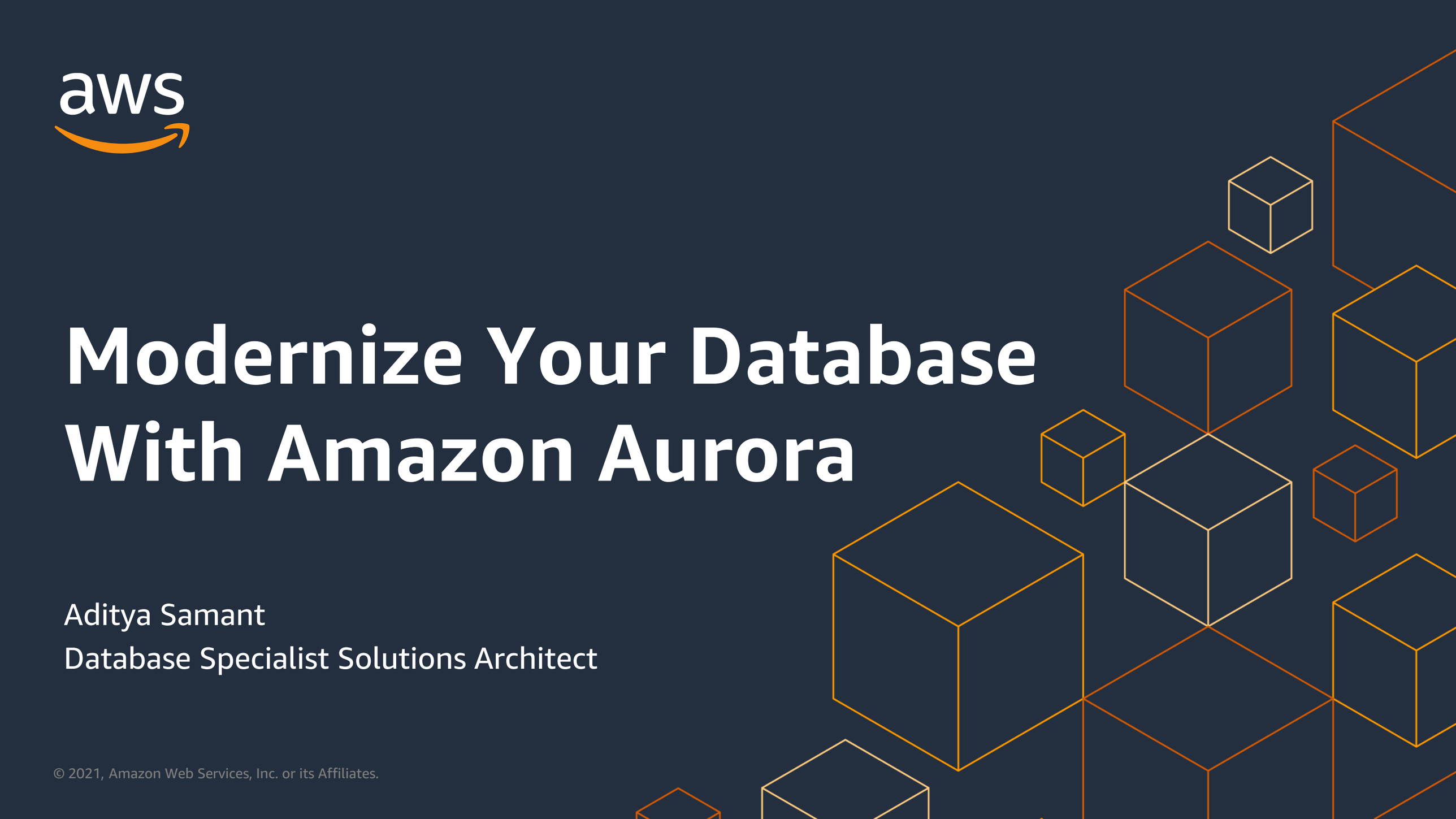


# Modernize Your Database With Amazon Aurora

Aditya Samant
Database Specialist Solutions Architect

# Agenda

- What is Amazon Aurora?
- Distributed Storage fundamentals
- Aurora Key Innovations
- Monitoring performance
- Minimizing Disruptions
- What’s new?
- What’s coming?
- Summary



aws

# A cost-effective, enterprise database…

Amazon Aurora - enterprise database
at open source price, delivered as a managed service


**Speed** and **availability** of high-end commercial databases

**Simplicity** and **cost-effectiveness** of open source databases

Drop-in **compatibility** with MySQL and PostgreSQL

Simple **pay as you go** pricing



aws

# Re-imagining databases for the cloud…

1. Scale-out, distributed, design
2. Service-oriented architecture using AWS services
3. Fully managed service, automating administrative tasks



aws

# Aurora Customer Adoption

**Fastest growing service in AWS history**

**Aurora is used by ¾ of the top 100 AWS customers**




aws

# Amazon Aurora Fundamentals

## Storage System and Cluster Architecture



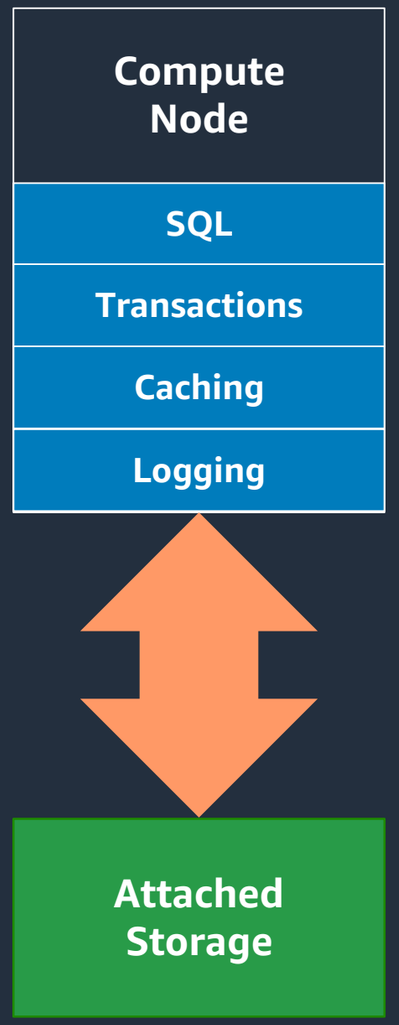
aws

# Traditional Database Architecture

**Databases are all about I/O…**

Design principles over the last 40+ years:

- **Increase** I/O bandwidth
- **Decrease** number of I/Os consumed





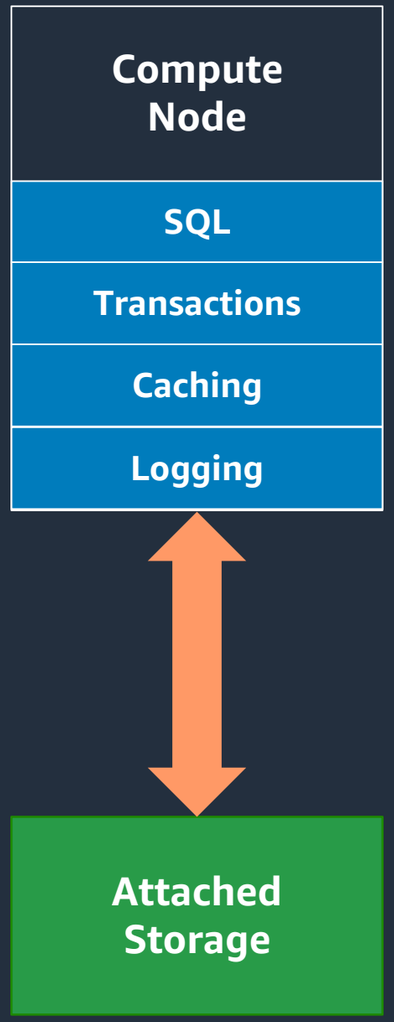

aws

# Traditional Database Architecture… in the Cloud

## Compute and storage have different lifetimes

- Instances fail and may be replaced
- Instances are shut down
- Instances are scaled up/down
- Instances are added to cluster to scale out

Compute and storage are best **decoupled** for scalability, availability and durability






aws

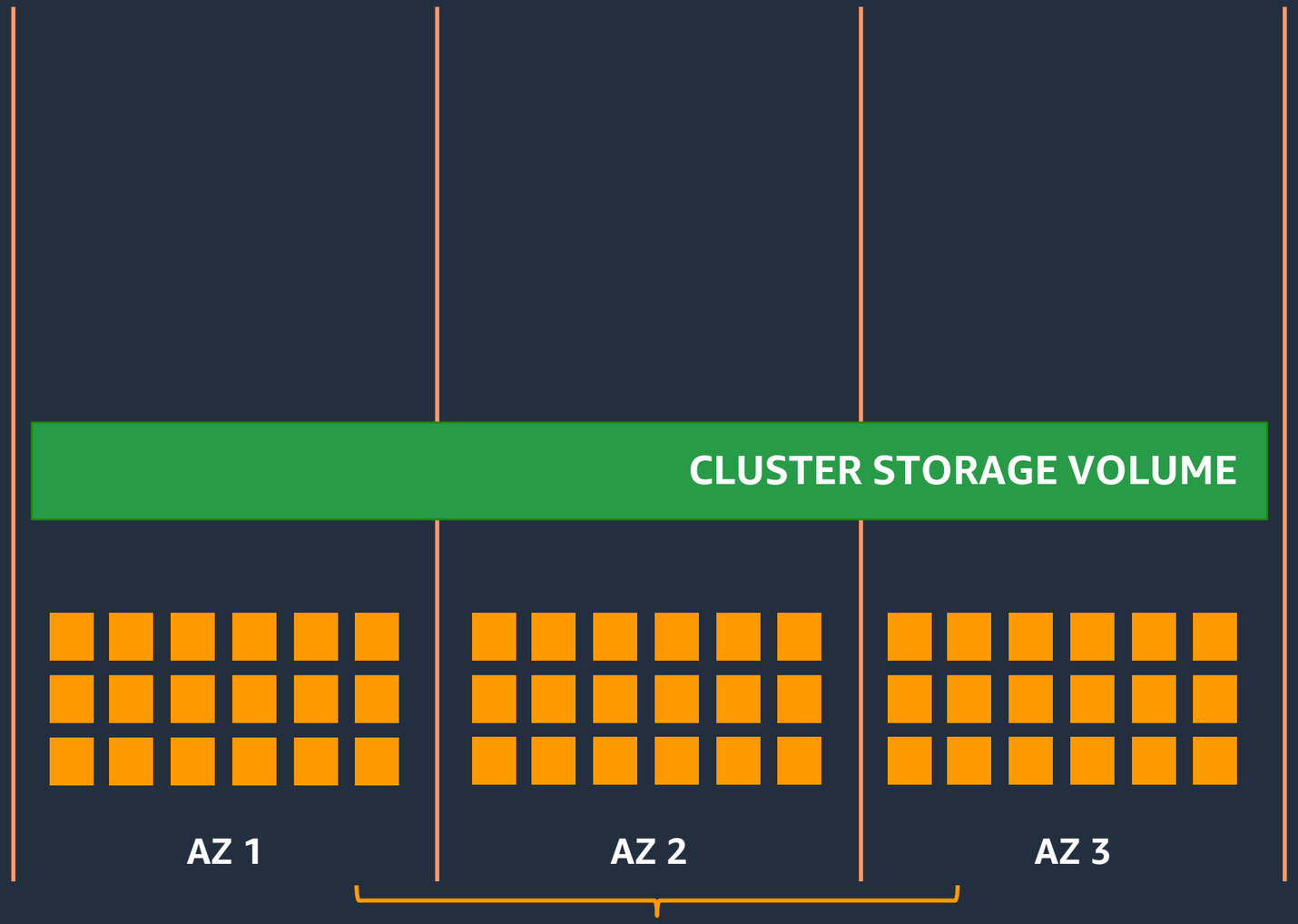
# Scale-out, distributed, multi-tenant storage architecture

Purpose-built log-structured distributed storage

Storage volume is striped across hundreds of storage nodes

Storage nodes with locally attached SSDs

Continuous backup to Amazon S3.



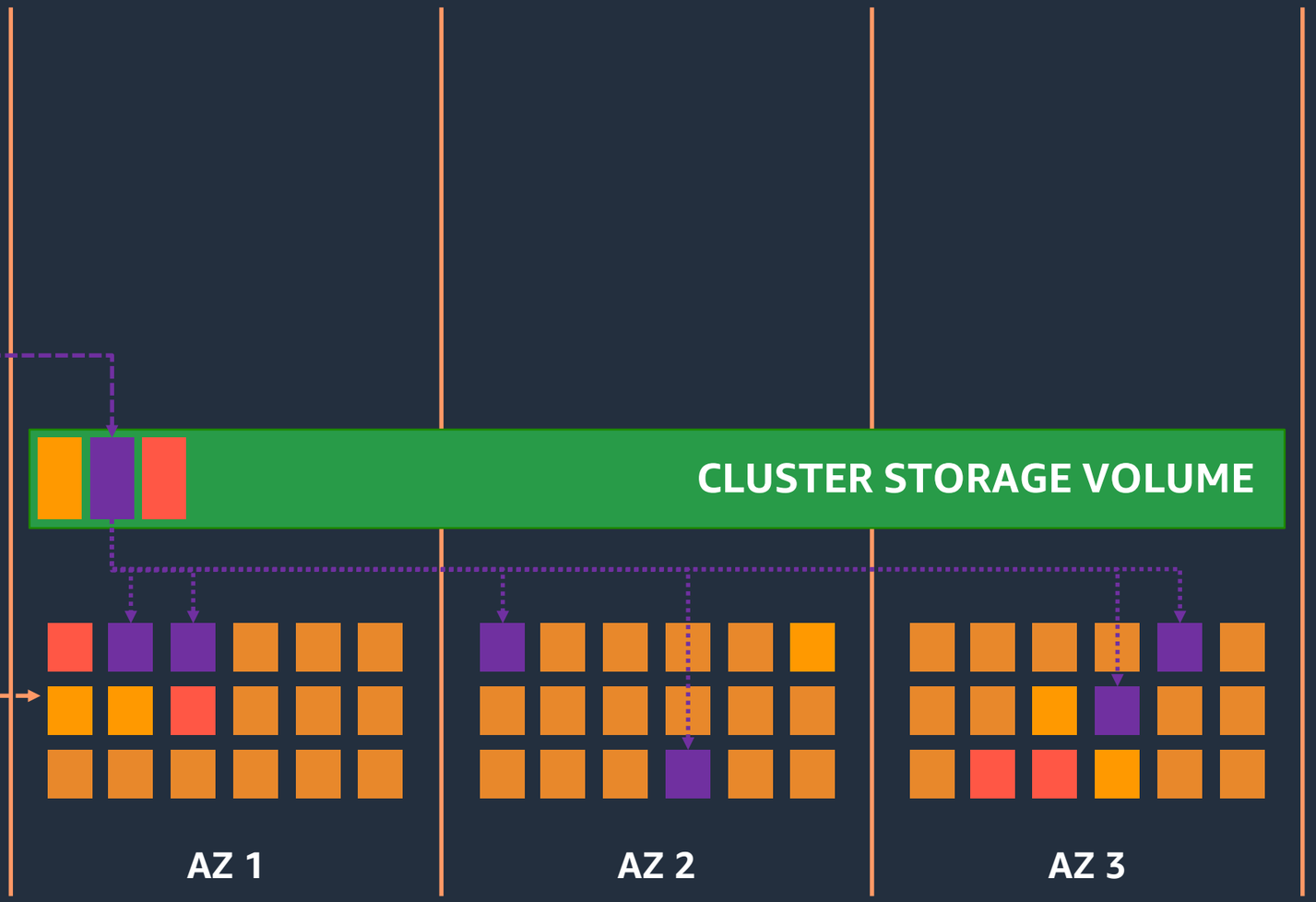
# Scale-out, distributed, multi-tenant storage architecture

Six copies of data, two in each Availability Zone to protect against AZ+1 failure modes

Storage volume segmented in 10 GB **protection groups (PG)**

Each PG contains six 10 GB **segments**, copies of the same data on different storage nodes, two in each AZ.

