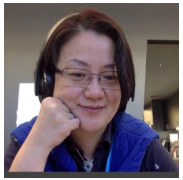
AWS re:Invent

Getting started with Amazon Aurora

Debanjan Saha
GM, Amazon Aurora



Linda Xu
Principal Architect, Ticketmaster

11/29/2016

amazon web services

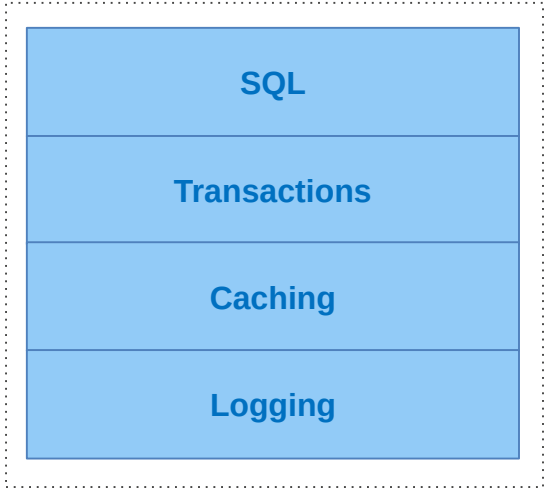
Outline

- What is Amazon Aurora
  - Background and history
- What you need to know about Aurora
  - Differentiators; use cases; cost of ownership
- Hear directly from one of our customers
  - Ticketmaster will share their experience

A bit of history …

Re-imagining relational databases for the cloud era

Relational databases were not designed for the cloud

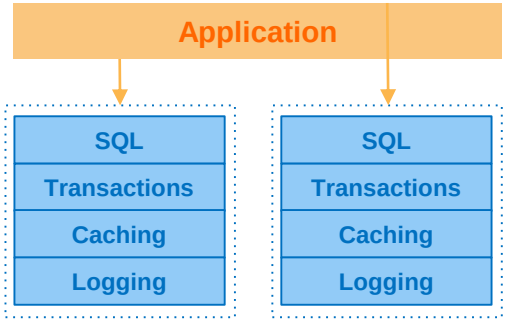


Multiple layers of functionality all in a monolithic stack

Not much has changed in last 20 years

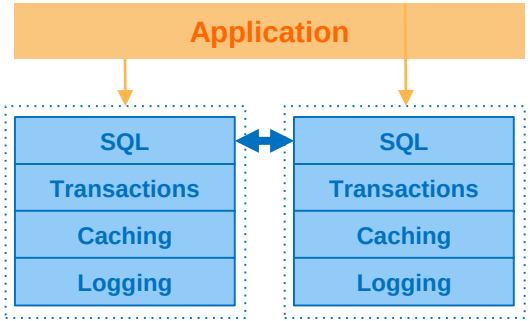
Sharding

Coupled at the application layer



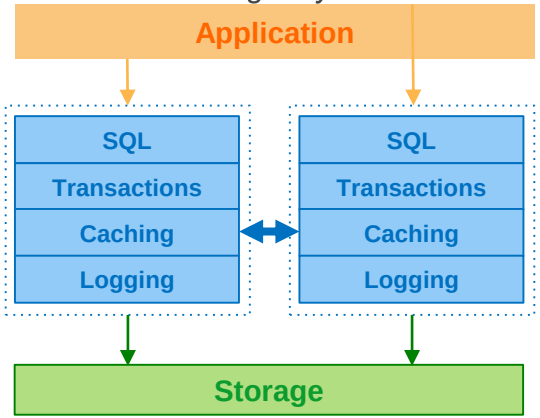
Shared Nothing

Coupled at the SQL layer



Shared Disk

Coupled at the caching and storage layer



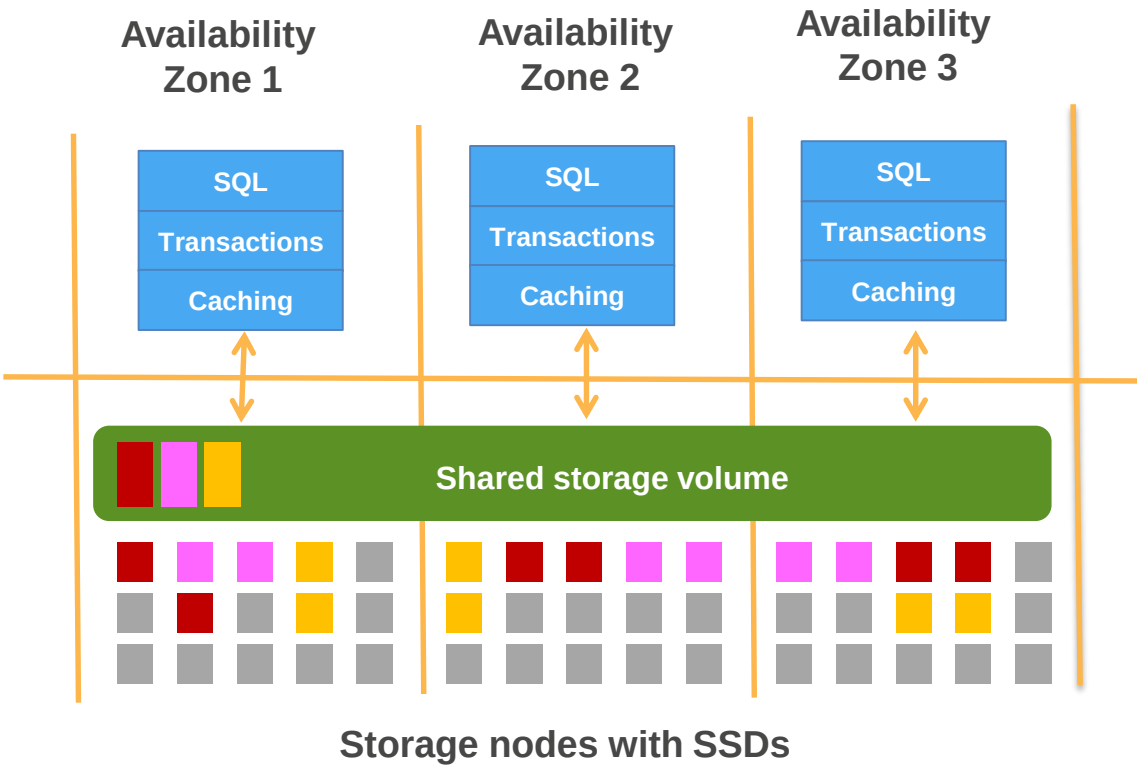
Even when you scale it out, you're still replicating the same stack

