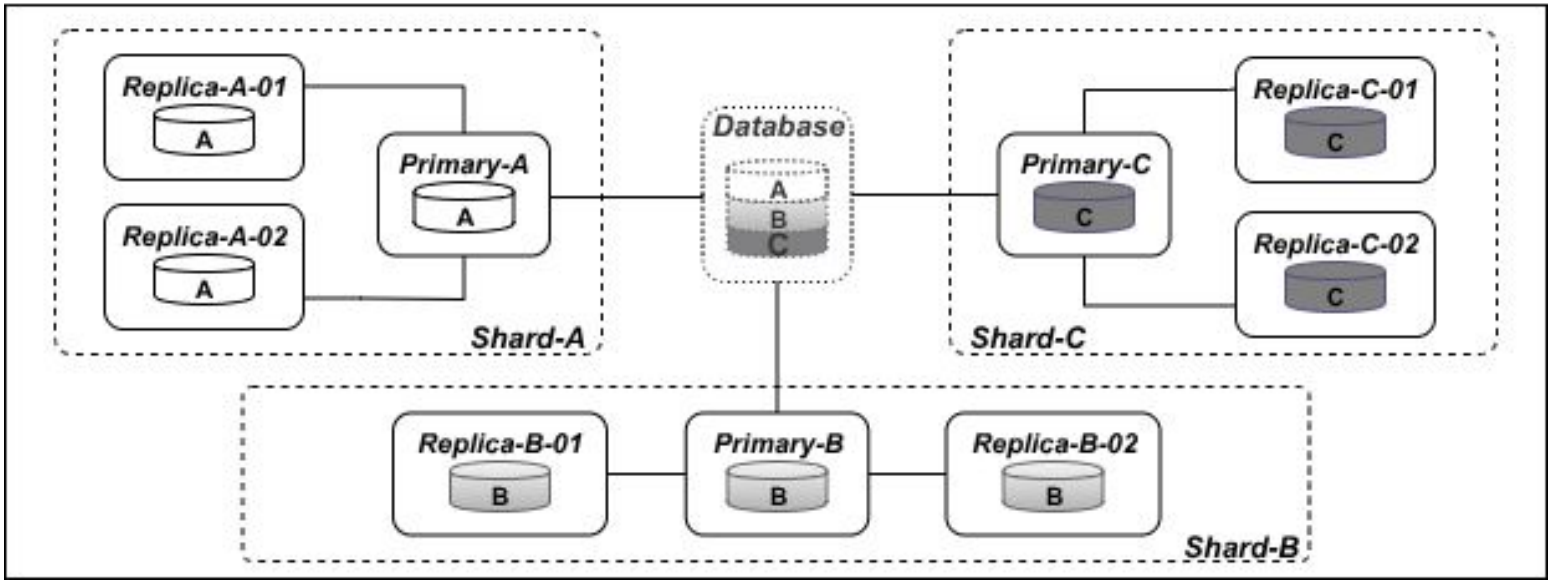
What is “latency” exactly?



RDBMS: Replication and Partitioning



Replication and Partitioning (combined)

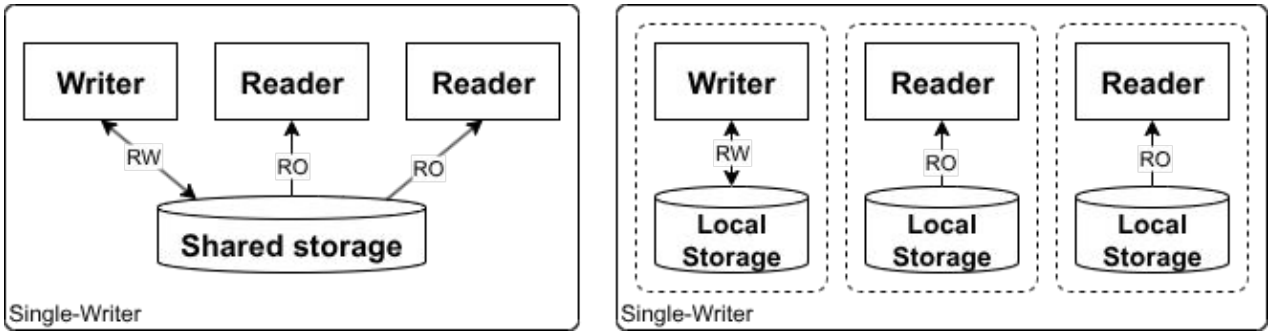


DBMS designs: data access path viewpoint

See: "Is Scalable OLTP in the Cloud a Solved Problem?" by T. Ziegler et al., 2023