Storage volume grows automatically by adding PGs, up to 128 TB






aws

# High durability storage system, tolerant of AZ+1 failures

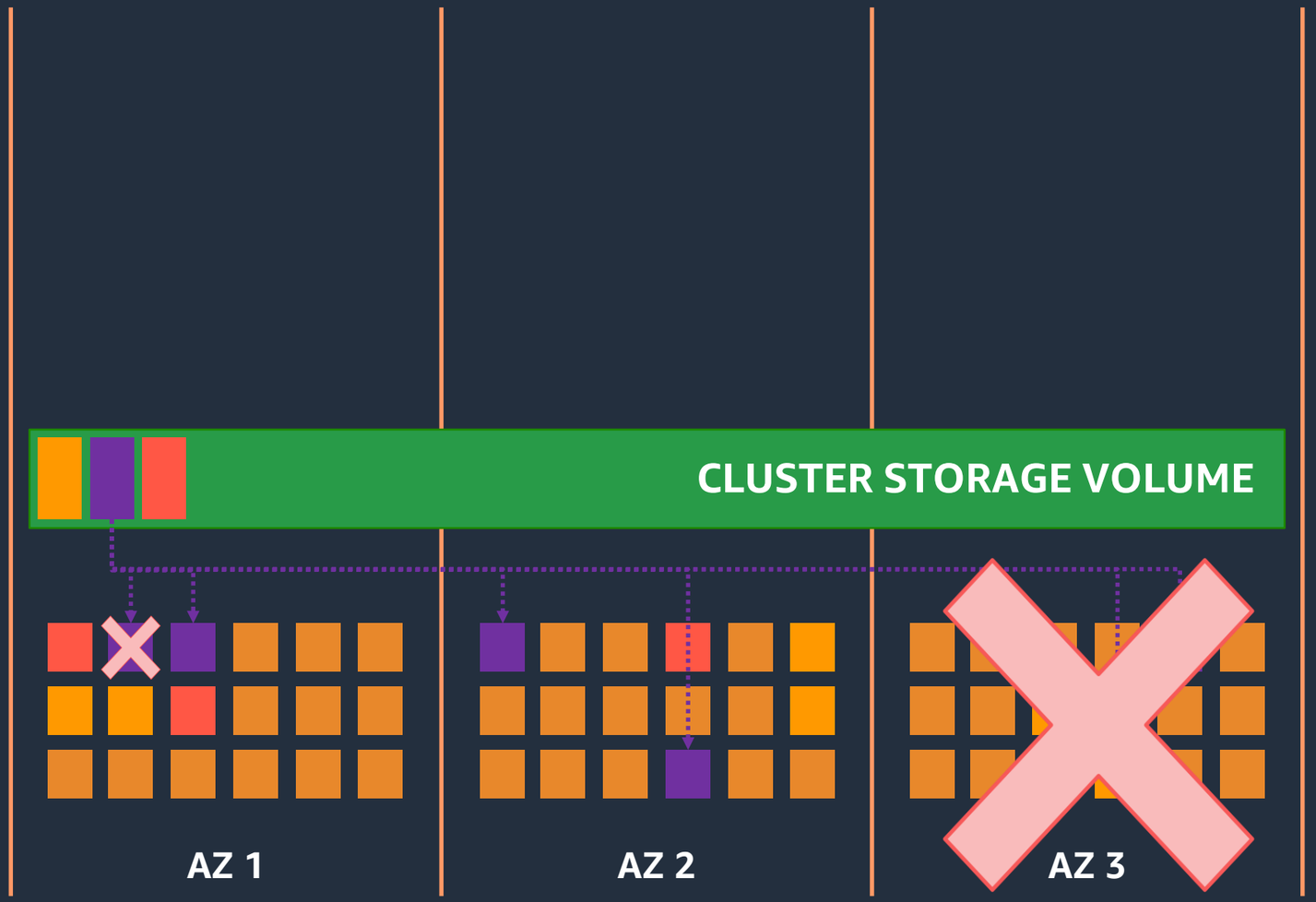
Using quorum model for writes and reads:

- 4 out of 6 for writes
- 3 out of 6 for reads (recovery)

Maintains write capability if an AZ fails, maintains read capability if AZ + 1 storage node fails.

Self-healing architecture rebalances hot storage nodes, rebuilds segments from failed hardware

Peer to peer “gossip protocol” is used for repairs






aws

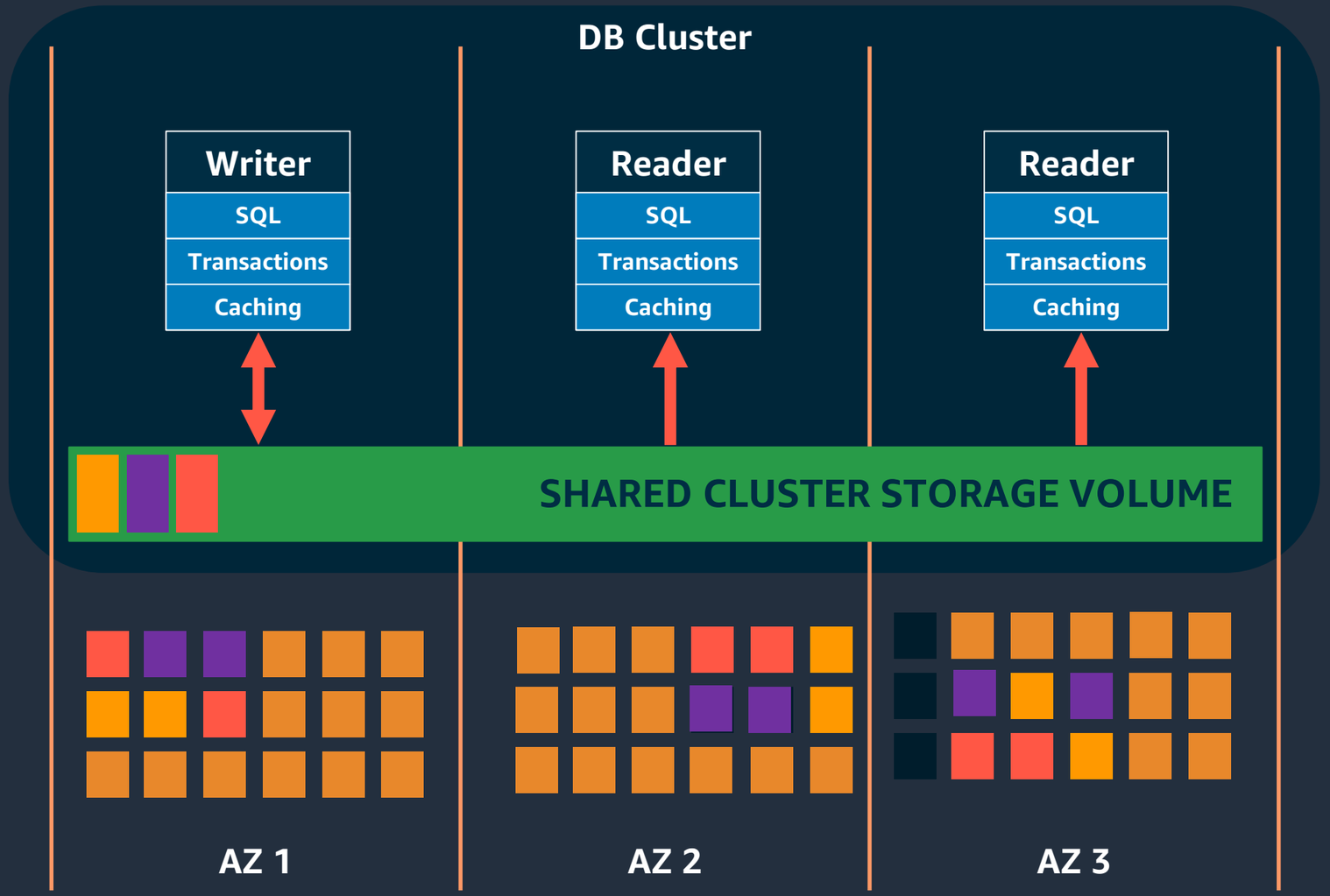
# Amazon Aurora cluster topology

Up to 16 DB instances/nodes in a **regional** cluster, spanning multiple AZs

One (first) is always the writer node.

Storage volume shared with readers. Readers open volume in read only mode (MySQL: **innodb_read_only = 1** PostgreSQL: **transaction_read_only=on**).

Low reader lag < ~20ms






aws

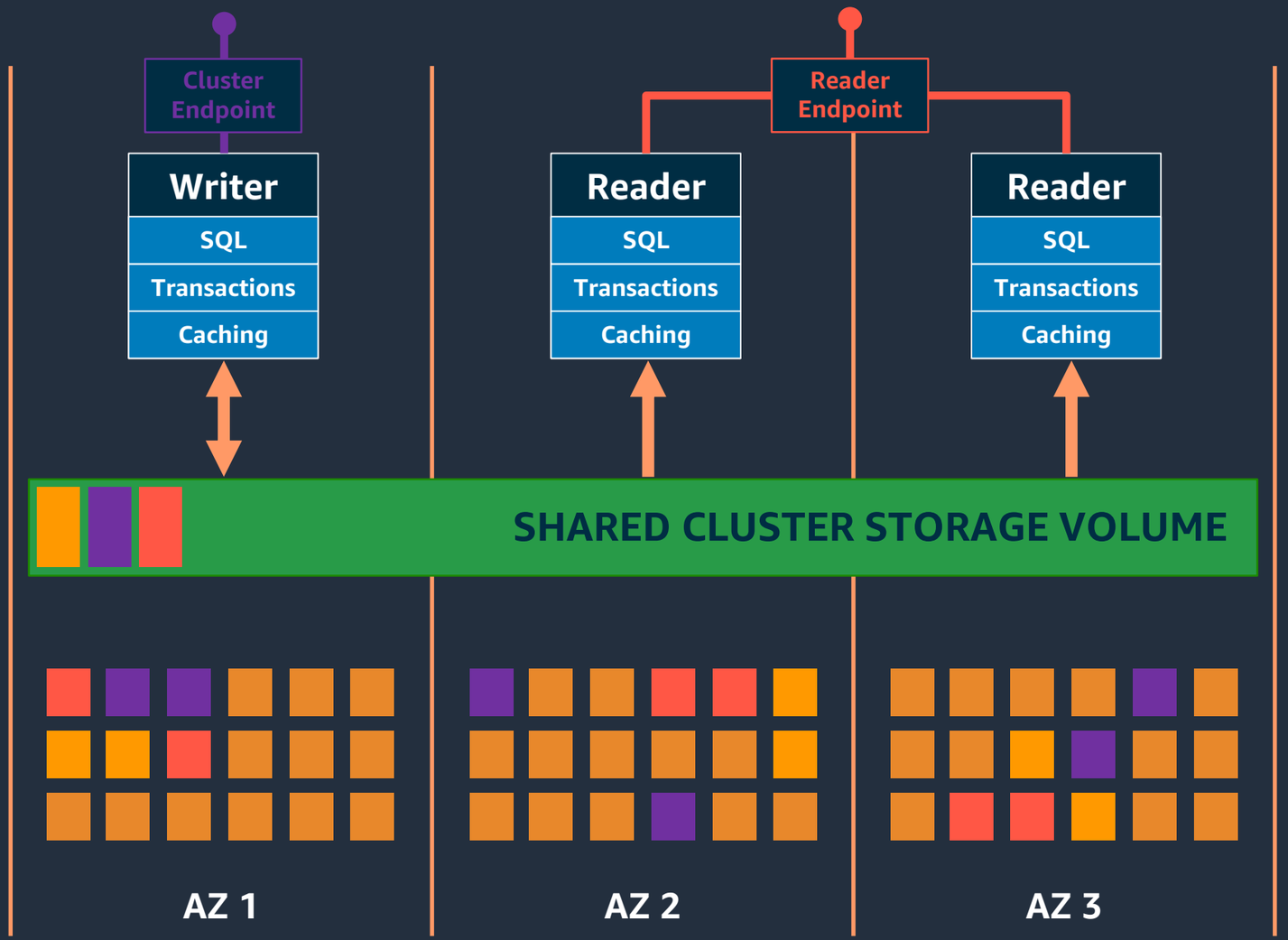
# Accessing your Aurora databases

Managed DB service, no OS or filesystem level access

Connect to writer using **Cluster (DNS) Endpoint** – always points to writer!

Round robin load balancing for reads using **Reader (DNS) Endpoint** (excludes writer except on single node clusters)

**Custom (DNS) Endpoints, read replica auto scaling** supported as well






aws

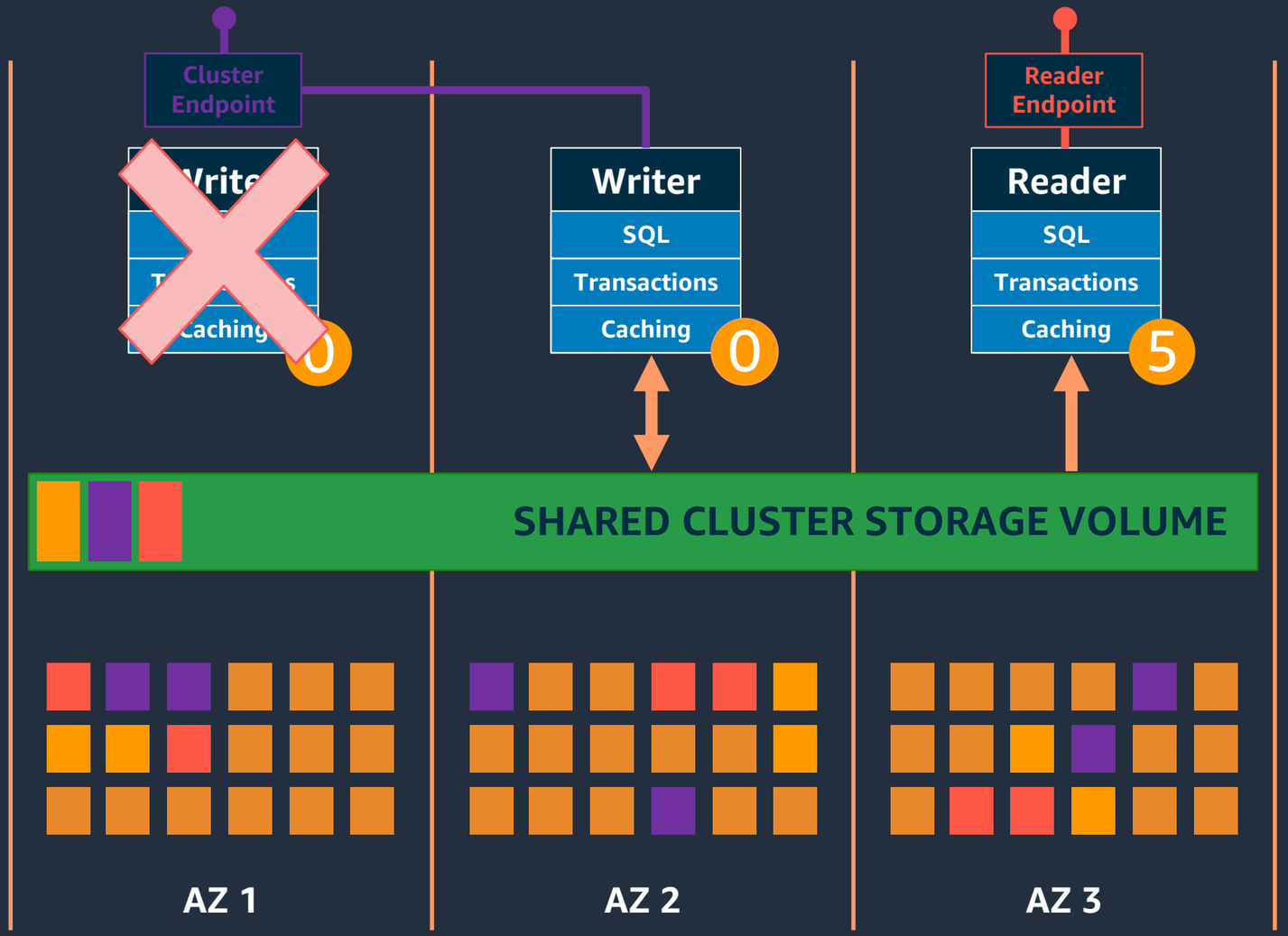
# Tolerating compute failures

Any reader node can be promoted to writer/master

**Failover tier** determines preference on failover reader candidates. Lower values more preferred.

Failed instances/nodes will be replaced after failover and come online as readers.

Readers reboot on writer failover.