Re-imagining relational database

1. Scale-out and distributed design
2. Service-oriented architecture leveraging AWS services
3. Automate administrative tasks – fully managed service

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 ecosystem

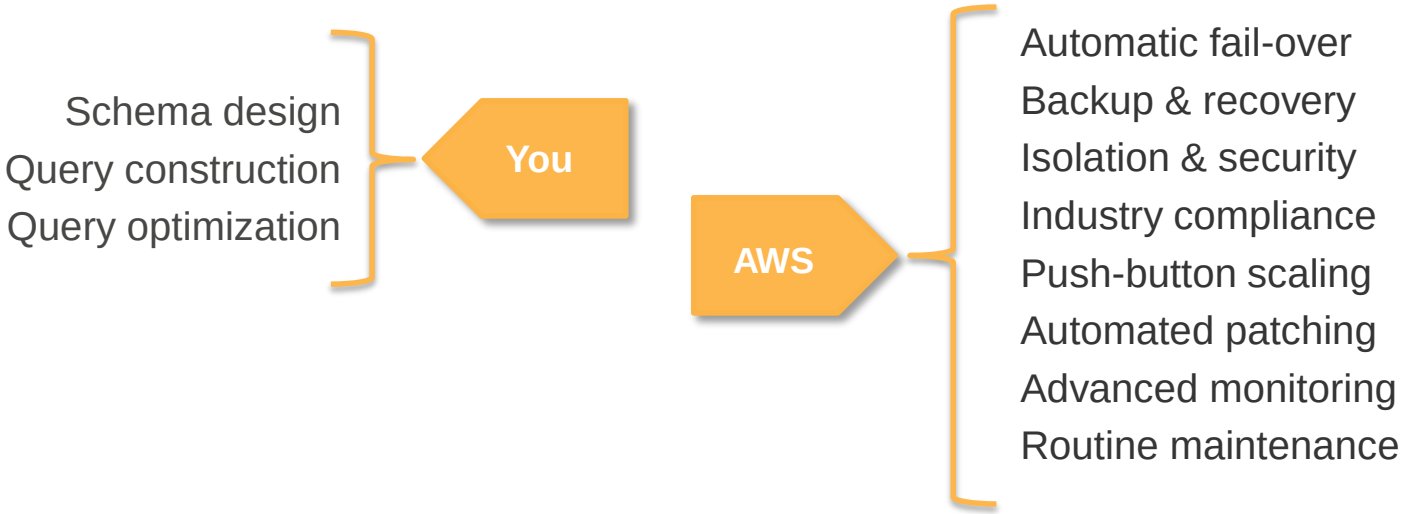
Lambda  Invoke Lambda events from stored procedures/triggers.

S3  Load data from S3, store snapshots and backups in S3.

IAM  Use IAM roles to manage database access control.

CloudWatch  Upload systems metrics and audit logs to CloudWatch.

Automate administrative tasks



Takes care of your time-consuming database management tasks, freeing you to focus on your applications and business

Meet Amazon Aurora ……

Database 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
- ☑ Simple **pay as you go** pricing

Delivered as a **managed** service

Aurora customer adoption

Aurora is used by:

2/3 of top 100 AWS customers

8 of top 10 gaming customers

Fastest growing service in AWS history

Who are moving to Aurora and why?

Customers using MySQL engines

- Higher performance – up to 5x
- Better availability and durability
- Reduces cost – up to 60%
- Easy migration; no application change

Customers using commercial engines

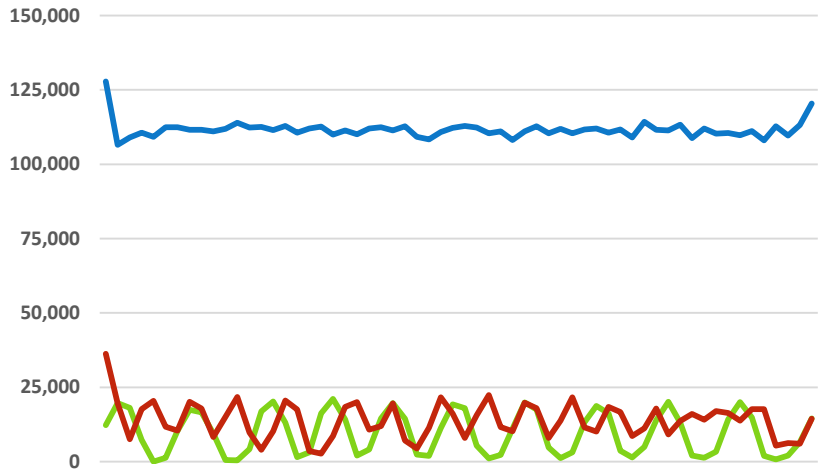
- One tenth of the cost; no licenses
- Integration with cloud ecosystem
- Comparable performance and availability
- Migration tooling and services

