yugabyteDB

Distributed SQL Databases Deconstructed

Understanding Amazon Aurora, Google Spanner & the Spanner Derivatives

Sid Choudhury

Bryn Llewellyn

NoCOUG Summer 2019 Conference

Types of Data Stores

Today's Focus

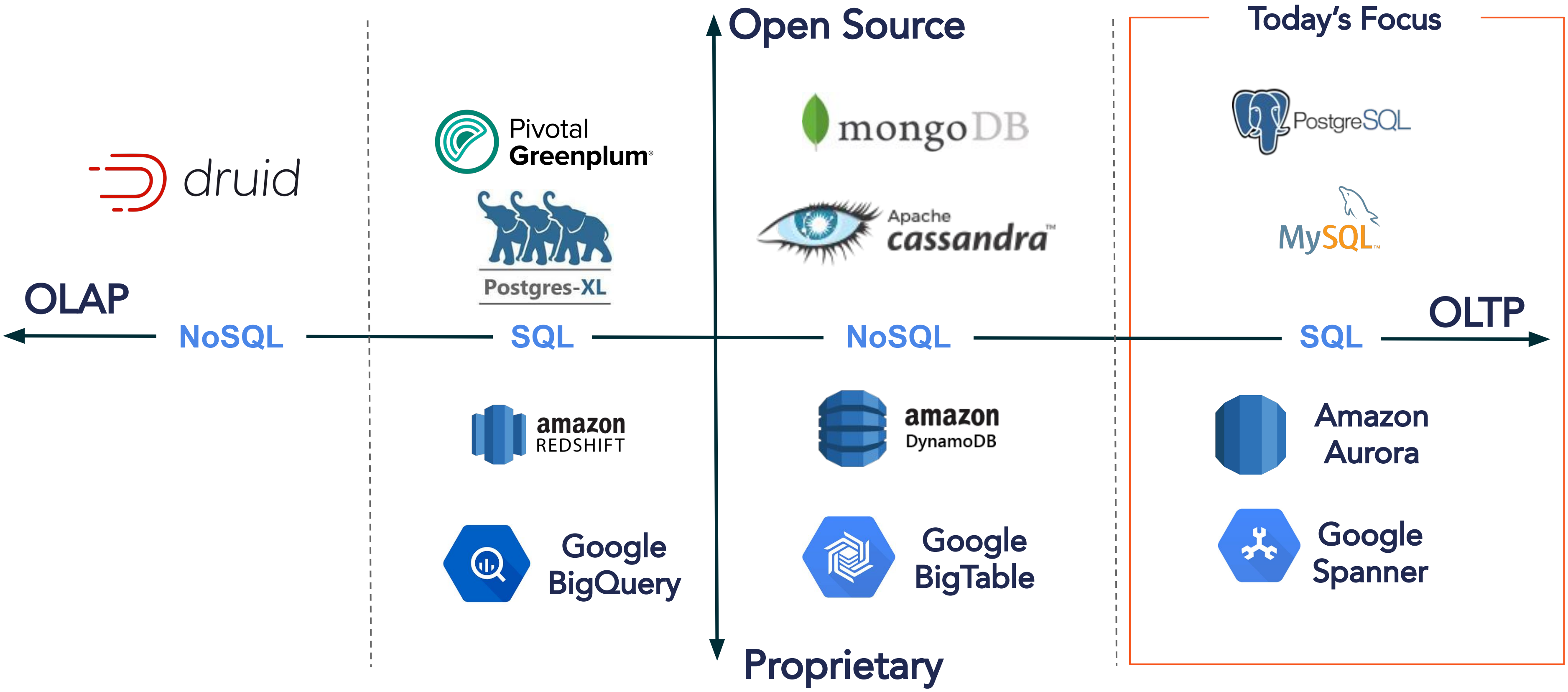
OLAP

OLTP

Write once, Read many
Few concurrent sessions
Long running, ad-hoc queries
Large table scans
Petabyte-scale data storage

Mixed reads & writes
Many concurrent sessions
Single-digit ms query latency
Point reads & short-range scans
Terabyte-scale data storage

Types of Data Stores



Open Source

Today's Focus

druid

Pivotal Greenplum

Postgres-XL

mongoDB

Apache cassandra

PostgreSQL

MySQL

OLAP

NoSQL

SQL

NoSQL

SQL

OLTP

amazon REDSHIFT

amazon DynamoDB

Amazon Aurora

Google BigQuery

Google BigTable

Google Spanner

Proprietary

Why Devs 😍 SQL?

1. Query Flexibility 💪
   - Model data once, change queries as business changes
   - Balance modeling richness with performance needs

2. Rich Ecosystem 🔌
   - Data modeling & query examples
   - Developer IDEs & data visualization tools
   - Easy to reuse & build integrations

3. Universal Standard for Data Access 😇
   - Learn once, use forever

Why Devs 😡 SQL?

1. Large Dataset? 📈
    - No horizontal write scalability
    - Use manually sharded SQL or non-transactional NoSQL
2. Infrastructure Failures? 🚨
    - No native failover & repair, SPOF w/ Single Node DB
    - Use complex replication schemes
3. Multi-Region/Geo-Distributed App? 🌏
    - Multi-master deployment is the only option
    - Data inconsistency w/ Last Writer Wins conflict resolution

Distributed SQL = Keep 😍 & Remove 😡

1. SQL Features
   - ACID, JOINs, foreign keys, serializable isolation

2. Horizontal Write Scalability
   - Scale write throughput by adding/removing nodes

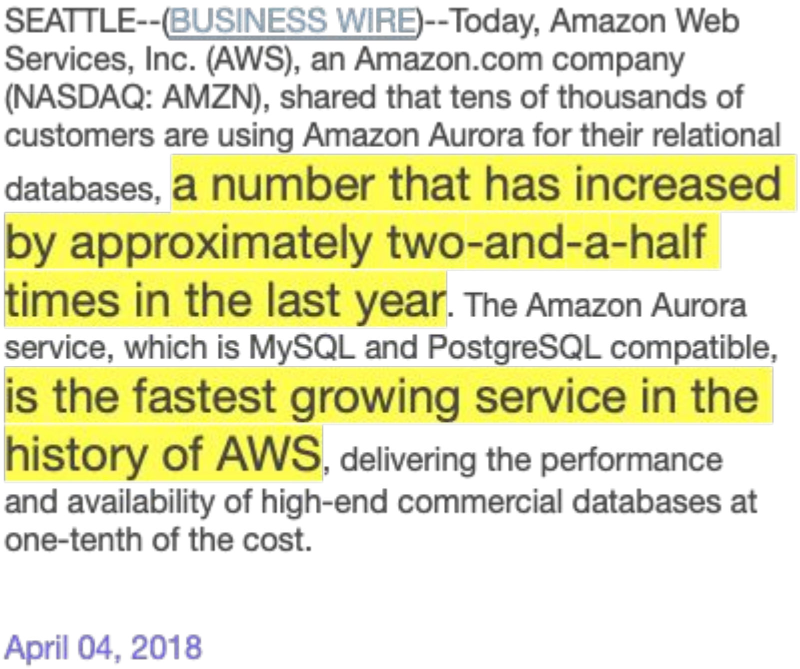
3. Fault Tolerance With High Availability
   - Native failover & repair

4. Globally Consistent Writes
   - Lower end user latency and tolerate region failures

5. Low Read Latency
   - Strongly consistent (aka correct) reads

What’s the fastest growing service in AWS?

Amazon Aurora



SEATTLE--(BUSINESS WIRE)--Today, Amazon Web Services, Inc. (AWS), an Amazon.com company (NASDAQ: AMZN), shared that tens of thousands of customers are using Amazon Aurora for their relational databases, a number that has increased by approximately two-and-a-half times in the last year. The Amazon Aurora service, which is MySQL and PostgreSQL compatible, is the fastest growing service in the history of AWS, delivering the performance and availability of high-end commercial databases at one-tenth of the cost.