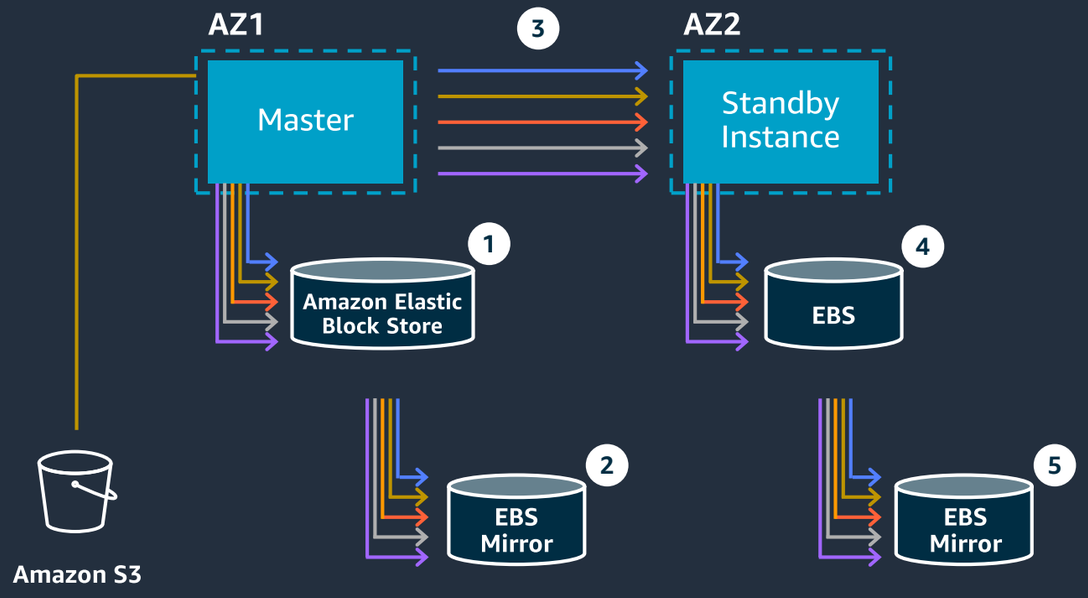




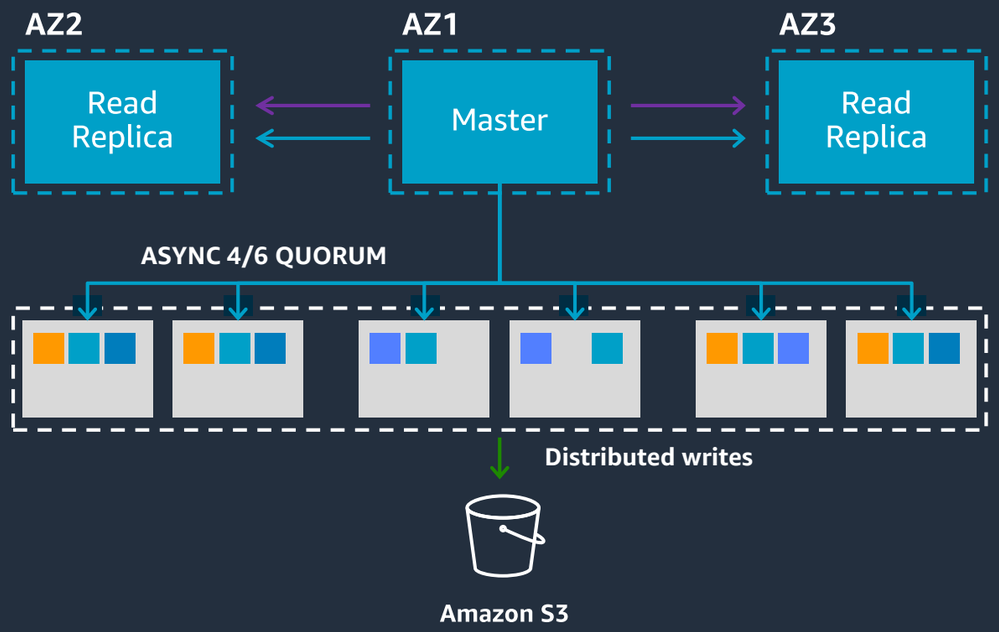

aws

# Aurora I/O profile compared (MySQL)

## Amazon RDS MySQL Multi-AZ




## Amazon Aurora MySQL




**MySQL I/O profile for 30 min Sysbench run**

- 780K transactions
- Average 7.4 I/Os per transaction

**Aurora IO profile for 30 min Sysbench run**

- 27,378K transactions – **35x more**
- 0.95 I/Os per transaction (6x amplification) – **7.7x less**

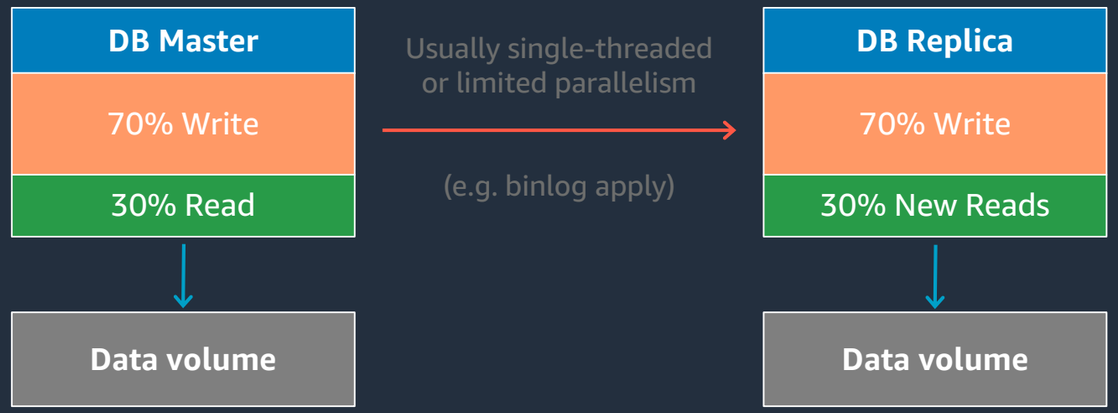
Log Binlog Data Double-write .frm files



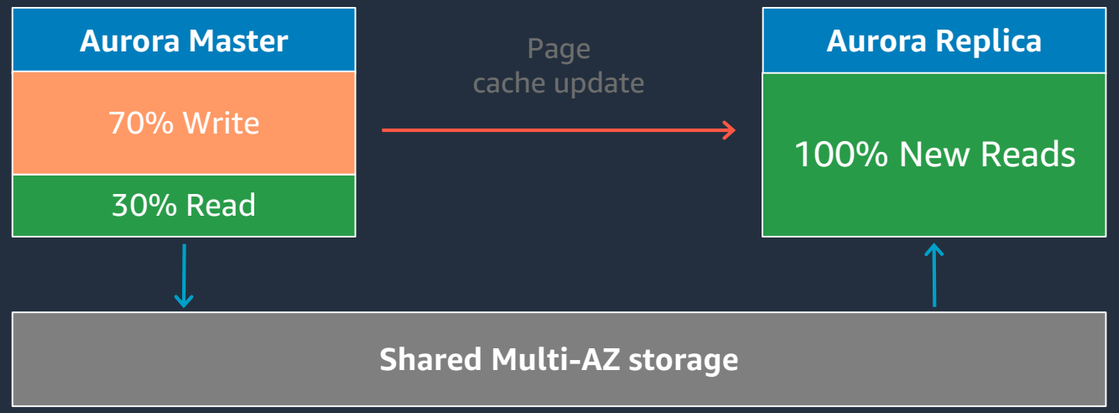
aws

# Aurora read replicas are dedicated to reads

## Traditional read scaling




## Amazon Aurora read scaling




Logical using binary log, or WAL

**Same write** workload on replica

**Independent** storage

Physical using page change vectors (redo log)

**No writes** on replica

**Shared** storage

**Shared**: Common MVCC state, efficient for OLTP reads
**Independent**: safe for ad-hoc, long running or risky queries



aws

# Aurora architectural improvements

## Do less work:

- Do fewer I/Os
- Minimize network packets
- Cache prior results
- Offload the database engine

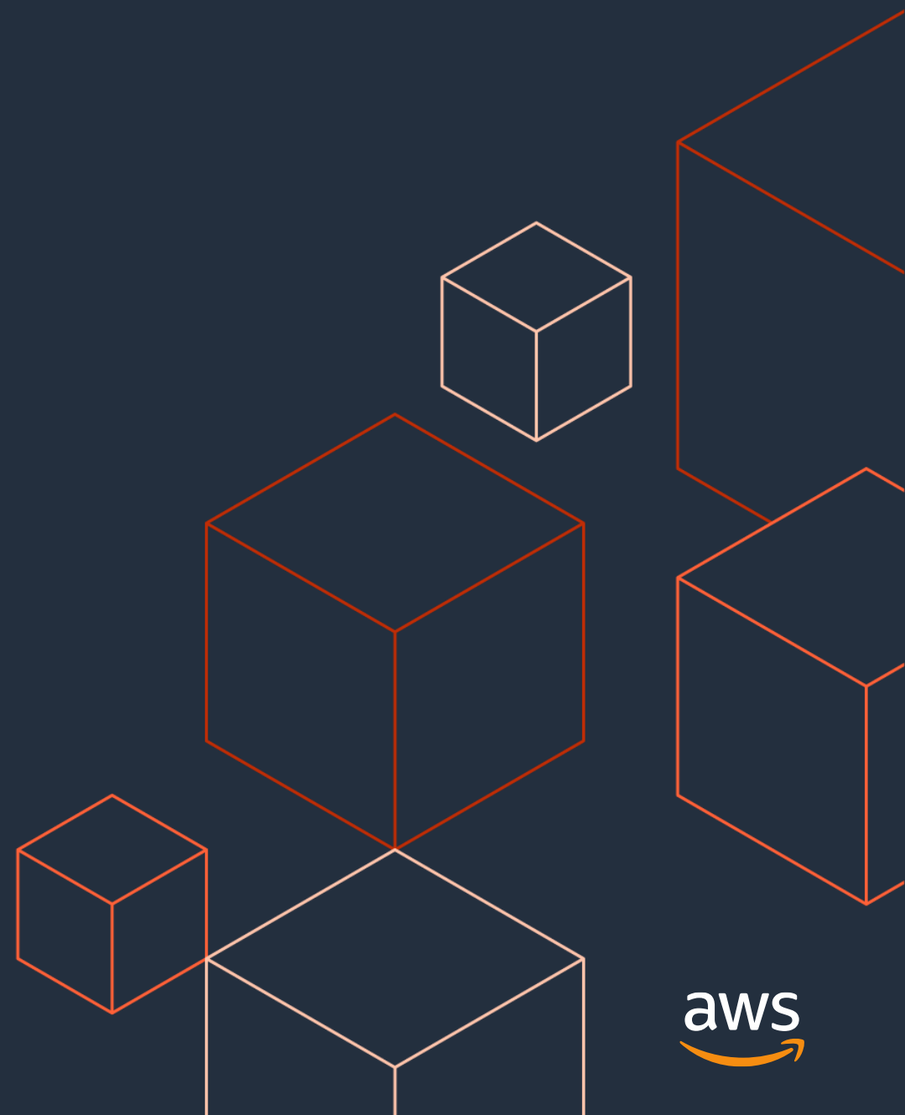
## Be more efficient:

- Process asynchronously
- Reduce latency path
- Use lock-free data structures
- Batch operations together



aws

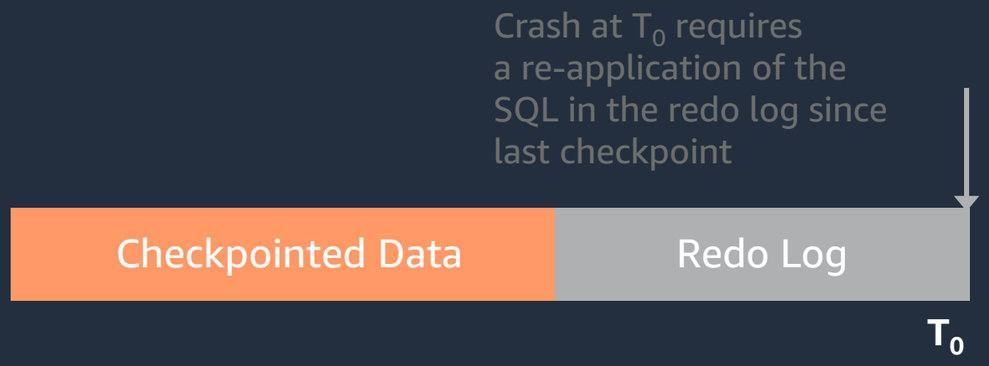
# Amazon Aurora Key Innovations



# Near-instant crash recovery

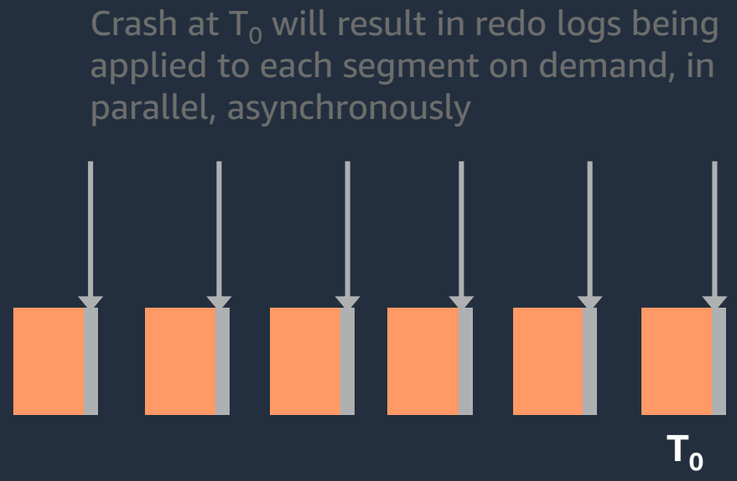
## Traditional database

- Have to replay logs since the last checkpoint
- Typically 5 minutes between checkpoints
- Single-threaded in MySQL; requires a large number of disk accesses




## Amazon Aurora

- Underlying storage replays redo records on demand as part of a disk read
- Parallel, distributed, asynchronous
- No replay for startup