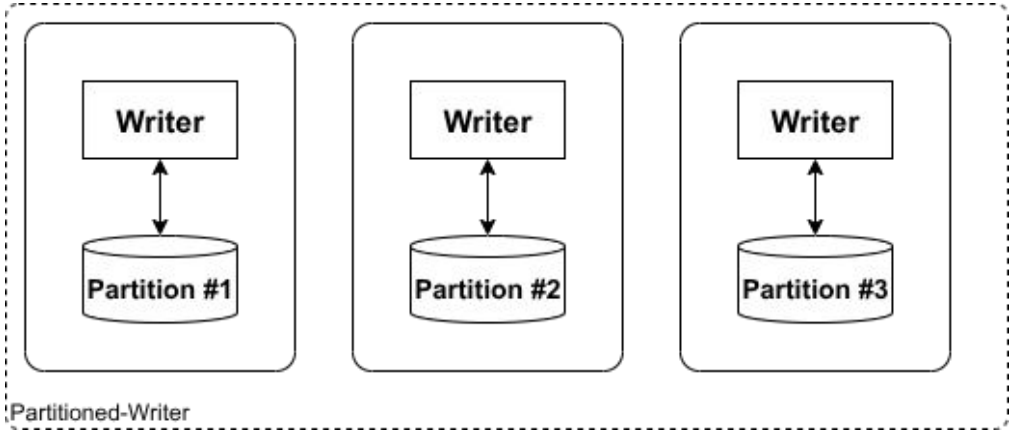
- **Single-Writer**: single Primary (RW) node, multiple Replica (RO) nodes
- **Partitioned-Writer**: one Primary per partition (each writes to its local shard)
- **Shared-Writer**: multiple RW nodes utilize shared storage:
  - no coherent cache: RW nodes write/read from the shared storage
  - coherent cache: RW nodes follow a cache coherence protocol

DBMS designs: Single-Writer



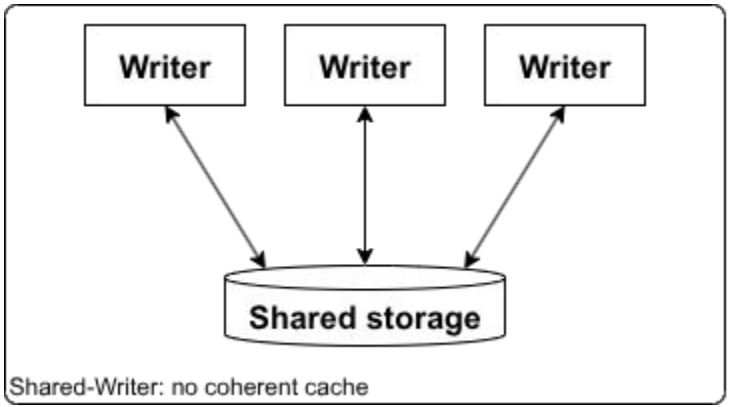
- **Data path**: single Writer node and multiple Reader nodes access storage
- **Latency impact**:
  - writes are bounded by the single RW node capacity
  - read latency increase when RO Nodes sync state updates via network
- **Examples**: MySQL, PostgreSQL, ... (typical Active/Passive setup)