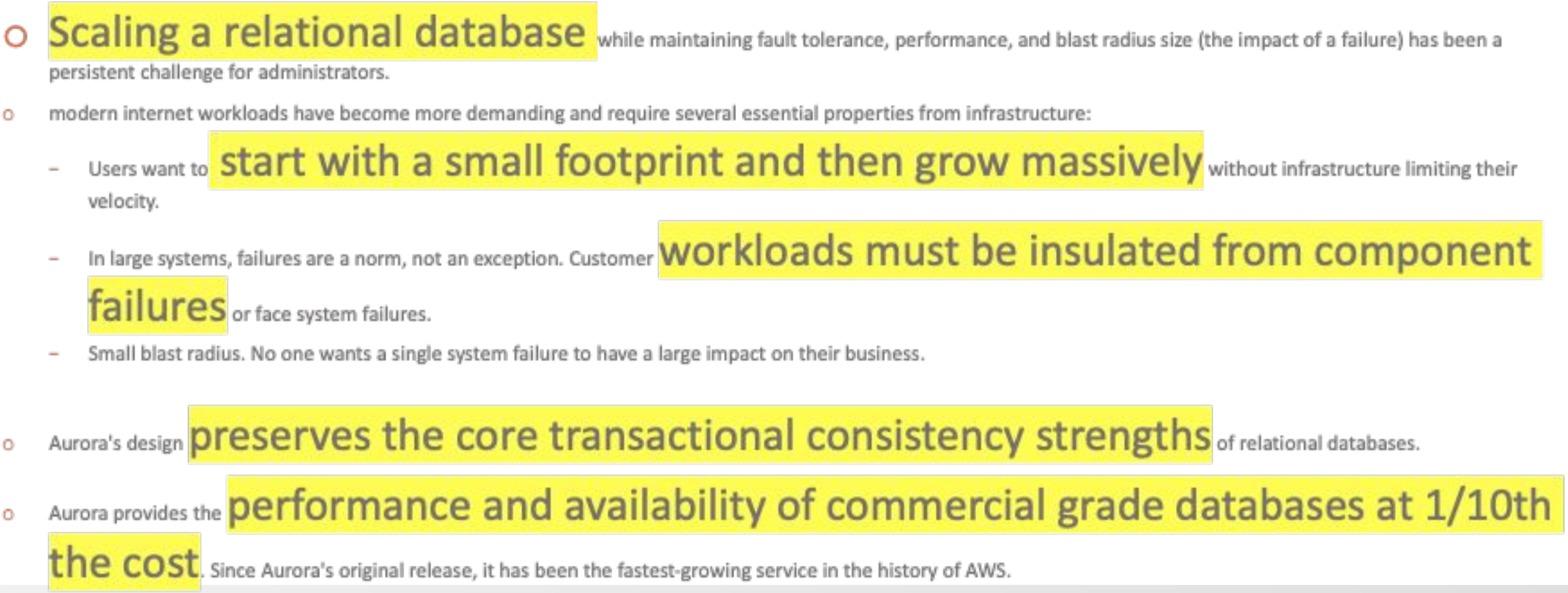
April 04, 2018

https://www.allthingsdistributed.com/2019/03/Amazon-Aurora-design-cloud-native-relational-database.html

Excerpts from Vogels blog post

Amazon Aurora ascendant: How we designed a cloud-native relational database

By Werner Vogels on 13 March 2019 10:00 AM | Permalink | Comments (4)



- Scaling a relational database while maintaining fault tolerance, performance, and blast radius size (the impact of a failure) has been a persistent challenge for administrators.
- modern internet workloads have become more demanding and require several essential properties from infrastructure:
  - Users want to start with a small footprint and then grow massively without infrastructure limiting their velocity.
  - In large systems, failures are a norm, not an exception. Customer workloads must be insulated from component failures or face system failures.
  - Small blast radius. No one wants a single system failure to have a large impact on their business.
- Aurora's design preserves the core transactional consistency strengths of relational databases.
- Aurora provides the performance and availability of commercial grade databases at 1/10th the cost. Since Aurora's original release, it has been the fastest-growing service in the history of AWS.

yugabyteDB

What database powers Google AdWords and Google Play?

Google Spanner

“At Google, Spanner supports tens of millions of queries per second and runs some of our most critical services, including AdWords and Google Play.”

https://ai.google/research/pubs/pub39966

Distributed SQL Architectures - Aurora vs Spanner

Shared Storage

Amazon Aurora

*"A **highly available** MySQL and PostgreSQL-compatible relational database service"*

Available on AWS since 2015

Shared Nothing

Google Cloud Spanner

"The first horizontally scalable, strongly consistent, relational database service"

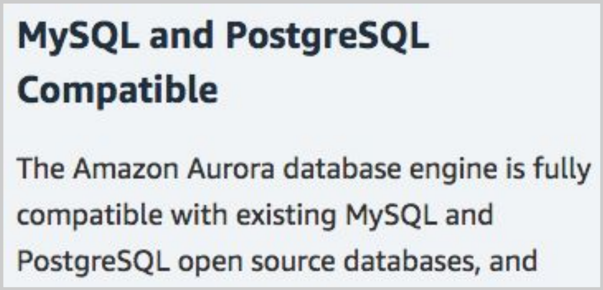
Available on Google Cloud since 2017

#1 SQL Features and Completeness

Depth of SQL Support

Amazon Aurora

✓ MySQL and PostgreSQL-compatible

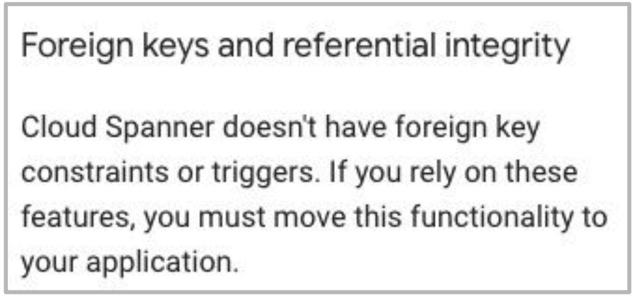


MySQL and PostgreSQL Compatible

The Amazon Aurora database engine is fully compatible with existing MySQL and PostgreSQL open source databases, and

Google Spanner

⚠ Subset of MySQL/PostgreSQL features



Foreign keys and referential integrity

Cloud Spanner doesn't have foreign key constraints or triggers. If you rely on these features, you must move this functionality to your application.

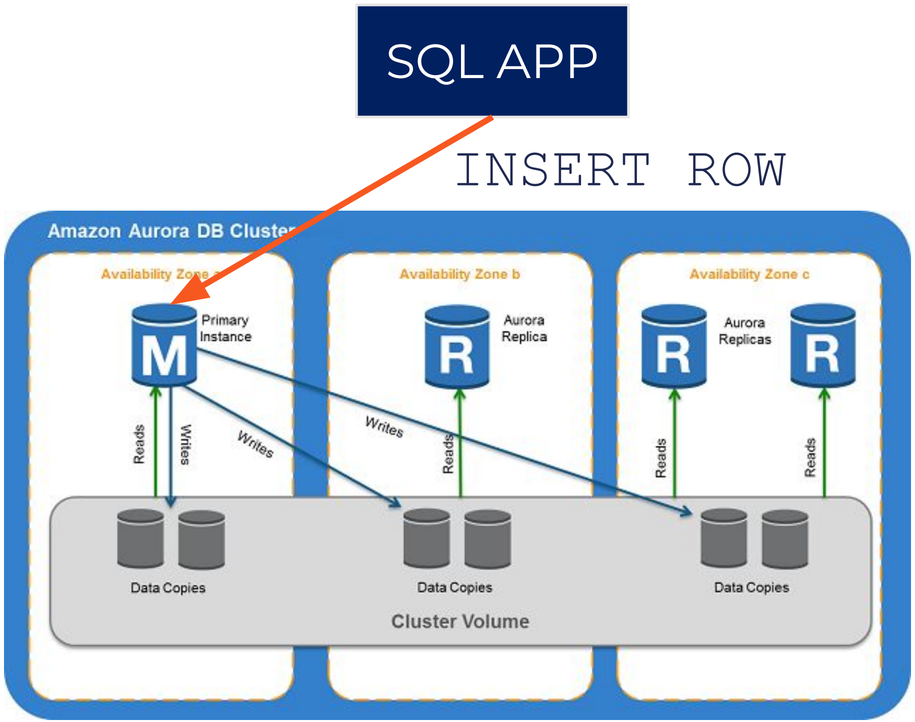
Aurora vs Spanner

Feature	Amazon Aurora	Google Spanner
SQL Features	✓	⚠
Horizontal Write Scalability		
Fault Tolerance with HA		
Globally Consistent Writes		
Low Read Latency		

