What’s New in Amazon Aurora

Steve Abraham
Principal Solutions Architect

Agenda

Introduction	What is Aurora?
Performance	Aurora performance: design and enhancements
Availability	Aurora availability architecture
Recent Announcements	New features and capabilities

Amazon Aurora ……

Databases reimagined for the cloud

☑ **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

Delivered as a **managed** service

Re-imagining the relational database

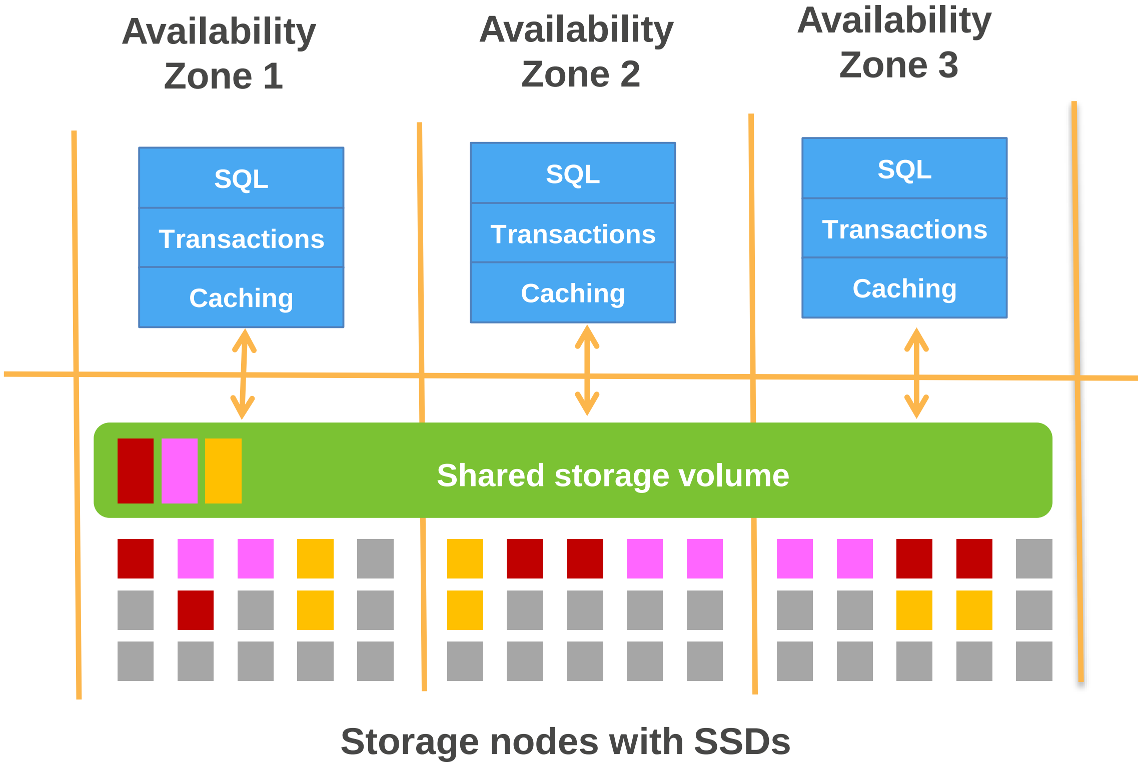
1 Scale-out, distributed, multi-tenant design

2 Service-oriented architecture leveraging AWS services

3 Fully managed service – automate administrative tasks

Scale-out, distributed, multi-tenant architecture

- Purpose-built log-structured distributed storage system designed for databases
- Storage volume is striped across hundreds of storage nodes distributed over 3 different Availability Zones
- Six copies of data, two copies in each Availability Zone to protect against AZ+1 failures
- Plan to apply same principles to other layers of the stack



Leveraging cloud eco-system

AWS Lambda

Invoke AWS Lambda events from stored procedures/triggers.

Amazon S3

Load data or Select into Amazon S3; store snapshots and backups in S3.

IAM

Use AWS Identity & Access Management (IAM) roles to manage database access control.

Amazon CloudWatch

Upload systems metrics and audit logs to Amazon CloudWatch.

MySQL compatibility

MySQL 5.6 / InnoDB compatible

- No application compatibility issues reported in last 18 months
- MySQL ISV applications run pretty much as is
- Back ported 100 fixes from different MySQL releases

MySQL 5.7 compatibility coming soon.

tableau

> “We ran our compatibility test suites against Amazon Aurora and everything just worked."
>
> - Dan Jewett, VP, Product Management at Tableau

Business Intelligence	Data Integration	Query and Monitoring
tableau	talend	Webyog
ZOOMDATA	ATTUNITY	
looker	informatica	Navicat Premium for Aurora

Who is moving to Aurora and why?

Customers using open source engines

Higher performance - up to 5x
Reduces cost at scale

Better availability and durability

Easy migration; no application change

Customers using commercial engines

1/10th of the cost; no licenses

Integration with cloud ecosystem

Comparable performance and availability

Migration tooling and services

Customers using NoSQL

Improved maintainability – Aurora automatically manages hotspots

Cost reduction – pay only for used IO; eliminate read costs for memory bound workloads

Customers

Aurora is used by:

2/3 of top 100 AWS customers

8 of top 10 gaming customers

Expedia
PEARSON Education
Nasdaq
ticketmaster
ancestry.com
FANDUEL
TalentBin BY MONSTER
INTERCOM
zynga
igt DOUBLE DOWN
gumi One Step Beyond.

Fastest growing service in AWS history

Security and compliance

- Amazon Aurora gives each database instance IP firewall protection
- Aurora offers transparent encryption at rest and SSL protection for data in transit
- Amazon VPC lets you isolate and control network configuration and connect securely to your IT infrastructure
- AWS Identity and Access Management provides resource-level permission controls

AICPA Service Organization Control Reports
AICPA
SERVICE ORGANIZATIONS
SOC
aicpa.org/soc
Formerly SAS 70 Reports
SOC3
International Organization for Standardization
ISO
9001
Information Security Management System
ISO
27001
Certified
PAYMENT CARD INDUSTRY
PCI DSS
LEVEL 1 SERVICE PROVIDER
DATA SECURITY STANDARDS
HIPAA