Amazon Aurora is fast …

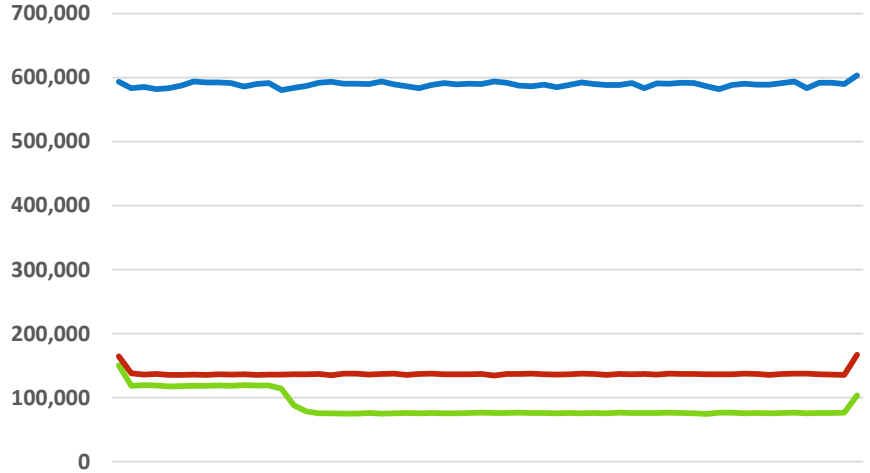
5x faster than MySQL

5X faster than RDS 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.

Aurora Scaling

With user connection

Connections	Amazon Aurora	RDS MySQL w/ 30K IOPS
50	40,000	10,000
500	71,000	21,000
5,000	110,000	13,000

UP TO
8x
FASTER

With number of tables

Tables	Amazon Aurora	MySQL I2.8XL local SSD	RDS MySQL w/ 30K IOPS (single AZ)
10	60,000	18,000	25,000
100	66,000	19,000	23,000
1,000	64,000	7,000	8,000
10,000	54,000	4,000	5,000

UP TO
11x
FASTER

With database size - SYSBENCH

DB Size	Amazon Aurora	RDS MySQL w/ 30K IOPS
1GB	107,000	8,400
10GB	107,000	2,400
100GB	101,000	1,500
1TB	26,000	1,200