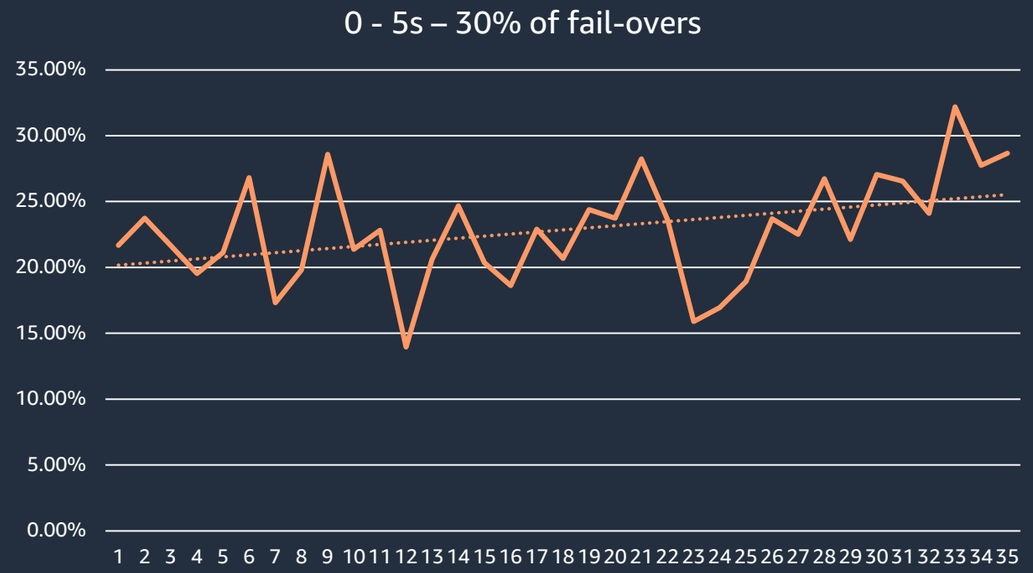


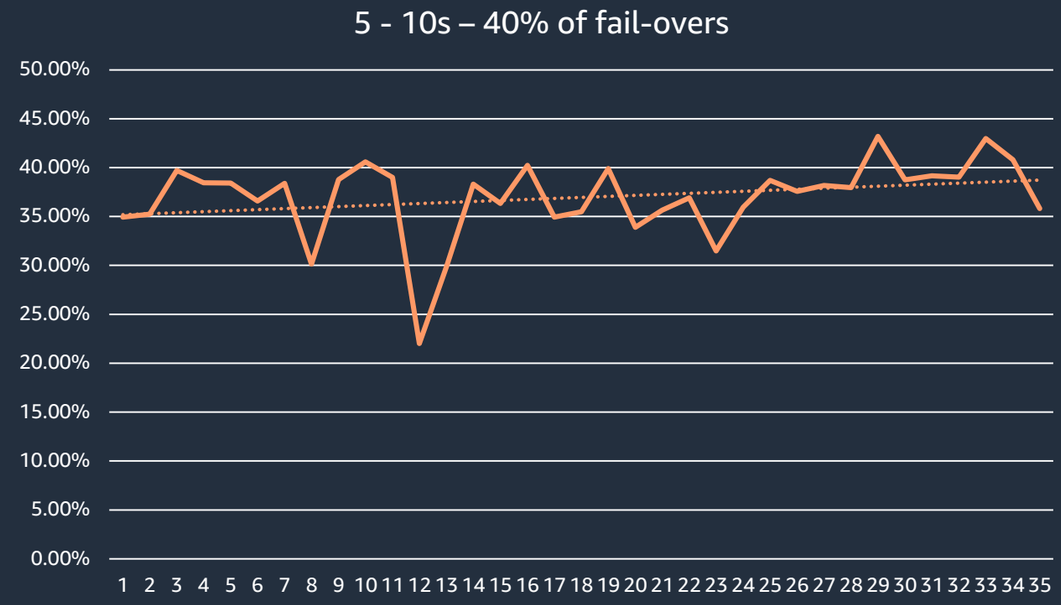

**MySQL Note:** Binlogs (master), undo logs, catalog, FTS indexes, XA still require crash recovery

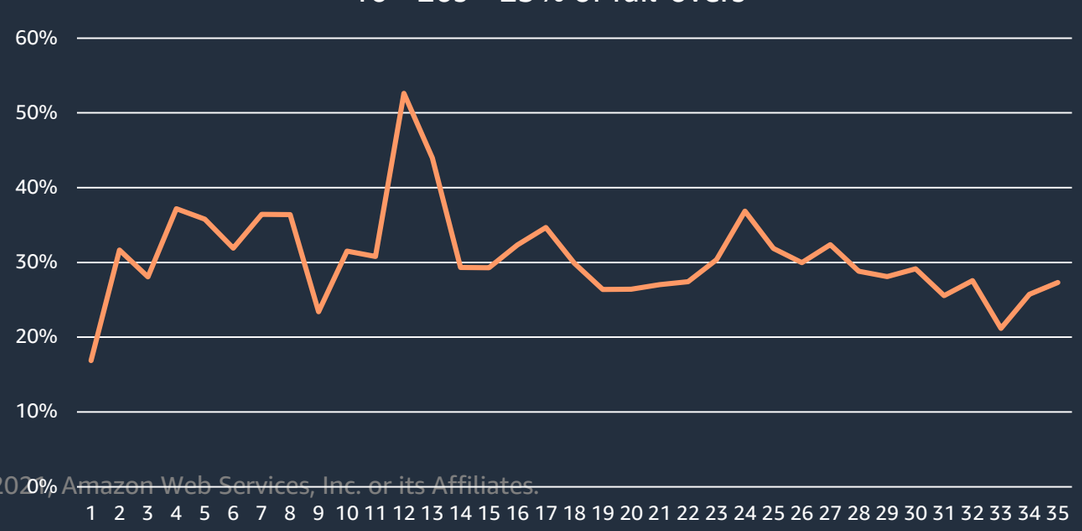


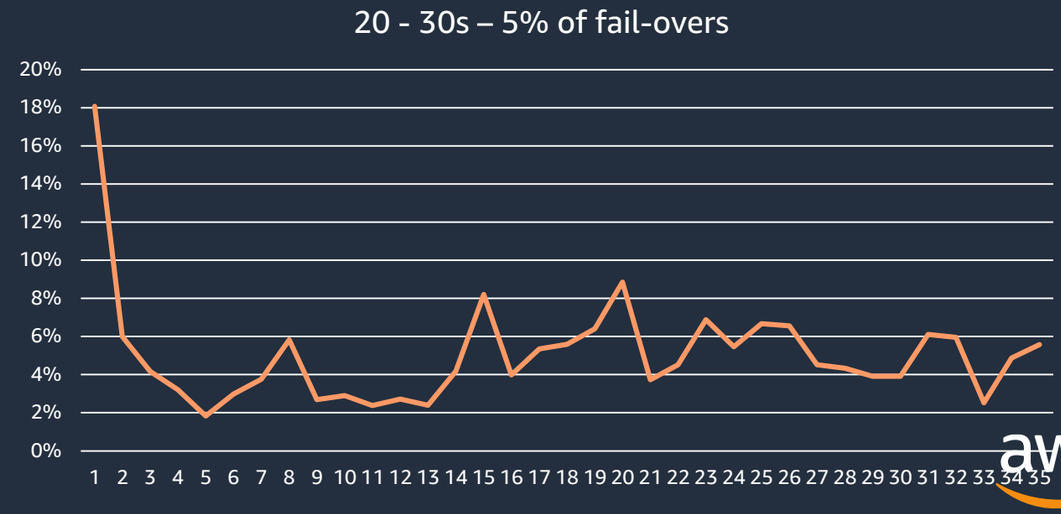
http://blog.symedia.pl/2016/03/crash-recovery-mysql-aurora.html

# When the database fails – recovery is fast <30 seconds



# Cluster Cache Management (CCM)

PostgreSQL Only

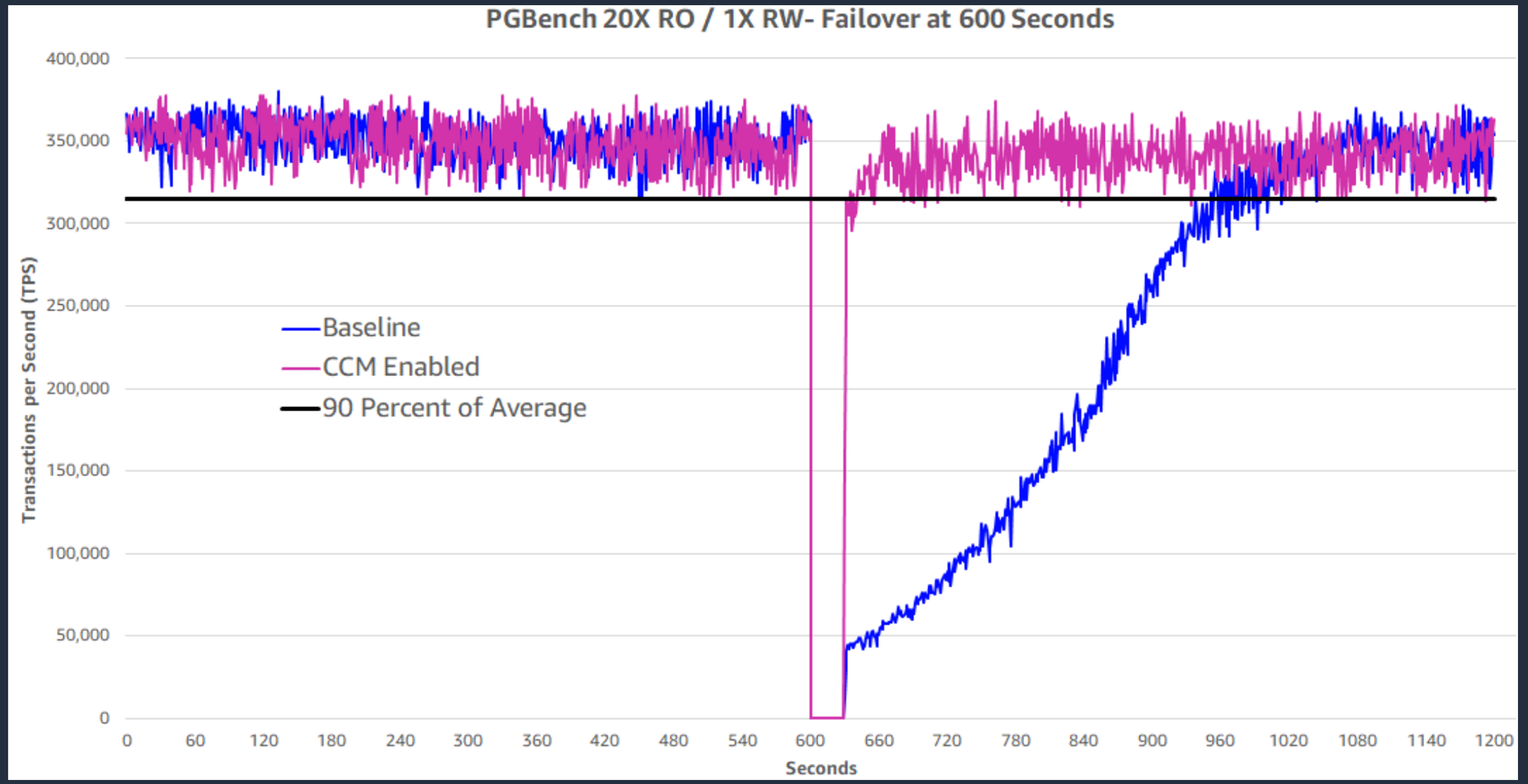
- Improves the customer experience with regards to the performance of the new writer node after failover.
- Lowest failover tier reader will send the set of buffers that are currently cached to the writer node as a bloom filter (avoid duplicate sends).
- On receipt writer node compares the blocks cached in the buffer cache and sends frequently used buffers to that reader node .
- If a failover occurs, the designated reader (now promoted to new writer) uses values in its warm cache immediately.



aws

# CCM in practice

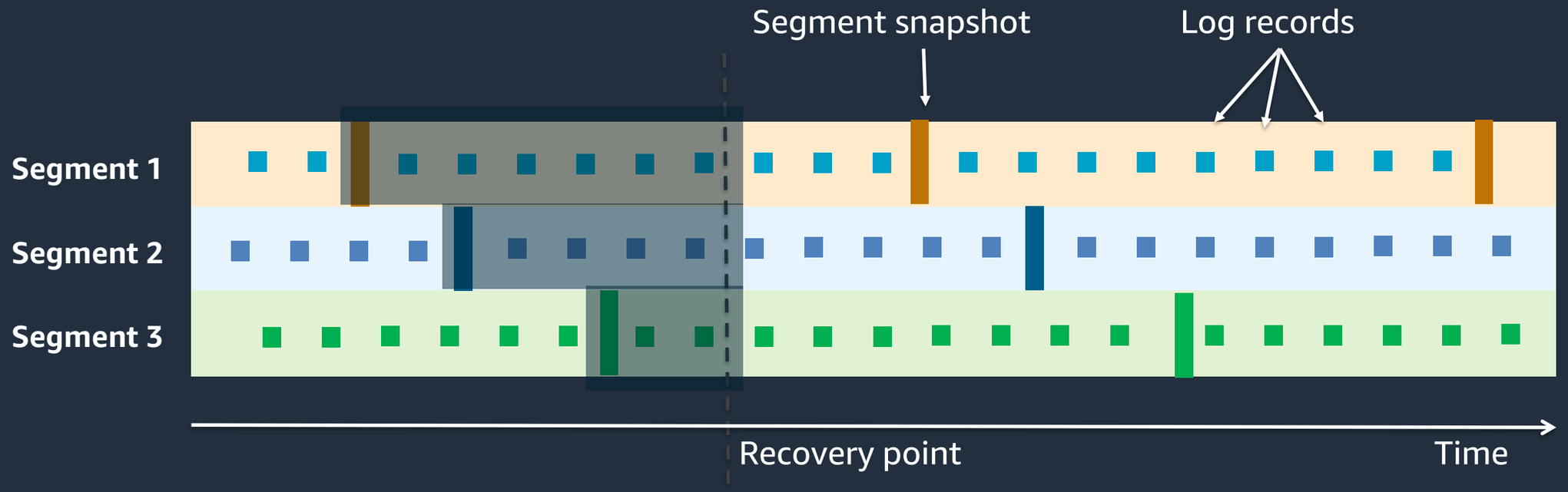
PostgreSQL Only






aws

# Continuous backup with point-in-time restore



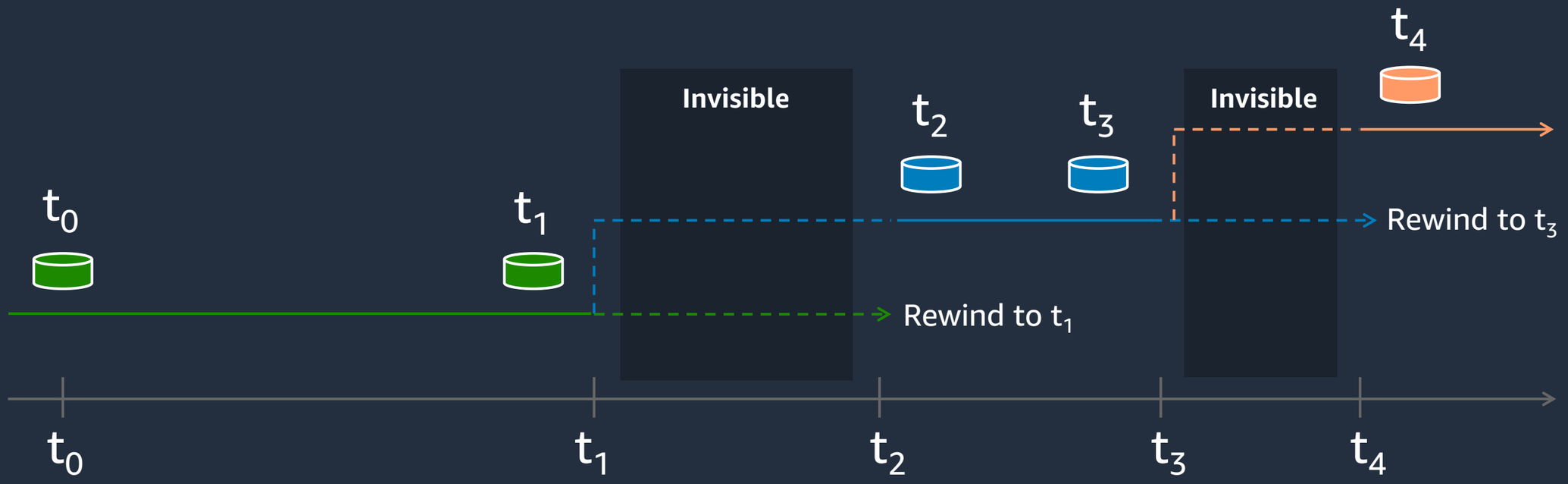

- Backup is performed without performance or availability impact by storage nodes
- Periodic snapshot of each segment is taken in parallel, and stored in S3
- Logs records streamed to Amazon S3 continuously
- At restore, appropriate segment snapshots and log records are retrieved by storage nodes
- At restore, log records are applied to segment snapshots in parallel and asynchronously



aws

# Aurora Backtrack

MySQL Only




**Backtrack brings the database to a point in time without requiring restore from backups**

Recover from an unintentional DML or DDL operation

Backtrack is not destructive; you can backtrack multiple times to find the right point in time

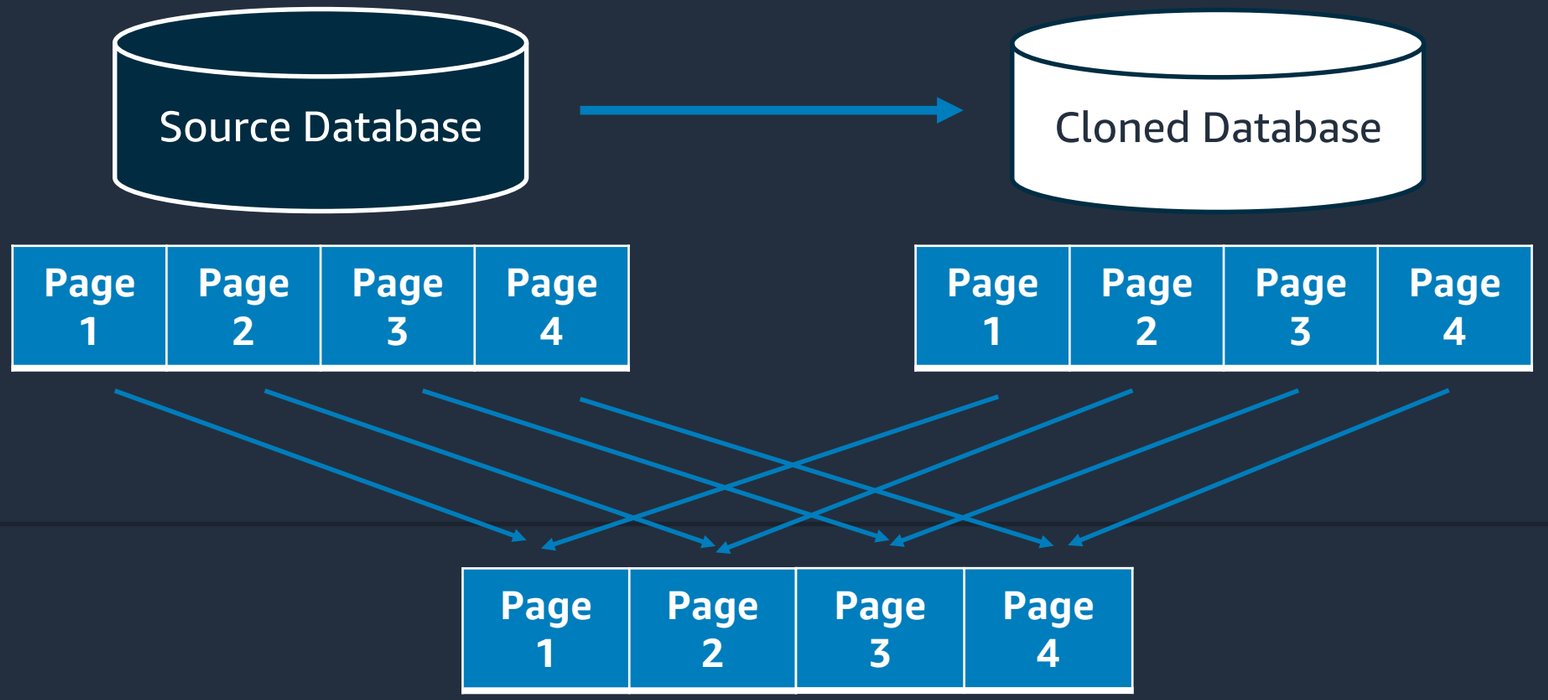
Also useful for QA (rewind your DB between test runs)



aws

# Fast database cloning

Create a copy of a DB cluster (storage volume) without duplicate storage cost
Creation is fast – we don't physically copy data