Aurora performance

Aurora performance tenets

DO LESS WORK

Do fewer IOs

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

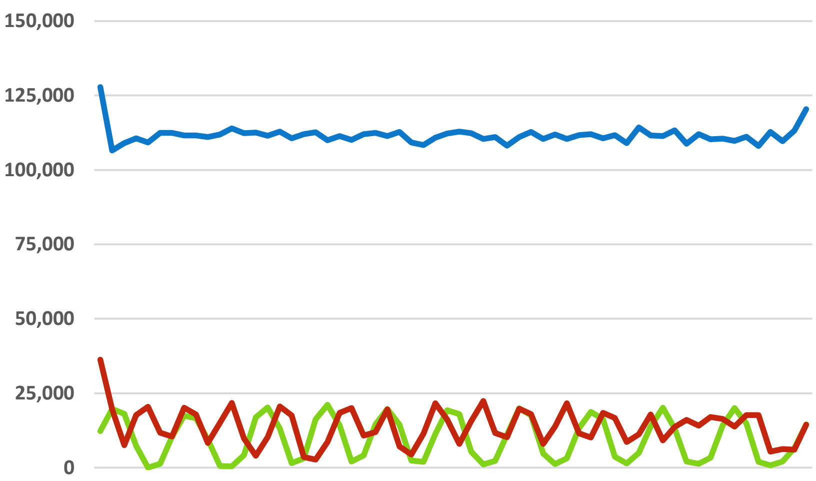
DATABASES ARE ALL ABOUT I/O

NETWORK-ATTACHED STORAGE IS ALL ABOUT PACKETS/SECOND

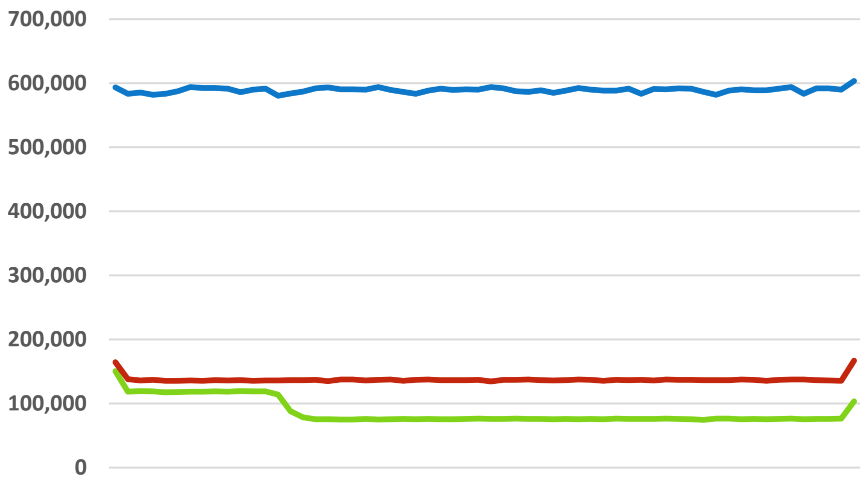
HIGH-THROUGHPUT PROCESSING IS ALL ABOUT CONTEXT SWITCHES

5X more throughput than MySQL 5.6 & 5.7

WRITE PERFORMANCE



READ PERFORMANCE



MySQL SysBench results

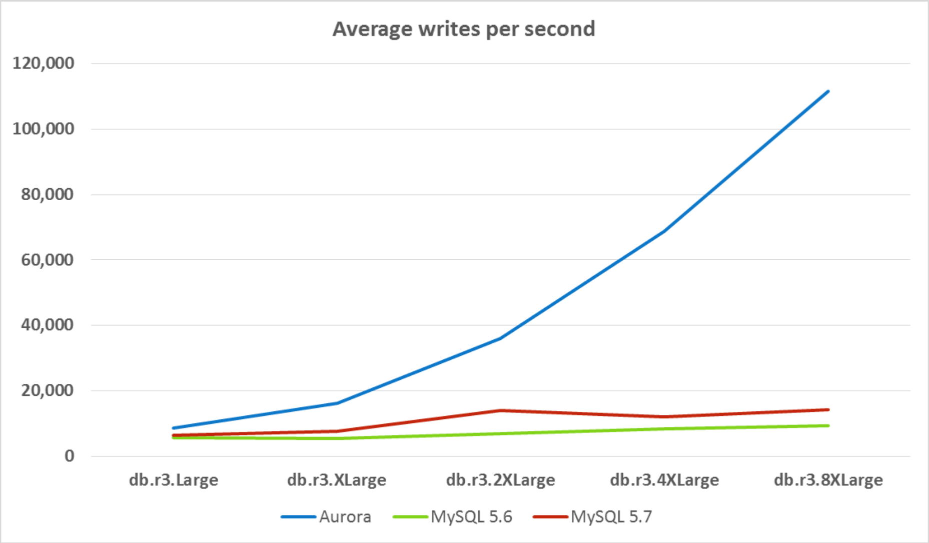
R3.8XL: 32 cores / 244 GB RAM

Aurora 🟦 **MySQL 5.6** 🟩 **MySQL 5.7** 🟥

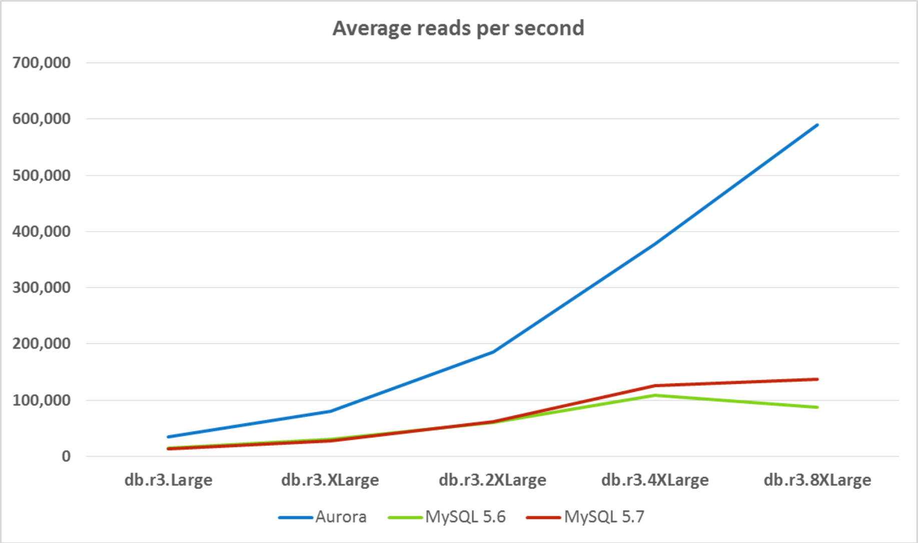
Five times higher throughput than stock MySQL based on industry standard benchmarks.

Scaling with instance sizes

WRITE PERFORMANCE



READ PERFORMANCE

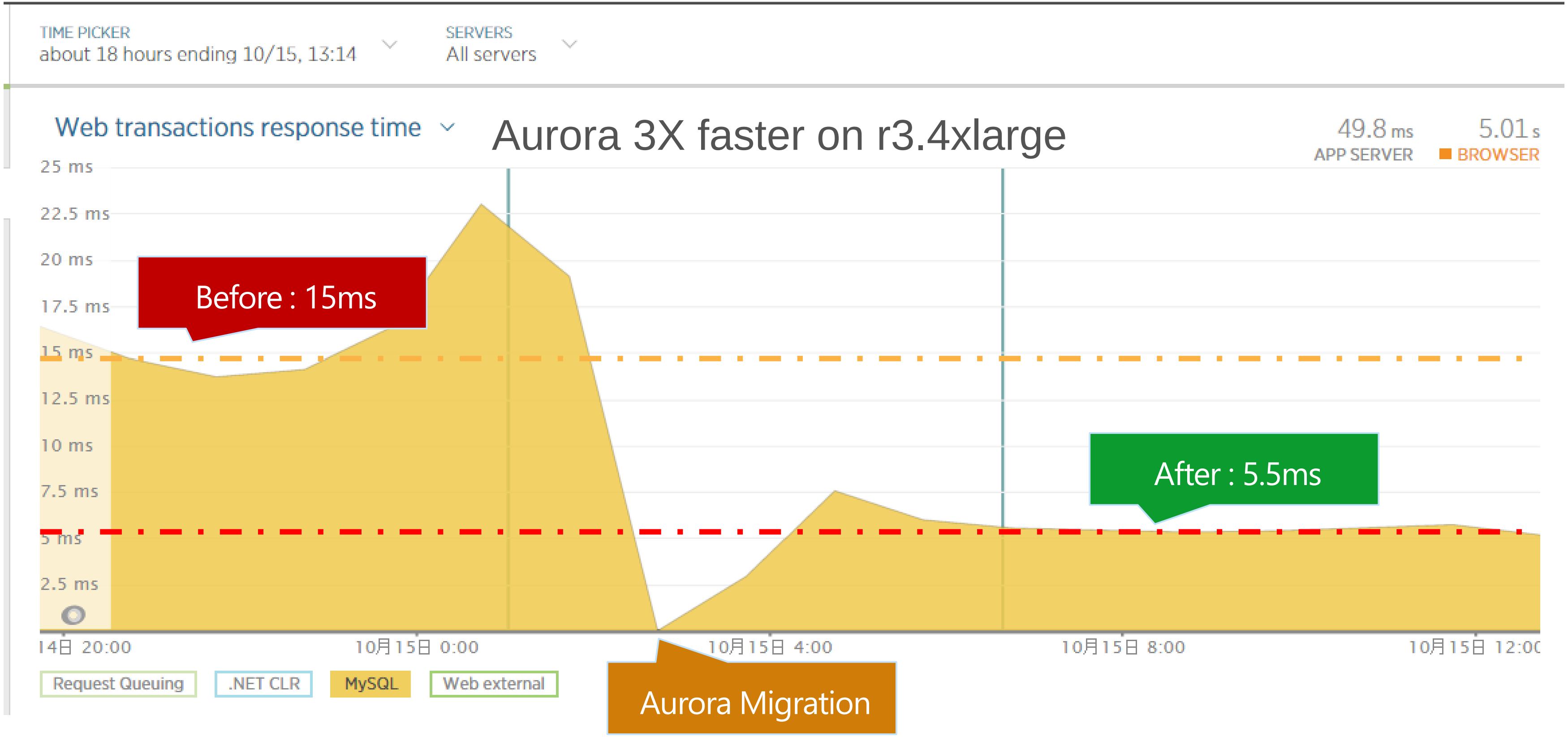


Aurora 🟦 **MySQL 5.6** 🟩 **MySQL 5.7** 🟥

Aurora scales with instance size for both read and write.

Real-life data – gaming workload

Aurora vs. RDS MySQL – r3.4XL, MAZ



TIME PICKER
about 18 hours ending 10/15, 13:14

SERVERS
All servers

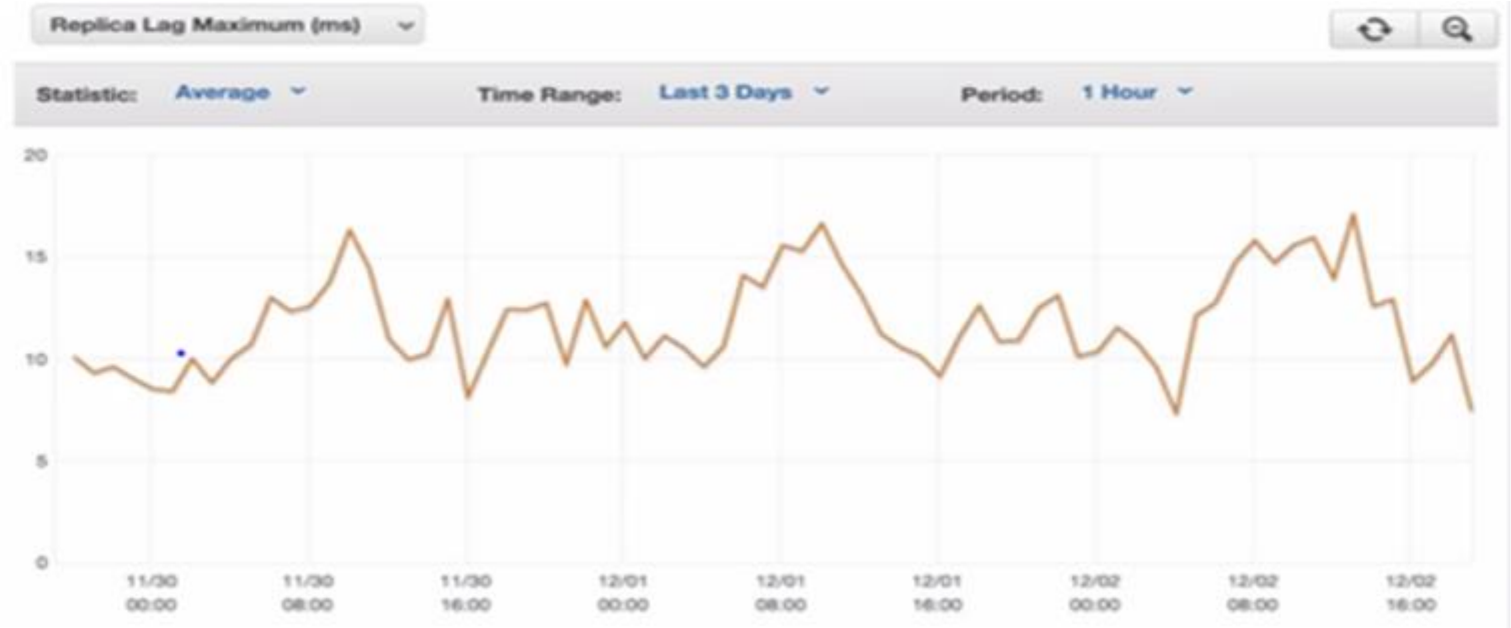
Aurora 3X faster on r3.4xlarge

Before : 15ms

After : 5.5ms

Aurora Migration

Real-life data - read replica latency



Replica Lag Maximum (ms)