UP TO
21
FASTER

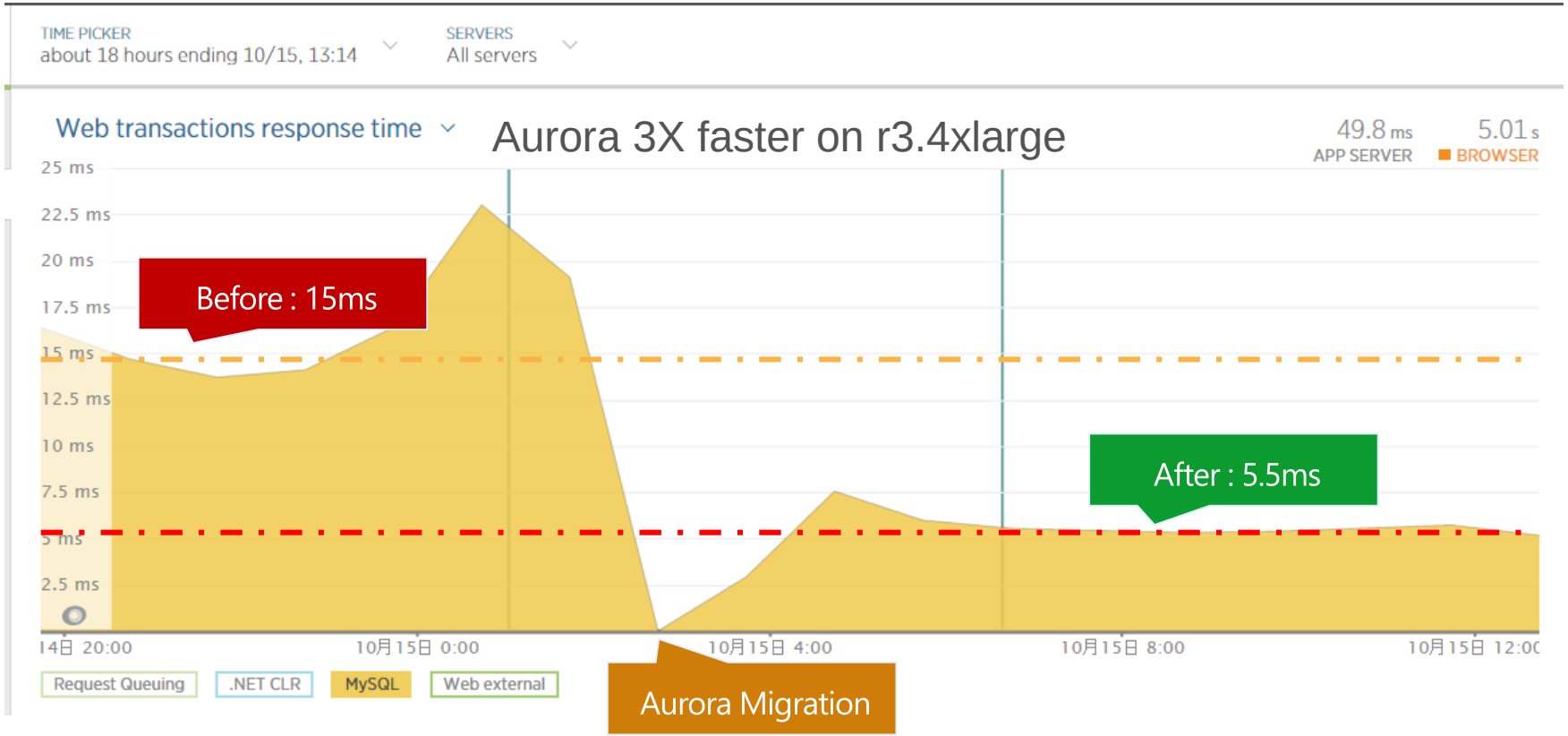
With database size - TPCC

DB Size	Amazon Aurora	RDS MySQL w/ 30K IOPS
80GB	12,582	585
800GB	9,406	69

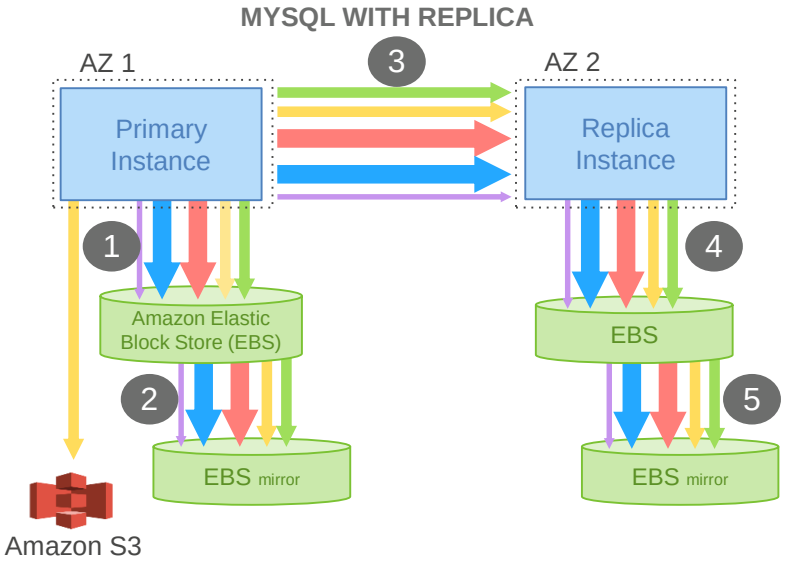
UP TO
136x
FASTER

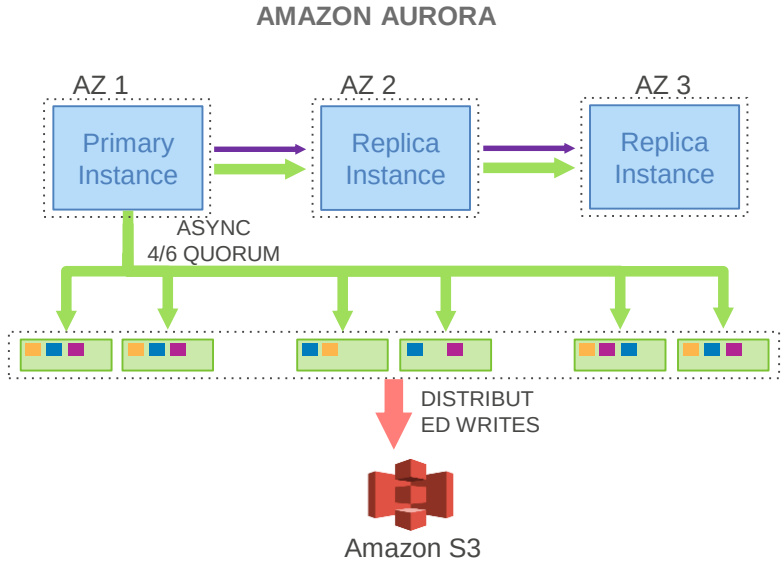
Real-life data – gaming workload

Aurora vs. RDS MySQL – r3.4XL, MAZ



How did we achieve this?





MySQL IO profile for 30 min Sysbench run

780K transactions

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

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**

TYPE OF WRITE

LOG BINLOG DATA DOUBLE-WRITE FRM FILES

New performance enhancements

Read performance

- Smart selector
- Logical read ahead
- Read views

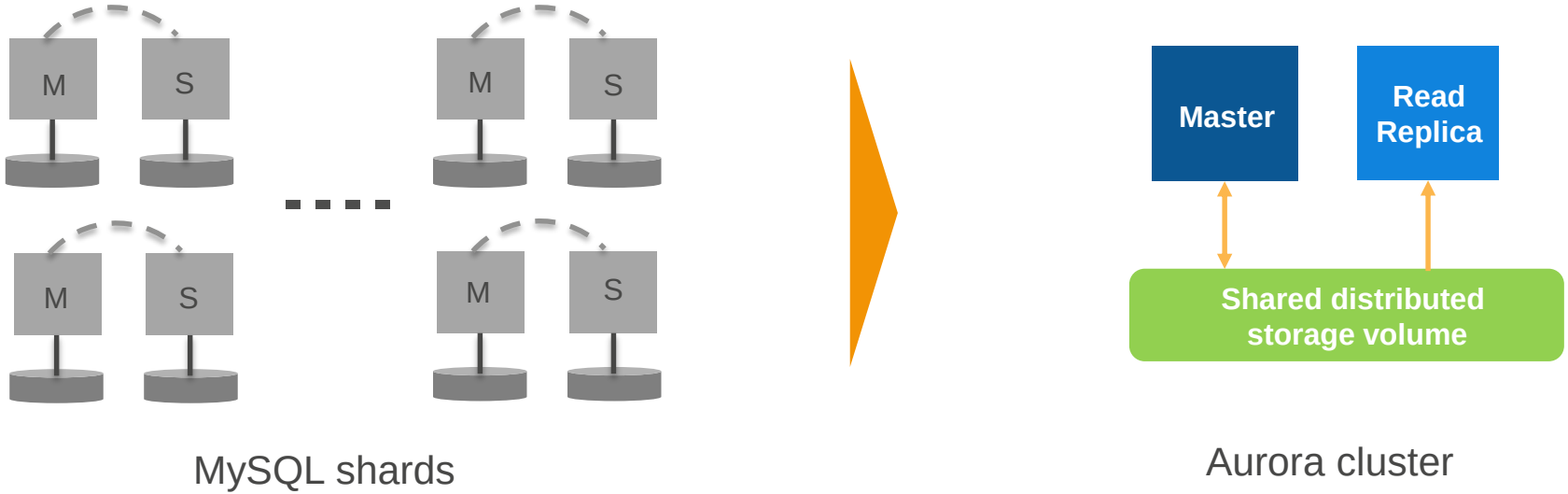
Write performance

- NUMA aware scheduler
- Latch-free lock manager

Meta-data access

- Instant schema update
- B-Tree concurrency
- Catalog concurrency
- Faster index build