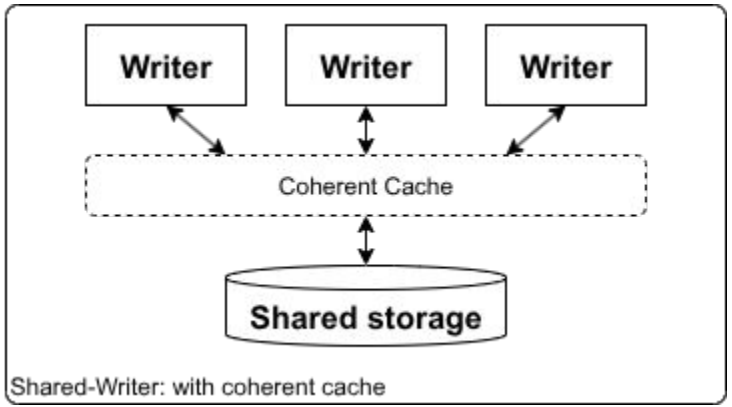
DBMS designs: Partitioned-Writer



- **Data path**: one Writer per data partition (shard)
- **Latency impact**:
  - cross-partition TXs require Two-Phase Commit (2PC). E.g. standard 2PC requires up to 5(!) RTTs. High latency is inevitable in multi-DC setups
- **Examples**: Spanner, CockroachDB, YugabyteDB, Vitess, ...

DBMS designs: Shared-Writer





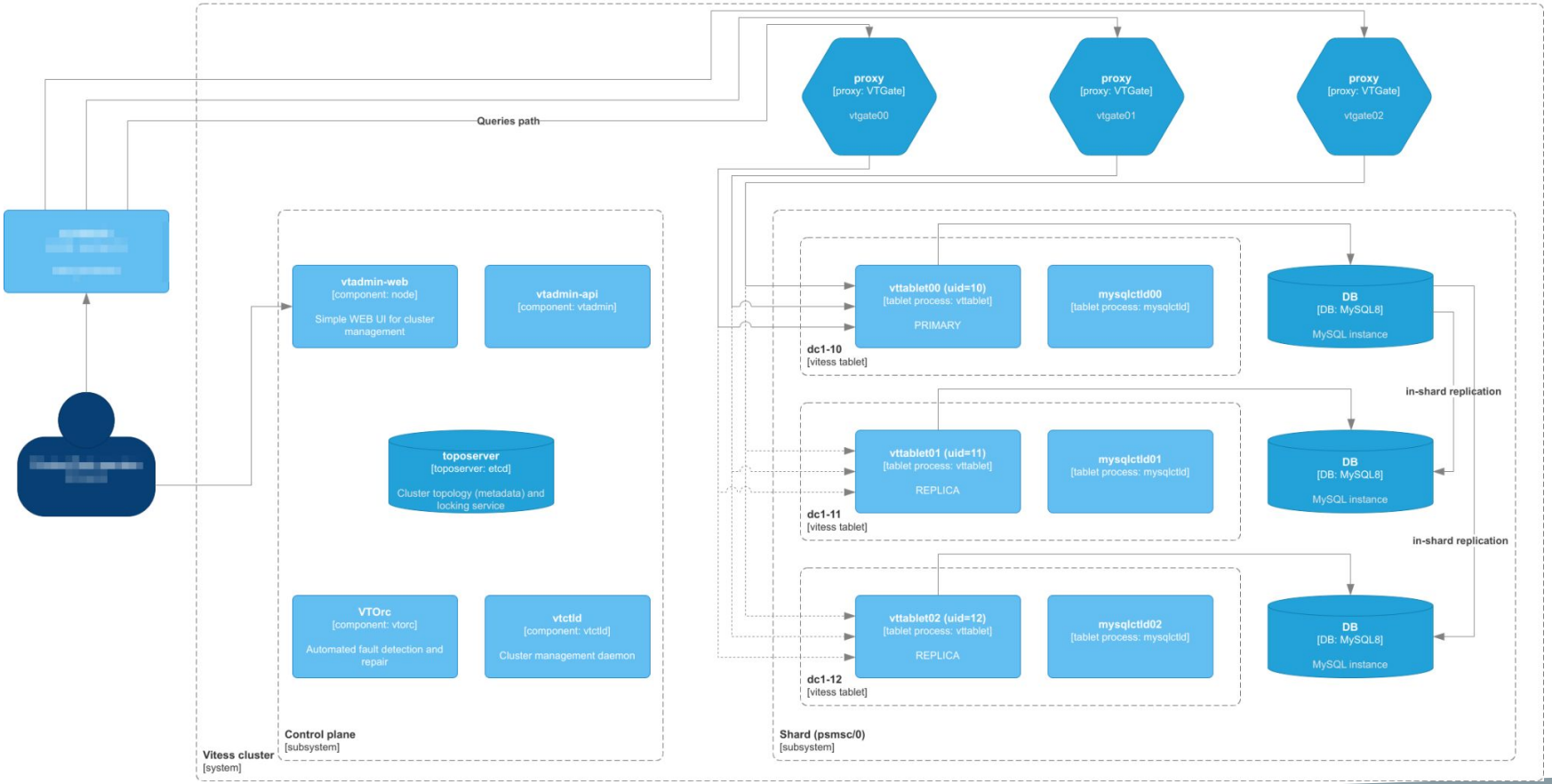
- **Data path**: multiple Writers access Storage via cache coherence protocol
- **Latency impact**: consistency is maintained by cross-network cache coherence messaging (invalidation/update). Every cache page modification requires cross-node coordination (high latency is inevitable).
- **Examples**: Oracle RAC