#2 Horizontal Write Scalability

Amazon Aurora

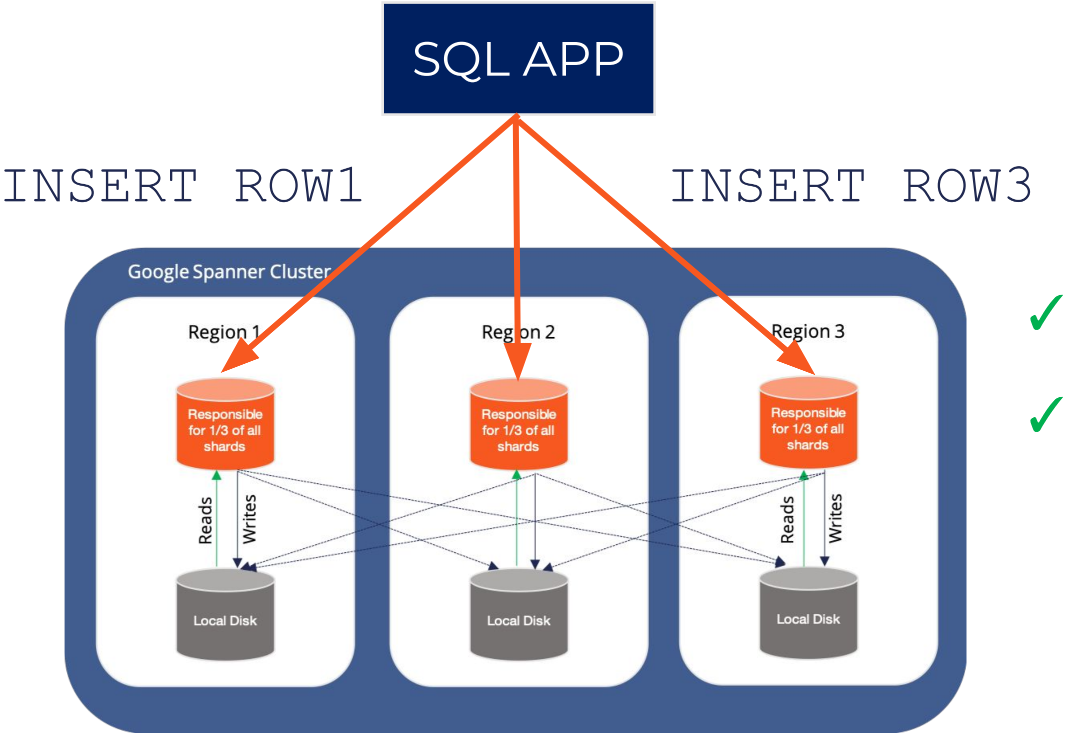
Single Node SQL on Multi-Zone Distributed Storage



- ❌ Add Primary Instances for Write Scaling
- ✓ Add Read Replicas for Read Scaling

Google Spanner

Multi-Node SQL on Multi-Region Distributed Storage



- ✓ Add Primary Instances for Write Scaling
- ✓ Add Read Replicas for Read Scaling

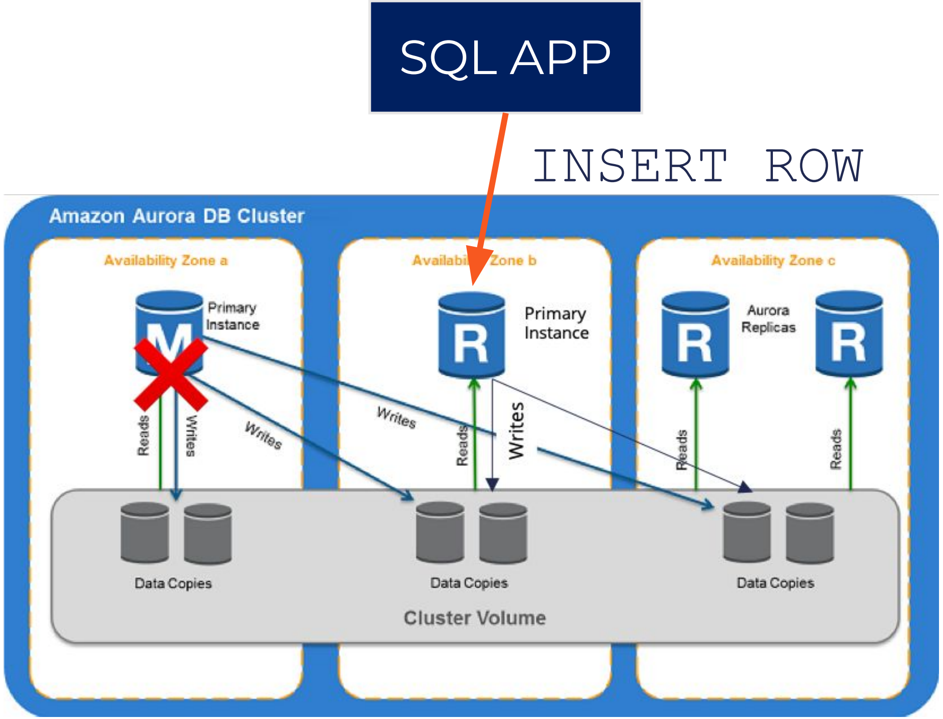
Aurora vs Spanner

Feature	Amazon Aurora	Google Spanner
SQL Features	✓	⚠
Horizontal Write Scalability	✗	✓
Fault Tolerance with HA		
Globally Consistent Writes		
Low Read Latency		