Use case: MySQL shard consolidation



MySQL shards

Aurora cluster

Customer, a global SAAS provider, was using hundreds of MySQL shards in order to avoid MySQL performance and connection scalability bottlenecks

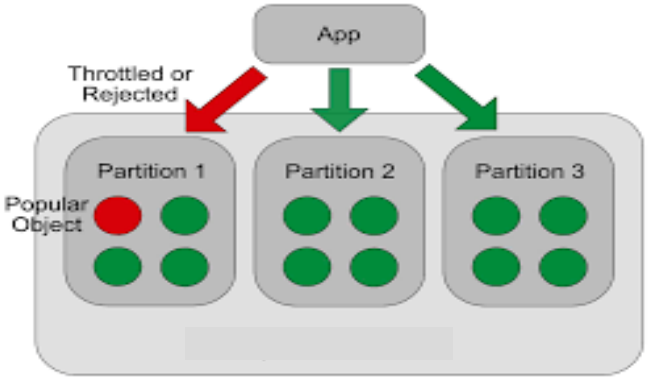
- Consolidated multiple 29 MySQL shards to single r3.4xlarge Aurora cluster
- Even after consolidation cluster utilization is still 30% with plenty of headroom to grow.

Use case: Massively concurrent event store

For messaging, gaming, IoT

New Aurora-backed data store reduces operational costs by **40%**

- The cost of reading data (**70%** of user traffic) almost eliminated due to memory-bound nature of the workload.
- Only pay for IO used, not provisioned. Also, Aurora does automatic hot spot management. So, no need to over provision IOPS based on IO requirements of hottest partition.



Customer, a global mobile messaging platform, was using NoSQL key-value database for user messages:

- **~22 million** accesses per hour (70% read, 30% write) - billing grows linearly with the traffic.
- Scalability bottleneck where certain portions (partitions) of data became “hot” and overloaded with requests.

What about availability

“Performance only matters if your database is up”

6-way replicated storage

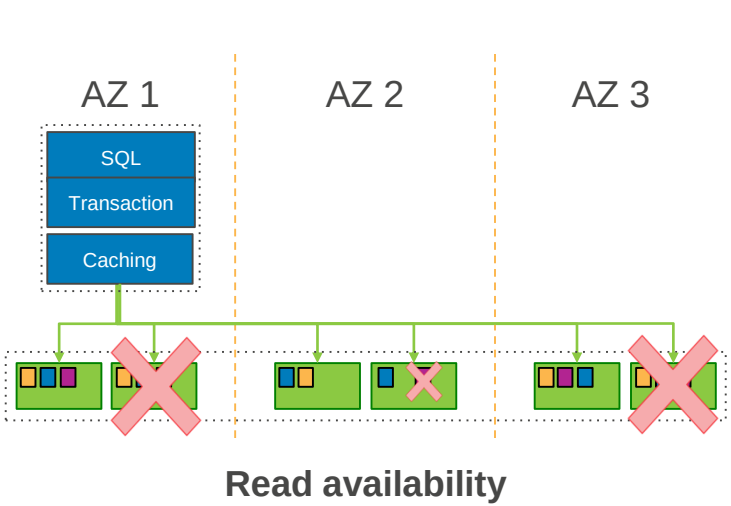
Survives catastrophic failures

Six copies across three availability zones

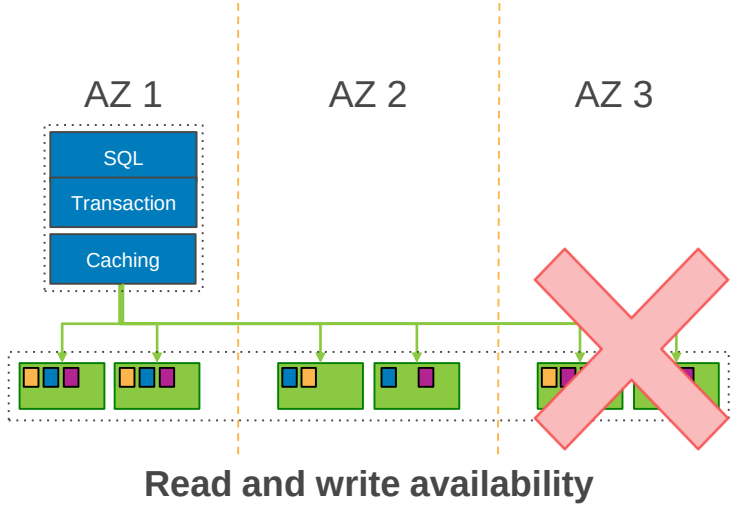
4 out 6 write quorum; 3 out of 6 read quorum

