Creating Highly Available and Resilient Databases on Amazon Aurora

Chandra sekhar Pathivada, Adarsha Kuthuru

Database Specialist Solutions Architects

Agenda

- Aurora overview
- Distributed Storage Fundamentals
- High Availability
  - Tolerating Storage failures
  - Tolerating compute failures
  - Crash recovery
  - Cluster Cache Management(CCM)
- Disaster Recovery
  - Aurora backup and restore
  - Aurora Backtrack
  - Aurora Global Database
- Testing workload fault resilience
- Improve application scalability, availability, and security with Amazon RDS Proxy

Amazon Aurora

Enterprise database at open-source price

Delivered as a **managed** service

Amazon Aurora

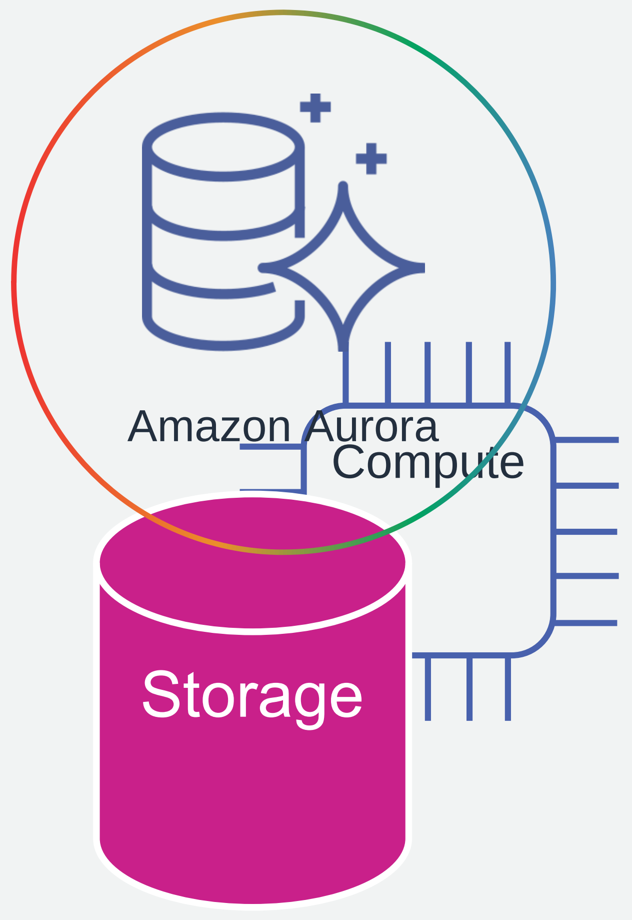
Build reliable applications with multi-AZ with a 99.99% uptime SLA and automatic global replication

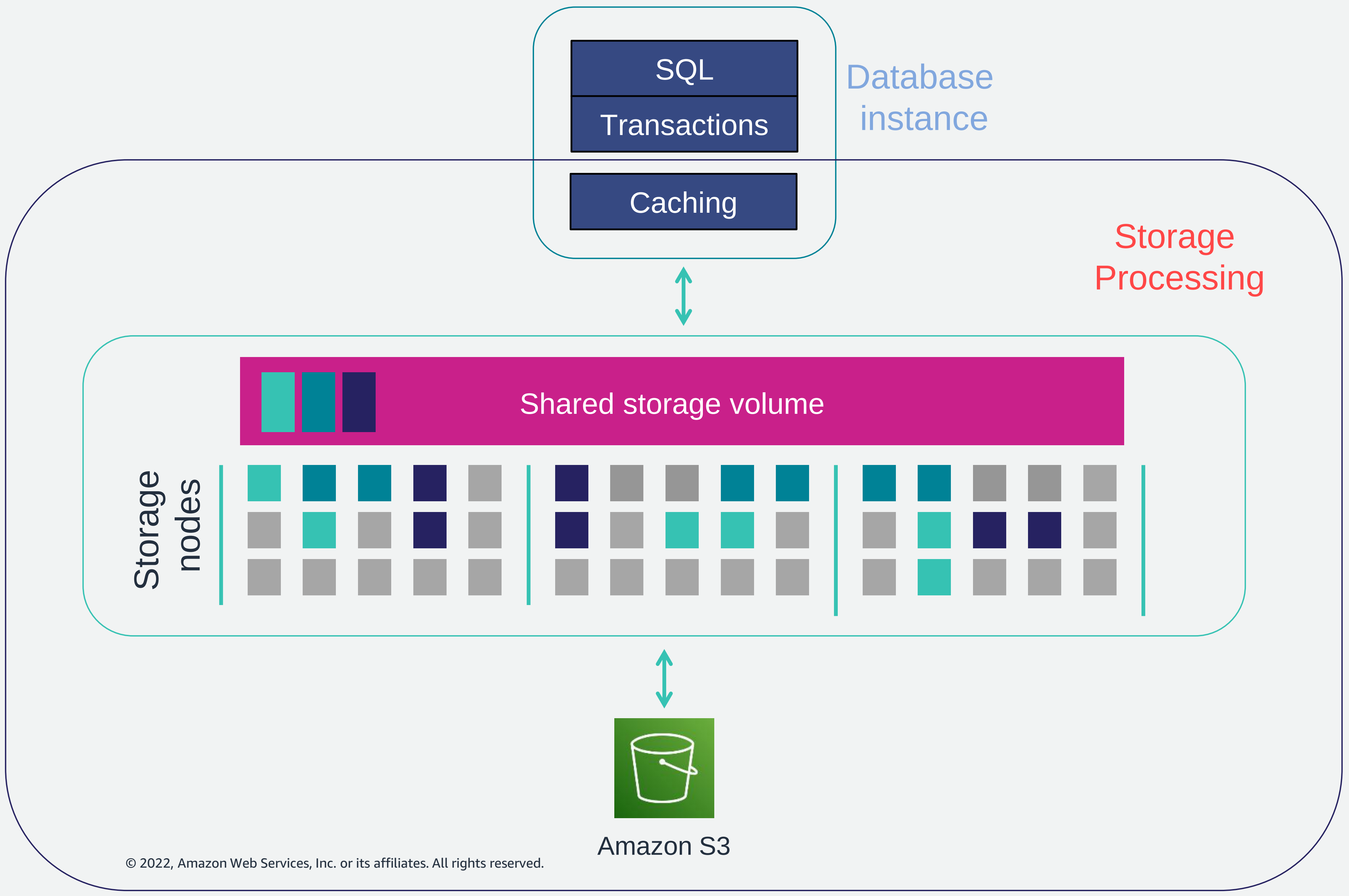
Power performance-intensive apps and maintain full compatibility with MySQL or PostgreSQL at 1/10th the cost of commercial databases

Take advantage of automated database administration and innovative capabilities

Easily migrate MySQL or PostgreSQL databases to and from Aurora

Aurora decouples storage and query processing





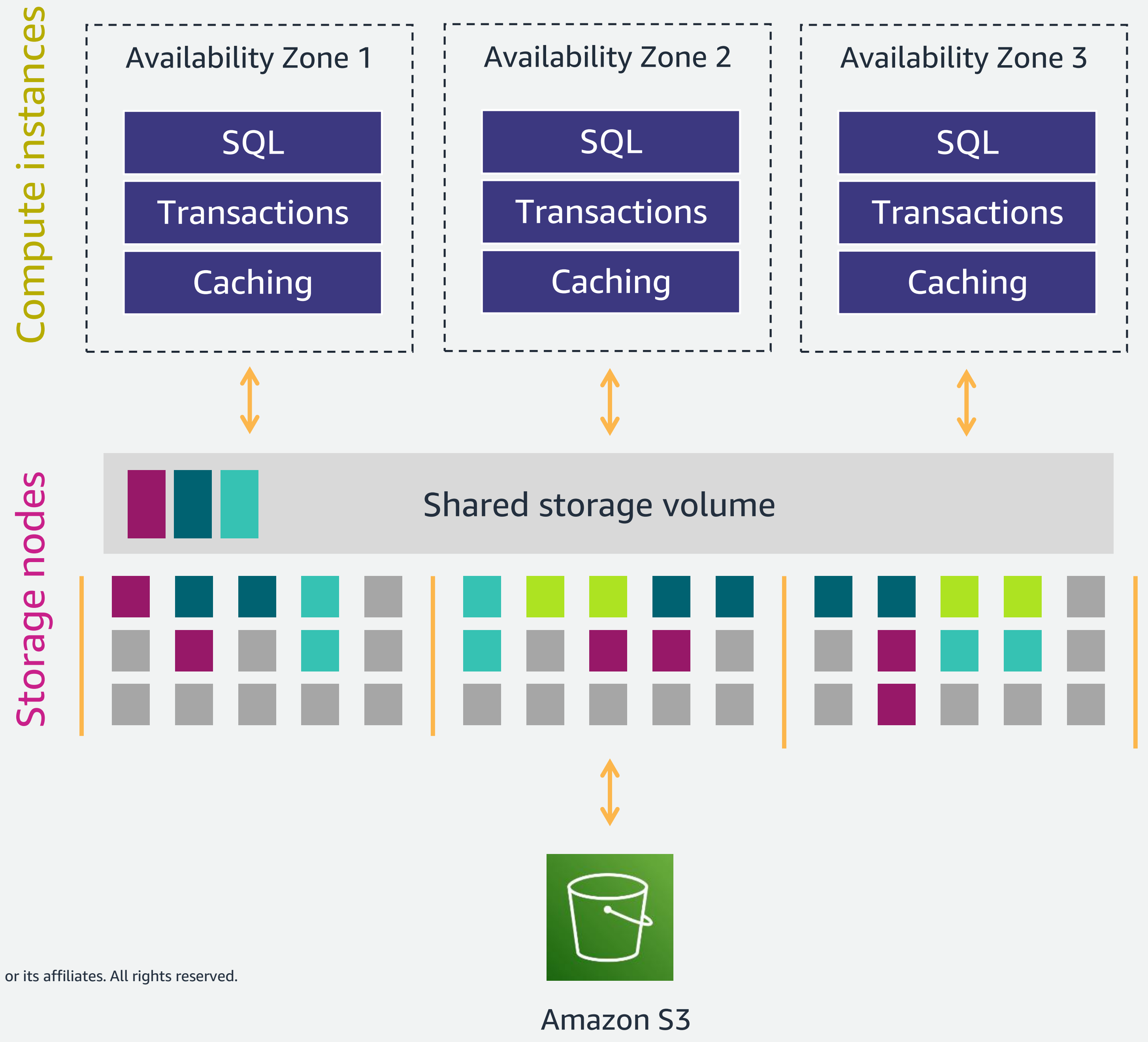
Scale-out, distributed storage processing architecture