#3 Fault Tolerance with HA

Amazon Aurora

Native Failover & Repair Through Primary Auto Election

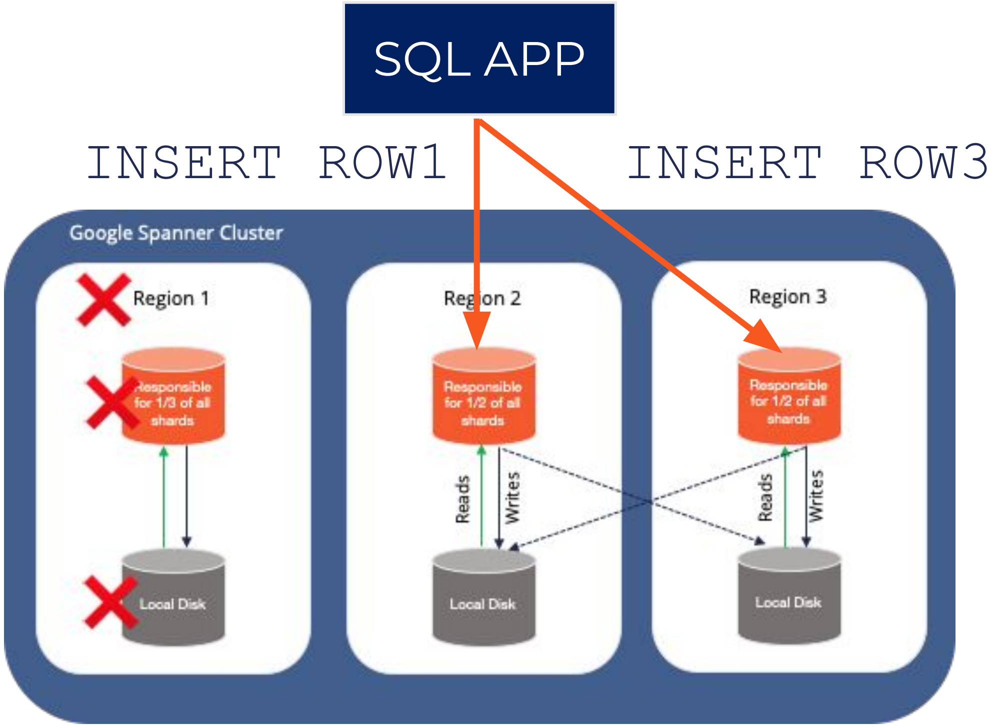


✓ HA When Primary Instance Fails

✓ HA When Read Replica Fails

Google Spanner

Native Failover & Repair Through Shard Leader Auto Election



✓ HA When Any Primary Node Fails

✓ HA When Read Replica Fails

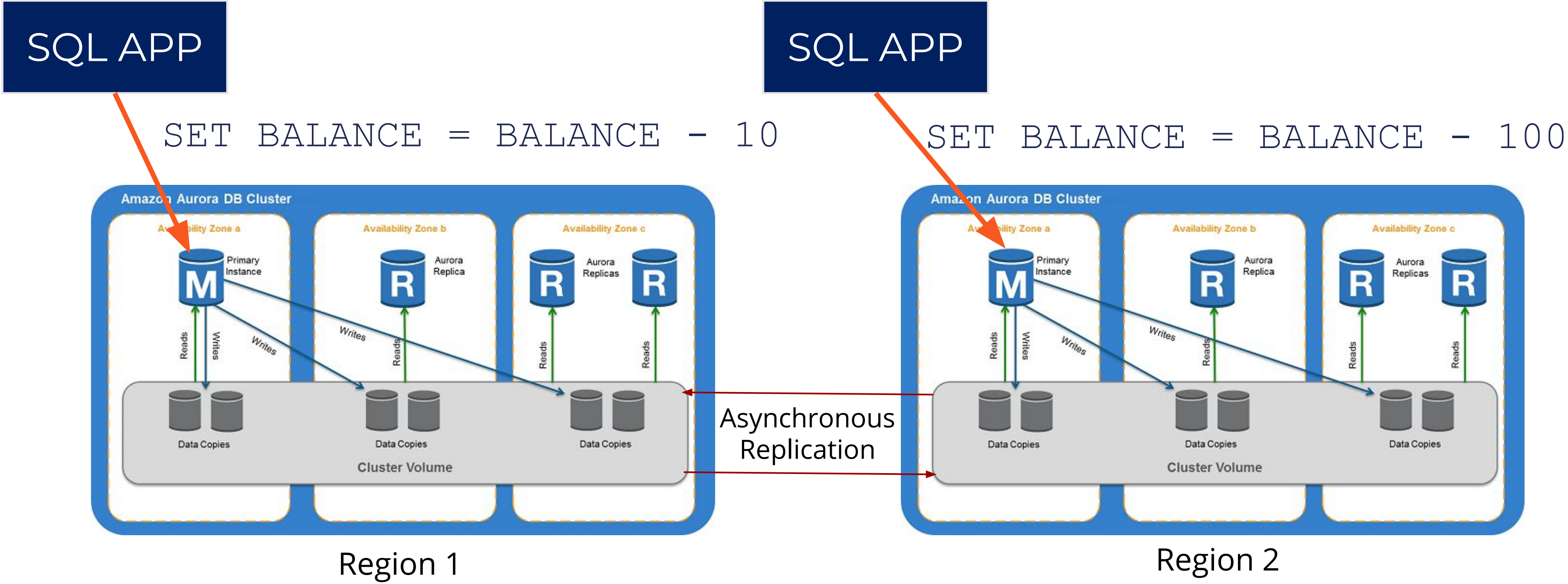
Aurora vs Spanner

Feature	Amazon Aurora	Google Spanner
SQL Features	✓	⚠
Horizontal Write Scalability	✗	✓
Fault Tolerance with HA	✓	✓
Globally Consistent Writes		
Low Read Latency		

#4 Global Write Consistency

Amazon Aurora

Multi-Master Last Writer Wins Conflict Resolution Leads to Inconsistencies



SQL APP

SET BALANCE = BALANCE - 10

SQL APP

SET BALANCE = BALANCE - 100

Asynchronous Replication

Region 1

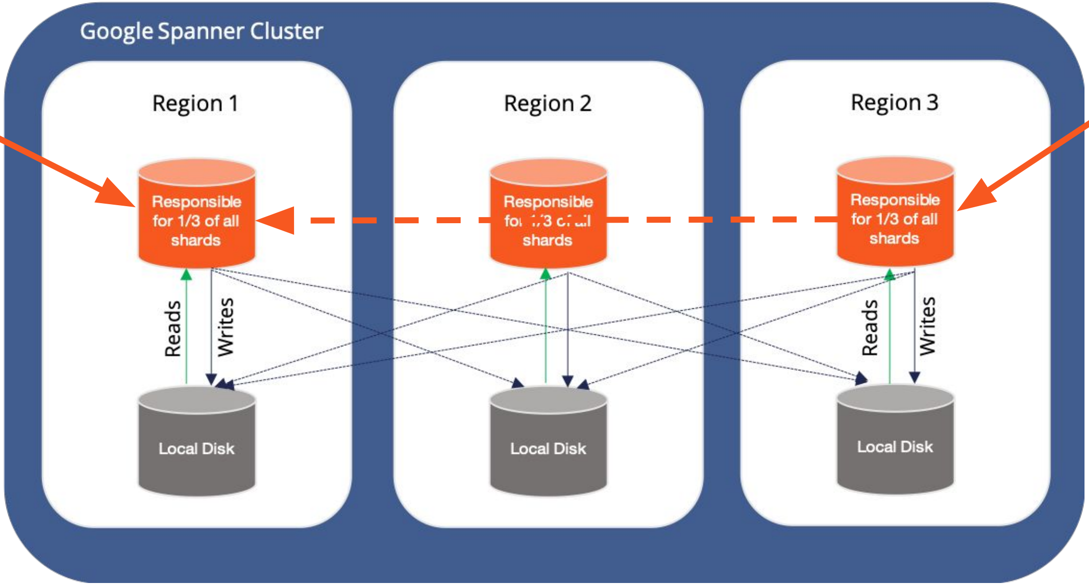
Region 2

Google Spanner

Purpose-Built for Globally Consistent Writes

SQL APP

SET BALANCE = BALANCE - 10


Google Spanner Cluster
Region 1
Responsible for 1/3 of all shards
Reads
Writes
Local Disk
Region 2
Responsible for 1/3 of all shards
Local Disk
Region 3
Responsible for 1/3 of all shards
Reads
Writes
Local Disk

SQL APP

SET BALANCE = BALANCE - 100

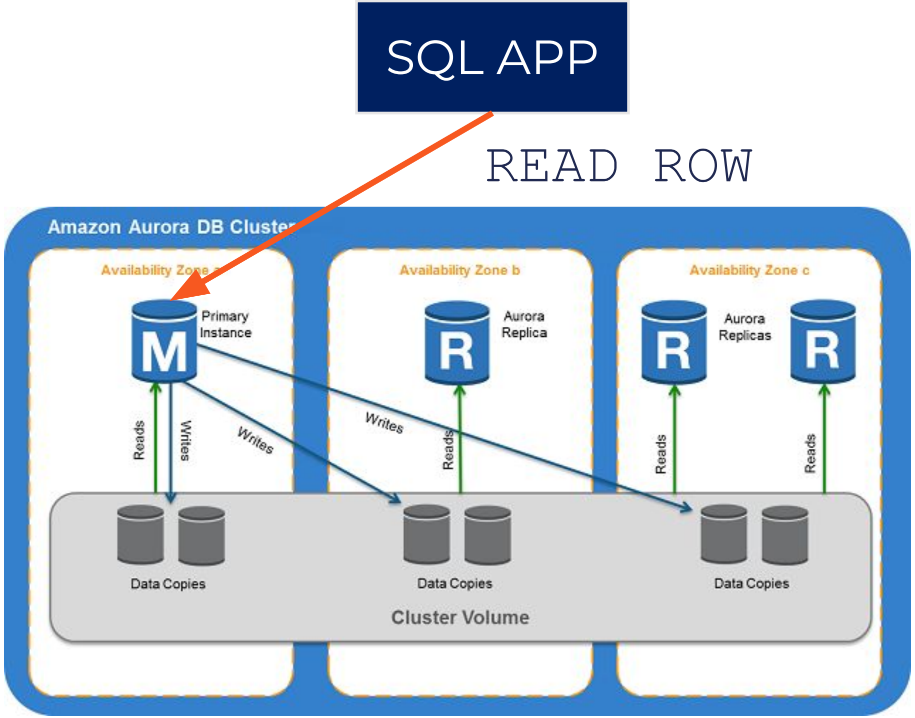
Aurora vs Spanner

Feature	Amazon Aurora	Google Spanner
SQL Features	✓	⚠
Horizontal Write Scalability	❌	✓
Fault Tolerance with HA	✓	✓
Globally Consistent Writes	❌	✓
Low Read Latency		

#5 Low Read Latency

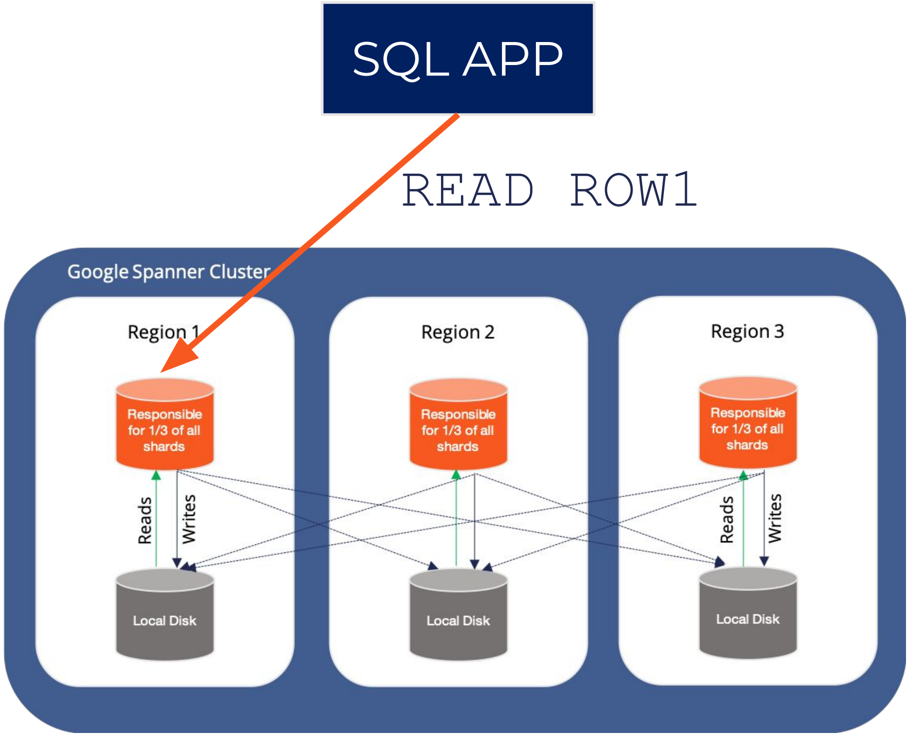
Amazon Aurora

Strongly Consistent Reads Served By Primary Instance



Google Spanner

Strongly Consistent Reads Served By Shard Leaders w/o Read Quorum



Aurora vs Spanner

Feature	Amazon Aurora	Google Spanner
SQL Features	✓	⚠
Horizontal Write Scalability	✗	✓
Fault Tolerance with HA	✓	✓
Global Write Consistency	✗	✓
Low Read Latency	✓	✓