Statistic: Average

Time Range: Last 3 Days

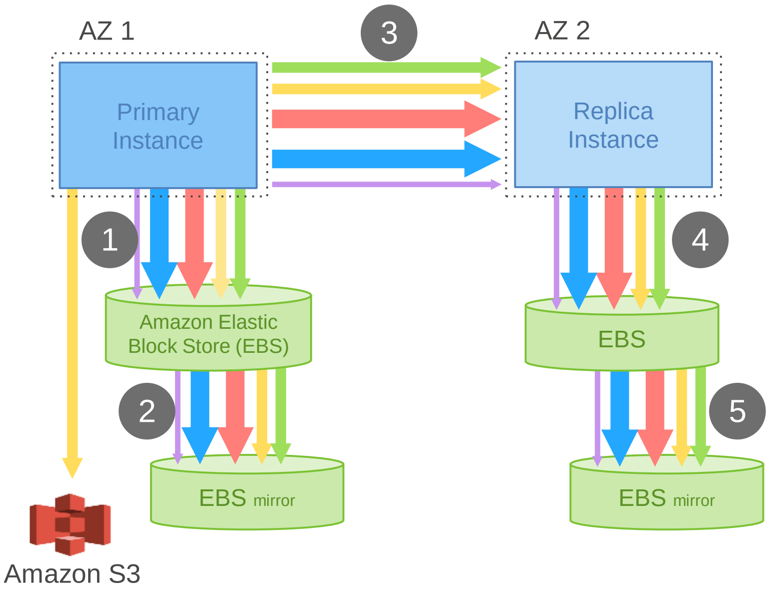
Period: 1 Hour

GoGuardian

“In MySQL, we saw replica lag spike to almost 12 minutes which is almost absurd from an application’s perspective. With Aurora, the maximum read replica lag across 4 replicas never exceeded 20 ms.”

IO traffic in MySQL

MYSQL WITH REPLICA



IO FLOW

Issue write to Amazon EBS – EBS issues to mirror, ack when both done

Stage write to standby instance using synchronous replication

Issue write to EBS on standby instance

OBSERVATIONS

Steps 1, 3, 4 are sequential and synchronous

This amplifies both latency and jitter

Many types of writes for each user operation

Have to write data blocks twice to avoid torn writes

PERFORMANCE

780K transactions

7,388K I/Os per million txns (excludes mirroring, standby)

Average 7.4 I/Os per transaction

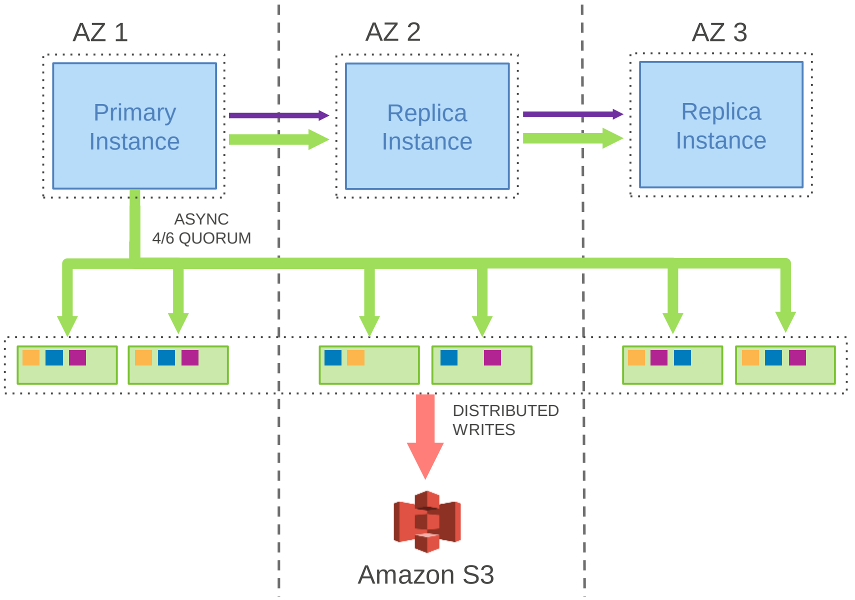
30 minute SysBench write-only workload, 100GB dataset, **RDS MultiAZ**, 30K PIOPS

TYPE OF WRITE

→ LOG   → BINLOG   → DATA   → DOUBLE-WRITE   → FRM FILES

IO traffic in Aurora

AMAZON AURORA



IO FLOW

Boxcar redo log records – fully ordered by LSN

Shuffle to appropriate segments – partially ordered

Boxcar to storage nodes and issue writes

OBSERVATIONS

Only write redo log records; all steps asynchronous

No data block writes (checkpoint, cache replacement)

6X more log writes, but **9X** less network traffic

Tolerant of network and storage outlier latency

PERFORMANCE

27,378K transactions	**35X**	**MORE**
950K I/Os per 1M txns (6X amplification)	**7.7X**	**LESS**

TYPE OF WRITE

→ LOG

→ BINLOG

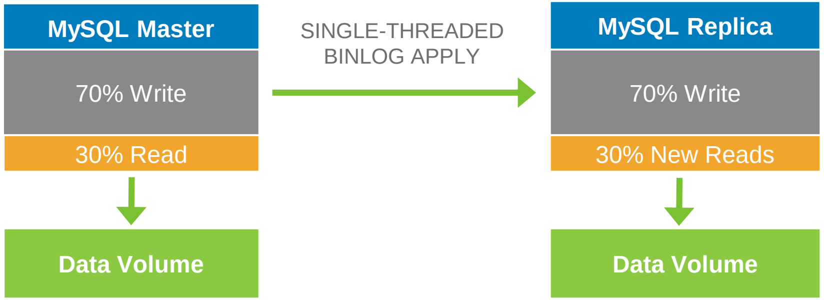
→ DATA

→ DOUBLE-WRITE

→ FRM FILES

IO traffic in Aurora Replicas

MYSQL READ SCALING



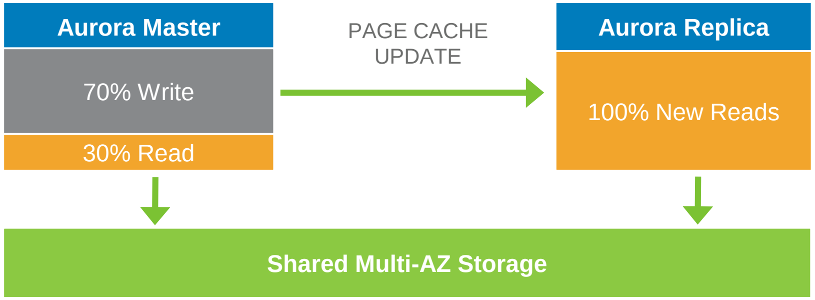
Logical: Ship SQL statements to Replica

Write workload similar on both instances

Independent storage

Can result in data drift between Master and Replica

AMAZON AURORA READ SCALING



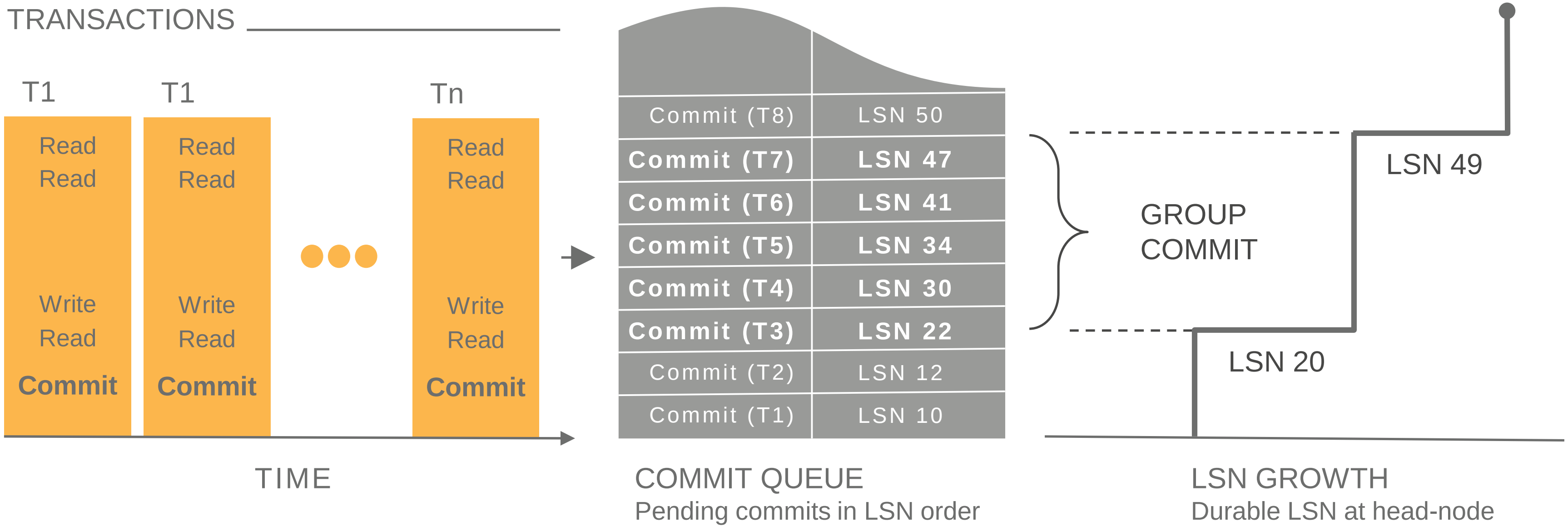
Physical: Ship redo from Master to Replica

Replica shares storage. No writes performed

Cached pages have redo applied

Advance read view when all commits seen

Asynchronous group commits



TRANSACTIONS

TIME

Commit (T8)	LSN 50
Commit (T7)	**LSN 47**
Commit (T6)	**LSN 41**
Commit (T5)	**LSN 34**
Commit (T4)	**LSN 30**
Commit (T3)	**LSN 22**
Commit (T2)	LSN 12
Commit (T1)	LSN 10