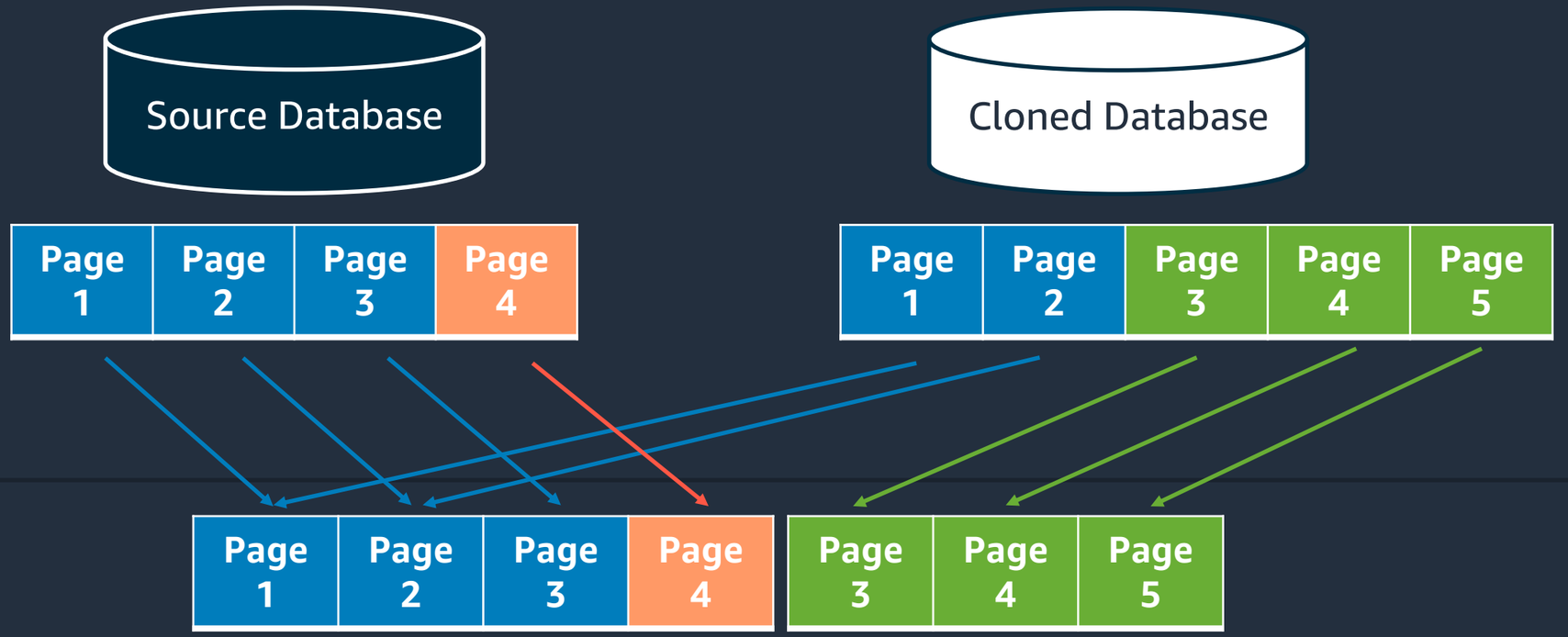

State: Created a clone, made no storage changes
Both databases reference **same** pages on the shared distributed storage system



aws

# Fast database cloning (continued)

Isolation: Activity on the clone doesn’t impact performance of the source (and vice-versa)



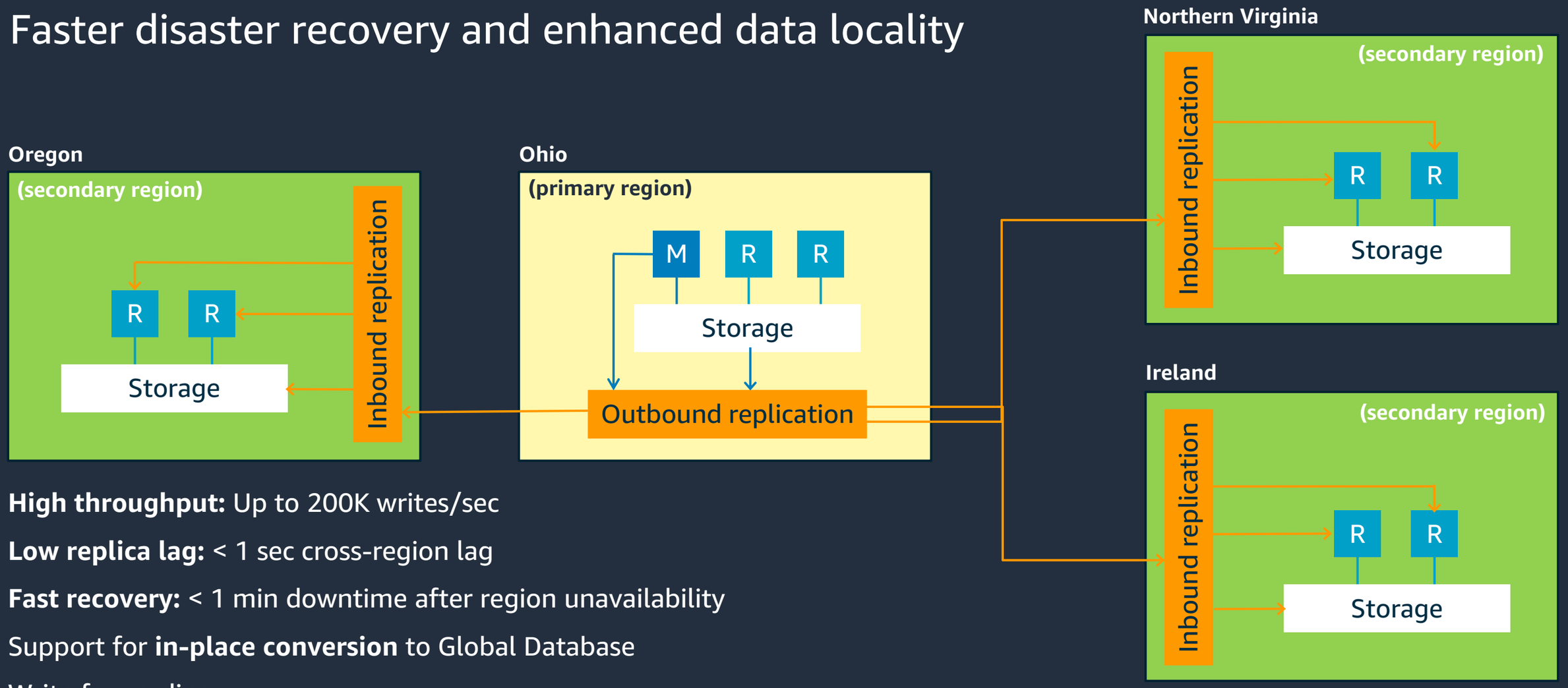

State: Created a clone, made storage changes on both source & clone
Both databases reference **common** pages on the shared distributed storage system



aws

# Aurora Global Database

## Faster disaster recovery and enhanced data locality




**High throughput:** Up to 200K writes/sec

**Low replica lag:** < 1 sec cross-region lag

**Fast recovery:** < 1 min downtime after region unavailability

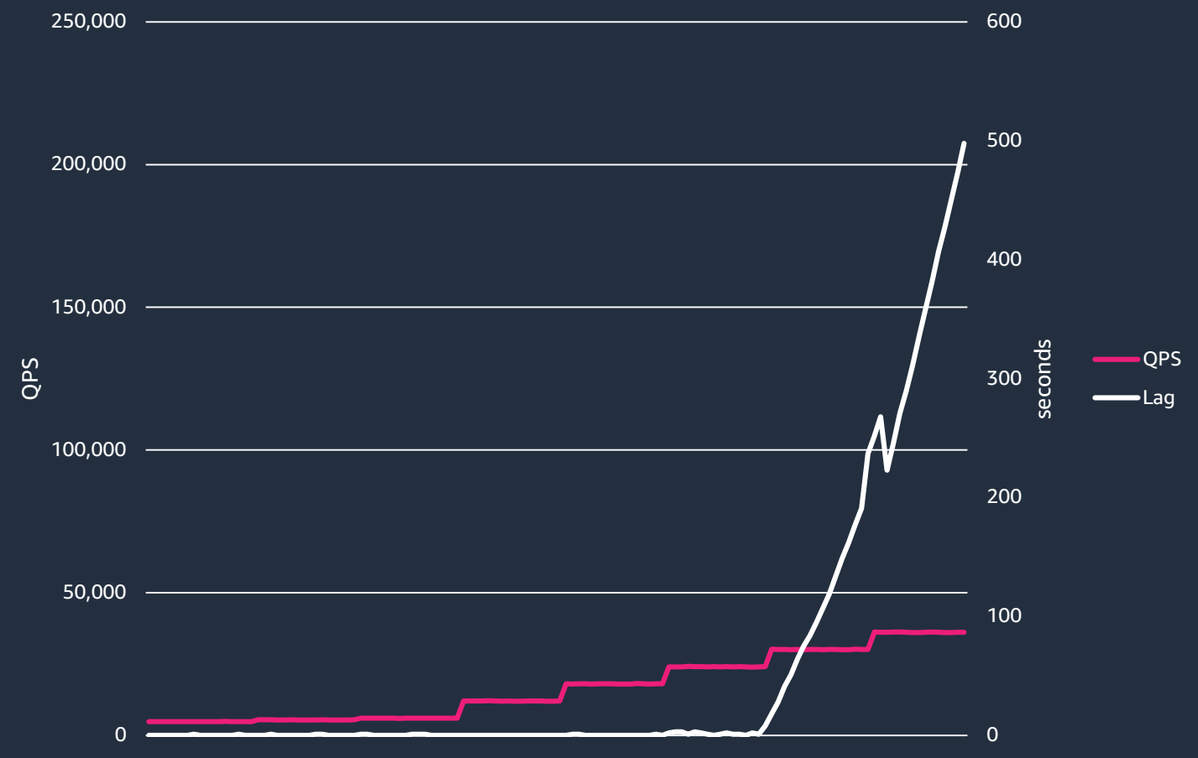
Support for **in-place conversion** to Global Database

Write forwarding

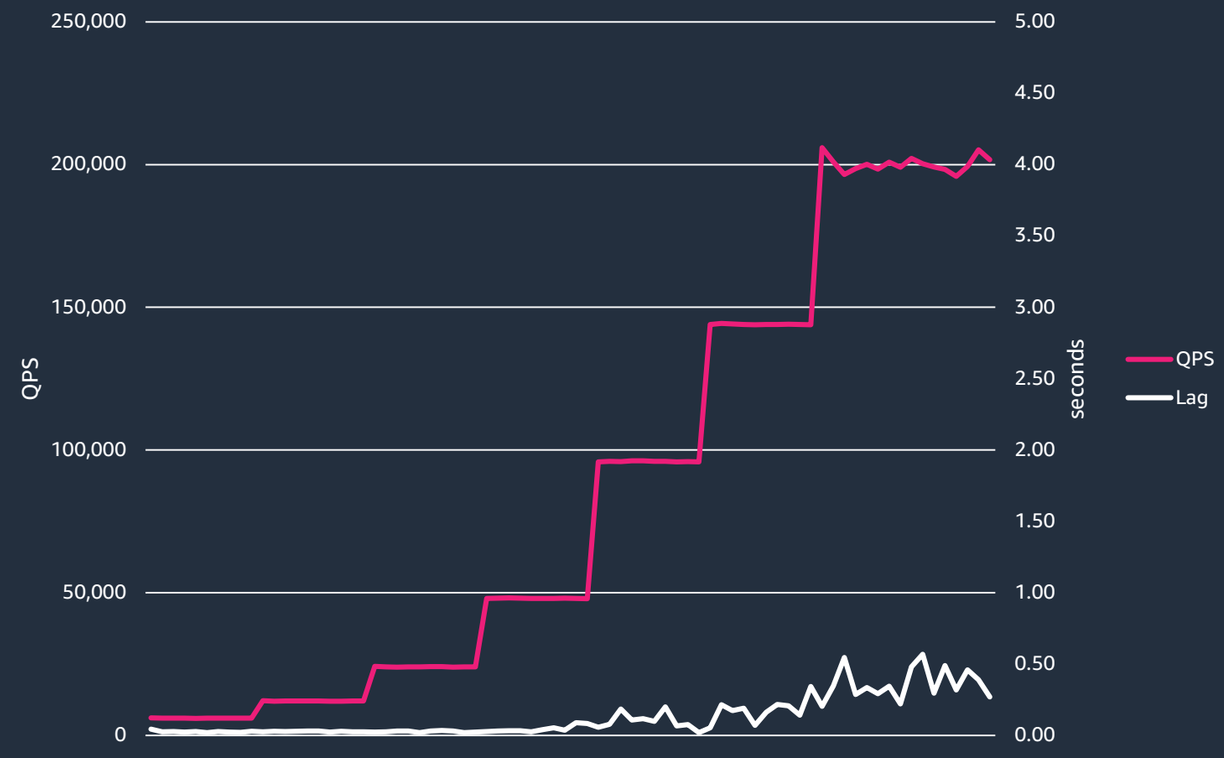


aws

# Performance: logical vs. physical replication




**Logical replication**
**MySQL Binary logs (MTS)**




**Physical replication**
**Global Database**

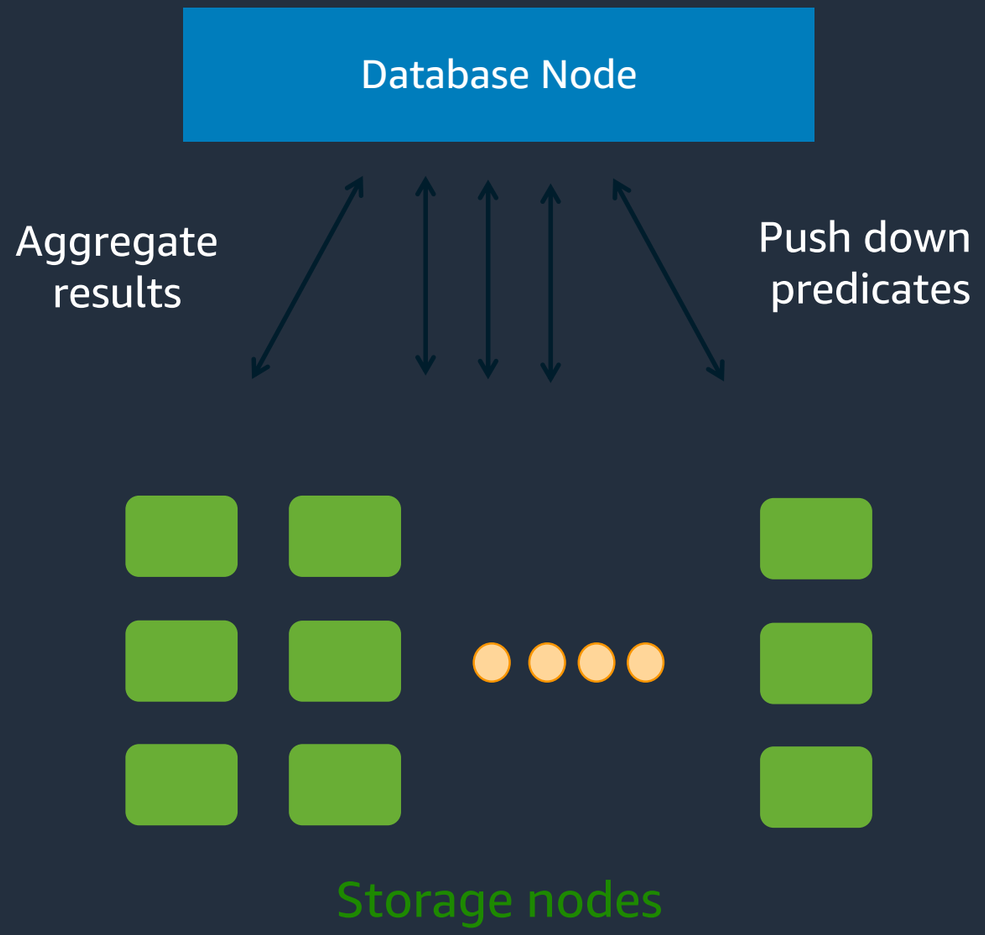
SysBench OLTP (write-only) stepped every 600 seconds on R4.16xlarge



aws

# Parallel query processing

MySQL Only




## Aurora storage has thousands of CPUs

- Opportunity to push down and parallelize query processing
- Moving processing close to data reduces network traffic and latency

## However, there are significant challenges

- Data is not range partitioned – require full scans
- Data may be in-flight
- Read views may not allow viewing most recent data
- Not all functions can be pushed down