Battle of Architectures - Spanner Beats Aurora

No Performance & Availability Bottlenecks

Scale to Large Clusters while Remaining Highly Available

Built for Geo-Distributed Apps

Future Proofs Data Tier at Global Businesses

Complex to Engineer

Needs Clock Skew Tracking Across Instances

Analyzing Open Source Spanner-Inspired Derivatives

Spanner Brought to Life in Open Source

yugabyteDB

Cockroach DB

No Longer Open Source as of June 2019

TiDB

YugaByte DB Design Principles

- **CP in CAP Theorem**
  - Consistent
  - Partition Tolerant
  - HA on failures
    (new leader elected in seconds)

- **ACID Transactions**
  - Single-row linearizability
  - Multi-row ACID
    - Serializable & Snapshot
    - No bottlenecks even for geo-distributed rows

- **High Performance**
  - All layers in C++ to ensure high perf
  - Run on large memory machines
  - Optimized for SSDs

- **Deploy Anywhere**
  - No IaaS specific dependencies
  - No atomic clocks
  - Bare metal, VM and Kubernetes

Functional Architecture

YSQL

PostgreSQL-Compatible Distributed SQL API

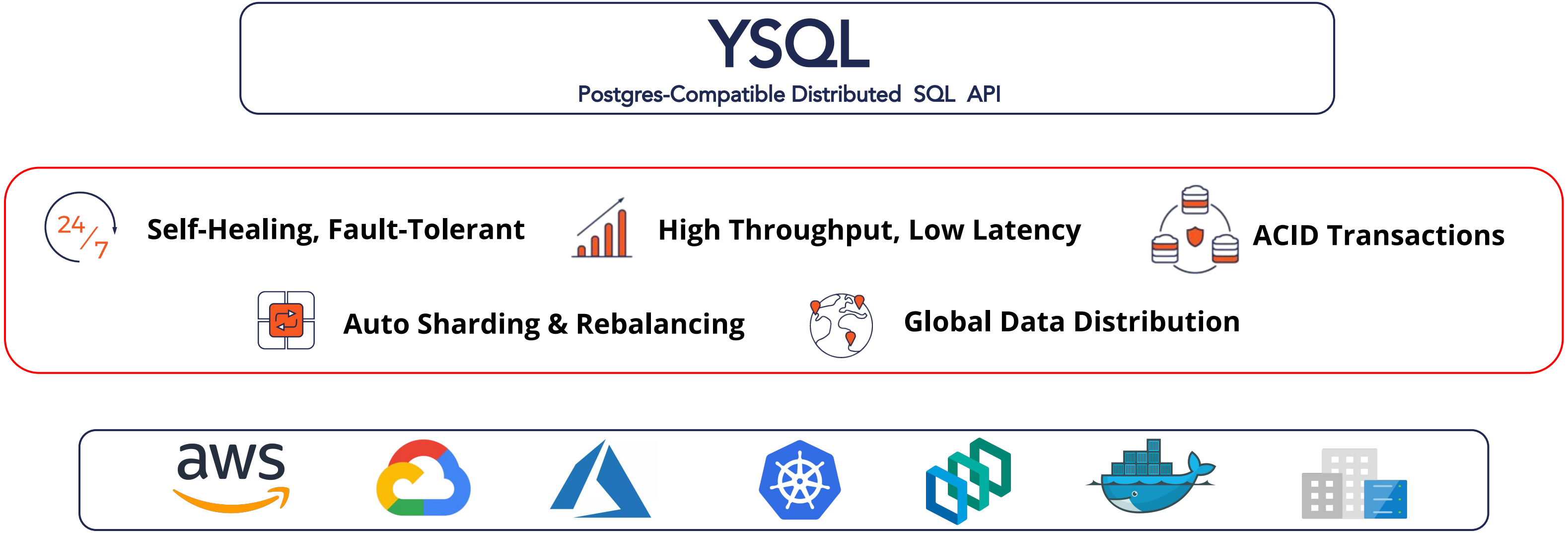
DOCDB

Spanner-Inspired Distributed Document Store

CLOUD NEUTRAL

No Specialized Hardware Needed

Design Follows a Layered Approach



YSQL

Postgres-Compatible Distributed SQL API

Self-Healing, Fault-Tolerant

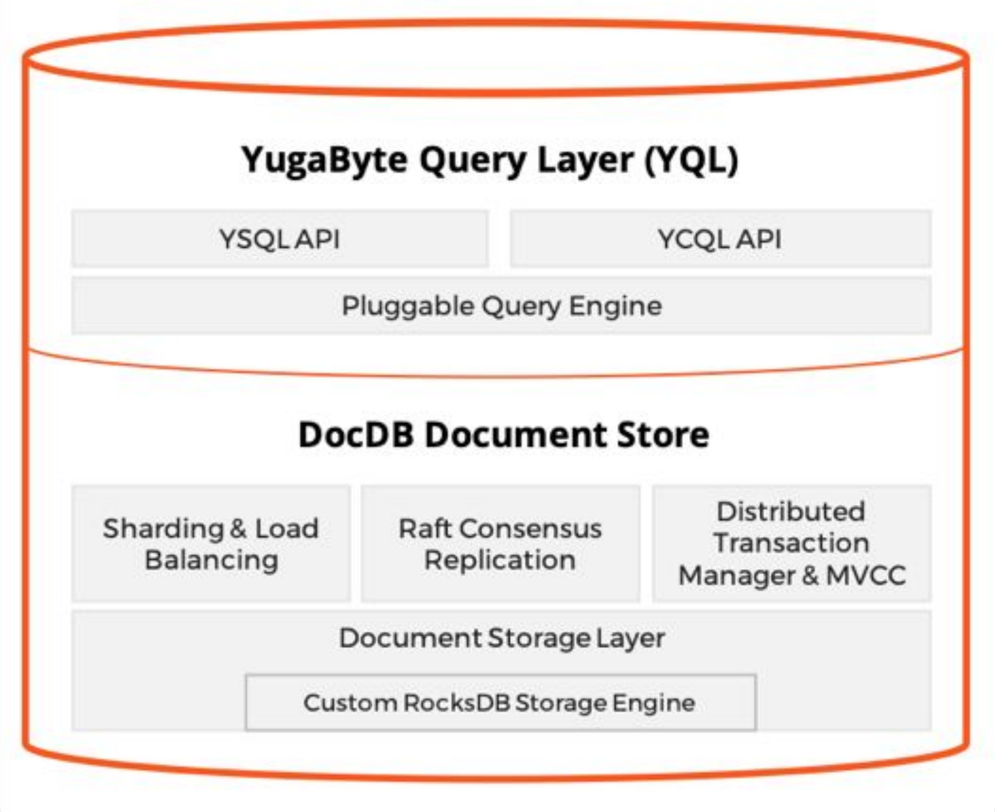
High Throughput, Low Latency

ACID Transactions

Auto Sharding & Rebalancing

Global Data Distribution

YugaByte DB Architecture



YugaByte Query Layer (YQL)

YSQL API

YCQL API

Pluggable Query Engine

DocDB Document Store

Sharding & Load Balancing

Raft Consensus Replication

Distributed Transaction Manager & MVCC

Document Storage Layer

Custom RocksDB Storage Engine

Distributed SQL = Keep 😍 & Remove 😡

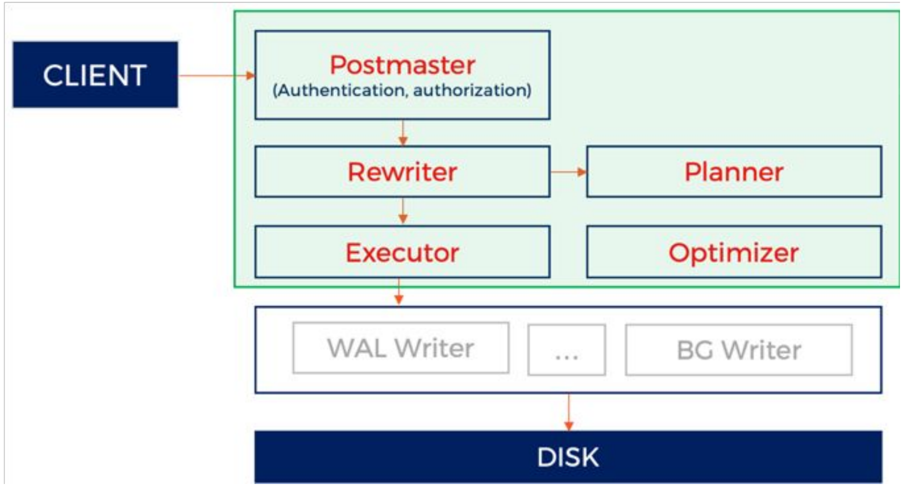
1. SQL Features
2. Replication Protocol
3. Clock Skew Tracking
4. Geo-Distributed Transactions