Peer-to-peer replication for repairs

Volume striped across hundreds of storage nodes

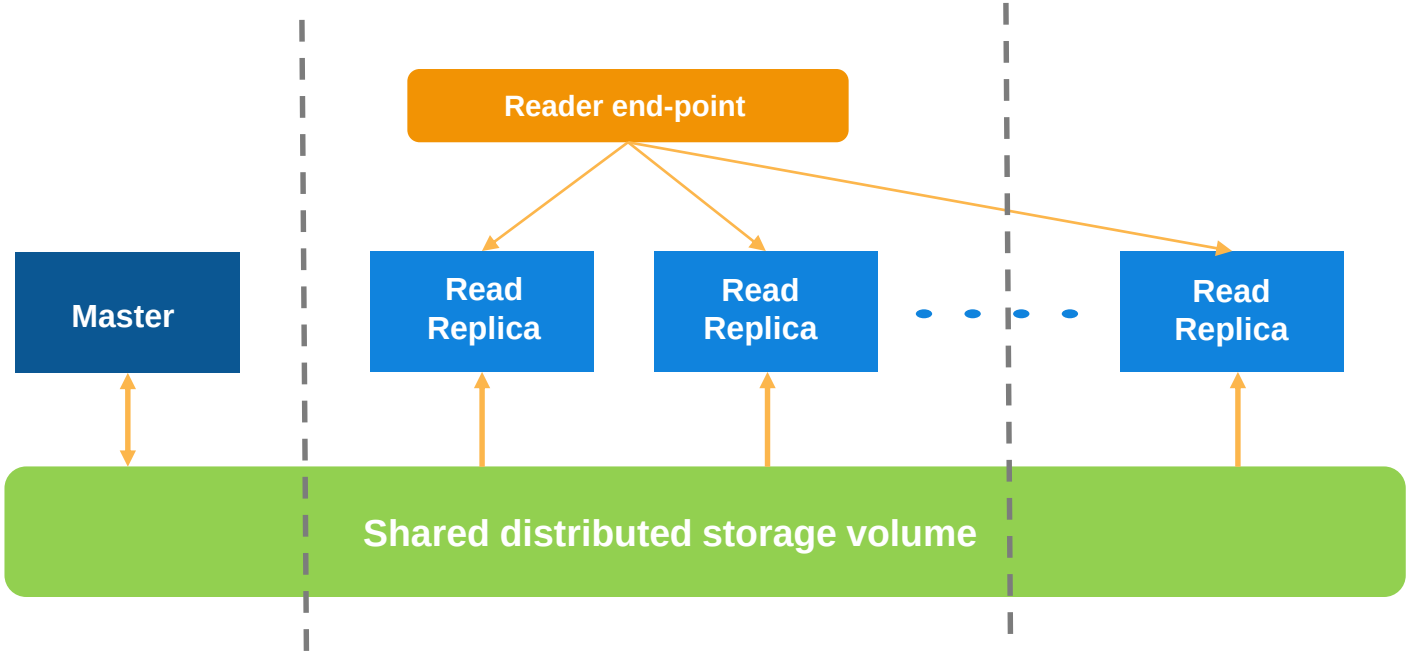


Read availability



Read and write availability

Up to 15 promotable read replicas

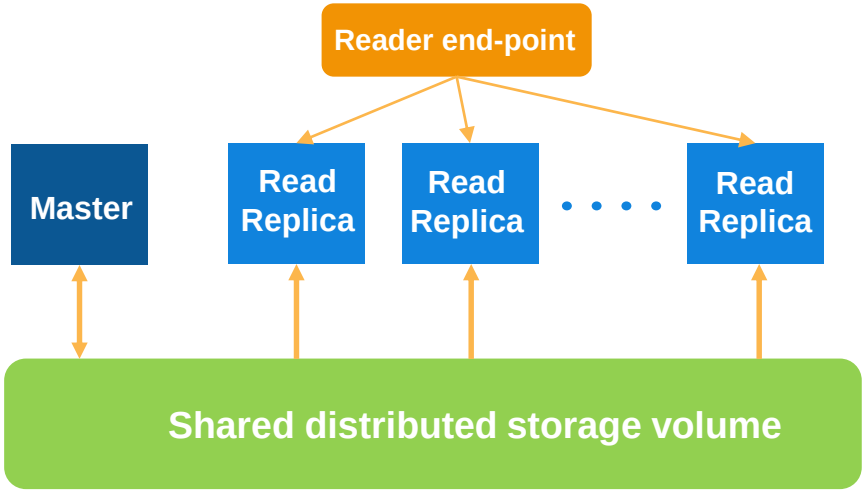


- Up to 15 promotable read replicas across multiple availability zones
- Re-do log based replication leads to low replica lag – typically < 10ms
- Reader end-point with load balancing; customer specifiable failover order

Use case: Near real-time analytics and reporting

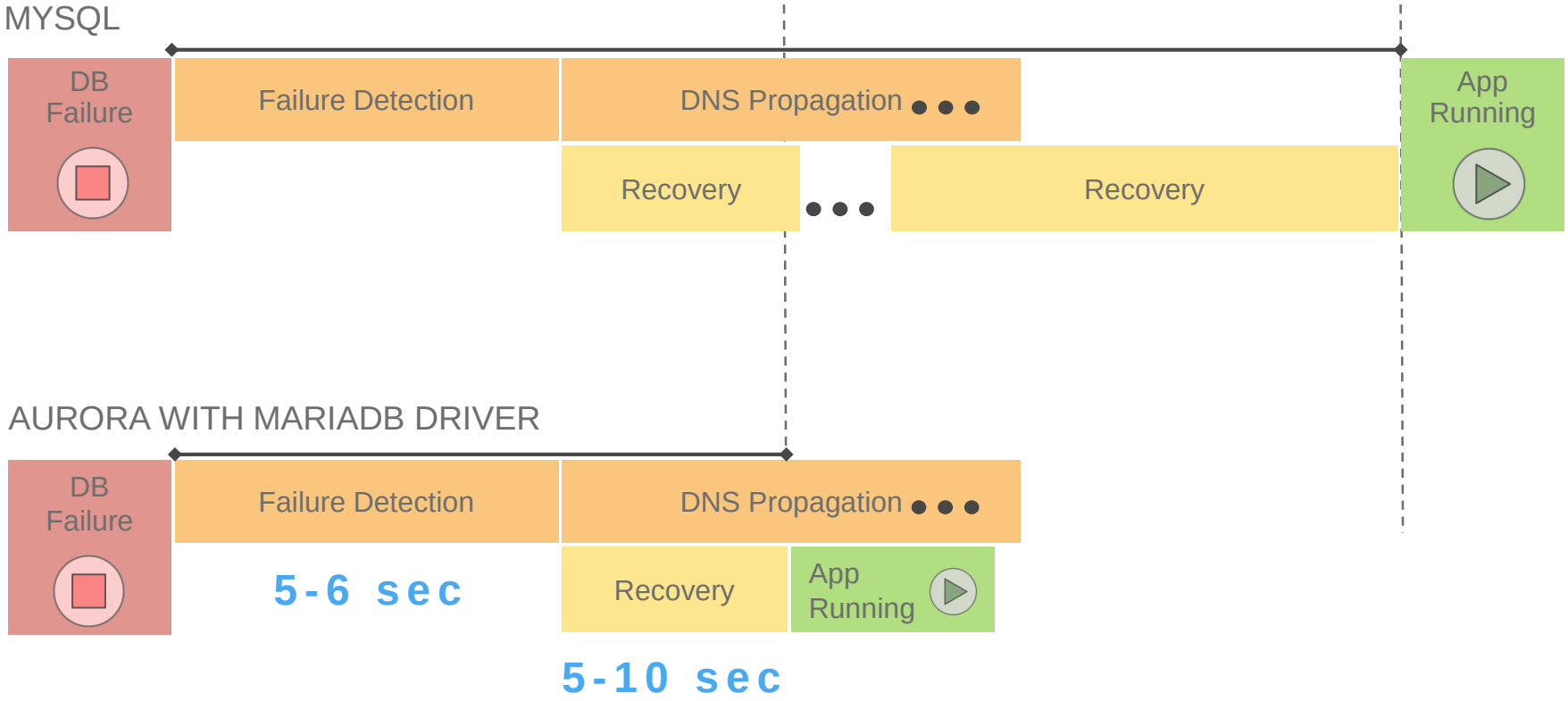
A customer in the travel industry migrated to Aurora for their core reporting application, which is accessed by ~1,000 internal users.