COMMIT QUEUE
Pending commits in LSN order

LSN GROWTH
Durable LSN at head-node

TRADITIONAL APPROACH

Maintain a buffer of log records to write out to disk

Issue write when buffer full or time out waiting for writes

First writer has latency penalty when write rate is low

AMAZON AURORA

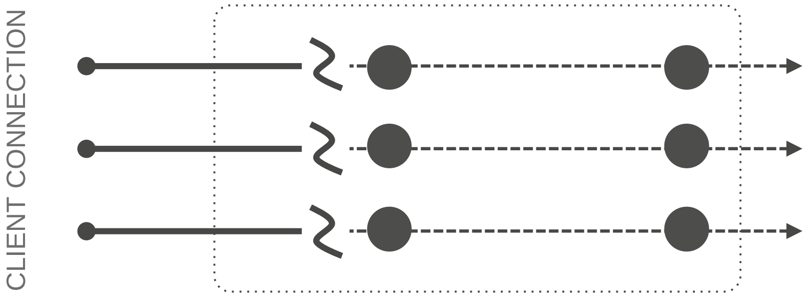
Request I/O with first write, fill buffer till write picked up

Individual write durable when 4 of 6 storage nodes ACK

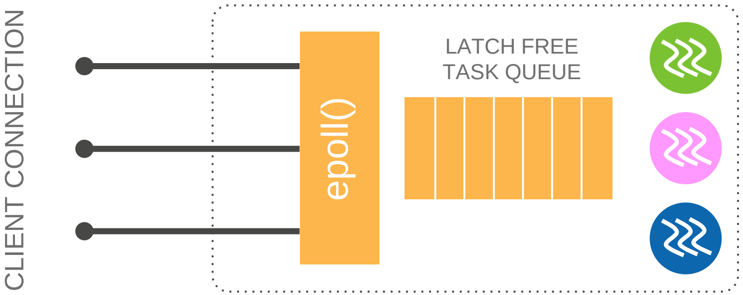
Advance DB Durable point up to earliest pending ACK

Adaptive thread pool

MYSQL THREAD MODEL



AURORA THREAD MODEL



Standard MySQL – one thread per connection

Doesn’t scale with connection count

MySQL EE – connections assigned to thread group

Requires careful stall threshold tuning

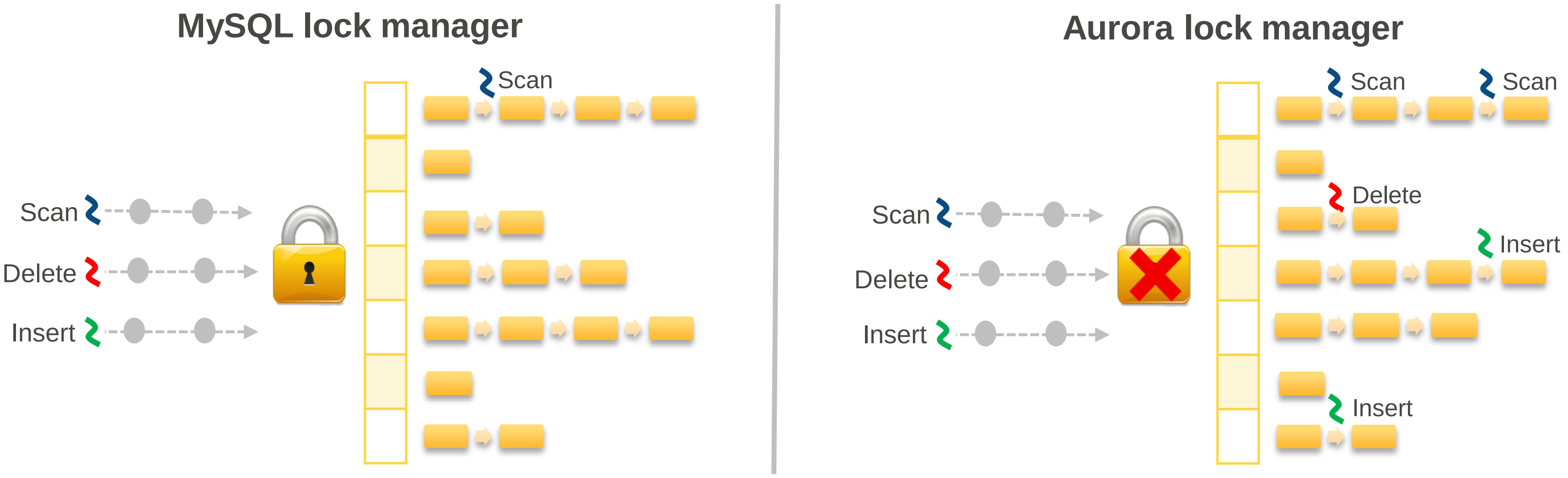
Re-entrant connections multiplexed to active threads

Kernel-space epoll() inserts into latch-free event queue

Dynamically size threads pool

Gracefully handles 5000+ concurrent client sessions on r3.8xl

Aurora lock management



Same locking semantics as MySQL

Concurrent access to lock chains

Multiple scanners allowed in an individual lock chains

Lock-free deadlock detection

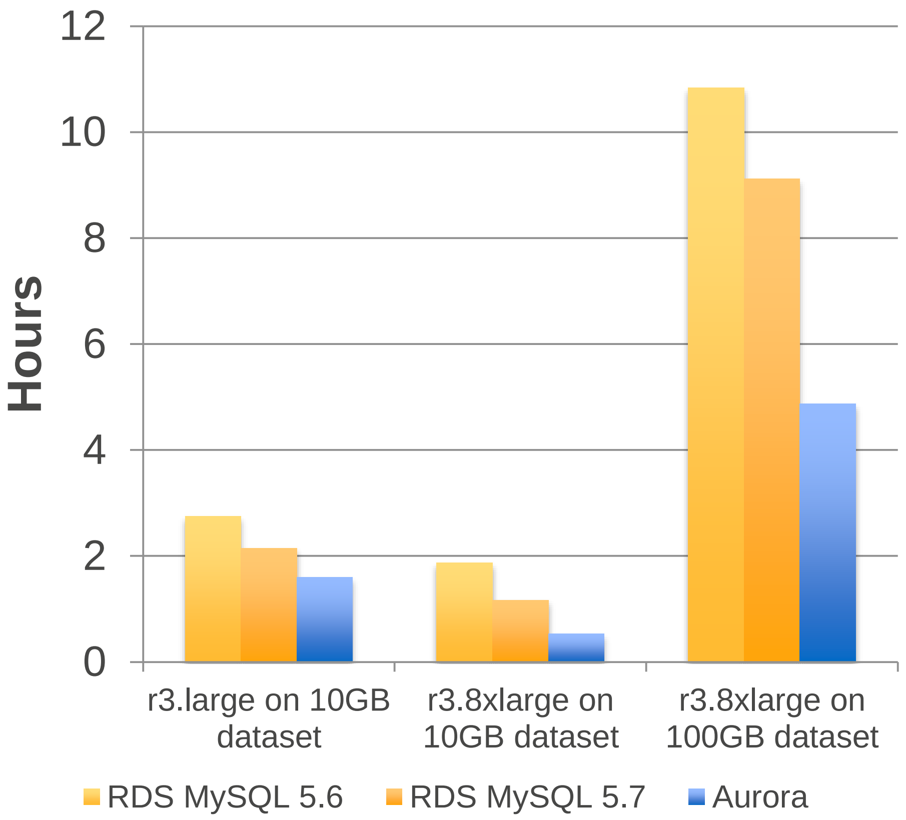
Needed to support many concurrent sessions, high update throughput

Performance enhancements

Faster index build

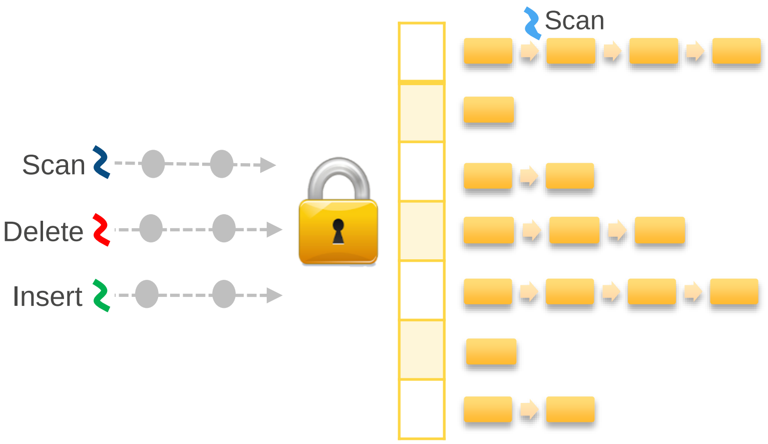
- MySQL 5.6 leverages Linux read ahead – but this requires consecutive block addresses in the btree. It inserts entries top down into the new btree, causing splits and lots of logging.
- Aurora's scan pre-fetches blocks based on position in tree, not block address.
- Aurora builds the leaf blocks and then the branches of the tree.
  - No splits during the build.
  - Each page touched only once.
  - One log record per page.