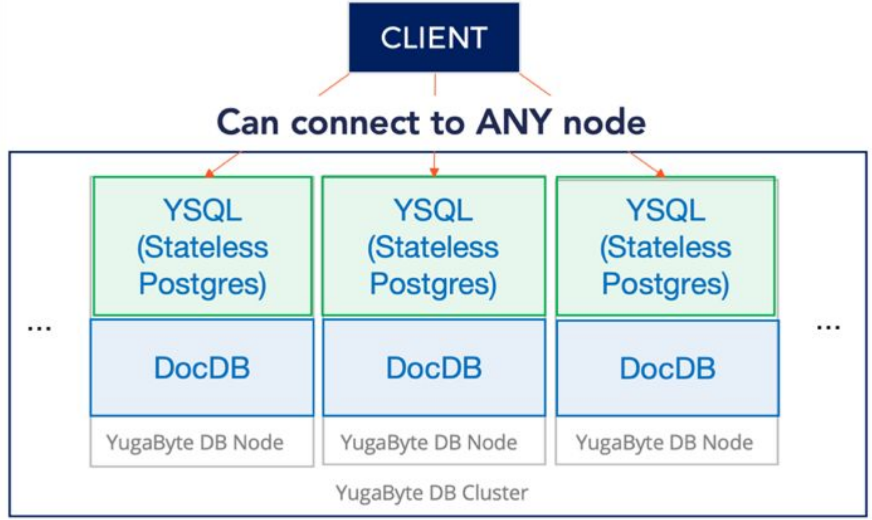
Spanner vs. its Open Source Derivatives

Feature	Google Spanner	YugaByte DB	CockroachDB	TiDB
Cost	Expensive Proprietary	Free Open Source	Free Proprietary	Free Open Source
SQL API Compatibility				
Replication Protocol				
Clock Skew Tracking				
Geo-Distributed Txns				
Tunable Read Latency				
Official Jepsen Tests				

SQL API Compatibility

PostgreSQL Transformed into Distributed SQL





Depth of SQL Support – YugaByte DB

- **SQL Features**
  - Data Types
  - Relational Integrity (Foreign Keys)
  - Built-in Functions
  - Expressions
  - JSON Column Type
  - Secondary Indexes
  - JOINs
  - Transactions
  - Views

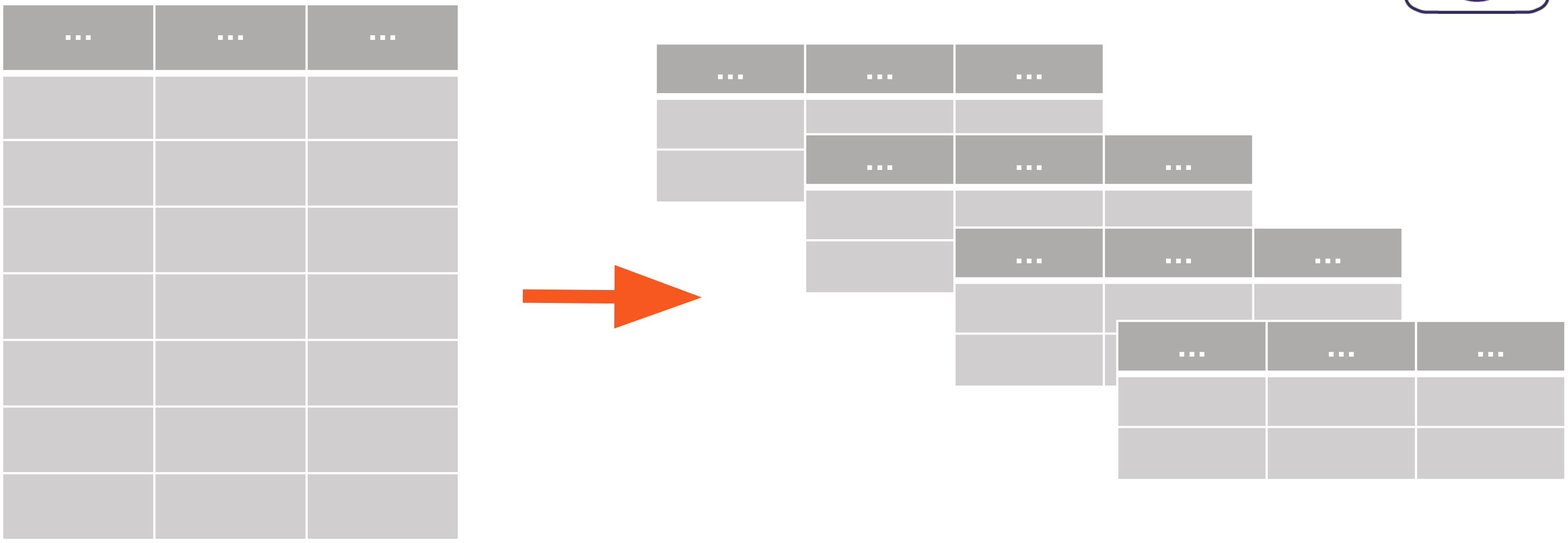
- **Advanced SQL Features**
  - Partial Indexes
  - Stored Procedures
  - Triggers
  - And more ...

Spanner vs. its Open Source Derivatives

Feature	Google Spanner	YugaByte DB	CockroachDB	TiDB
Cost	Expensive Proprietary	Free Open Source	Free Proprietary	Free Open Source
SQL API Compatibility	Proprietary	PostgreSQL	PostgreSQL No Stored Procedures	MySQL No Foreign Keys
Replication Protocol				
Clock Skew Tracking				
Transaction Manager				
Tunable Read Latency				
Official Jepsen Tests				