- **Fast provisioning:** replicas can be created, deleted and scaled within minutes based on load.
- **Load balancing:** read-only queries are load balanced across replica fleet through a DNS endpoint – no application configuration needed when replicas are added or removed.
- **Low replication lag:** allows mining for fresh data with no delays, immediately after the data is loaded.
- **Faster, concurrent access:** significant performance gains for core analytics queries - some of the queries executing in 1/100th the original time.



- Up to 15 promotable read replicas
- Low replica lag – typically < 10ms
- Reader end-point with load balancing

Automated failover in 15 secs



Cross-region read replicas

Faster 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

New availability features

Read replicas

- Read replica end-point
- Specifiable fail-over order
- Faster fail-overs < 15 secs

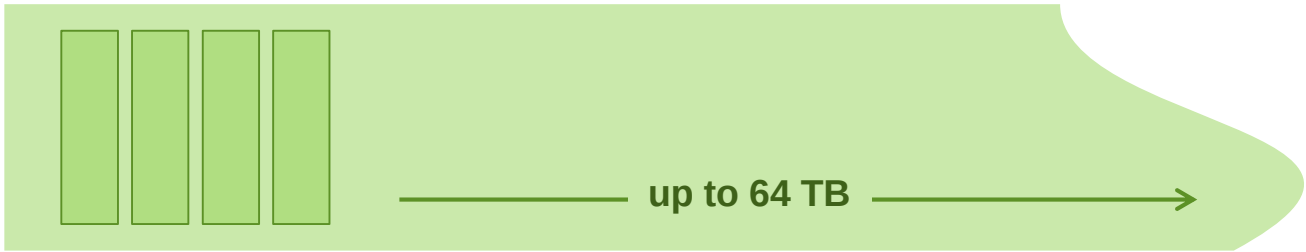
X-region DR

- Cross-region replication
- Cross-region snapshot copy ***Coming soon***
- Cross-account snapshot sharing ***Coming soon***

Amazon Aurora is easy to use

Automated storage management, security and compliance, advanced monitoring, database migration.