Purpose-built log-structured distributed storage system designed for databases

Storage volume is striped across hundreds of storage nodes distributed over three different Availability Zones

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

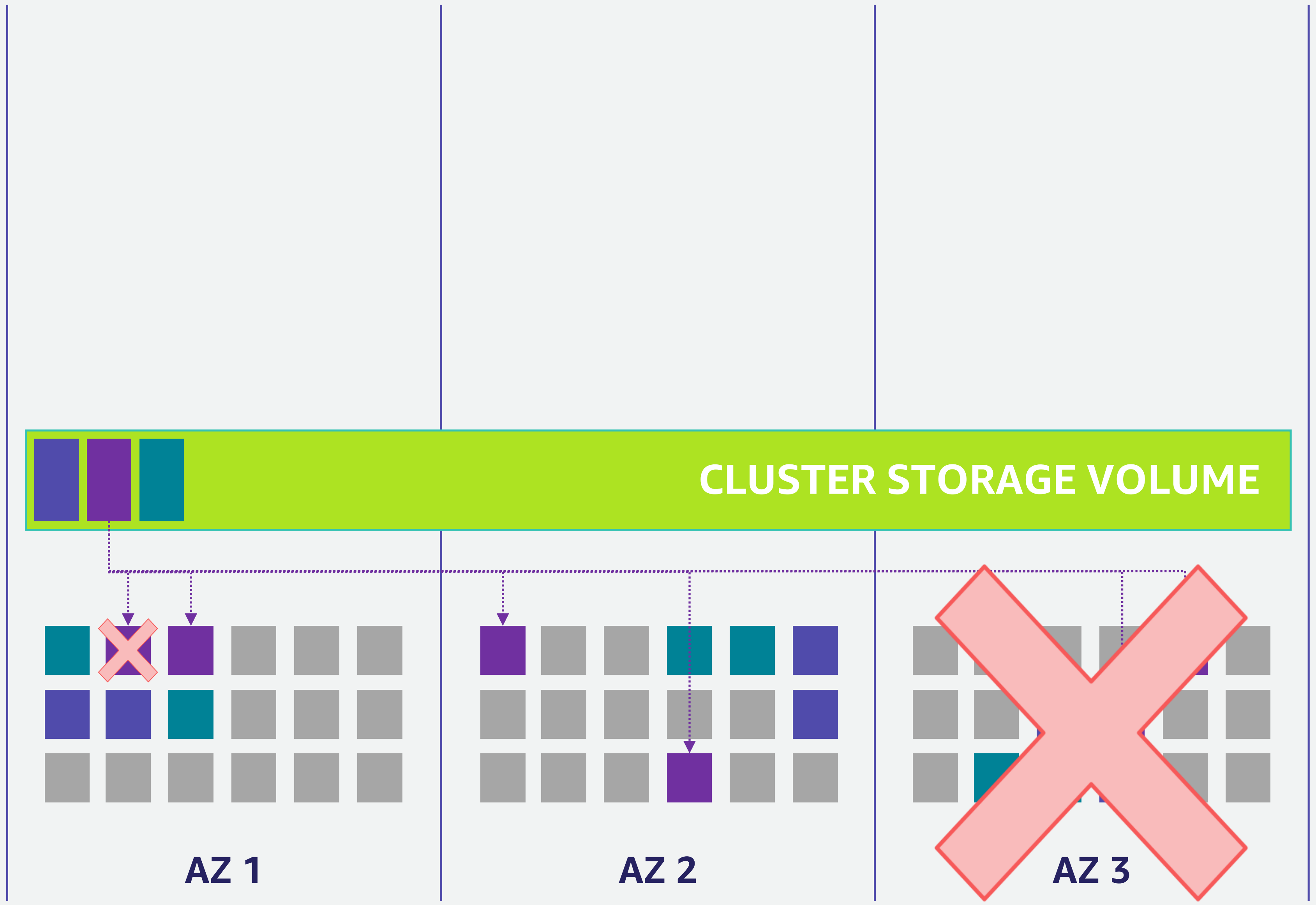
Data is written in 10 GB "protection groups," growing automatically when needed



High Availability

High durability storage system, tolerant of AZ+1 failures

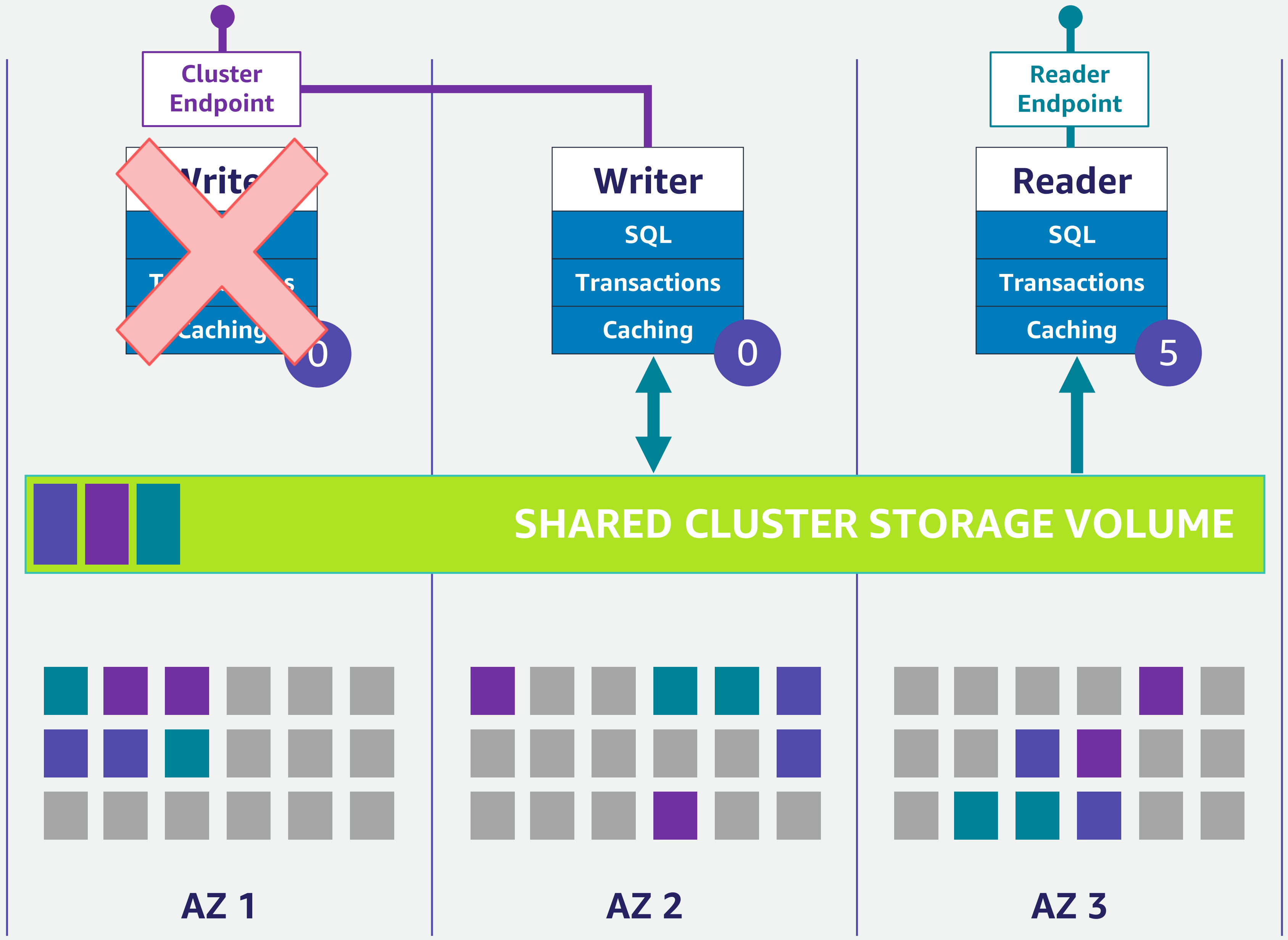
- Using quorum model for writes and reads:
  - 4 out of 6 for writes
  - 3 out of 6 for reads (recovery)
- Maintains write capability if an AZ fails, maintains read capability if AZ + 1 storage node fails.
- Self-healing architecture rebalances hot storage nodes, rebuilds segments from failed hardware
- Peer to peer “gossip protocol” is used for repairs



More: https://aws.amazon.com/blogs/database/amazon-aurora-under-the-hood-quorum-and-correlated-failure/