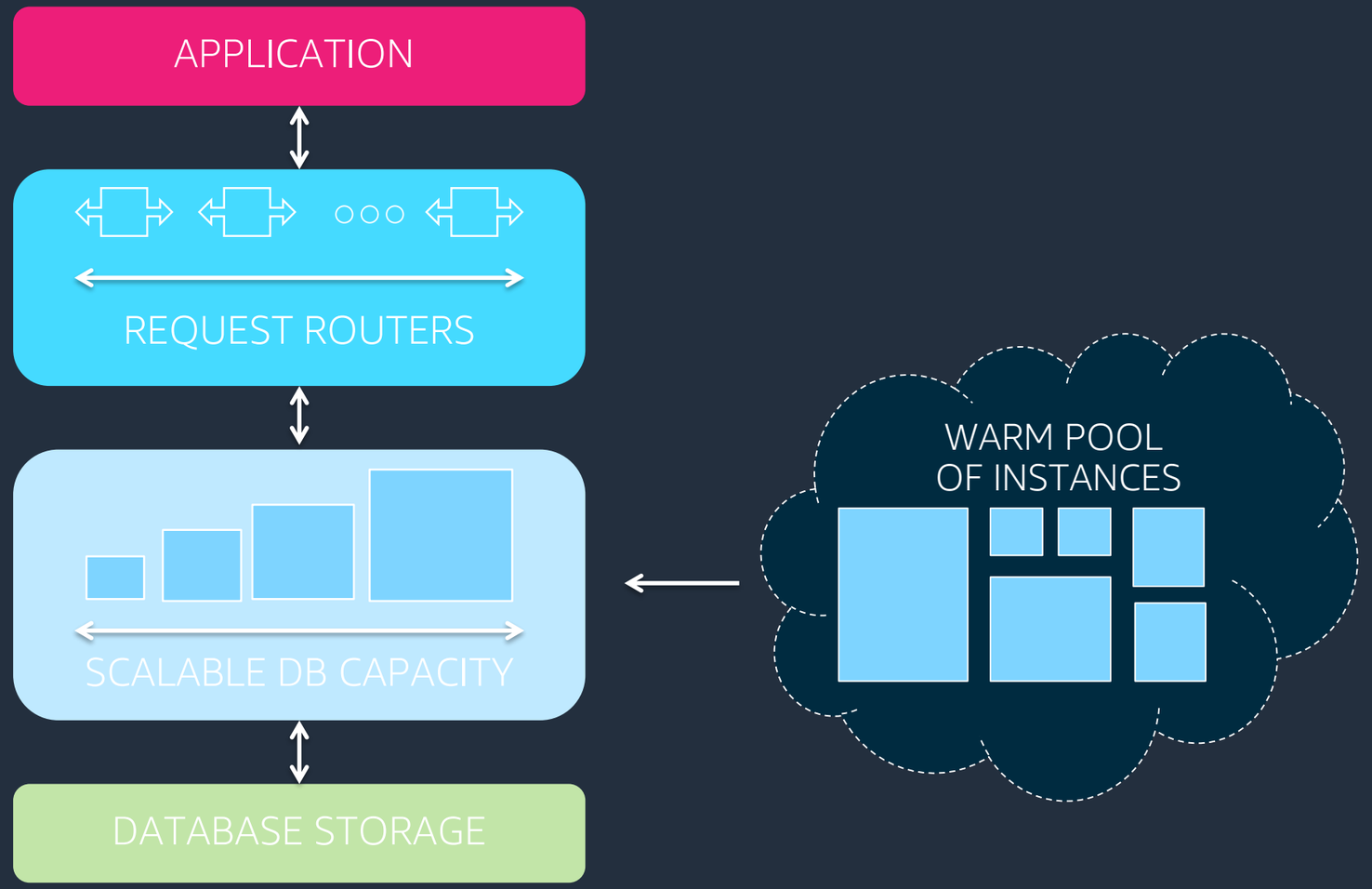
https://aws.amazon.com/blogs/aws/new-parallel-query-for-amazon-aurora/



aws

# Aurora Serverless v1
## Warm pool architecture

- Starts up on demand, shuts down when not in use
- Scales up/down automatically
- No application impact when scaling
- Pay per second, 1 minute minimum

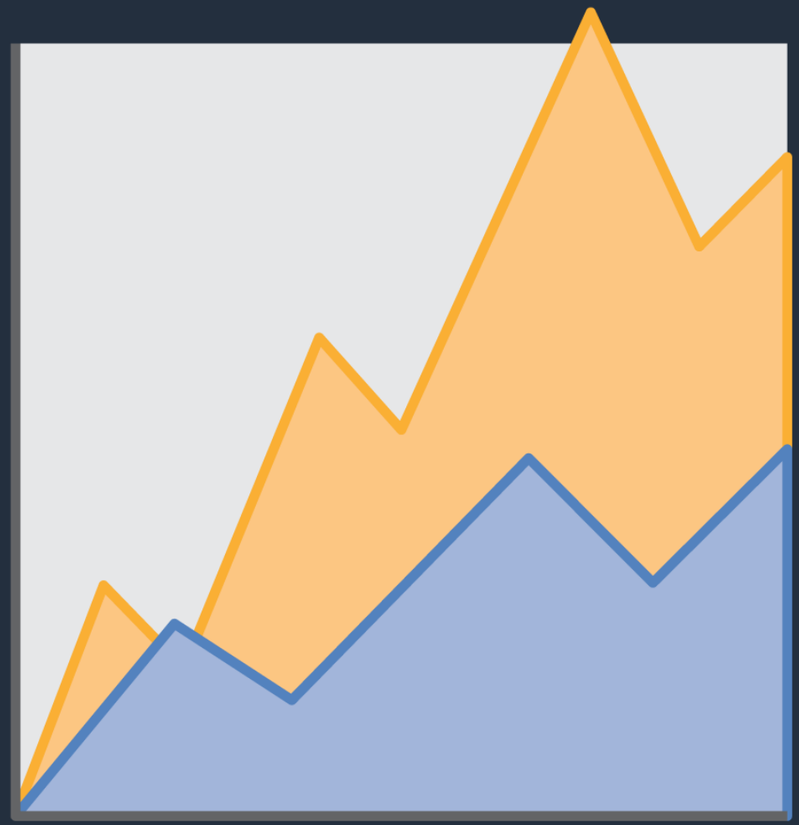





aws

# Aurora Serverless
## Use cases

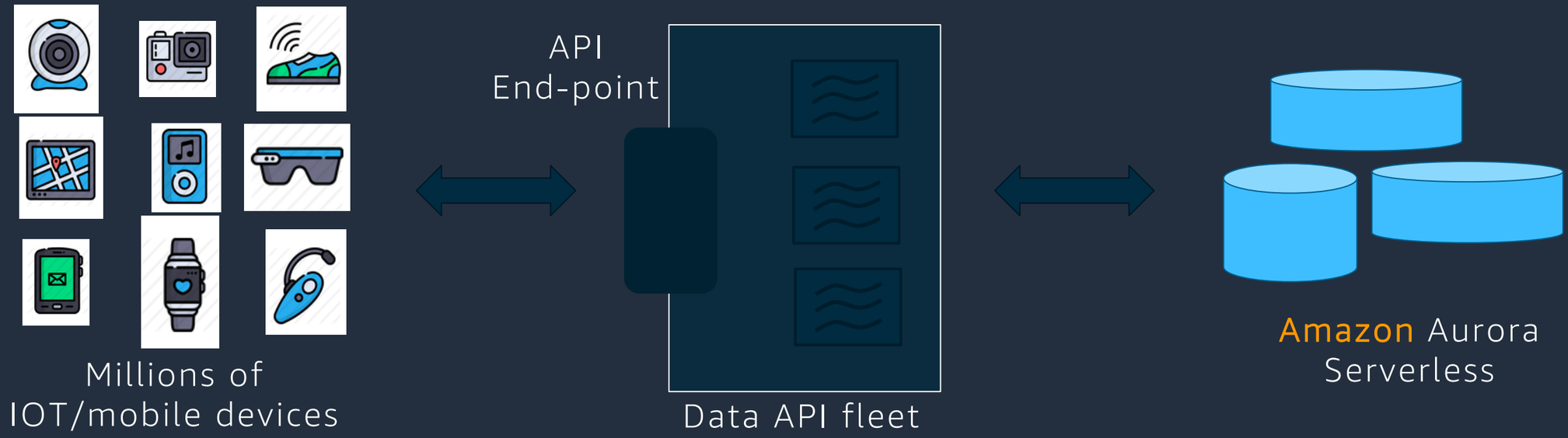
- Infrequently used applications (e.g., low-volume blog site)
- Applications with variable load—peaks of activity that are hard to predict (e.g., news site)
- Development or test databases not needed on nights or weekends
- Consolidated fleets of multi-tenant SaaS applications





aws

# Amazon RDS Data API for serverless applications




Access through simple web interface

- Public endpoint addressable from anywhere
- No client configuration required
- No persistent connections required

Ideal for Serverless applications (Lambda)

Ideal for light-weight applications (IOT)

https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/data-api.html



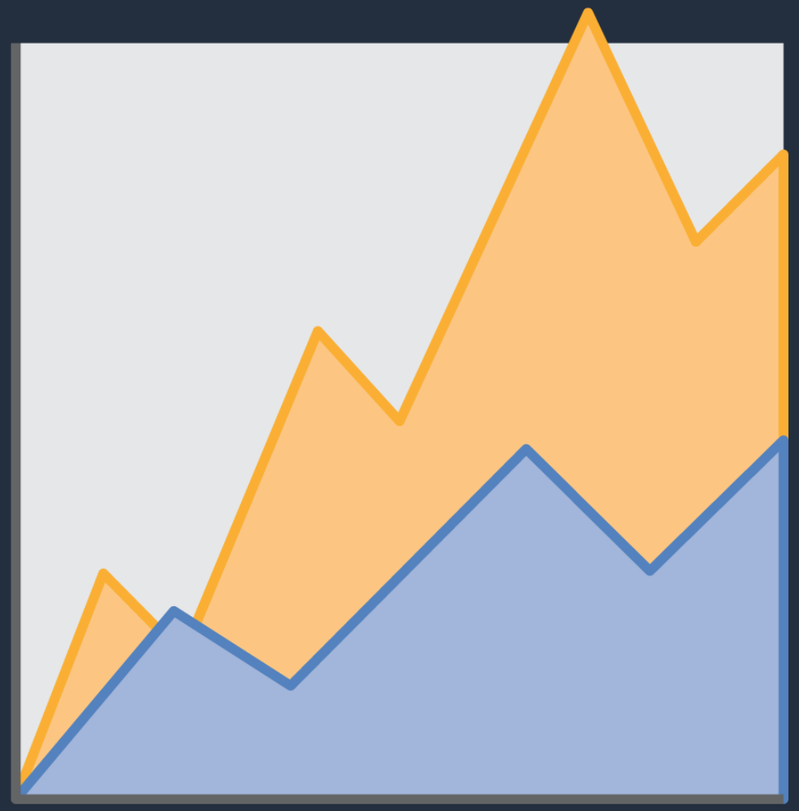
aws

# Monitoring Performance



aws

# Performance Insights



**Why Database Tuning?**

RDS is all about managed databases

Customers also want performance managed:

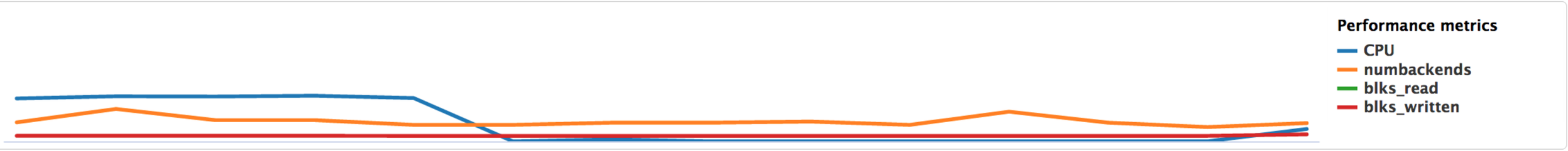
❑ Want easy tool for optimizing cloud database workloads
❑ May not have deep tuning expertise

→ Want a single pane of glass to achieve this

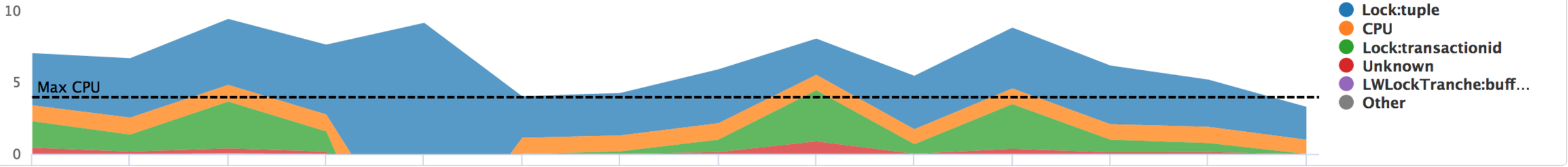


aws

Last 5 minutes




DB load by: **Waits** SQL Hosts Users




Waits | **SQL** | Hosts | Users

Search SQL Queries

| SQL Digest | DB Load | SQL |
|---|---|---|
| 4e15b546005d9489980349e399cc1d24 | | UPDATE pgbench_tellers SET tbalance = tbalance + ? WHERE tid = ?; |
| 9037de313c04df497488ab3670c2466b | | UPDATE pgbench_branches SET bbalance = bbalance + ? WHERE bid = ?; |
| f64d0eecd0bac50e4d71b98c500599af | | ROLLBACK TO SAVEPOINT JDBC_SAVEPOINT_1 |
| a30112fac30fcf95bebbdc07e3e38573 | | select foo(); |
| dc20ac1a0efa57e29ebf7f3df136c600 | | SELECT * FROM LOGIN("username_in" := $1,"password_in" := $2) |
| 3e20d081813ac00ef7ecd3f778eaefa5 | | SELECT abalance FROM pgbench_accounts WHERE aid = ?; |

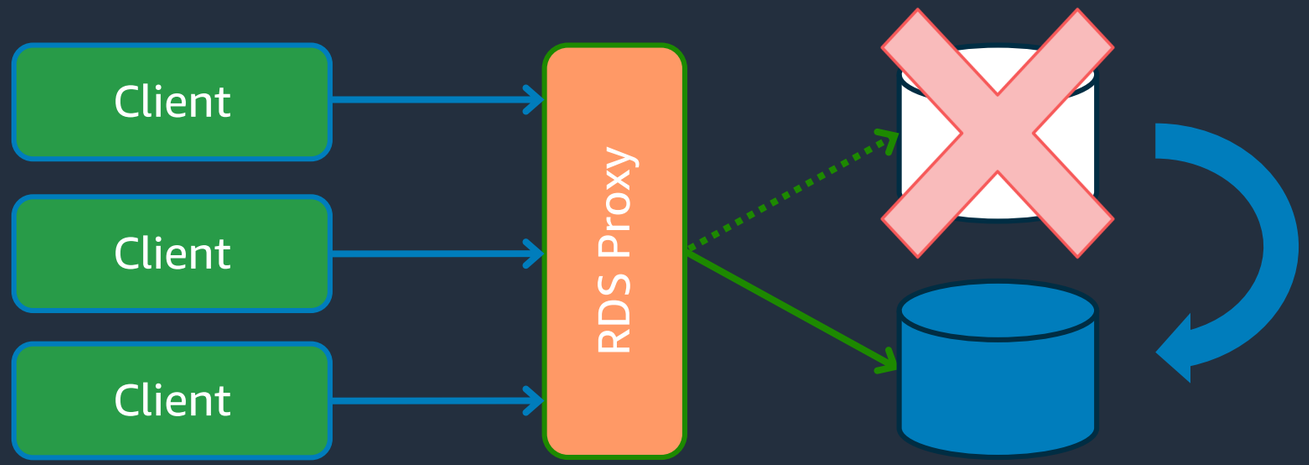
# Minimizing Disruptions



aws

# Minimize Failover Disruption with RDS Proxy

- Stabilizes and reduces connections against backend DB
- Multiplexes client requests at transaction boundaries
- Idle application connections are preserved by the proxy
- New requests will be queued up by the proxy during the failover
- Faster failure recovery, bypassing DNS (up to 66% faster)