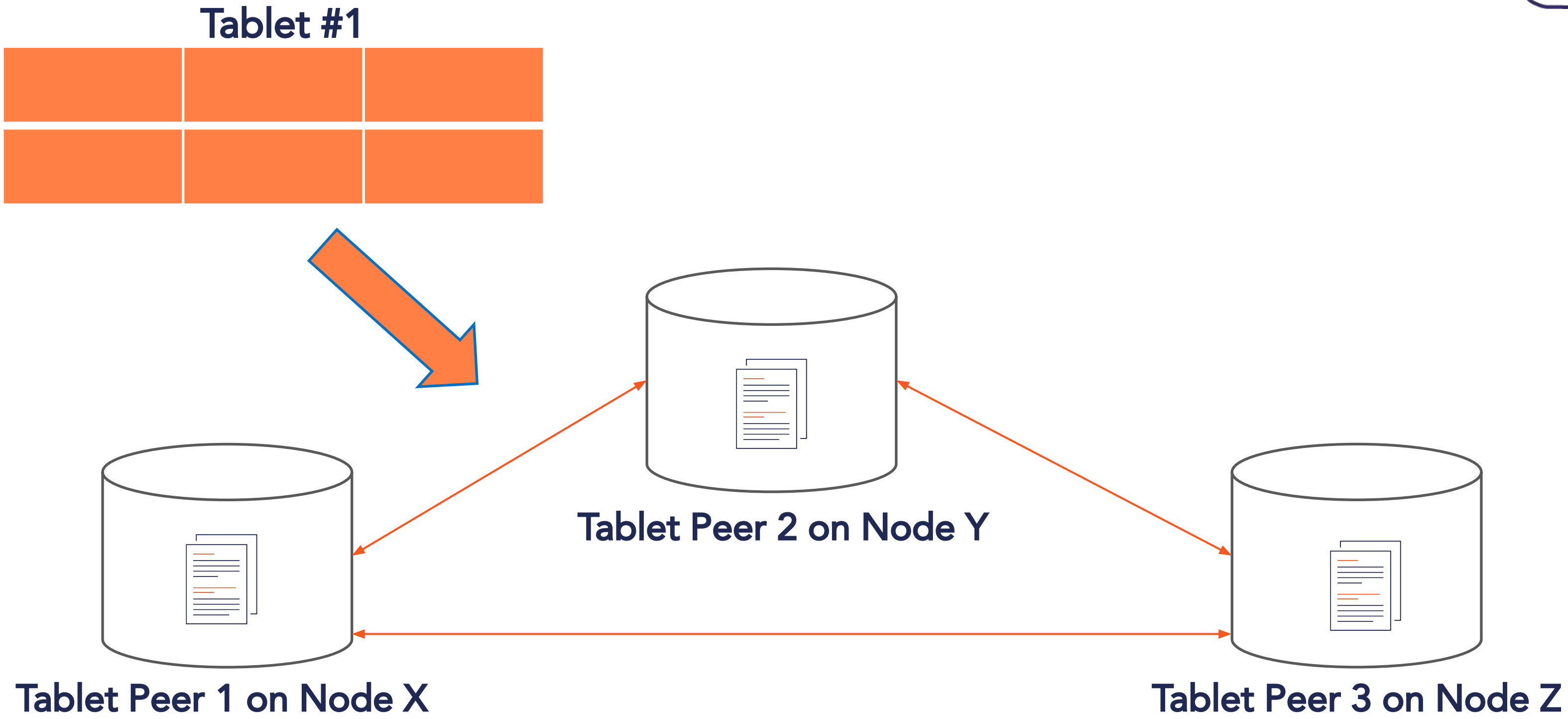
Replication Protocol

Every Table is Automatically Sharded

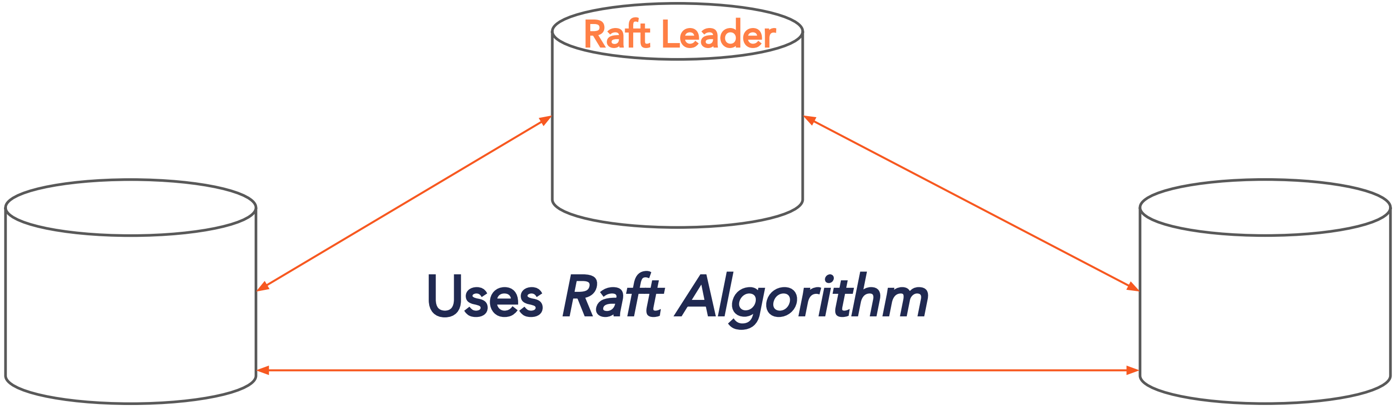


SHARDING = **AUTOMATIC PARTITIONING OF TABLES**

Replication Done at Shard Level



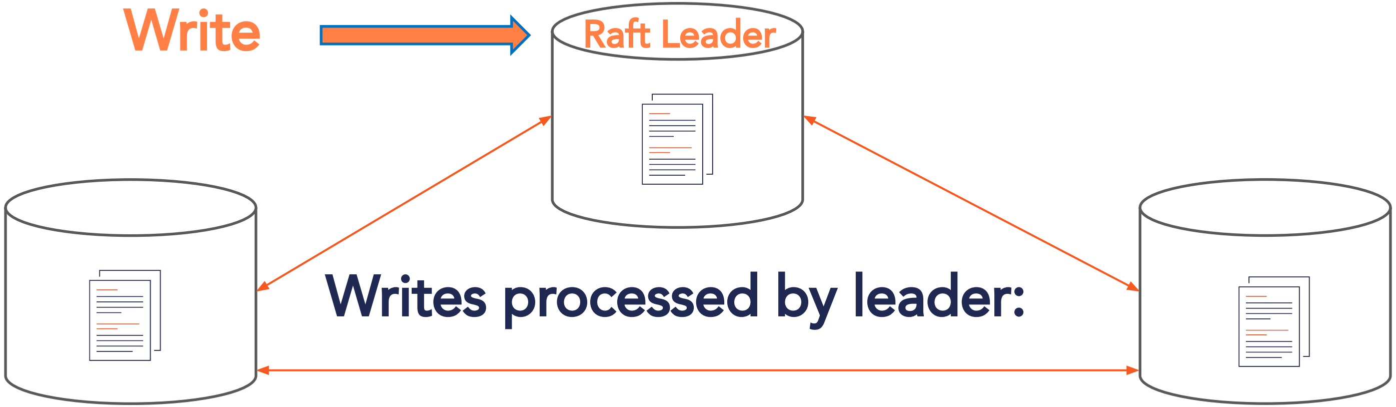
Replication uses a Consensus algorithm



Uses *Raft Algorithm*

First elect Tablet Leader

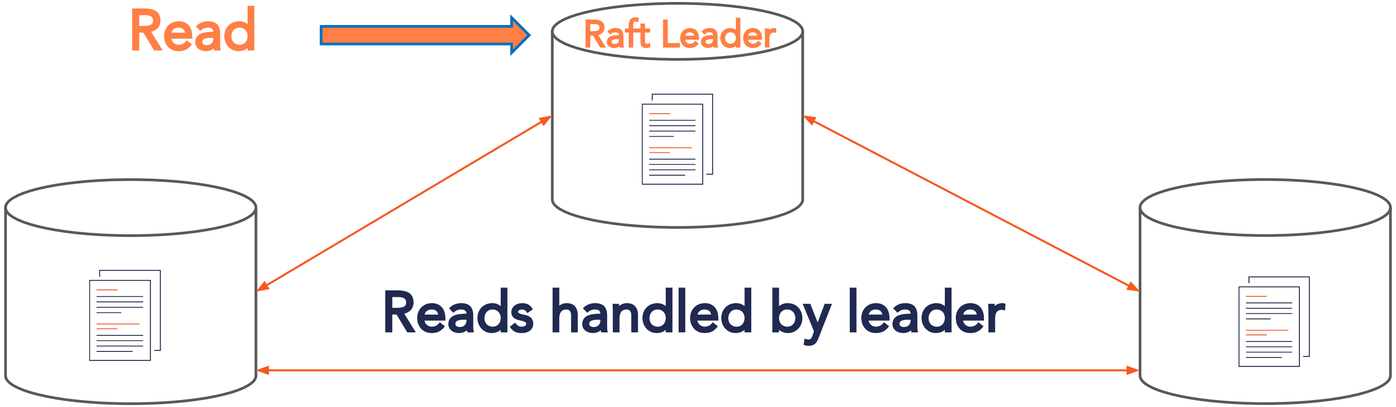
Writes in Raft Consensus



Writes processed by leader:

Send writes to all peers
Wait for majority to ack

Reads in Raft Consensus



Reads handled by leader

Uses *Leader Leases* for performance

Spanner vs. its Open Source Derivatives

Feature	Google Spanner	YugaByte DB	CockroachDB	TiDB
Cost	Expensive Proprietary	Free Open Source	Free Proprietary	Free Open Source
SQL API Compatibility	Proprietary	PostgreSQL	PostgreSQL No Stored Procedures	MySQL No Foreign Keys
Replication Protocol	Paxos	Raft	Raft	Raft
Clock Skew Tracking				
Geo-Distributed Txns				
Tunable Read Latency				
Official Jepsen Tests				