2-4X better than MySQL 5.6 or MySQL 5.7

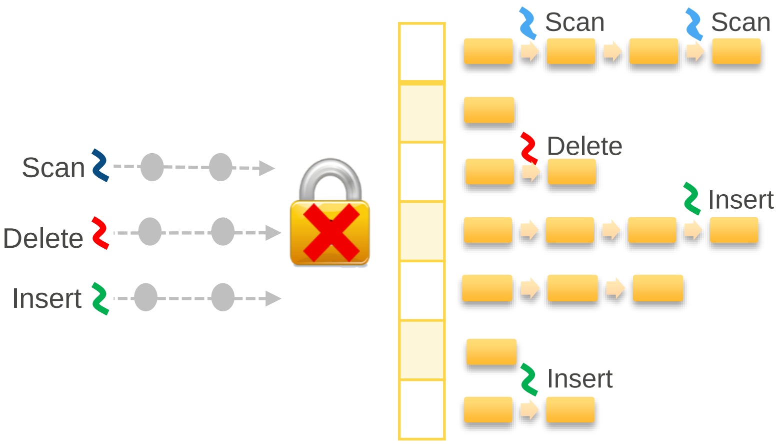


Hot row contention

MySQL lock manager



Aurora lock manager



Highly contended workloads had high memory and CPU

- Lock compression (bitmap for hot locks)
- Replace spinlocks with blocking futex – up to 12x reduction in CPU, 3x improvement in throughput
- Use dynamic programming to release locks: from O(totalLocks * waitLocks) to O(totalLocks)

Throughput on Percona TPC-C 100 improved 29x (from 1,452 txns/min to 42,181 txns/min)

Hot row contention

Percona TPC-C – 10GB

	MySQL 5.6	MySQL 5.7	Aurora	Improvement
500 connections	6,093	25,289	73,955	2.92x
5000 connections	1,671	2,592	42,181	16.3x

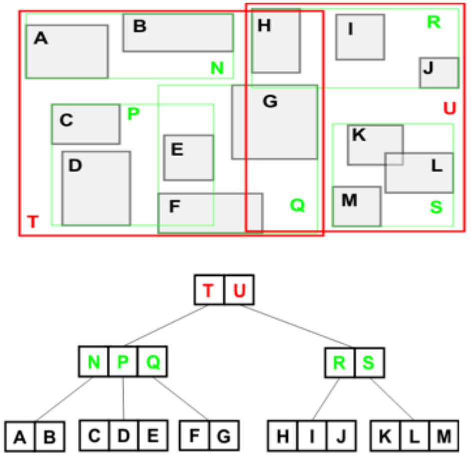
Percona TPC-C – 100GB

	MySQL 5.6	MySQL 5.7	Aurora	Improvement
500 connections	3,231	11,868	70,663	5.95x
5000 connections	5,575	13,005	30,221	2.32x

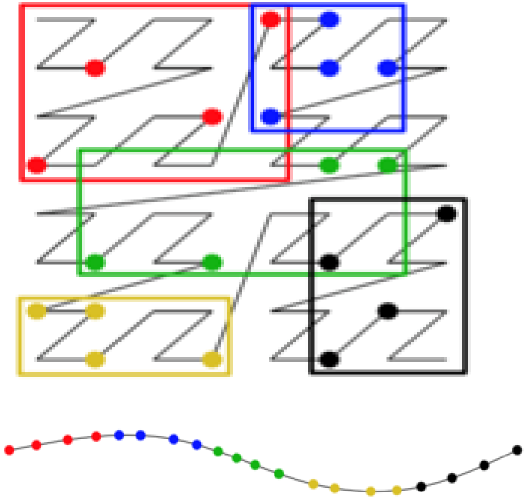
* Numbers are in tpmC, measured using release 1.10 on an R3.8xlarge, MySQL numbers using RDS and EBS with 30K PIOPS

Spatial indexes in Aurora

R-Tree used in MySQL 5.7



Z-index used in Aurora



Challenges with R-Trees

Keeping it efficient while balanced

Rectangles should not overlap or cover empty space

Degenerates over time

Re-indexing is expensive

Z-index (dimensionally ordered space filling curve)

Uses regular B-Tree for storing and indexing

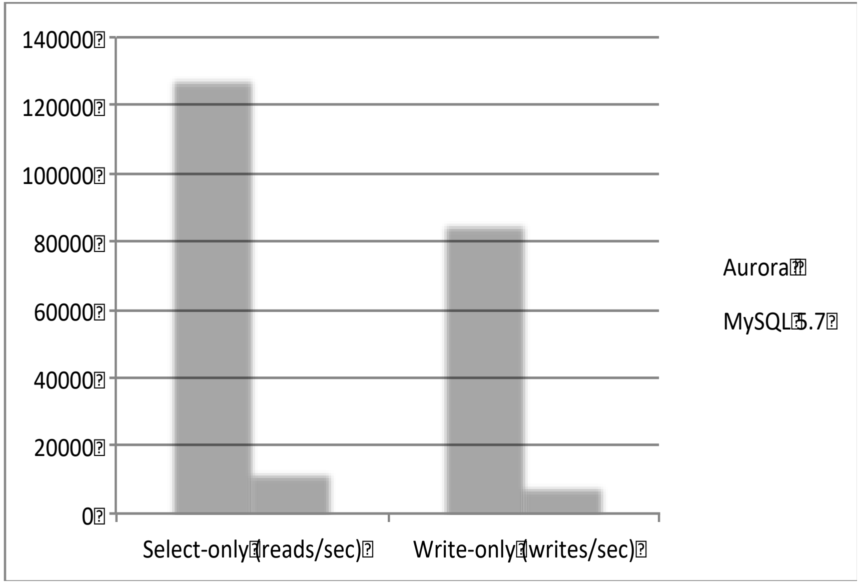
Removes sensitivity to resolution parameter

Adapts to granularity of actual data without user declaration

e.g. GeoWave (National Geospatial-Intelligence Agency)

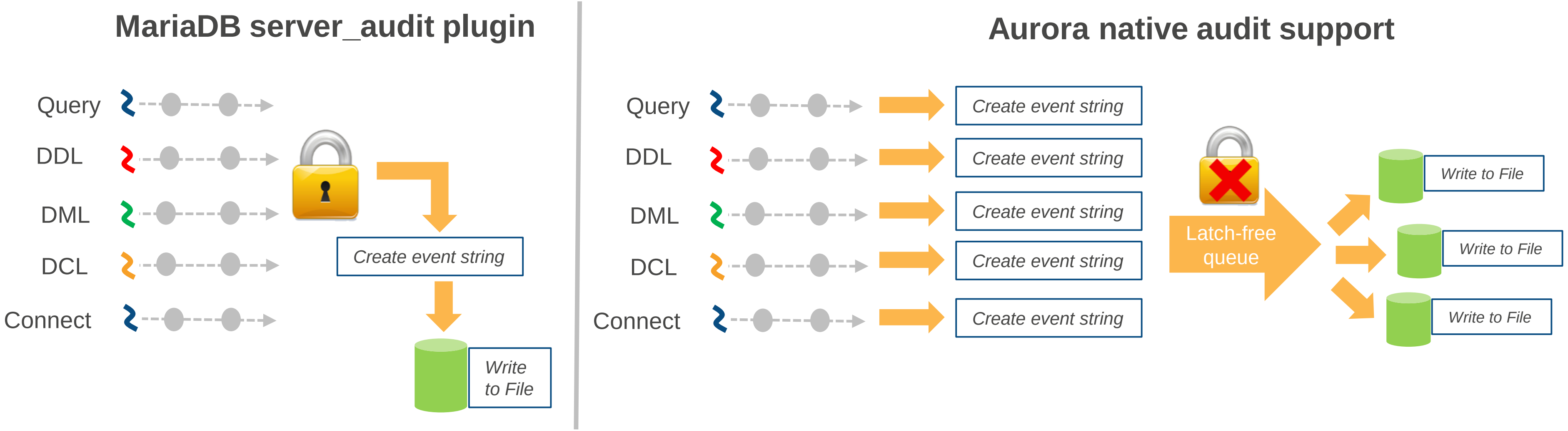
Spatial Index Benchmarks

Sysbench – points and polygons



* r3.8xlarge using Sysbench on <1GB dataset
* Write Only: 4000 clients, Select Only: 2000 clients, ST_EQUALS

High-performance auditing



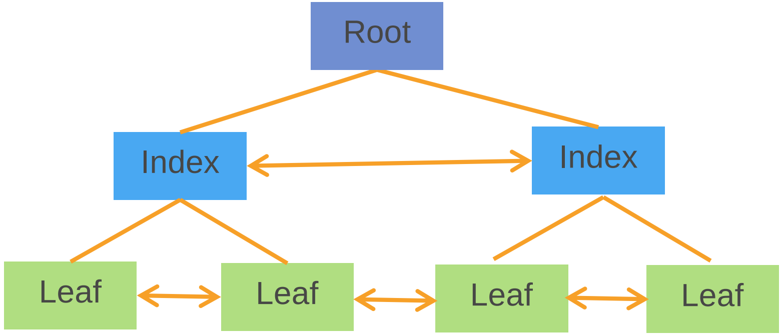
We can sustain over 500K events/sec

	MySQL 5.7	Aurora	
Audit Off	95K	615K	6.47x
Audit On	33K	525K	15.9x

Sysbench Select-only Workload on 8xlarge Instance

Fast DDL: Aurora vs. MySQL

MySQL



- Full Table copy; rebuilds all indexes – can take hours to complete.
- Needs temporary space for DML operations; table lock for DML changes.
- DDL operation impacts DML throughput.

Amazon Aurora

table name	operation	column-name	time-stamp
Table 1	add-col	column-abc	t1
Table 2	add-col	column-qpr	t2
Table 3	add-col	column-xyz	t3

- Add entry to metadata table - use schema versioning to decode the block.
- Modify-on-write to upgrade the block to latest schema when it is modified.