Tolerating compute failures

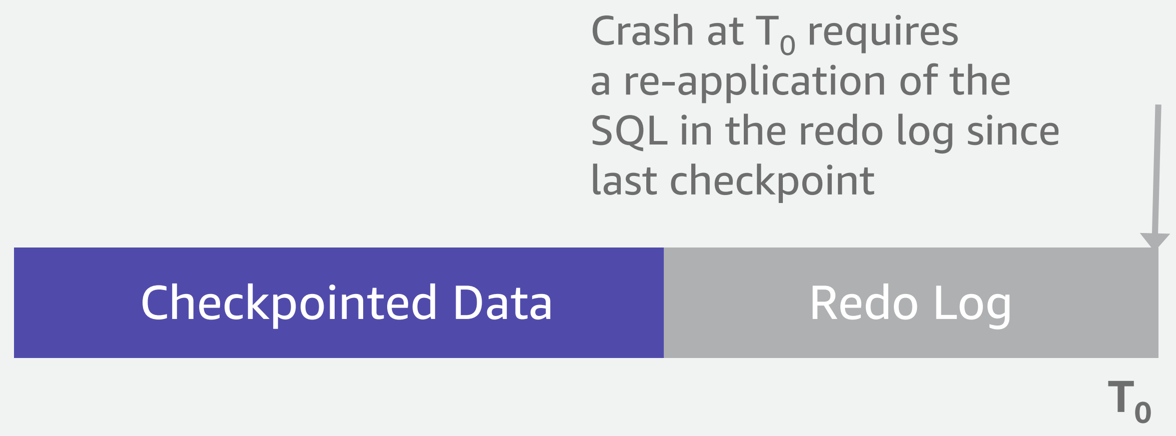
- Any reader instance can be promoted to writer/primary
- **Failover tier** determines preference on failover reader candidates. Lower values represent failover preference
- Failed instances will be replaced after failover and come online as readers
- In-Region readers do not reboot on writer restart/failover (In Aurora MySQL version 2.10 and higher)

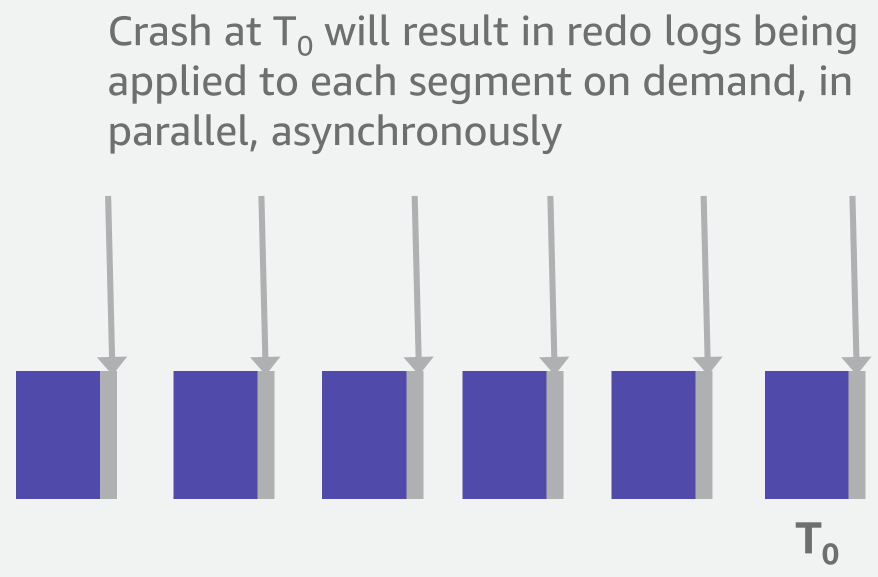


Crash recovery

- **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

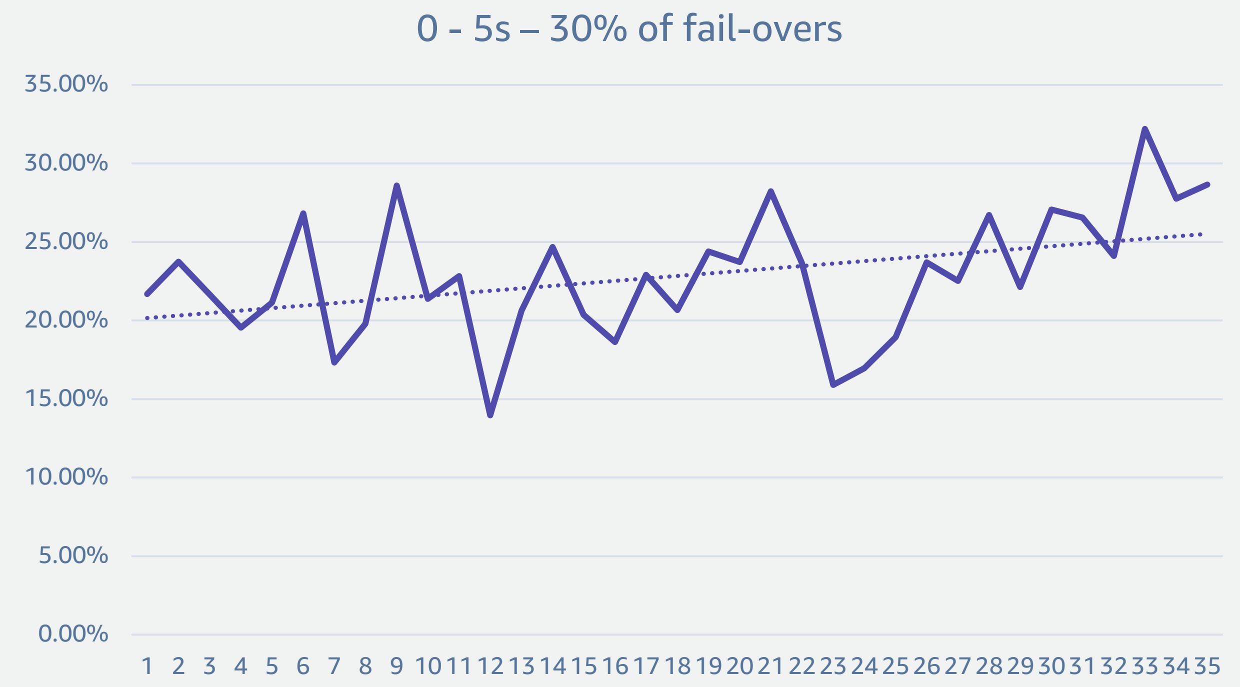

Crash at T_0 requires a re-application of the SQL in the redo log since last checkpoint
Checkpointed Data
Redo Log
T_0

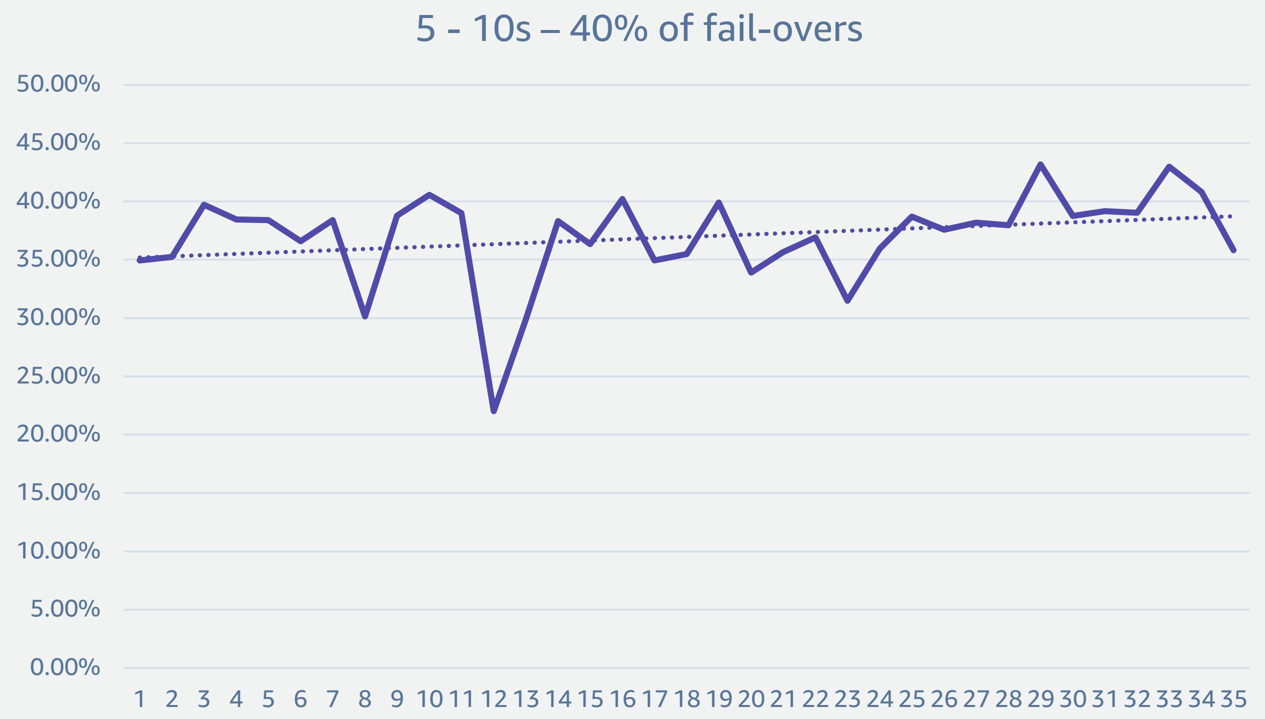

Crash at T_0 will result in redo logs being applied to each segment on demand, in parallel, asynchronously
T_0