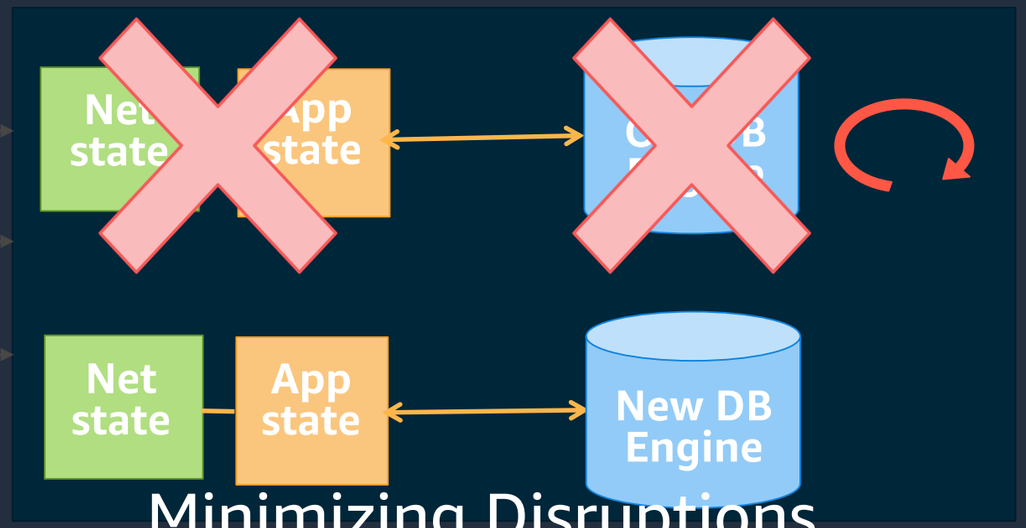



aws

# Zero Downtime Patching (ZDP)

Before ZDP

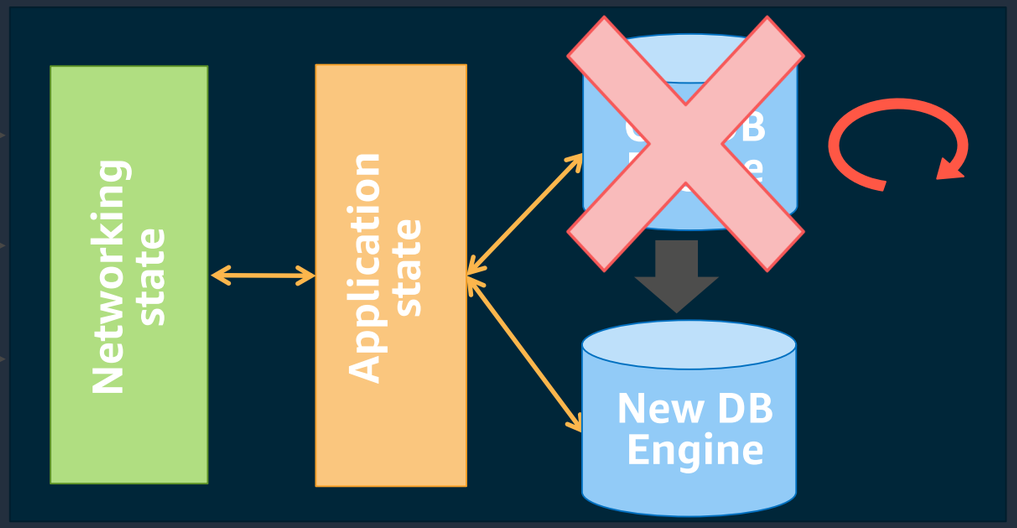
User sessions terminate during patching




Storage Service

With ZDP

User sessions remain active through patching

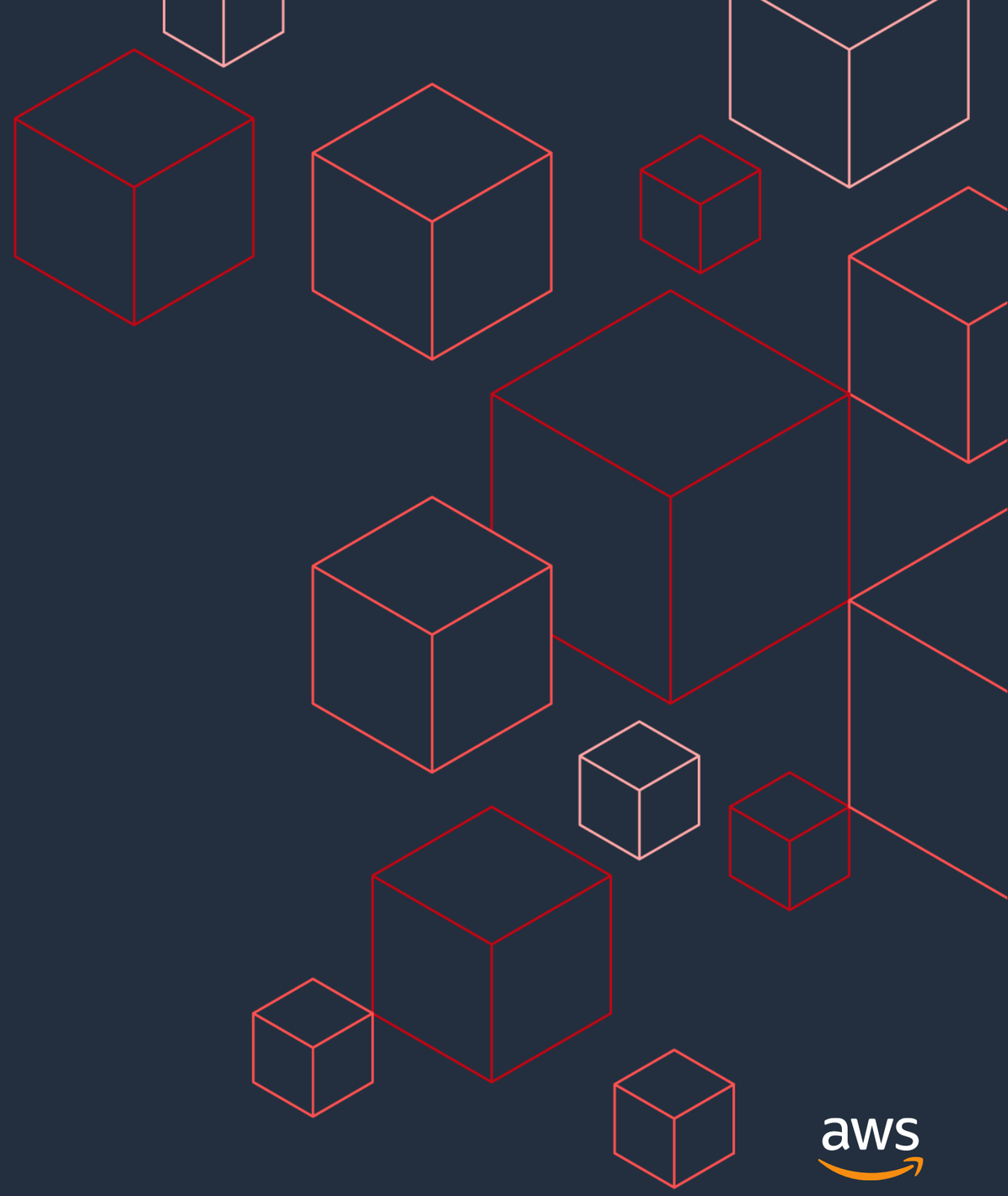



Storage Service



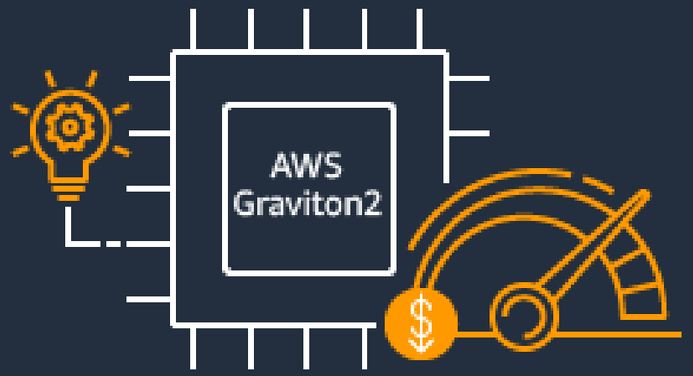
aws

# What’s New?



# AWS Graviton2 powered Aurora

## Aurora Graviton instances




**AWS Graviton2**

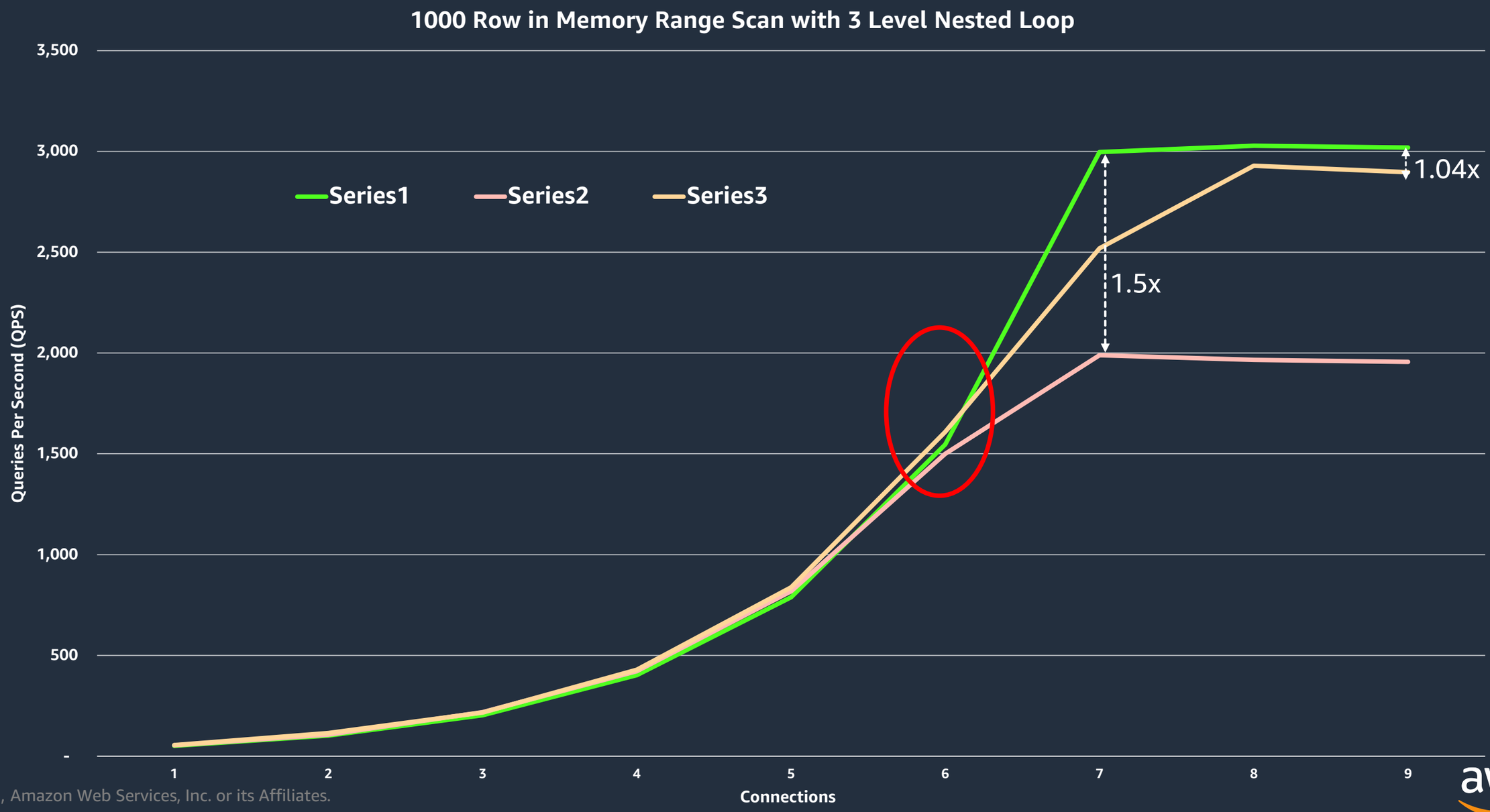
Powerful and efficient,
modern applications

- Custom designed by AWS utilizing Arm Neoverse N1 cores
- Graviton2, fully compatible with all existing Aurora features - No porting needed
- Up to 35% better performance



aws

# Graviton2



# Aurora Global Database: Managed Planned Failover

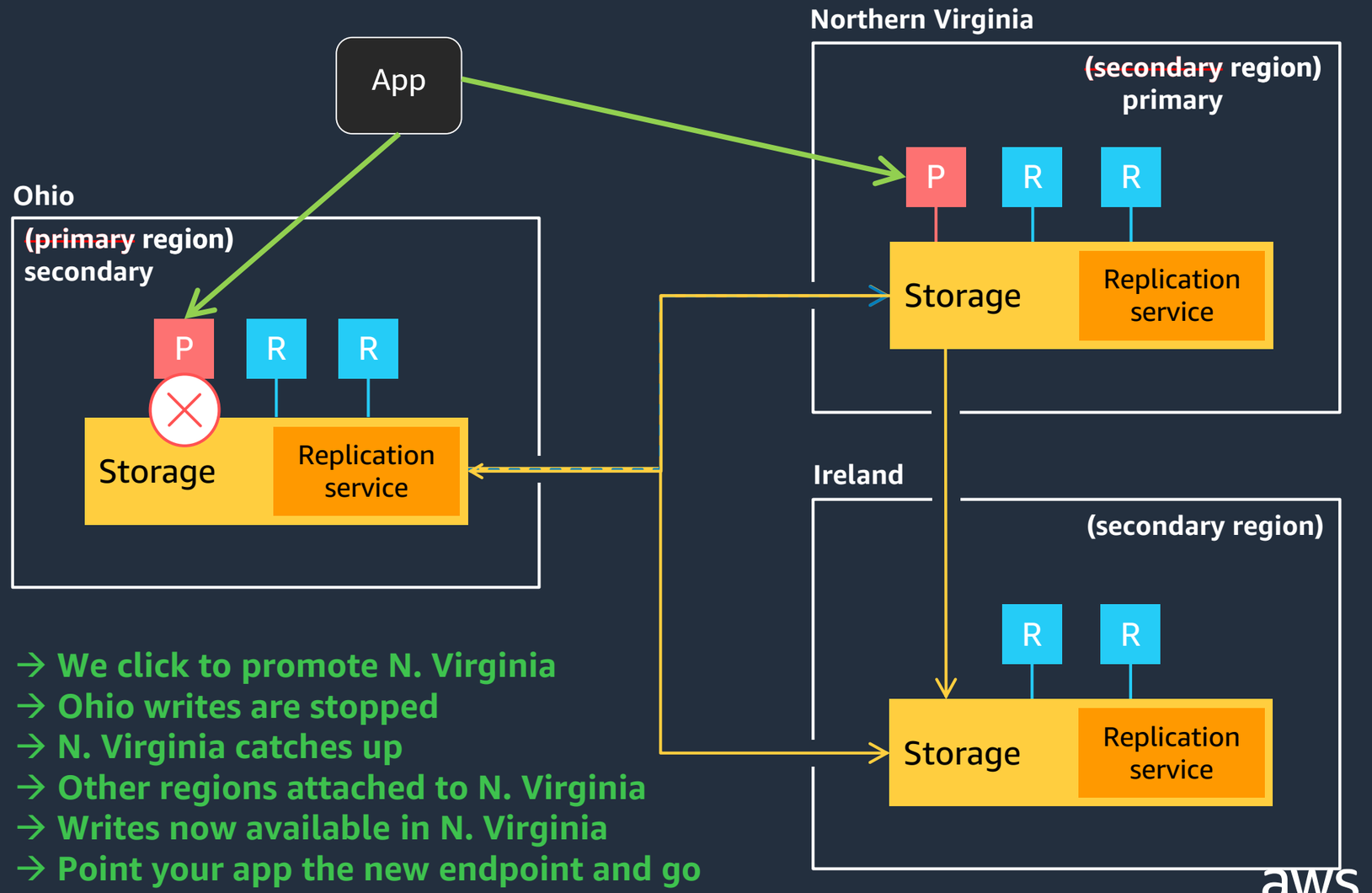
Automated way to promote secondary region to primary

For testing DR setup or relocating primary region

Fully maintains the topology

RPO=0; writes are stopped until new primary catches up

In this example we'll promote N. Virginia to primary




→ We click to promote N. Virginia
→ Ohio writes are stopped
→ N. Virginia catches up
→ Other regions attached to N. Virginia
→ Writes now available in N. Virginia
→ Point your app the new endpoint and go