Transactions & Clock Skew Tracking

Distributed (aka Multi-Shard) Transactions

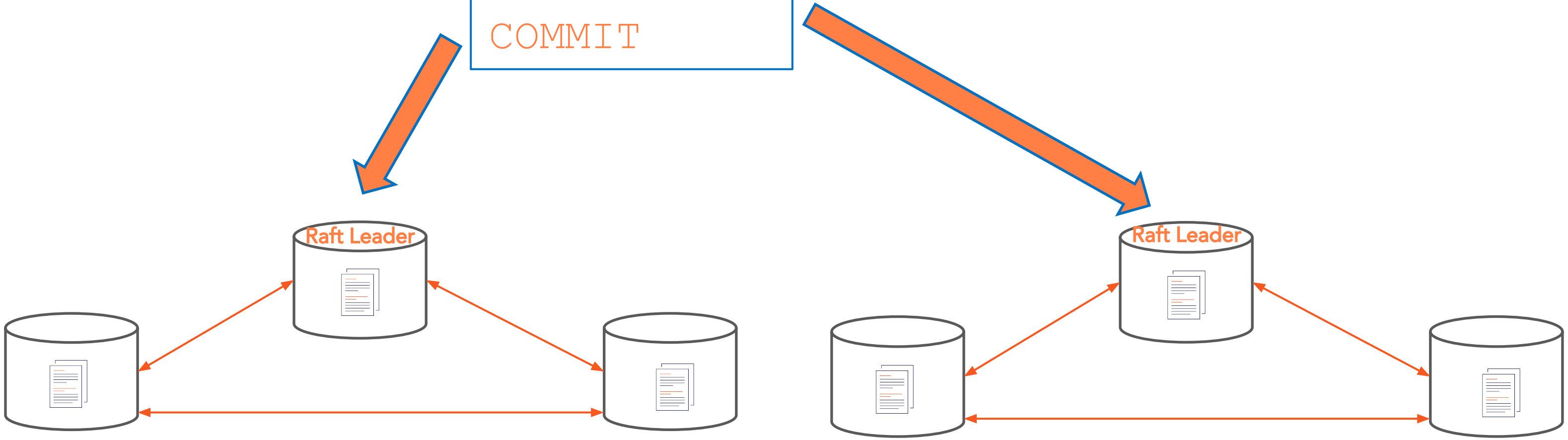
```
BEGIN TXN
  UPDATE k1
  UPDATE k2
COMMIT
```

k1 and k2 may belong to **different shards**

Belong to **different Raft groups** on completely **different nodes**

What do Distributed Transactions need?

```
BEGIN TXN
   UPDATE k1
   UPDATE k2
COMMIT
```



Updates should get written at the same physical time

But how will nodes agree on time?

Use a Physical Clock

You would need an Atomic Clock or two lying around

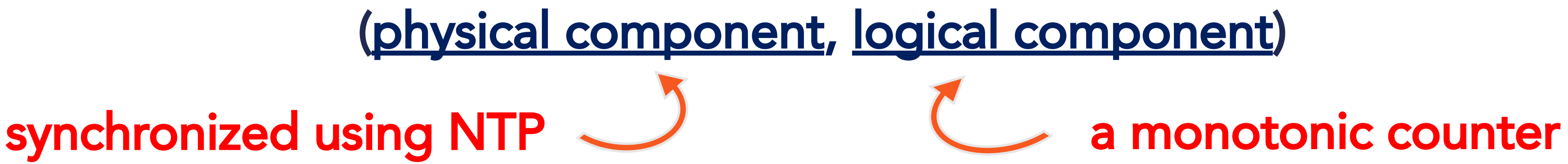
Atomic Clocks are highly available,
globally synchronized clocks with tight error bounds

Jeez! I’m fresh out of those.

Most of my physical clocks are never synchronized

Hybrid Logical Clock (HLC)

Combine coarsely-synchronized physical clocks with Lamport Clocks to track causal relationships



(physical component, logical component)

synchronized using NTP

a monotonic counter

Nodes update HLC on each Raft exchange for things like heartbeats, leader election and data replication

Spanner vs. its Open Source Derivatives

Feature	Google Spanner	YugaByte DB	CockroachDB	TiDB
Cost	Expensive Proprietary	Free Open Source	Free Proprietary	Free Open Source
SQL API Compatibility	Proprietary	PostgreSQL	PostgreSQL No Stored Procedures	MySQL No Foreign Keys
Replication Protocol	Paxos	Raft	Raft	Raft
Clock Skew Tracking	TrueTime Atomic Clock	Hybrid Logical Clock + Max Clock Skew	Hybrid Logical Clock + Max Clock Skew	Single Timestamp Gen ⇒ No Tracking Needed
Geo-Distributed Txns	✓	✓	✓	Not Recommended Given Single (Region) Timestamp Generator
Tunable Read Latency				
Official Jepsen Tests				

Miscellaneous

Jepsen Testing

Jepsen is an effort to improve the safety of distributed databases, queues, consensus systems, etc. led by Kyle Kingsbury

“YugaByte DB now passes tests for snapshot isolation, linearizable counters, sets, registers, and systems of registers, as long as clocks are well-synchronized”

Jepsen YugaByte DB Analysis:
https://jepsen.io/analyses/yugabyte-db-1.1.9

Spanner vs. its Open Source Derivatives

Feature	Google Spanner	YugaByte DB	CockroachDB	TiDB
Cost	Expensive Proprietary	Free Open Source	Free Proprietary	Free Open Source
SQL API Compatibility	Proprietary	PostgreSQL	PostgreSQL No Stored Procedures	MySQL No Foreign Keys
Replication Protocol	Paxos	Raft	Raft	Raft
Clock Skew Tracking	TrueTime Atomic Clock	Hybrid Logical Clock + Max Clock Skew	Hybrid Logical Clock + Max Clock Skew	Single Timestamp Gen ⇒ No Tracking
Geo-Distributed Txns	✓	✓	✓	Not Recommended Given Single (Region) Timestamp Generator
Tunable Read Latency	✓	✓	✗	✗
Official Jepsen Tests	**Unknown**	✓	✓	✓

yugabyteDB presents:

Distributed SQL Summit

September 20, 2019 Hilton, San Jose, CA

A full day of talks from experts on what it takes to build, deploy and scale distributed SQL databases in the cloud and on Kubernetes

Hear from the creators of Google Spanner, Amazon Aurora, Facebook DBs & YugaByte DB

Register today at distributedsql.org

Read more at blog.yugabyte.com

Storage Layer

blog.yugabyte.com/distributed-postgresql-on-a-google-spanner-architecture-storage-layer

Query Layer

blog.yugabyte.com/distributed-postgresql-on-a-google-spanner-architecture-query-layer

Questions?

Download
download.yugabyte.com

Join Slack Discussions
yugabyte.com/slack

Star on GitHub
github.com/YugaByte/yugabyte-db

Relevant Research Publications

- Bigtable:
  http://static.googleusercontent.com/media/research.google.com/en//archive/bigtable-osdi06.pdf
- Spanner:
  - http://static.googleusercontent.com/media/research.google.com/en//archive/spanner-osdi2012.pdf
- Megastore:
  - http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/36971.pdf
- Raft algorithm
  - http://ramcloud.stanford.edu/raft.pdf
  - https://raft.gixthub.io/
  - http://openlife.cc/system/files/3-modifications-for-Raft-consensus.pdf

Relevant Research Publications

- David Alves, Todd Lipcon, Vijay Garg. **Technical Report: HybridTime - Accessible Global Consistency with High Clock Uncertainty.** http://pdsl.ece.utexas.edu/david/hybrid-time-tech-report-01.pdf
- Sandeep Kulkarni, Murat Demirbas, Deepak Madeppa, Bharadwaj Avva, and Marcelo Leone. **Logical Physical Clocks and Consistent Snapshots in Globally Distributed Databases.** http://www.cse.buffalo.edu/tech-reports/2014-04.pdf
- Michael J. Cahill, Uwe Röhm, Alan D. Fekete. **Serializable Isolation for Snapshot Databases** (2008). https://courses.cs.washington.edu/courses/cse444/08au/544M/READING-LIST/fekete-sigmod2008.pdf
- Murat Demirbas, Sandeep Kulkarni. **Beyond TrueTime: Using AugmentedTime for Improving Spanner.** http://www.cse.buffalo.edu/~demirbas/publications/augmentedTime.pdf
- Dahlia Malkhi Jean-Philippe Martin. **Spanner's Concurrency Control.** (2) Ittay Eyal. **Fault Tolerant Transaction Architectures** https://www.cs.cornell.edu/~ie53/publications/DC-col51-Sep13.pdf

Relevant Research Publications

- Coordination Avoidance in Database Systems Peter Bailis, Alan Fekete, Michael J. Franklin, Ali Ghodsi, Joseph M. Hellerstein, Ion Stoica http://www.bailis.org/papers/ca-vldb2015.pdf

- RocksDB - RocksDB: A High Performance Embedded Key-Value Store for Flash Storage - Data@Scale , https://www.youtube.com/watch?v=plqVp_OnSzg

- Schema-Agnostic Indexing with Azure DocumentDB (VLDB paper): paper describes the Microsoft Azure's DocumentDB capabilities, including document representation, query language, document indexing approach, core index support, and early production experiences

- MergeOperator on RocksDB - https://github.com/facebook/rocksdb/wiki/Merge-Operator-Implementation

- Cluster scheduling blog post from cambridge: http://www.cl.cam.ac.uk/research/srg/netos/camsas/blog/2016-03-09-scheduler-architectures.html