Asymptotic scalability limits of RDBMS

	Single-Writer	Partitioned-Writer	Shared-Writer
Data access path	*1*RW + N*ROs*	*N shards * (1*RW + N*ROs)*	*N*RWs update coherent cache*
Complexity	Low	Medium	High
Elasticity	Only RO	Limited (needs rebalancing!)	Yes
Uniform Reads	Yes (infers more ROs)	Yes	Yes
Uniform Writes	No	Yes	Yes
Skewed Reads	Yes (infers more ROs)	Yes (infers more ROs)	Yes
Skewed Writes	No	No	No
Latency mitigation potential for OLTP	No (only OLAP)	Limited (region-aware partitioning)	Yes (CC mgmt overhead)

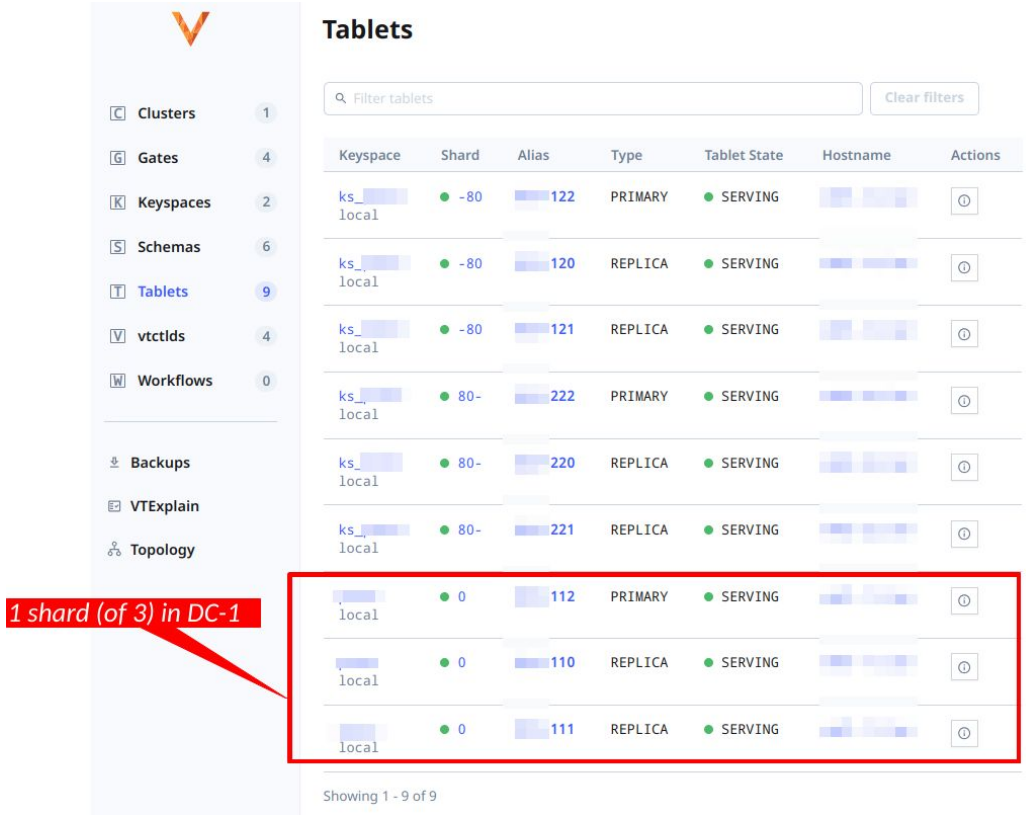
Partitioned-Writer: closer look

*Components and communication paths within Vitess cluster: 1 shard (of 3) with 1 * Primary + 2 * Replicas*