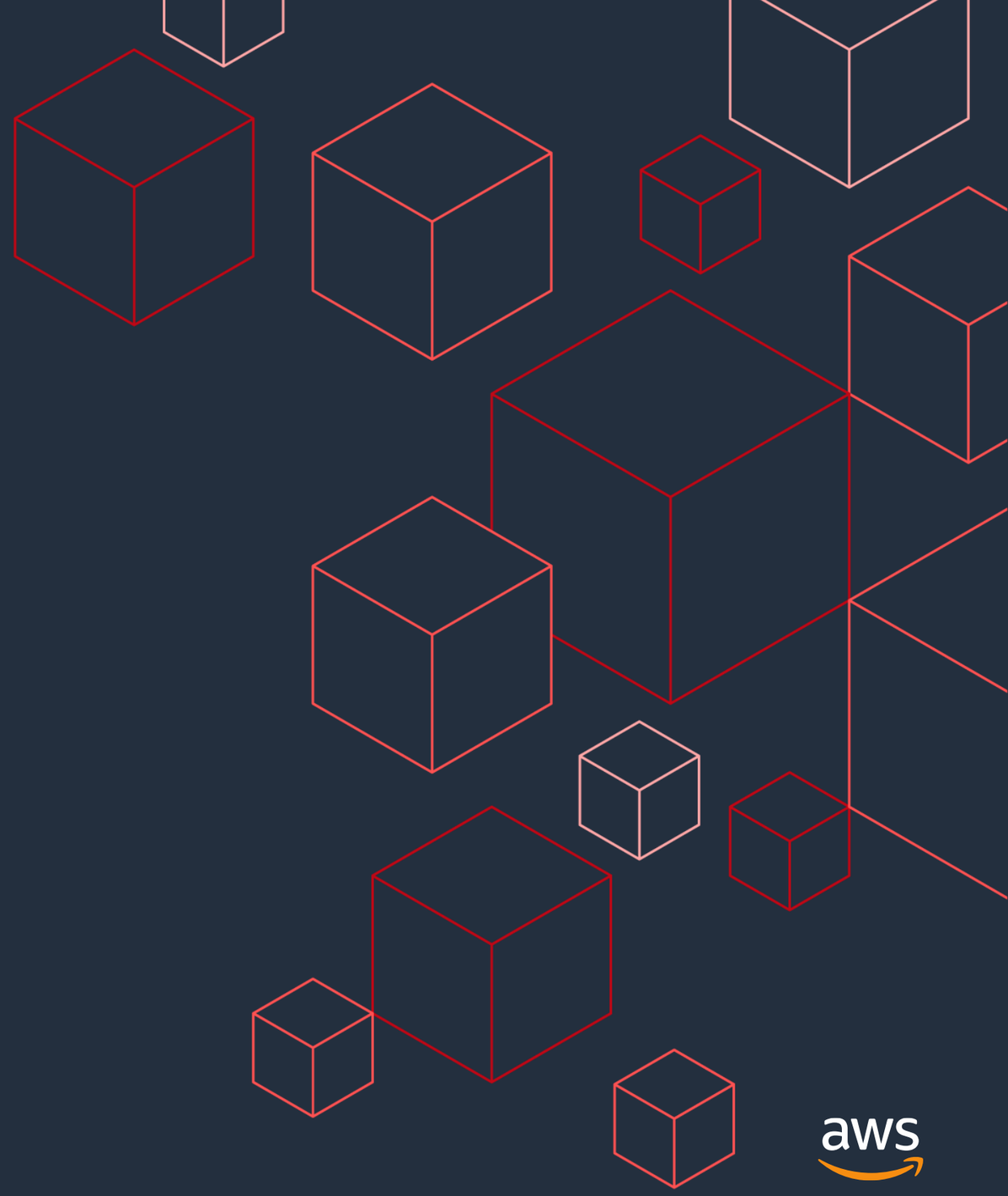
# What’s New?

- Graviton2 instance types – R6g
- Amazon Aurora now supports PostgreSQL 12
- In place upgrades support (MySQL 5.6 -> 5.7 and PostgreSQL 11 -> 12)
- Aurora Global Database Managed Planned Failover



aws

# What’s Coming?



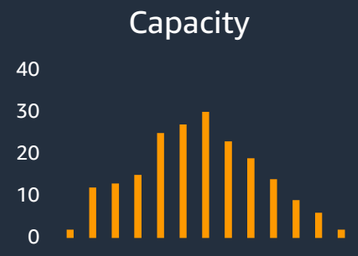


aws

# Introducing Amazon Aurora Serverless v2 (preview)

An auto-scaling configuration for Amazon Aurora that now supports even the most demanding applications and database workloads


Scale instantly, from hundreds to hundreds-of-thousands of transactions, in a fraction of a second




Scale in fine-grained increments to provide just the right amount of database capacity


Full breadth of Aurora capabilities including Multi-AZ, Global Database


Up to 90% cost savings when compared to provisioning for peak load



aws

# What’s Coming?

- Aurora Serverless V2
- Babelfish for Aurora PostgreSQL
- MySQL JDBC driver



aws

# Summary

- Amazon Aurora takes advantage of the cloud native features.
- Amazon Aurora uses a purpose built, log structured storage.
- Amazon Aurora has built in features like Automated backups, Multi-AZ, Global DB, and more that make it highly resilient to failures.
- Features like Backtrack, fast recovery, fast cloning makes it easy to recover from failures.
- Recovery from failure can be further managed by using RDS Proxy.
- New features like serverless V2 will improve scaling and performance.



aws

Aditya Samant
ausamant@amazon.com


# SugarCRM Cloud and AWS Aurora

Aliaksei Klimko

# SugarCRM

- 4,500 Customers
- 325,000 SaaS users
- Over 100 production database servers
- 7 AWS Regions
- Over 50 Clusters
- Over 1 billion requests per month
- Over 50 TB of managed data

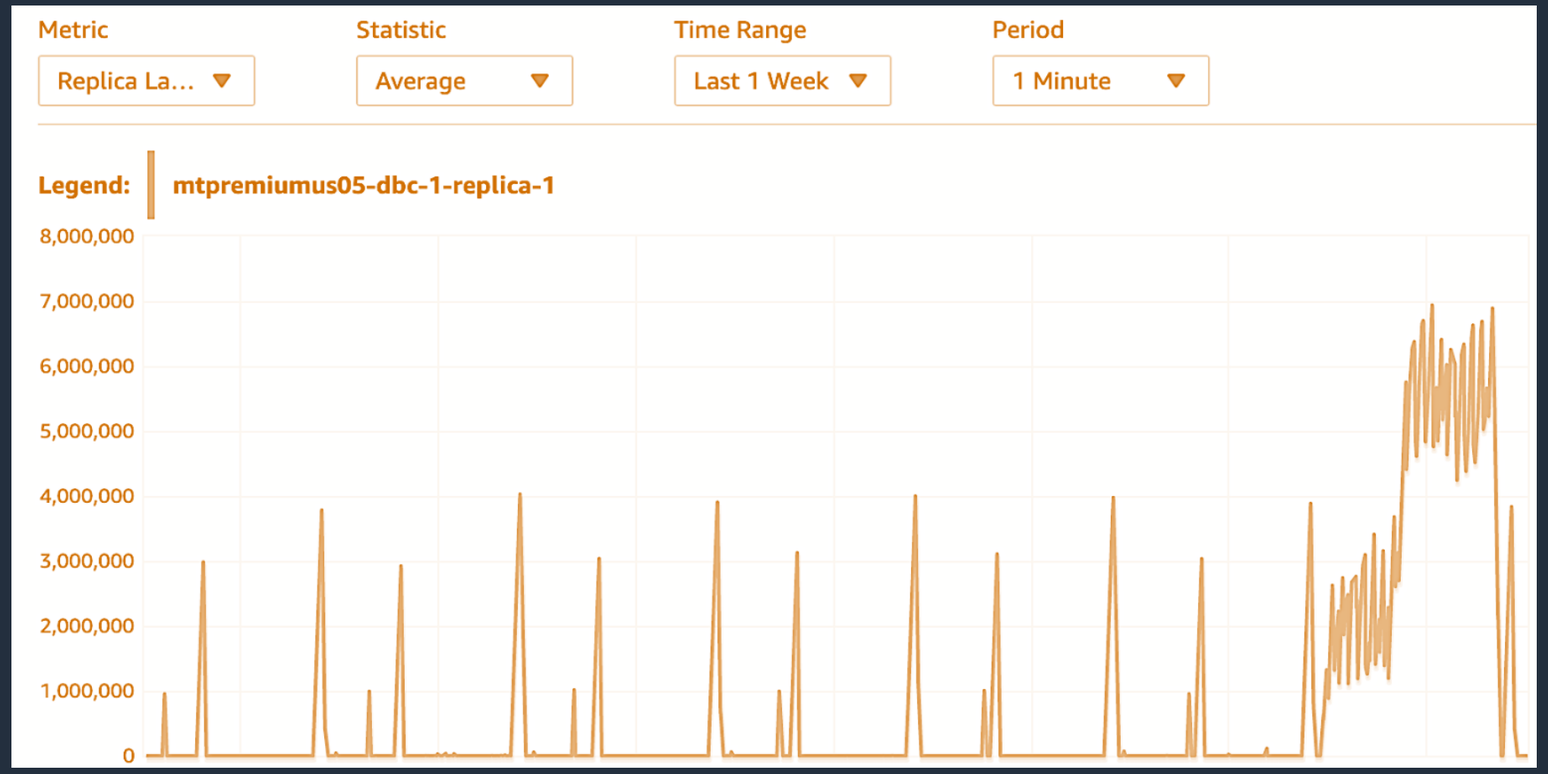


aws

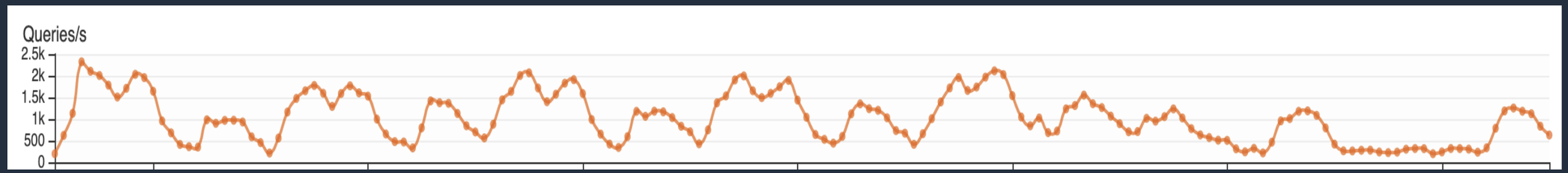
# Issues with MySQL cluster

- Our company was successfully using AWS RDS MySQL for years. But we had some issues:
- Replicas were used for disaster recovery and backups only.
- Even though the application is designed to utilize the reader database to server the analytical and complicated queries we were not able to do this.
- Due to the spikes in replica lag, the scalability in a single cluster was only possible vertically. The spikes sometimes were up to 48 hours.

# Why we migrated



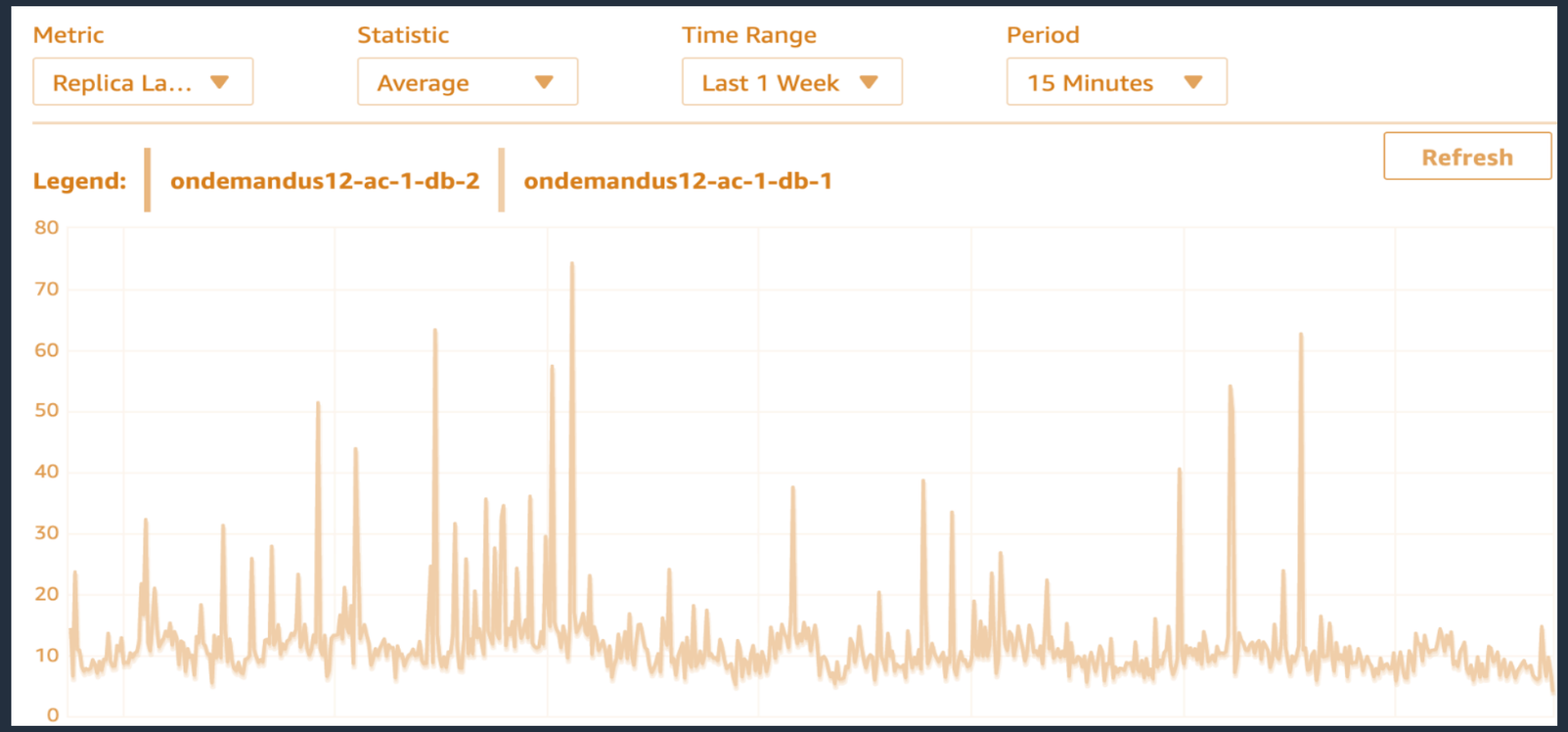


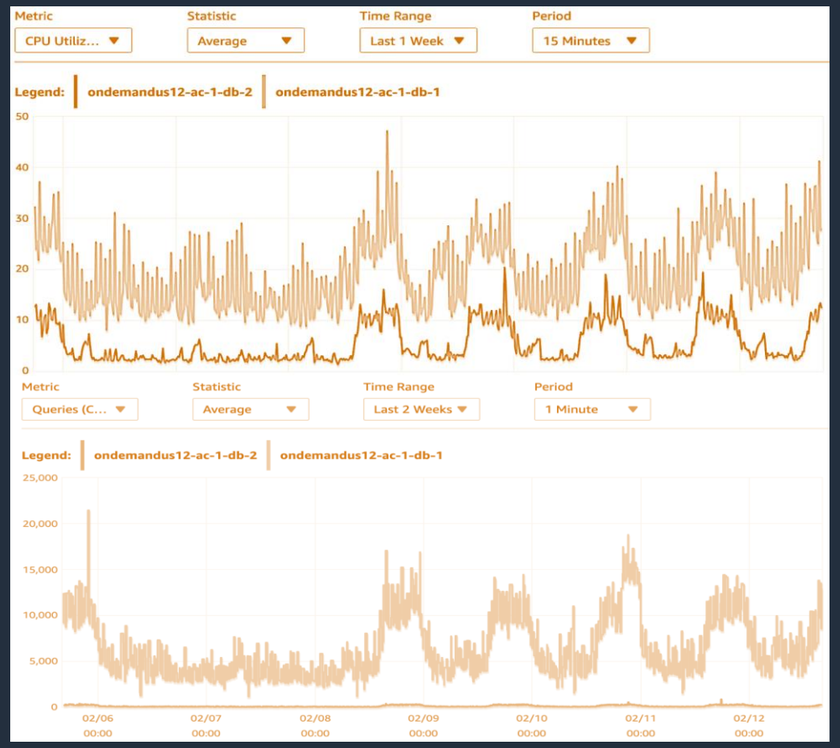



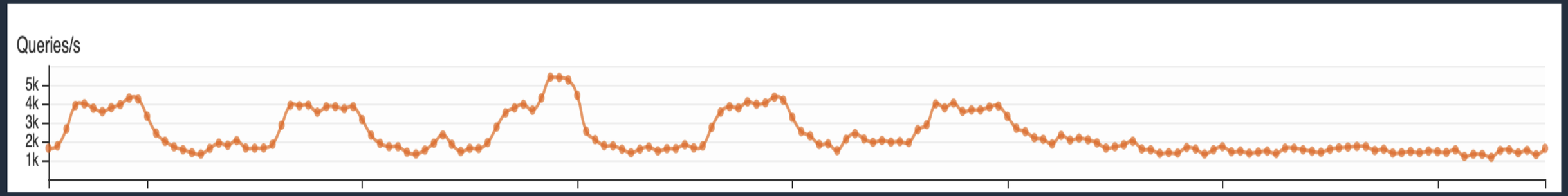



aws

# Aurora benefits



# Internal changes

*The thing that we have changed*

- Backups are now taken differently, using storage volume snapshots.
- Indices were tuned.
- A few queries were rewritten for better performance.

# Conclusion

*The thing that we have changed*

- Replicas are used for analytical and complicated queries, letting writes to serve more transactional queries per second.
- Replica lag is about 20ms with some very rare spikes to 500-700ms with is completely acceptable.
- We can consolidate instances, running more customers per a cloud stack and it allowed us to run fewer DB servers.



aws