Support NULLable column at the end; other add column, drop/reorder, modify data types

Fast DDL performance

DDL performance on r3.large

	Aurora	MySQL 5.6	MySQL 5.7
10GB table	0.27 sec	3,960 sec	1,600 sec
50GB table	0.25 sec	23,400 sec	5,040 sec
100GB table	0.26 sec	53,460 sec	9,720 sec

DDL performance on r3.8xlarge

	Aurora	MySQL 5.6	MySQL 5.7
10GB table	0.06 sec	900 sec	1,080 sec
50GB table	0.08 sec	4,680 sec	5,040 sec
100GB table	0.15 sec	14,400 sec	9,720 sec

Aurora availability

6-way replicated storage

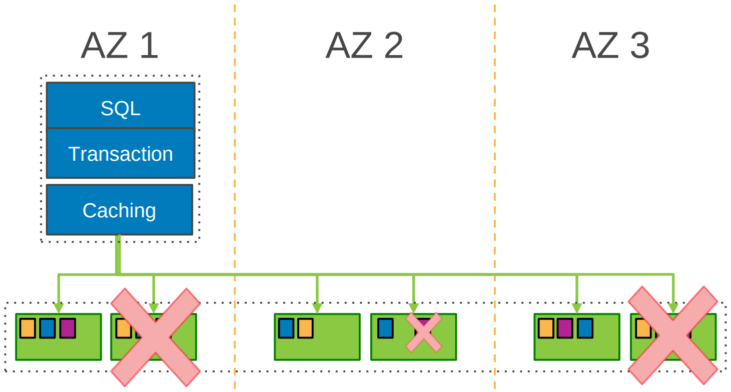
Survives catastrophic failures

Six copies across three Availability Zones

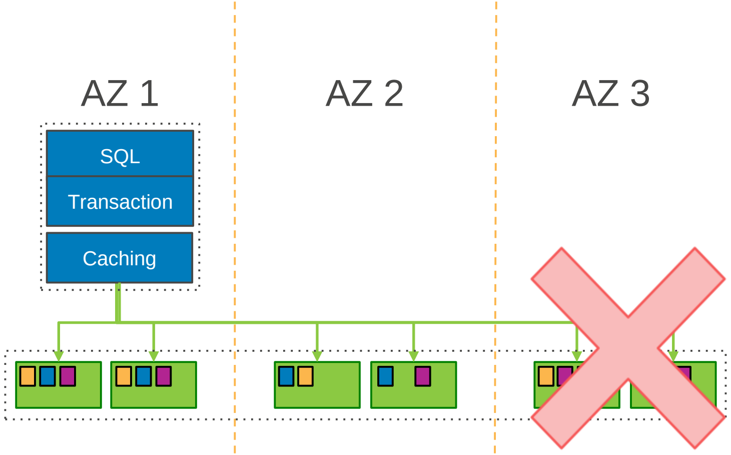
4 out 6 write quorum; 3 out of 6 read quorum

Peer-to-peer replication for repairs

Volume striped across hundreds of storage nodes



Read availability



Read and write availability

Storage Durability

Storage volume automatically grows up to 64 TB

Quorum system for read/write; latency tolerant

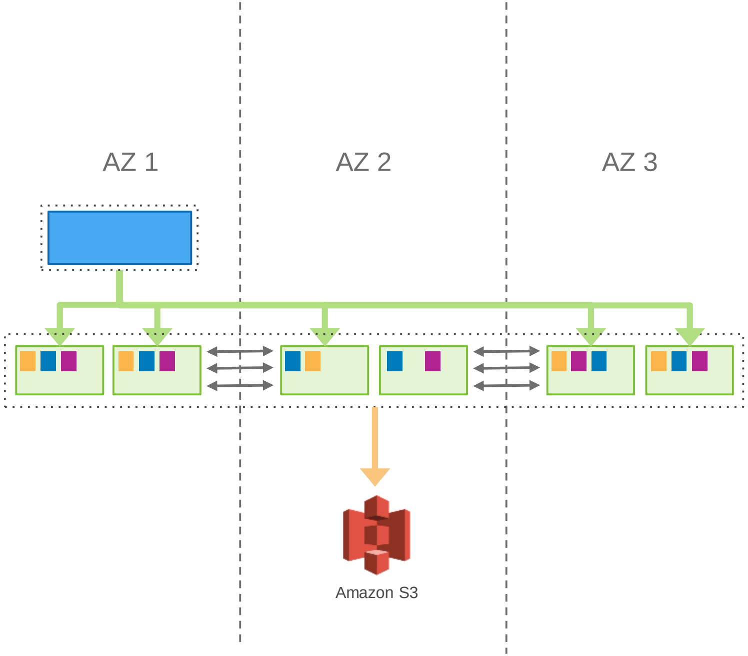
Peer to peer gossip replication to fill in holes

Continuous backup to S3 (built for 11 9s durability)

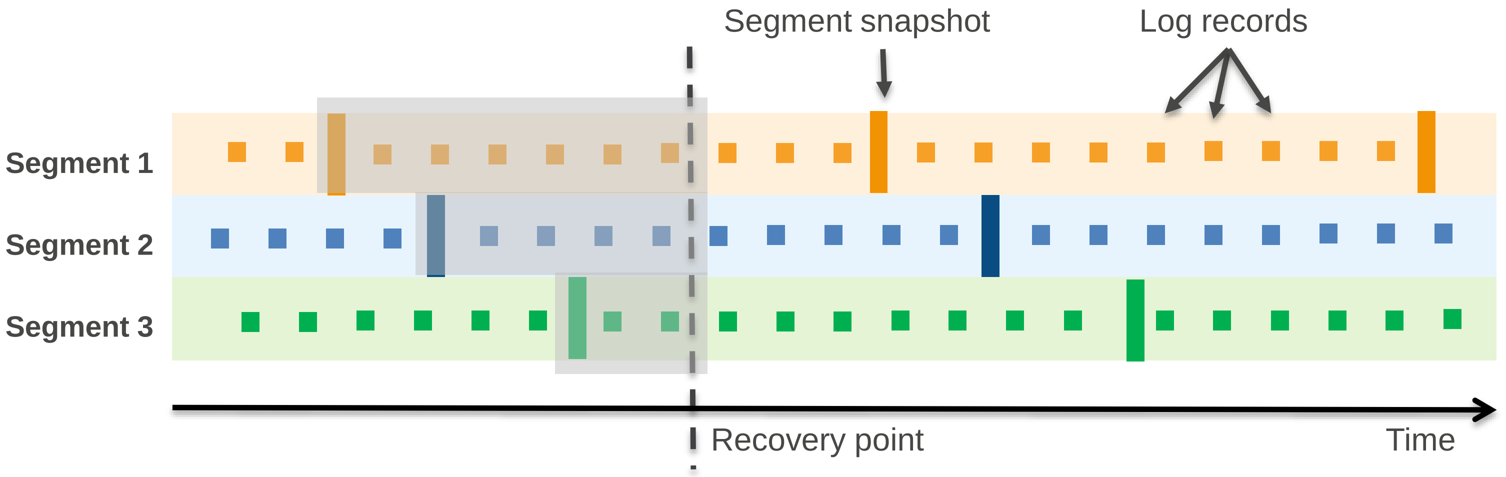
Continuous monitoring of nodes and disks for repair

10GB segments as unit of repair or hotspot rebalance

Quorum membership changes do not stall writes



Continuous backup and point-in-time recovery



- Take periodic snapshot of each segment in parallel; stream the redo logs to Amazon S3
- Backup happens continuously without performance or availability impact
- At restore, retrieve the appropriate segment snapshots and log streams to storage nodes
- Apply log streams to segment snapshots in parallel and asynchronously

Instant Crash Recovery

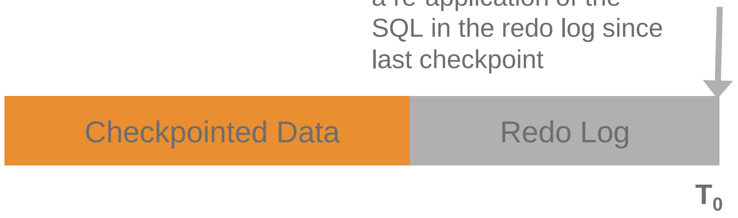
Traditional Databases

Have to replay logs since the last checkpoint

Typically 5 minutes between checkpoints

Single-threaded in MySQL; requires a large number of disk accesses

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



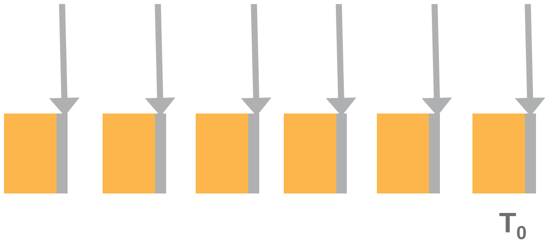
Amazon Aurora

Underlying storage replays redo records on demand as part of a disk read

Parallel, distributed, asynchronous

No replay for startup

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



Survivable Caches

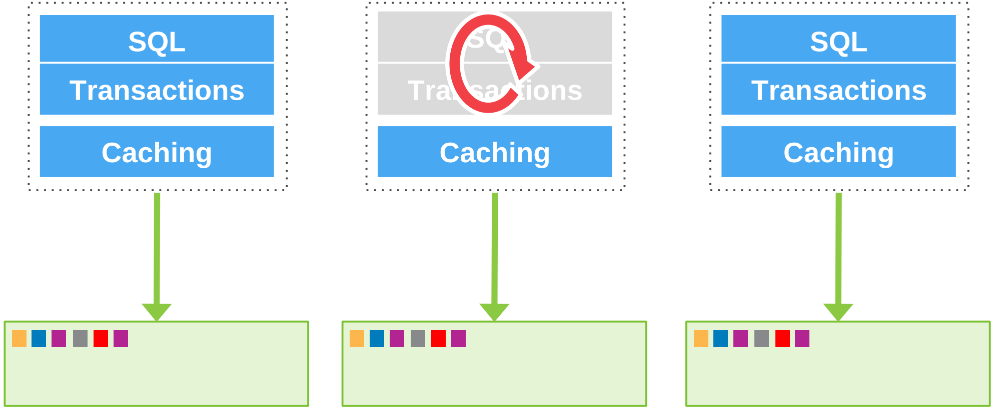
We moved the cache out of the database process