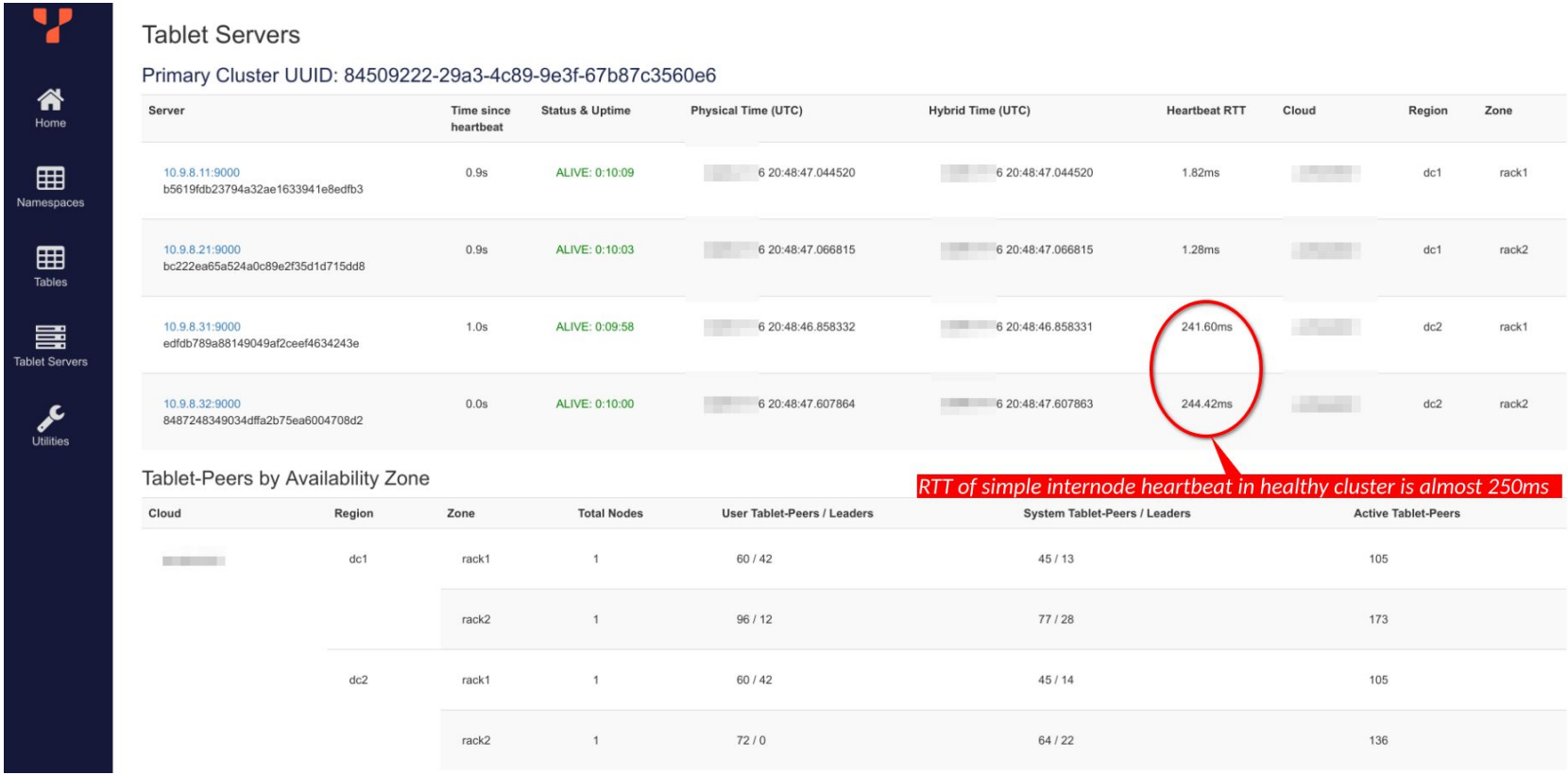
Partitioned-Writer: closer look

*Vitess cluster (3 shards): 3 * (1 * Primary + 2 * Replica)*



Partitioned-Writer: closer look

*YugabyteDB cluster (3 shards): 3 * (1 * Primary + 2 * Replica) (NB: 3-rd DC and three 2-nd Replicas are filtered out)*