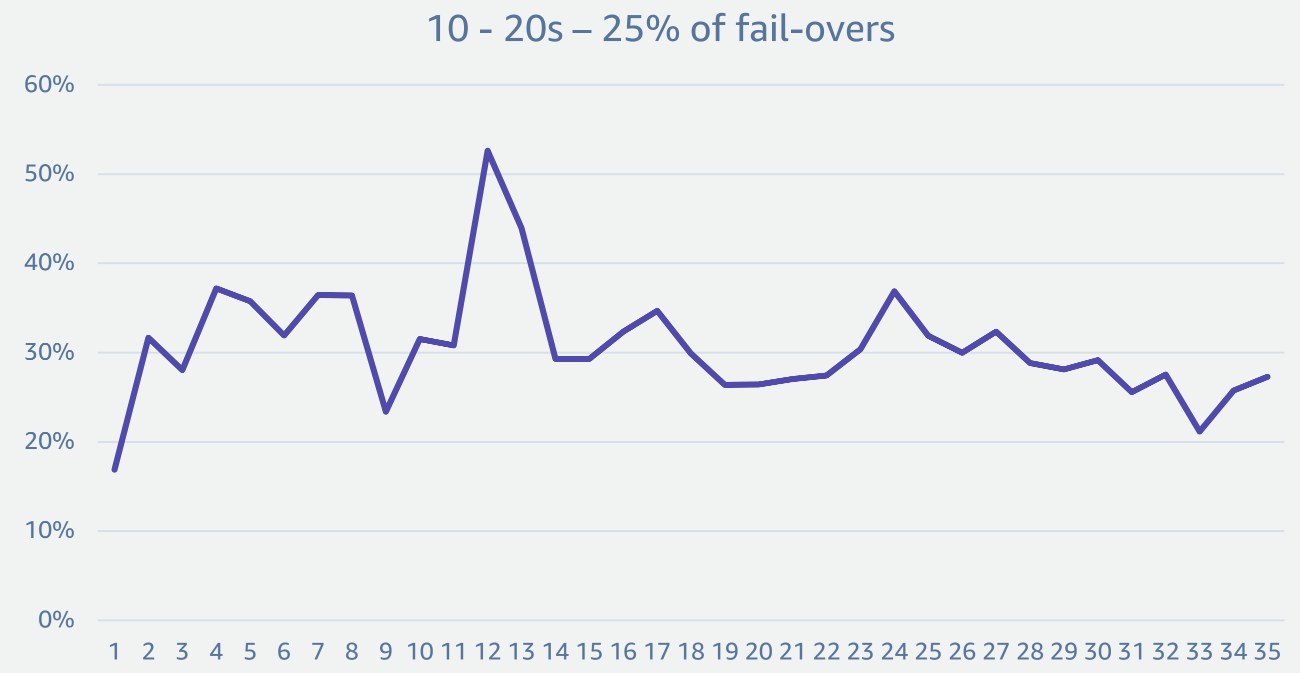
MySQL Note: Binlogs (primary), undo logs, catalog, FTS indexes, XA still require crash recovery

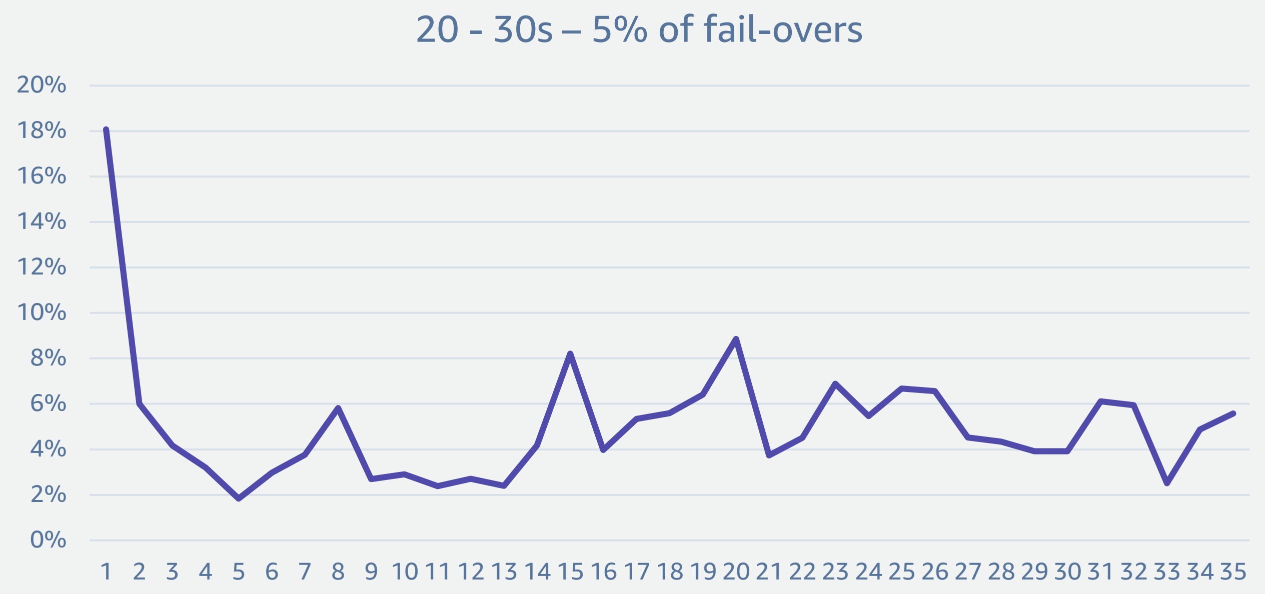
More: https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/Aurora.Overview.StorageReliability.html#Aurora.Overview.CrashRecovery

When the database fails – recovery is fast <30 seconds







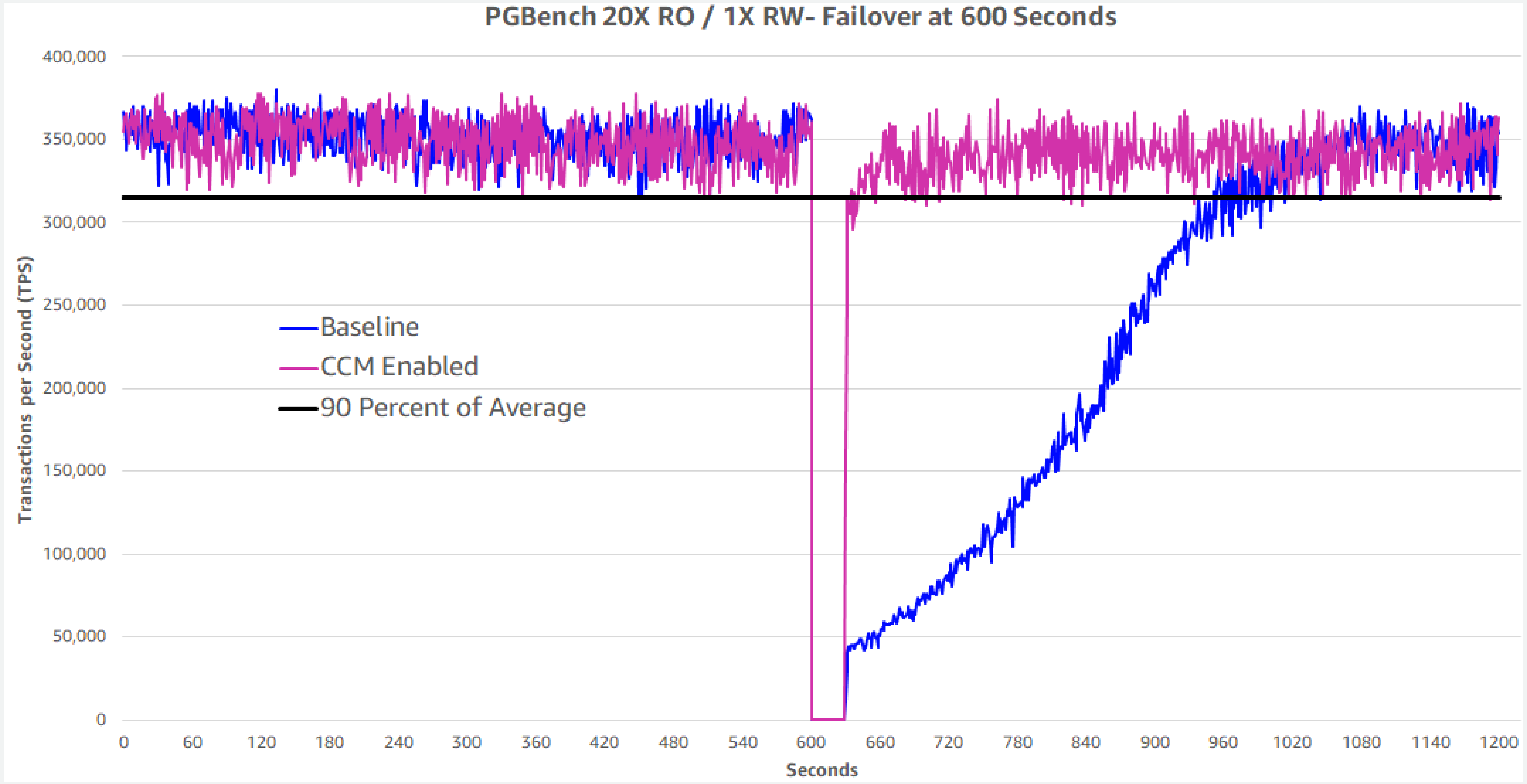


Cluster Cache Management(CCM) with Aurora PostgreSQL

- CCM works with AZ failovers and is not supported for cross-Region failovers
- Improves the customer experience with regards to the performance of the new writer instance after AZ failover
- Lowest failover tier reader will send the set of buffers that are currently cached to the writer instance as a bloom filter (avoid duplicate sends)
- On receipt writer instance compares the blocks cached in the buffer cache and sends frequently used buffers to that reader instance
- If a failover occurs, the designated reader (now promoted to new writer) uses values in its warm cache immediately
- Available for Aurora PostgreSQL only