Cache remains warm in the event of database restart

Lets you resume fully loaded operations much faster

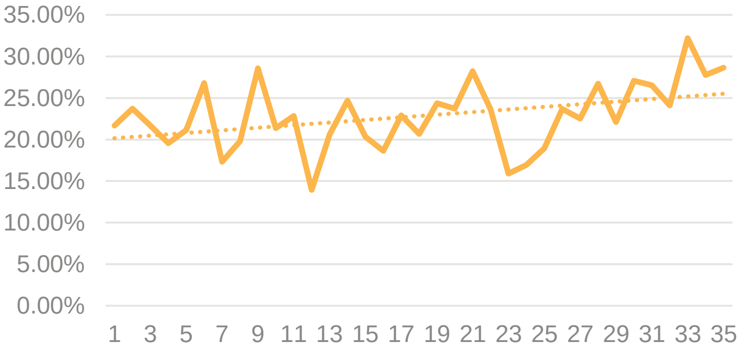
Instant crash recovery + survivable cache = quick and easy recovery from DB failures

Caching process is outside the DB process and remains warm across a database restart

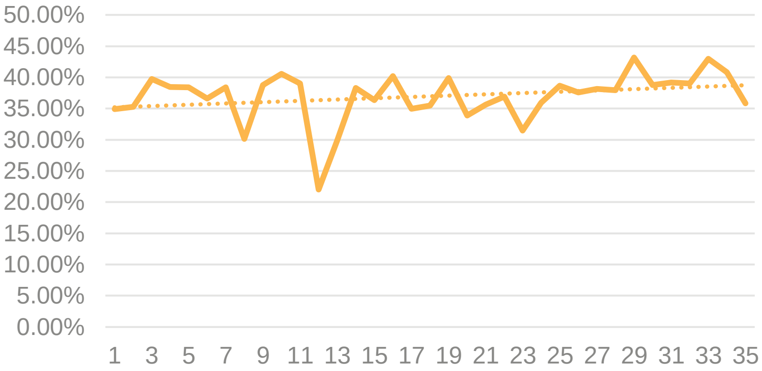


Database failover time

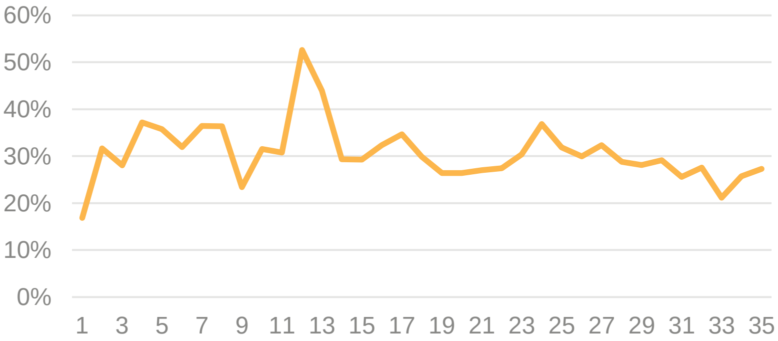
0 - 5s – 30% of fail-overs



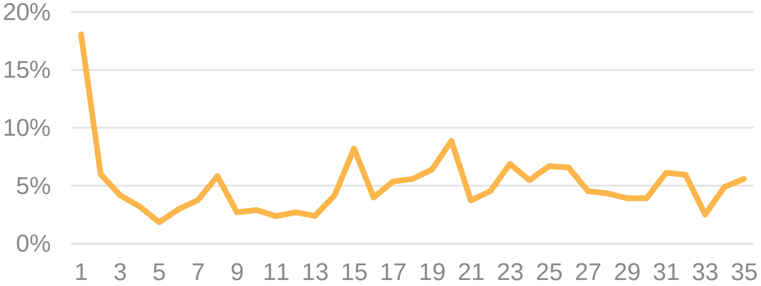
5 - 10s – 40% of fail-overs



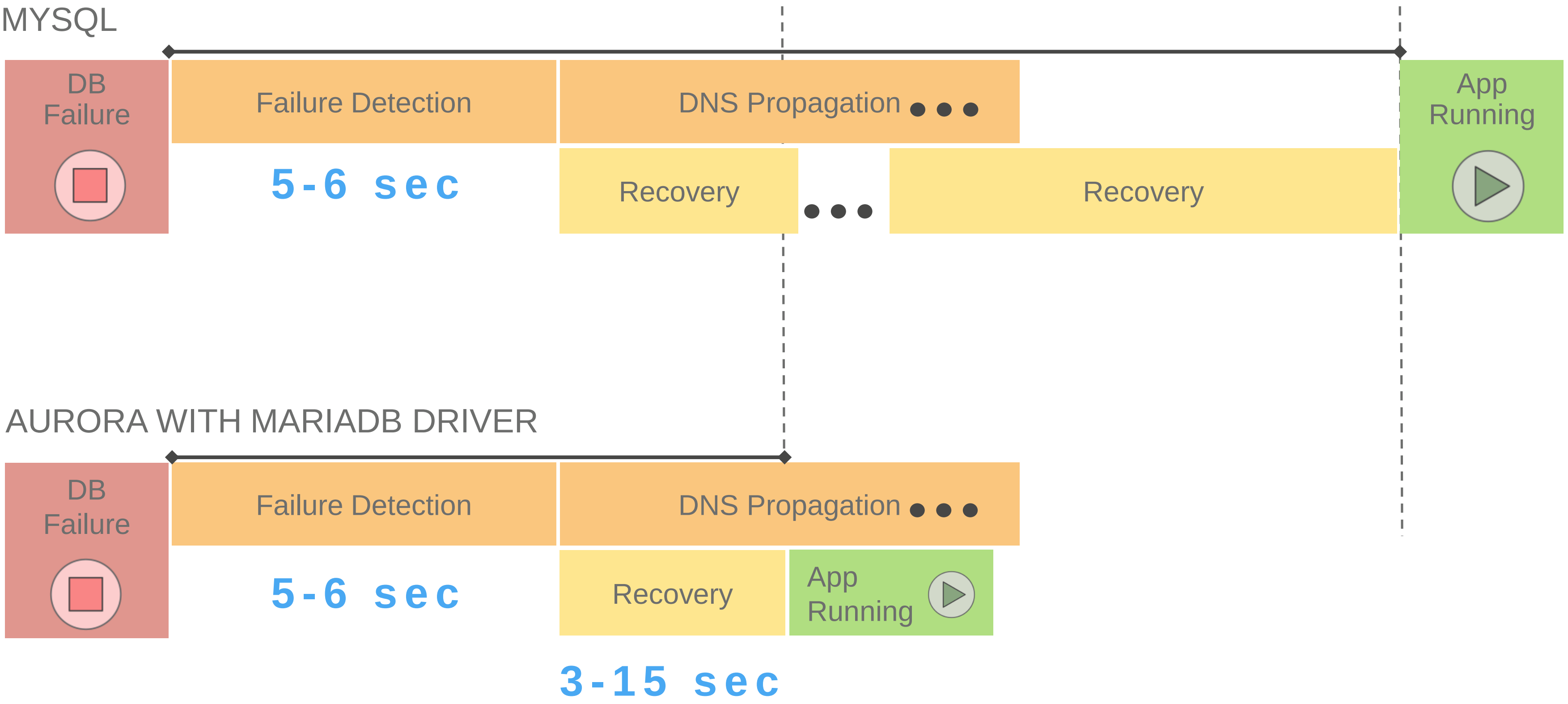
10 - 20s – 25% of fail-overs



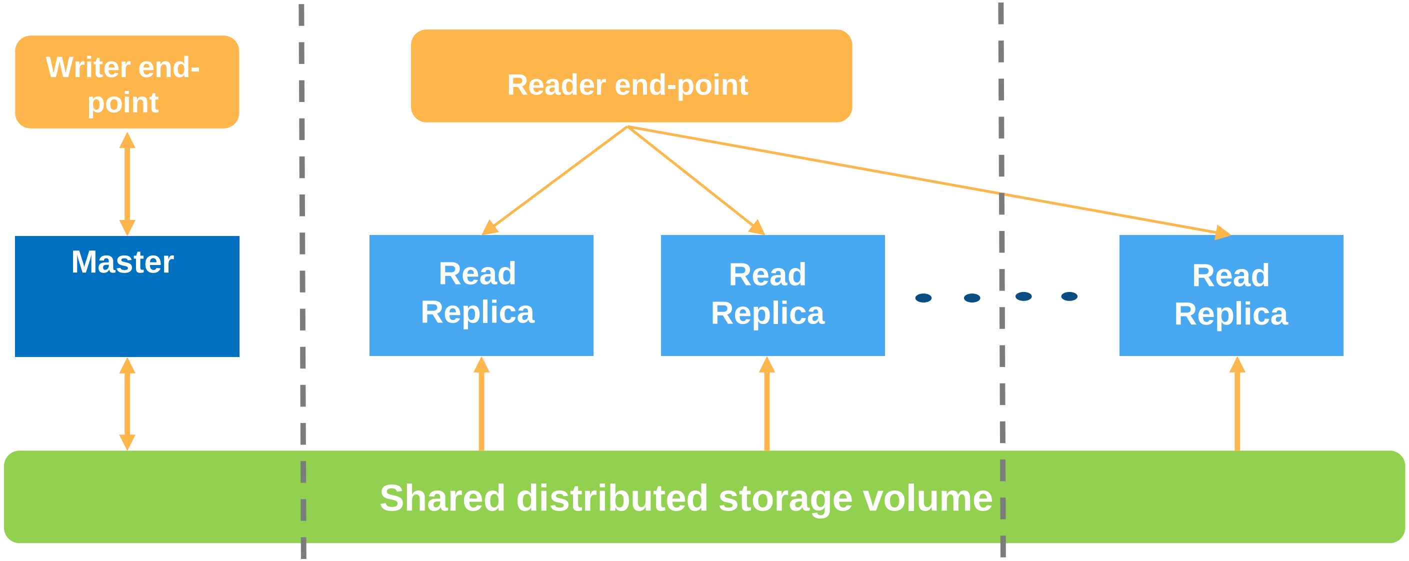
20 - 30s – 5% of fail-overs



Faster failover



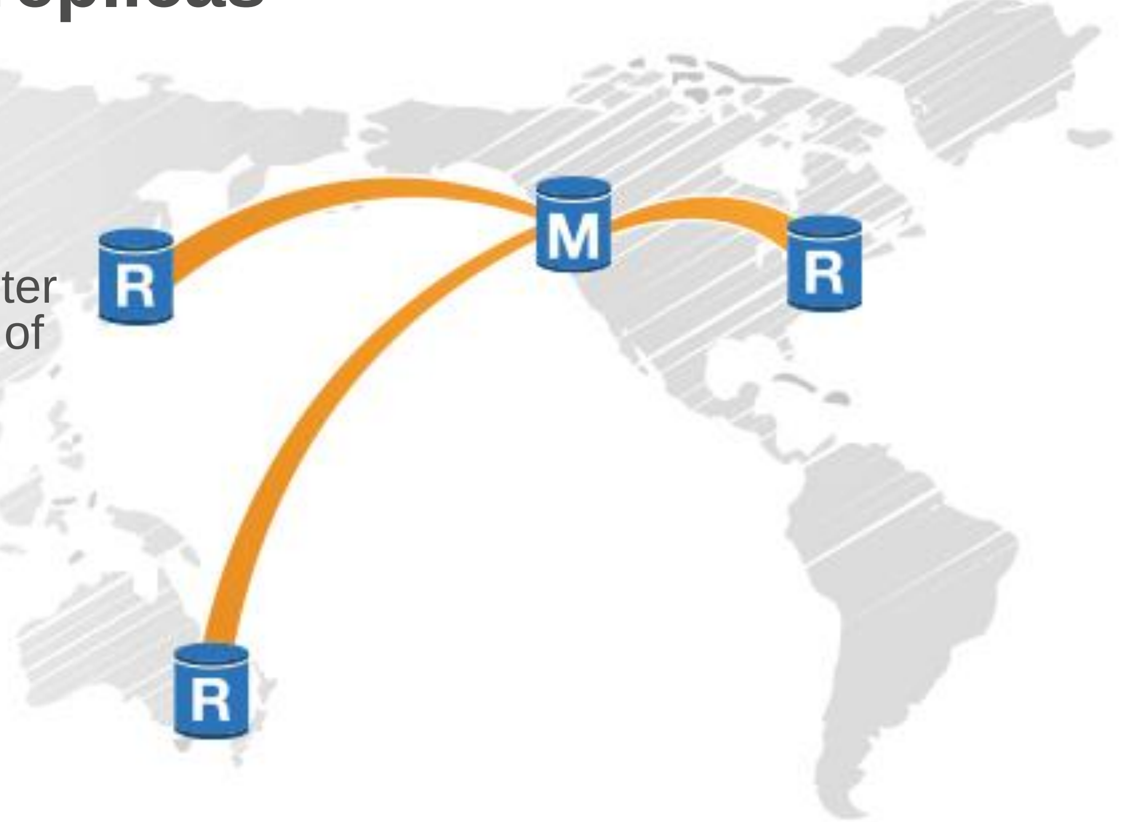
Up to 15 promotable read replicas



- Up to 15 promotable read replicas across multiple Availability Zones
- Re-do log based replication leads to low replica lag – typically < 10ms
- Reader end-point with load balancing; customer specifiable failover order

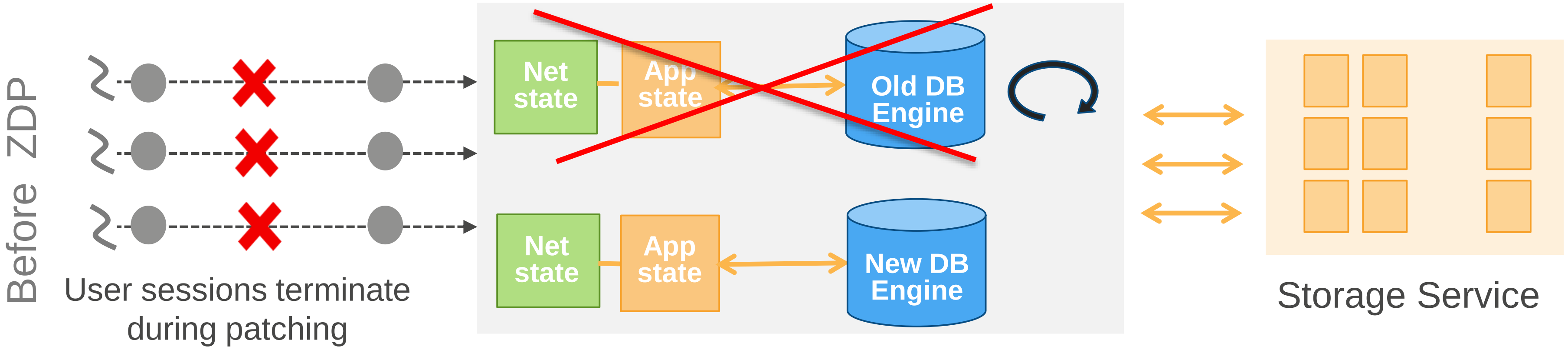
Cross-region read replicas

- MySQL binlog-based disaster recovery and enhanced data locality.
- Promote read-replica to a master for faster recovery in the event of disaster
- Bring data close to your customer’s applications in different regions
- Promote to a master for easy migration



Availability enhancements

Zero downtime patching



Before ZDP

User sessions terminate during patching