Latency penalty example (real one)

- Fact #2: according to this issue [redacted] latency between their sites is ~250ms+ [redacted]. And testing (see this and this) revealed that when network latency increases from 0.1ms to 256ms:
  - avg performance (TPS) drops from 167 to 0.53 -> which makes 315 times difference;
  - avg latency of a single trx (cost in ms of processing single call event) increases from 61.40ms to 7242.48 -> which makes 117 times difference;
  - simple math./logical calculation is this: the best avg trx latency observed for MySQL was 27ms. Start call trx includes 10 requests and stop call trx includes 6 requests (see details where it comes from) which makes avg 8 reqs per trx which in turn means that theoretical minimum of waiting time for single trx is 8*256ms = 2048ms. I.e. minimal theoretical "processing vs waiting" ratio can't be lesser then 2048/27=75 times (in reality it's substantially greater which is also confirmed in practice by two observations mentioned above).

db-unit-of-work: 1TX comprises 6 requests

db-unit-of-work: 1TX comprises 10 reqs

Why latency mitigation is “delayed”?

The theoretical barrier

- the speed of signal propagation (speed of light)
- the CAP theorem

The business and market barrier

- "Good Enough" is often good enough
- marketing vs. reality: "globally distributed" is often just a marketing claim

Engineers perspective

- OLTP systems are more often evolved than designed from the ground up
- reduced exposure to low-level system architecture (era of managed services)
- the "One Size Fits All" fallacy

Mitigation: research and practice

Research

- advanced replication and consistency models
- novel concurrency control protocols/MDCC (Multi-Data Center Consistency)
- novel data partitioning and geo-distribution strategies
- [?] leveraging modern hardware (e.g. RDMA)

Practice

- data locality and geo-partitioning: row-level partitioning, regional tables, ...
- app level design for latency tolerance: batching queries, async workflows,...
- (!!) relaxed consistency models: stale reads, per-statement or per-TRx consistency levels, ...

Useful resources

- Kleppmann, Martin. Designing Data-Intensive Applications: The Big Ideas behind Reliable, Scalable, and Maintainable Systems. First edition, O'Reilly Media, 2017. 591 c.
- Ziegler, T., et al. (2022). Is Scalable OLTP in the Cloud a Solved Problem? Analyzing Data Access for Distributed OLTP Architectures. URL: https://www.vldb.org/cidrdb/papers/2023/p50-ziegler.pdf
- Daniel Abadi, Anastasia Ailamaki, David Andersen, Peter Bailis, et. al. 2022. The Seattle report on database research. Commun. ACM 65, 8 (August 2022), 72–79. URL: https://doi.org/10.1145/3524284
- Pavlo, A. and Aslett, M., 2016. What's really new with NewSQL?. Harvard, ACM Sigmod Record, 45(2), pp.45–55. URL: https://dl.acm.org/doi/10.1145/3003665.3003674
- Collin Lee, Seo Jin Park, Ankita Kejriwal, Satoshi Matsushita, and John Ousterhout. 2015. Implementing linearizability at large scale and low latency. In Proceedings of the 25th Symposium on Operating Systems Principles (SOSP '15). Association for Computing Machinery, New York, NY, USA, 71–86. URL: https://doi.org/10.1145/2815400.2815416
- Pat Helland, San Francisco, California, USA, Scalable OLTP in the Cloud: What's the BIG DEAL? The Database AND the Application Have a BIG DEAL: Their Isolation Semantics. URL: https://www.cidrdb.org/cidr2024/papers/p63-helland.pdf
- Dan Pritchett. 2008. BASE: An Acid Alternative: In partitioned databases, trading some consistency for availability can lead to dramatic improvements in scalability. Queue 6, 3 (May/June 2008), 48–55. URL: https://doi.org/10.1145/1394127.1394128

Andrii Kosachenko

THANK YOU!