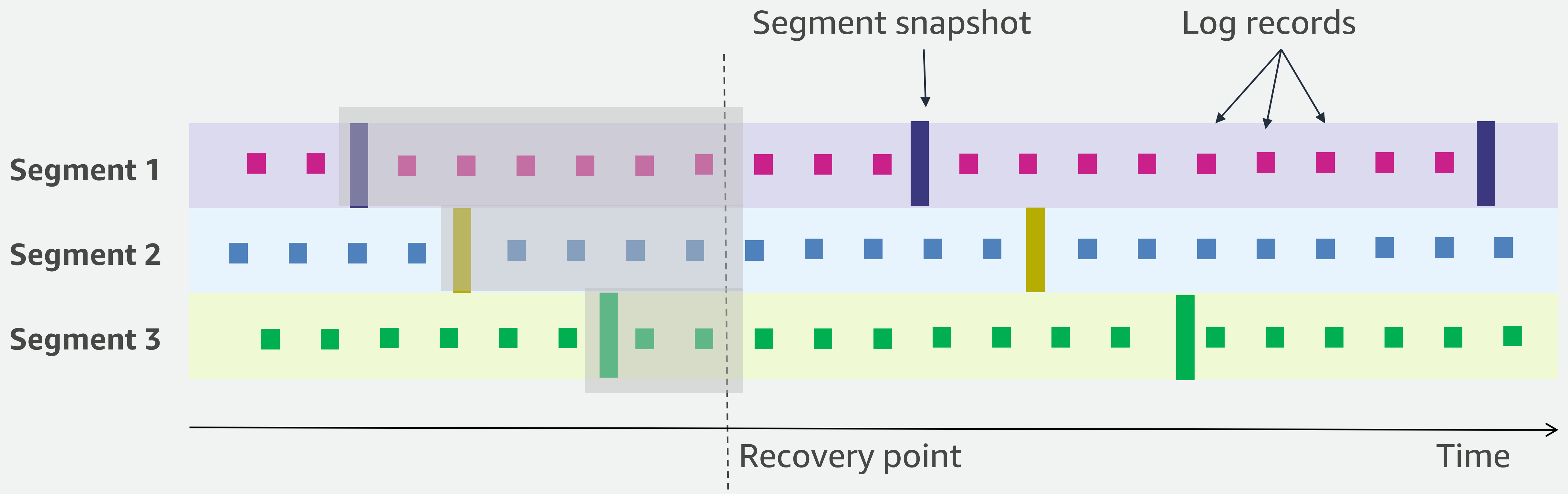
More: https://aws.amazon.com/blogs/database/introduction-to-aurora-postgresql-cluster-cache-management/

Cluster Cache Management(CCM) in practice with Aurora PostgreSQL



Disaster Recovery

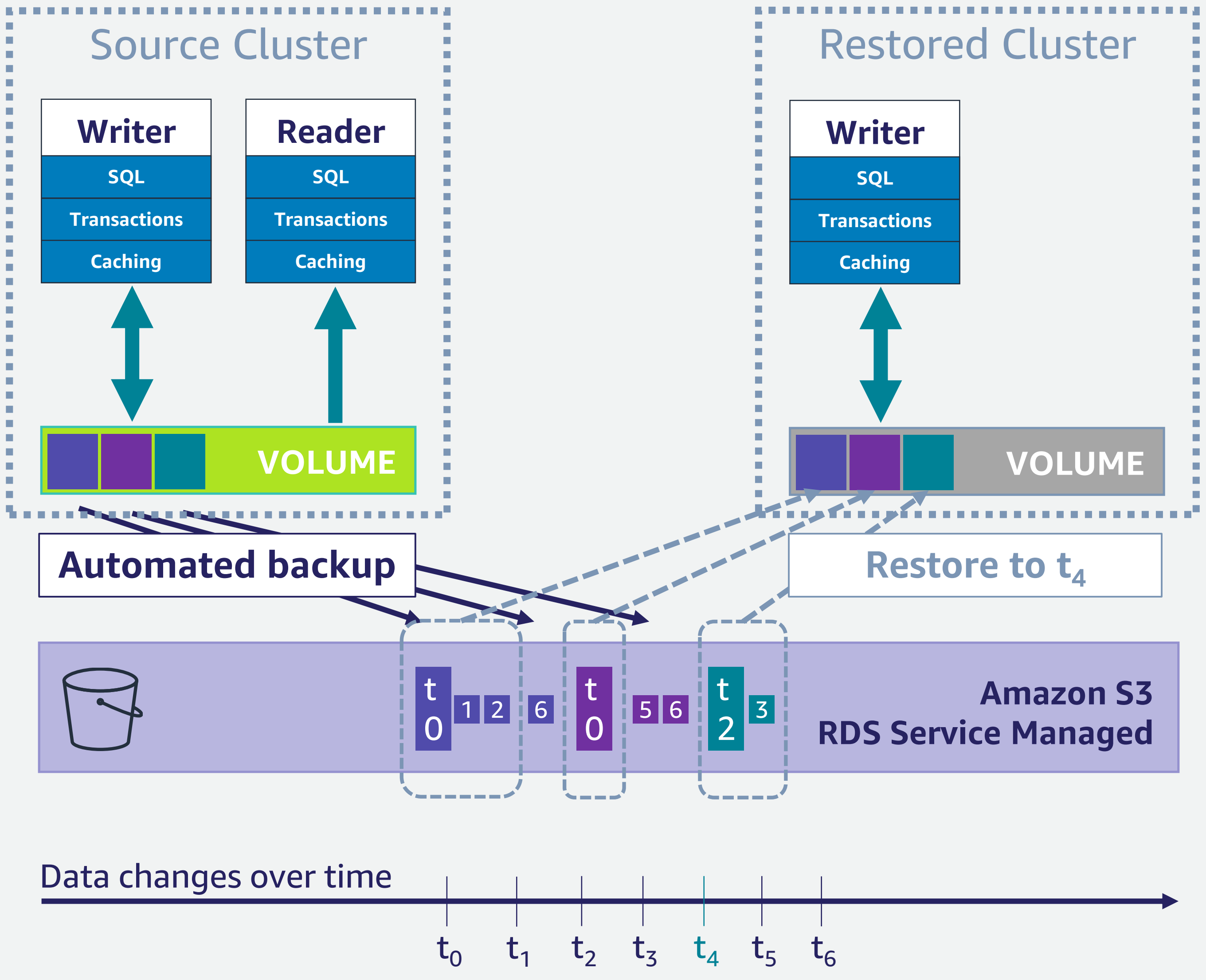
Continuous backup with point-in-time restore



- 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

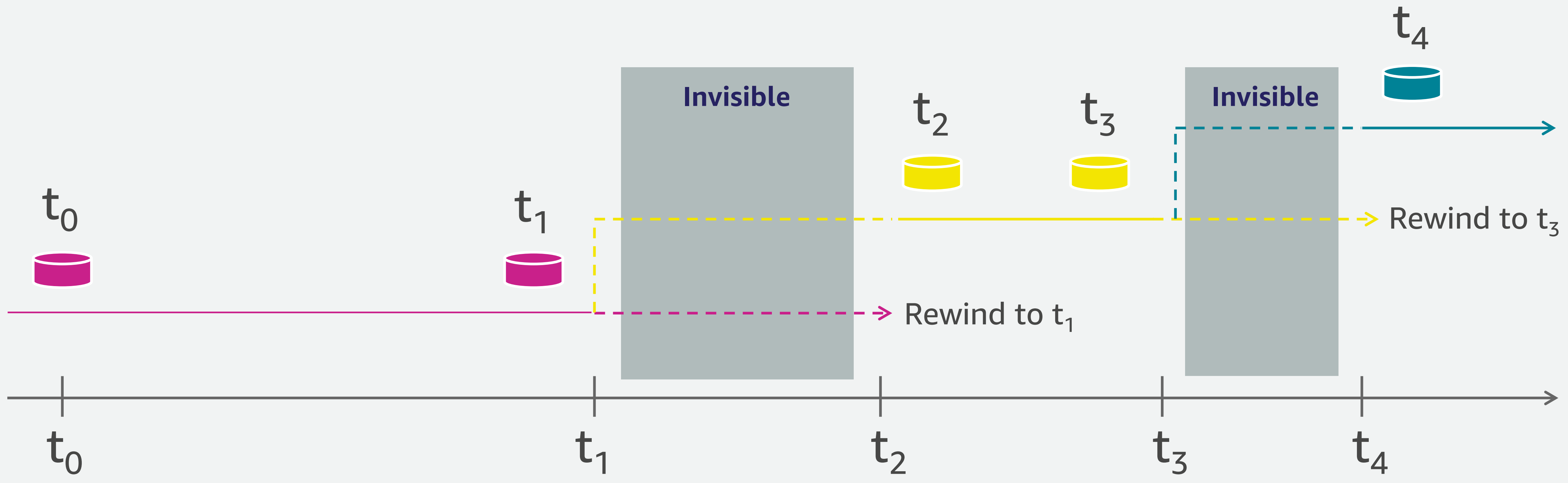
Aurora backup and restore

- **Automated backups:**
  - Between 1 and 35 days retention
  - Recover up to the last ~5 min point in time
- **Snapshots:**
  - Instantly create user snapshots
  - No performance impact
  - Copy snapshots to another region
  - Share snapshots with other AWS accounts
- **Restore:**
  - Time depends on cluster volume size
  - Always creates a new DB cluster



More: https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/Aurora.Managing.Backups.html