Net state

App state

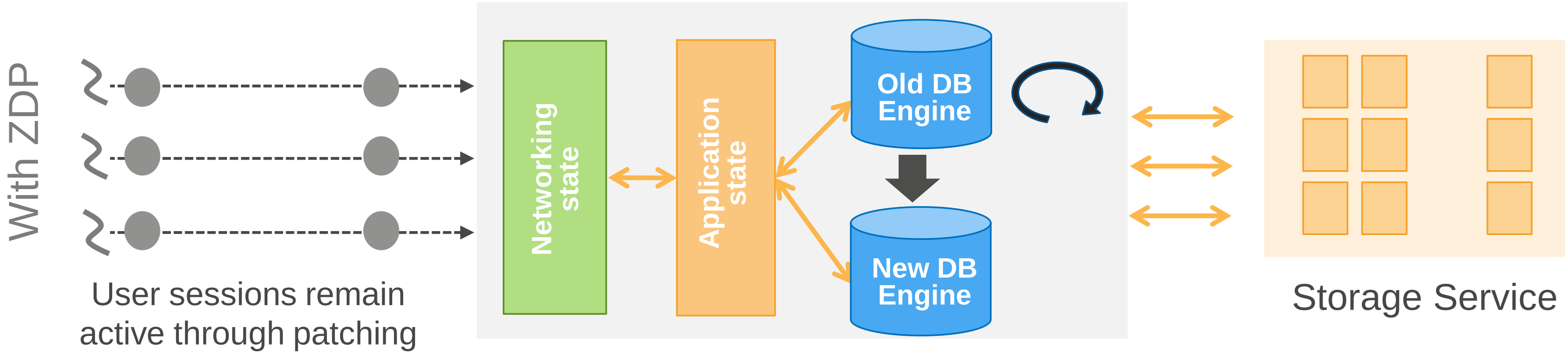
Old DB Engine

Net state

App state

New DB Engine

Storage Service



With ZDP

User sessions remain active through patching

Networking state

Application state

Old DB Engine

New DB Engine

Storage Service

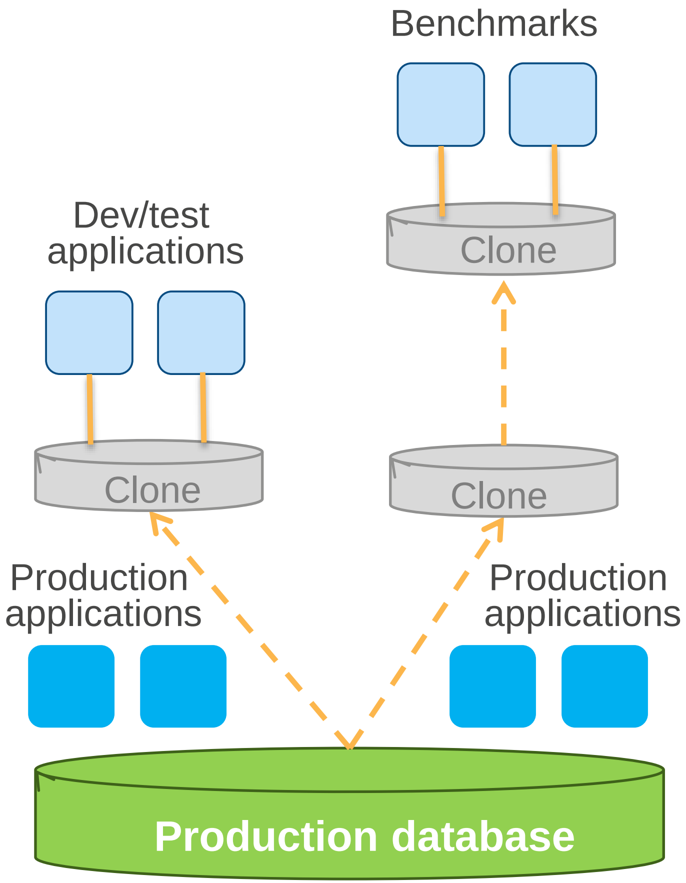
Database cloning

Create a copy of a database without duplicate storage costs

- Creation of a clone is nearly instantaneous – we don’t copy data
- Data copy happens only on write – when original and cloned volume data differ

Typical use cases:

- Clone a production DB to run tests
- Reorganize a database
- Save a point in time snapshot for analysis without impacting production system.



Recent announcements (2017)

Manageability

Advanced auditing; Cross-account encrypted snapshot sharing; Encryption support for cross-region replication; Database cloning; Encrypted migration from RDS MySQL to Aurora

Performance enhancements

Fast DDL for end of table ADD COLUMN operations

Cost reduction

T2.small – cuts cost of entry by half – run Aurora for $1 / day

Ecosystem integration

IAM for Aurora access management; SELECT INTO S3 (LOAD FROM already supported); Aurora audit activity monitoring with CloudWatch

Growing footprint

Launched in US West (N. California) and EU (Frankfurt) – now available in all 3-AZ regions

Thank you!

Reach out to
abrsteve@amazon.com
for questions