Aurora Backtrack with Aurora MySQL



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

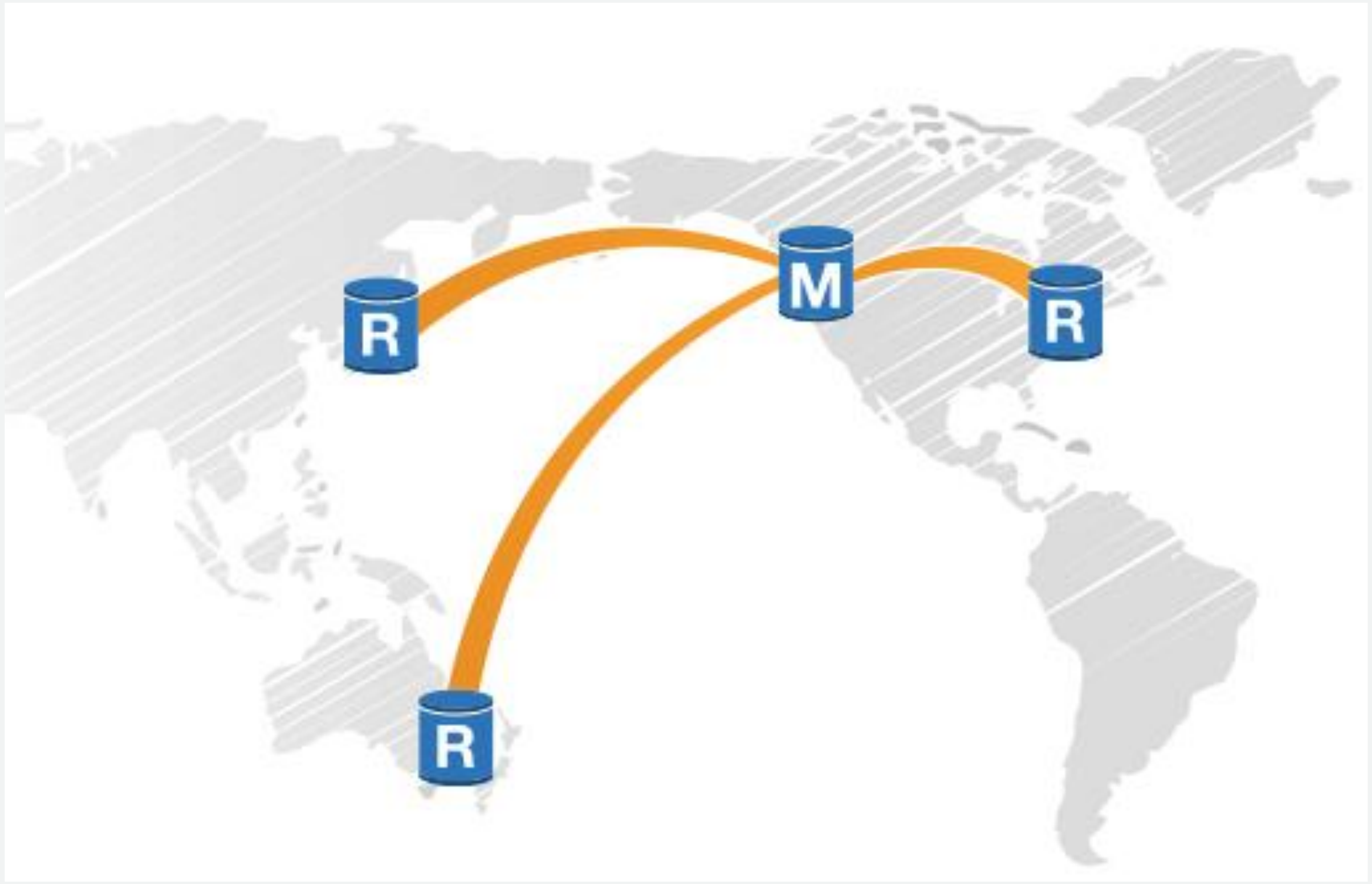
Also useful for QA (rewind your DB between test runs)

Available for Aurora MySQL only

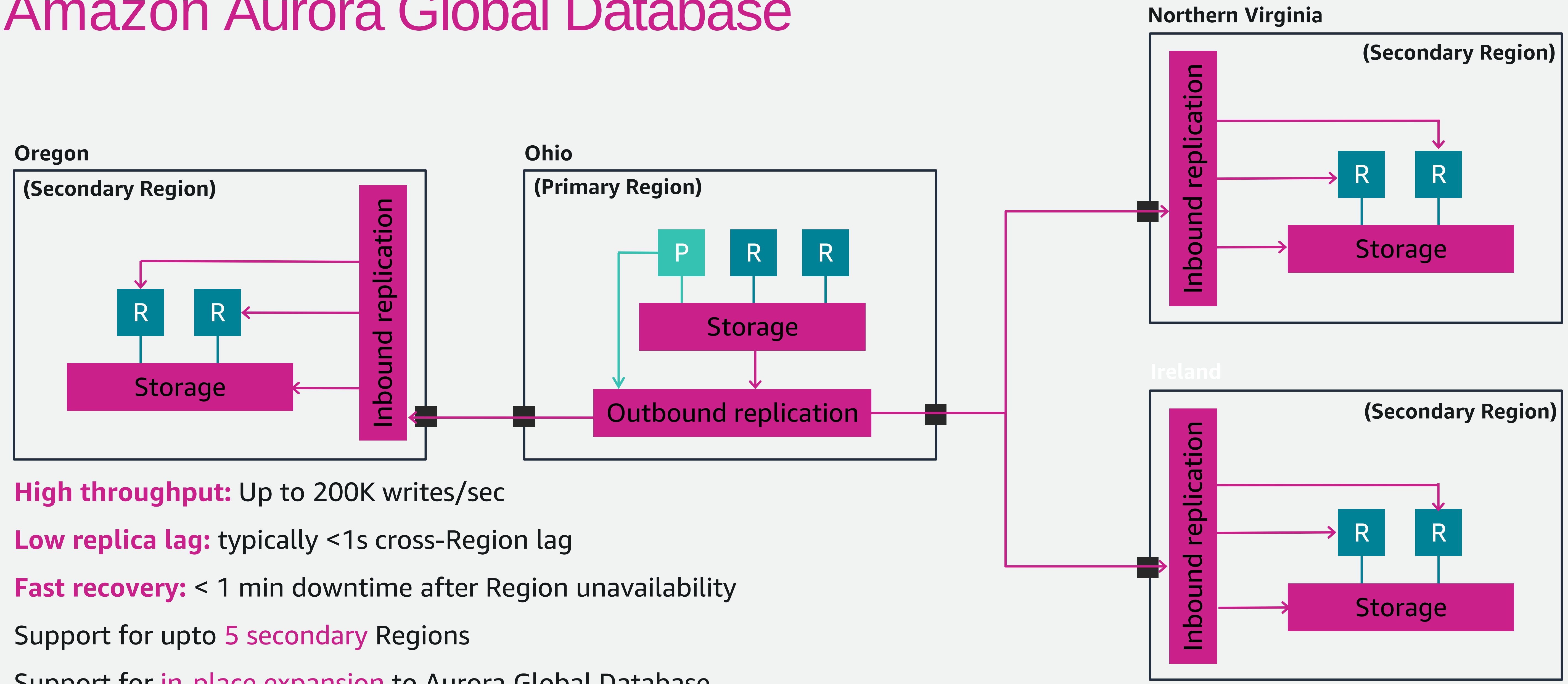
More: https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Managing.Backtrack.html

Improve DR and enhance data locality with Amazon Aurora Global Database

- Promote remote readers to a primary for faster cross-Region disaster recovery (DR)
- Bring data close to your customer's applications in different Regions
- Subsecond data replication cross-Region



Improve DR and enhance data locality with Amazon Aurora Global Database



High throughput: Up to 200K writes/sec

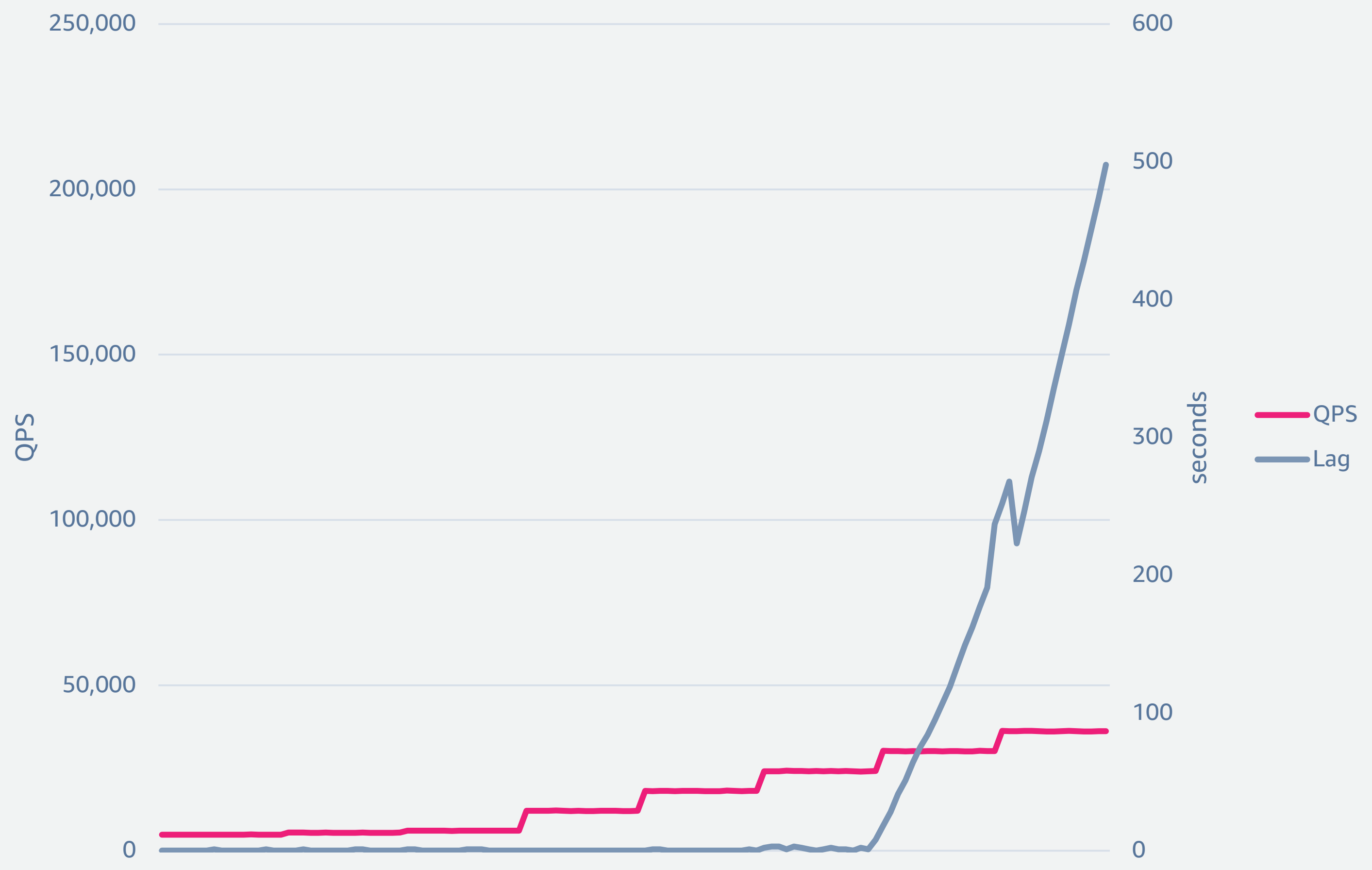
Low replica lag: typically <1s cross-Region lag

Fast recovery: < 1 min downtime after Region unavailability

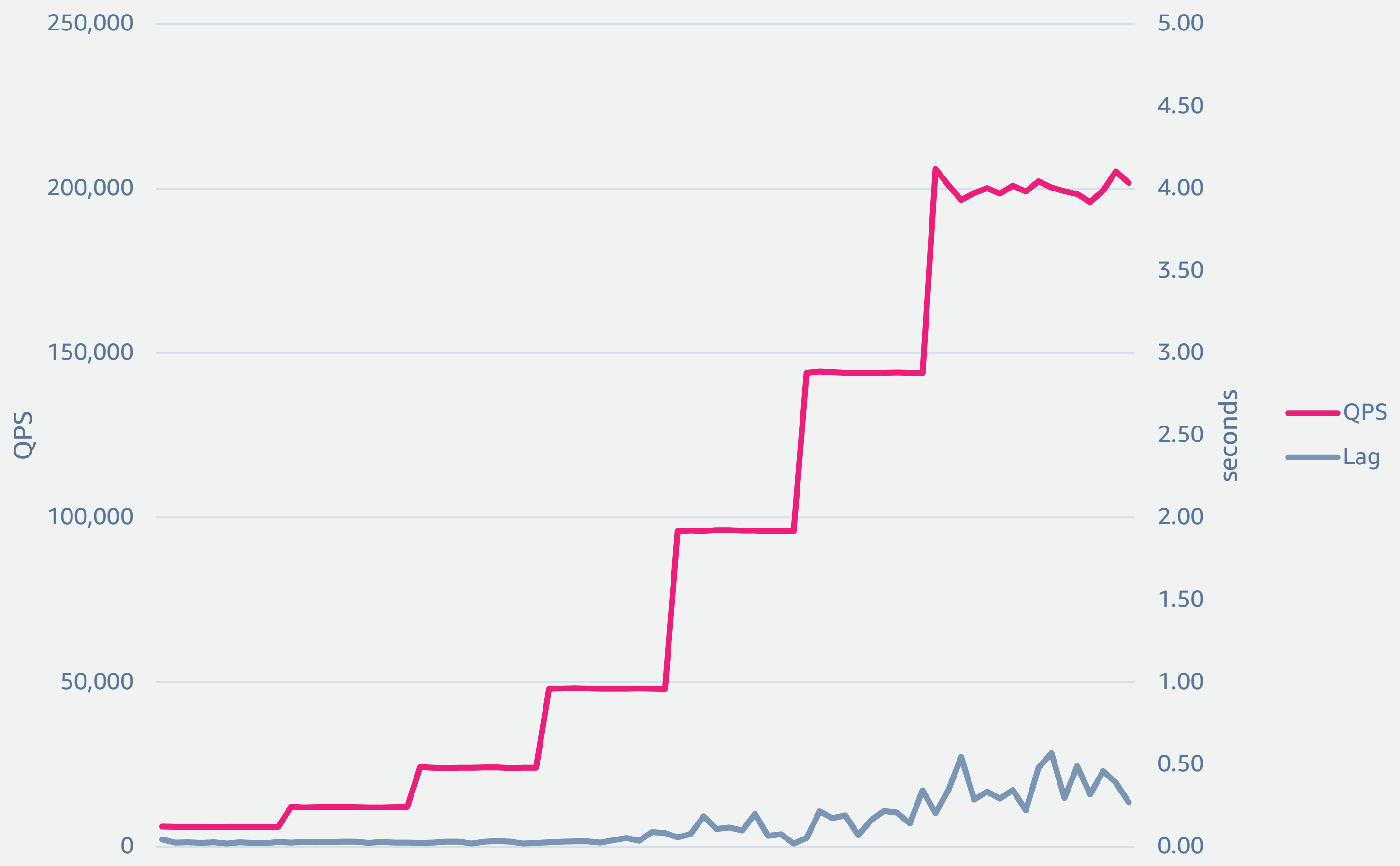
Support for upto 5 secondary Regions

Support for in-place expansion to Aurora Global Database

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

Managed Recovery point Objective (RPO) with Amazon Aurora Global Database

For applications with critical RPO requirements

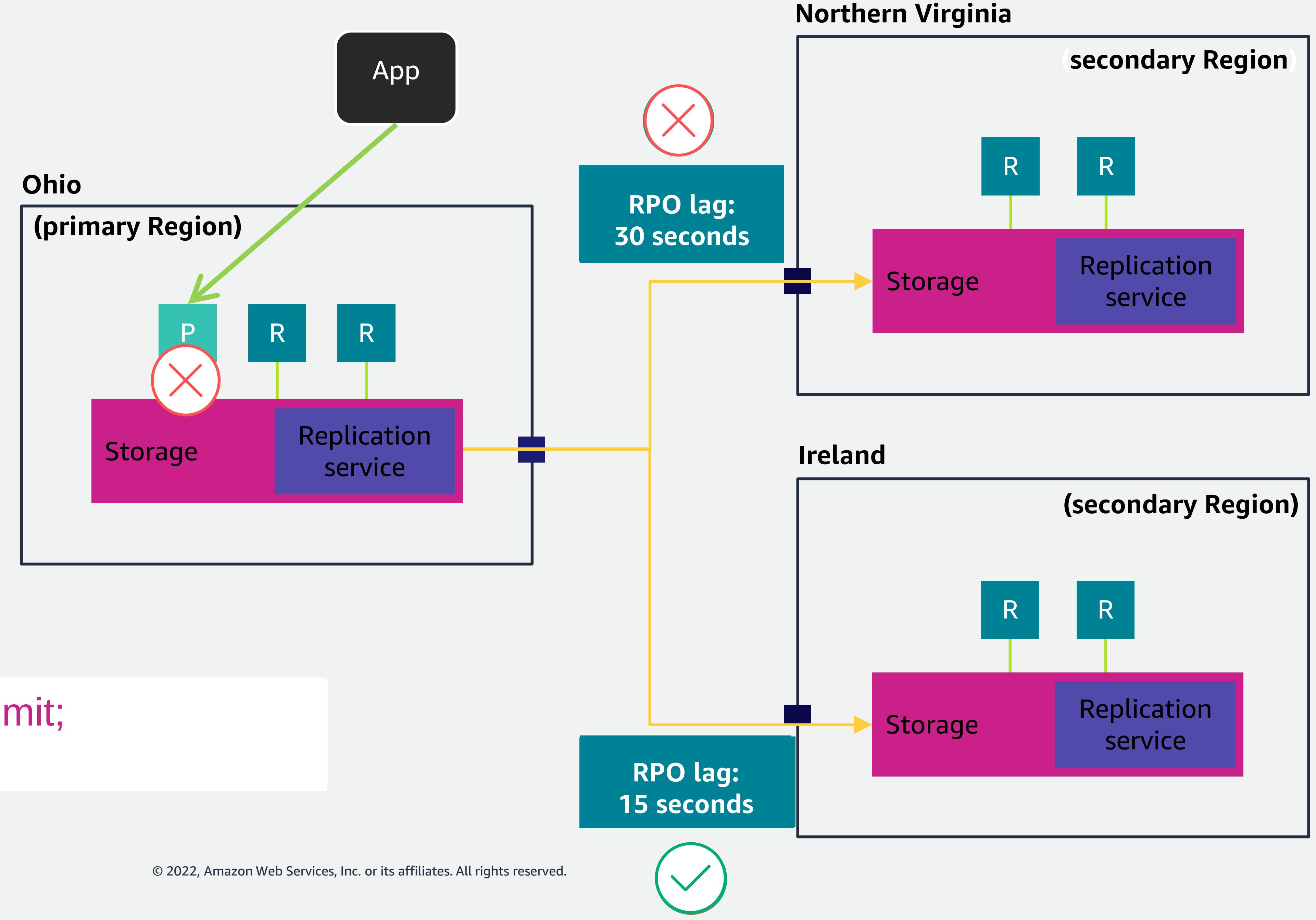
Available for Aurora PostgreSQL only

You define the maximum RPO that you allow

If RPO lag in **all** secondary Regions exceeds the limit, Aurora pauses writes until at least one Region catches up

Let's see an example where we set RPO = 20 seconds
→ Lag is within the limit

→ Lag in Ireland is back under the limit; writing has been resumed



Diving deeper into Amazon Aurora Global Database

- Up to 5 secondary regions
- Up to 90 Aurora Replicas across 6 Regions
- Replication is asynchronous, subject to lag (AuroraGlobalDBReplicationLag)
- Each secondary DB cluster (region), reduces the number of reader instances
- Optimized, parallelized and encrypted network flows between regions

More: https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/aurora-global-database.html

Global Database Write Forwarding with Aurora MySQL

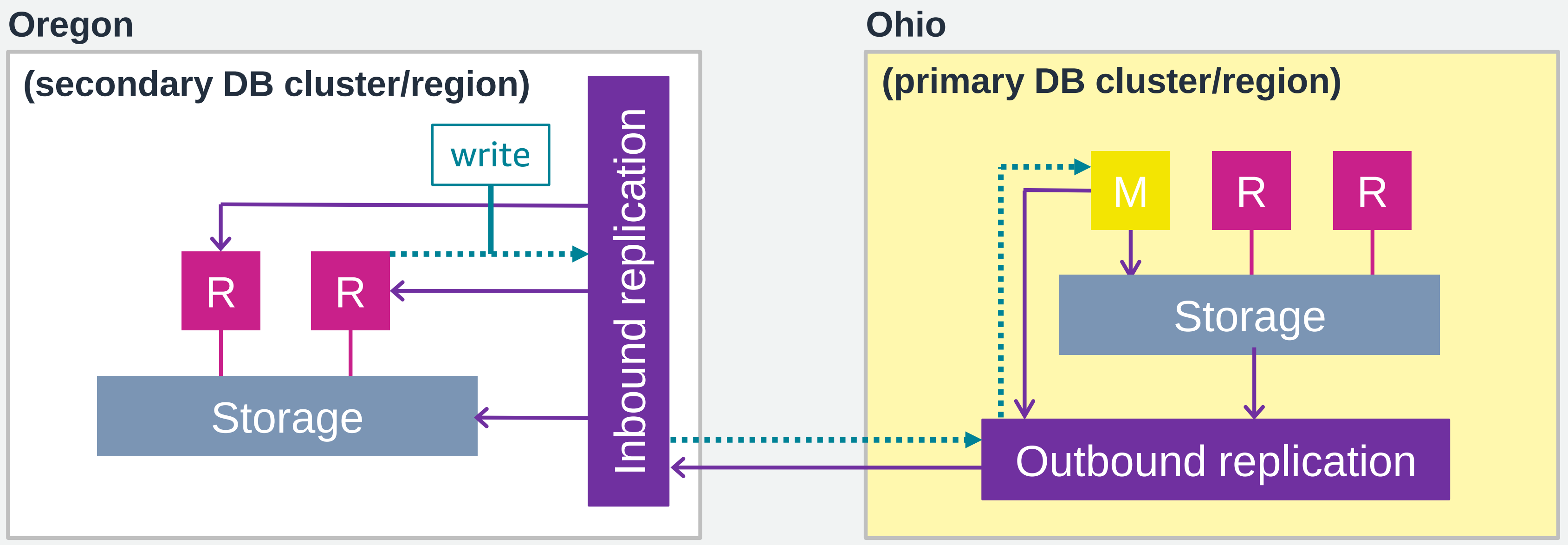
Readers in secondary DB clusters accept writes & and forward them to the primary DB cluster writer instance

Enable feature at secondary cluster level

Enable capability at session level, using:
SET aurora_replica_read_consistency

3 consistency levels:
EVENTUAL, SESSION, GLOBAL

Configurable max forwarded concurrent sessions, and session timeout



More: https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/aurora-global-database-write-forwarding.html

Planned Failover with Amazon Aurora Global Database

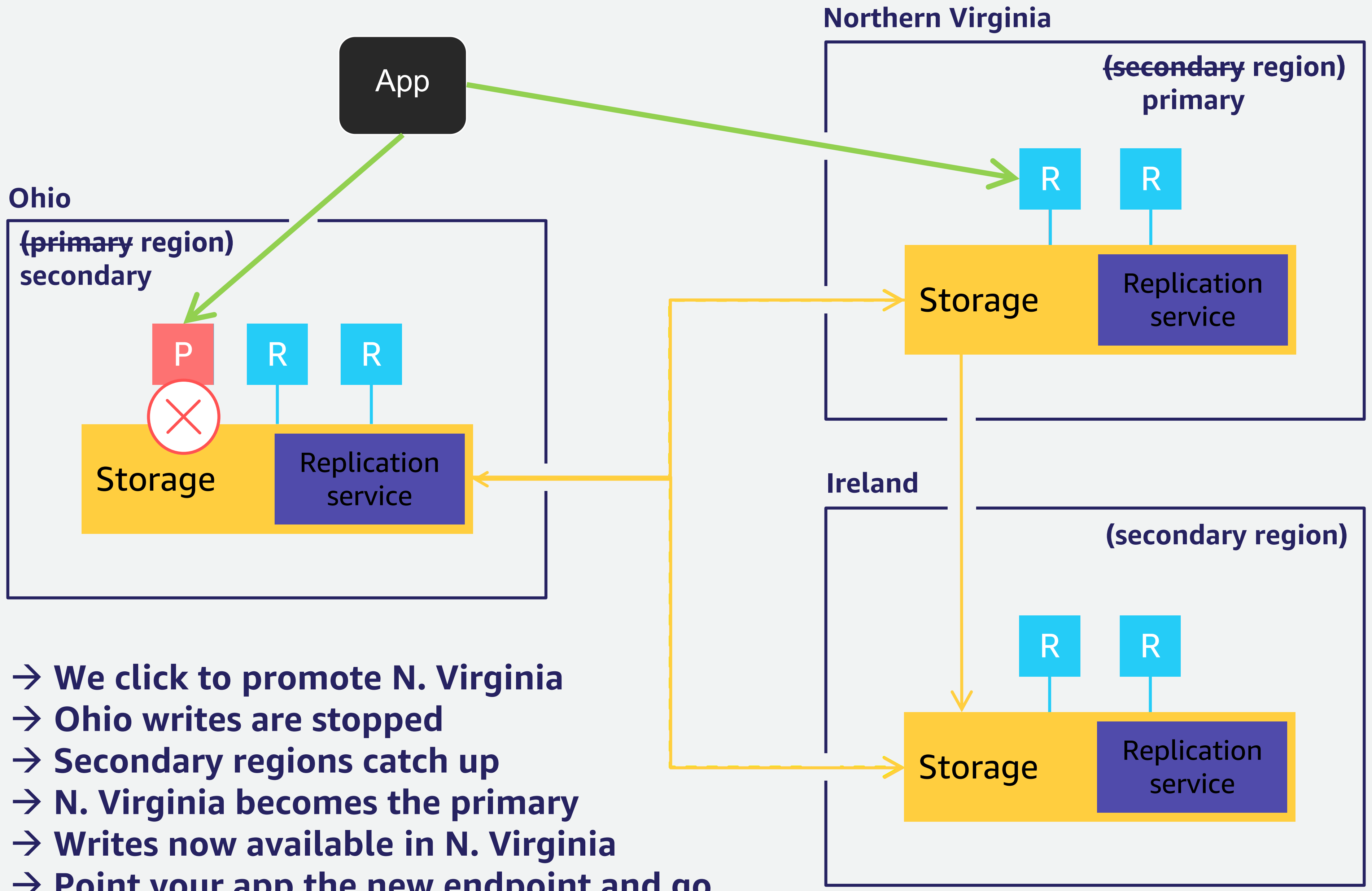
Automated way to promote secondary region to primary

For planned migration of primary.

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
→ Secondary regions catch up
→ N. Virginia becomes the primary
→ Writes now available in N. Virginia
→ Point your app the new endpoint and go

Unplanned Failover with Amazon Aurora Global Database

- Used to recover from an unplanned outage in an AWS region
- RPO depends on the AuroraGlobalDBReplicationLag metric value at the time of failure
- RTO depends on how quickly you can perform the manual failover related tasks
- Detach & Promote a secondary Region to be the primary
  - Stop writes to the Primary
  - Identify a secondary region to use as the new primary DB cluster based on least replication lag
  - Detach & promote the secondary region Aurora cluster
- Point your application to the new primary Aurora cluster
- Add secondary AWS regions as needed to re-create the global database topology

Testing workload fault resilience

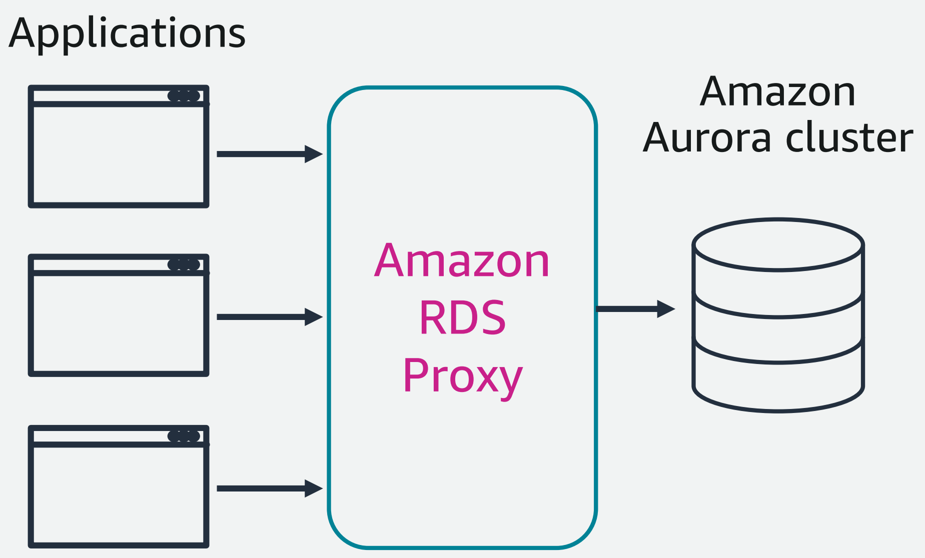
- **OBJECTIVES**
  - Understand app behavior
  - Understand failure duration
  - Understand recovery process
- **TYPES OF DOWNTIME**
  - Scheduled: software updates, instance scaling, non-dynamic parameter changes, hardware maintenance/retirement
  - Unscheduled: substrate failures (hardware, network, power), workload anomalies, overloads, OOM, unsafe usage

- **SERVER-SIDE MECHANISMS**
  - Instance reboots (API) Ⓦ Ⓡ
  - Manual failover (API) Ⓦ
  - Failure injection queries Ⓦ Ⓡ Ⓢ
  - Promotion of remote replica (API) Ⓦ Ⓡ
  - Contact Support/Service Team Ⓦ
- **CLIENT-SIDE MECHANISMS**
  - Networking disruption (NACLs) Ⓦ Ⓡ
  - Endpoint disruption (CNAME) Ⓦ Ⓡ
  - Connection pool disruption/reset Ⓦ Ⓡ
  - Application level fault injection Ⓦ Ⓡ

Ⓦ - Writer Ⓡ - Reader Ⓢ - Storage

More: https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Managing.FaultInjectionQueries.html

Improve application scalability, availability, and security with Amazon RDS Proxy



Connection pooling

Supports a large number of application connections

Deployed across multiple AZs and supports failover without losing a connection

Integrates with AWS Secrets Manager and AWS Identity and Access Management (IAM)

Get started with a few clicks in the console

More: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/rds-proxy.html

Thank You!