Simplify storage management

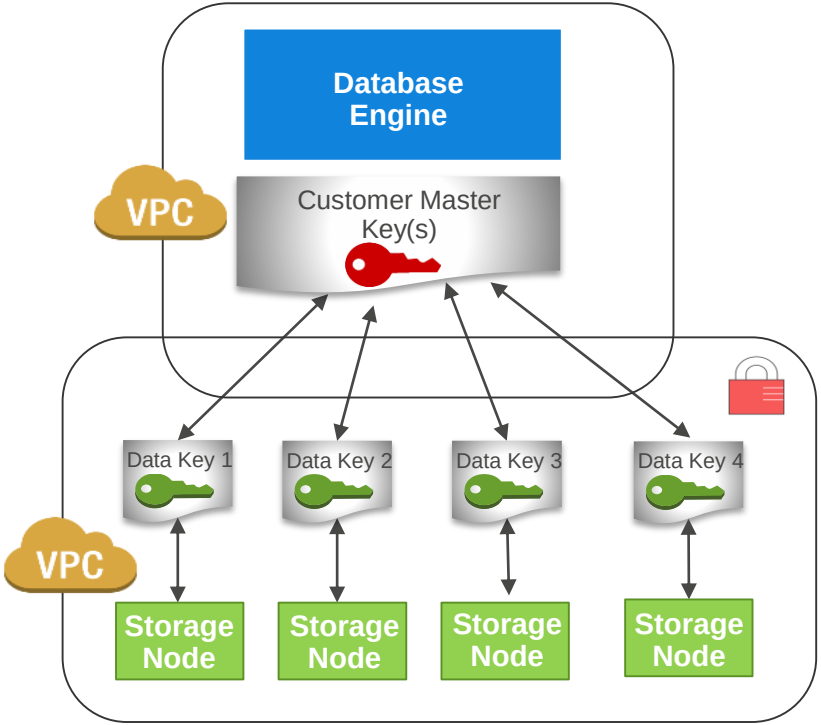


Up to 64TB of storage – auto-incremented in 10GB units

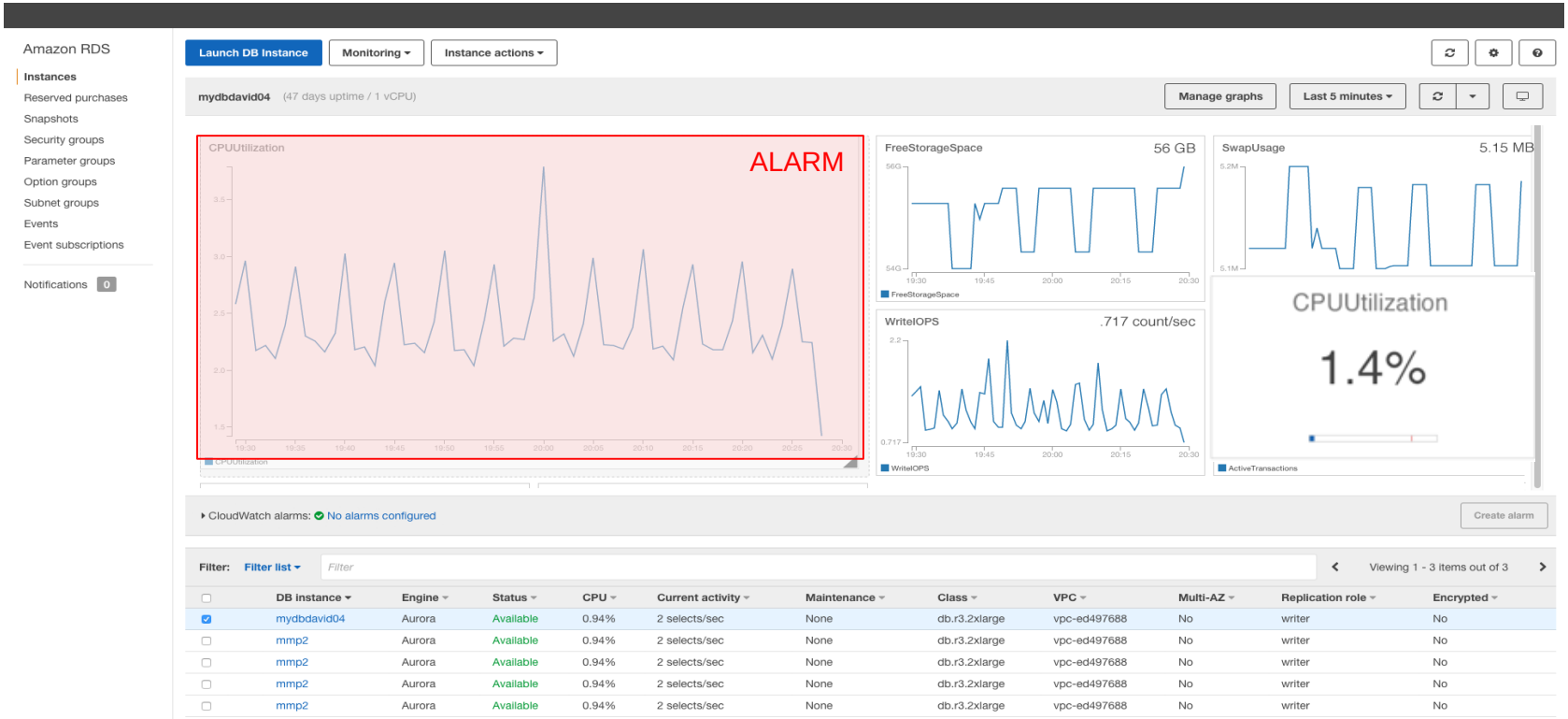
- Continuous, incremental backups to Amazon S3
- Instantly create user snapshots—no performance impact
- Automatic storage scaling up to 64 TB—no performance impact
- Automatic restriping, mirror repair, hot spot management, encryption

Security and compliance

- ☑ Encryption to secure data at rest using customer managed keys
  - AES-256; hardware accelerated
  - All blocks on disk and in Amazon S3 are encrypted
  - Key management via AWS KMS
- ☑ Encrypted cross-region replication, snapshot copy - SSL to secure data in transit
- ☑ Advanced auditing and logging without any performance impact
- ☑ Industry standard security and data protection – **SOC, ISO, PCI/DSS, HIPPA/BAA**



Advanced monitoring



50+ system/OS metrics | sorted process list view | 1-60 sec granularity

alarms on specific metrics | egress to CloudWatch Logs | integration with 3rd-party tools

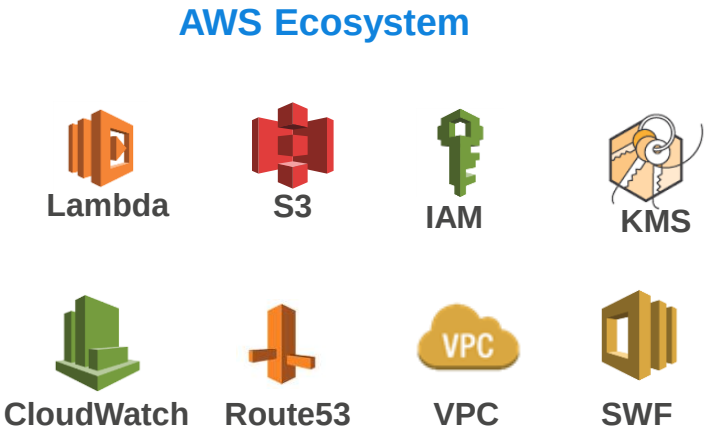
Amazon Aurora migration options

Source database	From where	Recommended option
	RDS	Console based automated snapshot ingestion and catch up via binlog replication.
  	EC2, on premise	Binary snapshot ingestion through S3 and catch up via binlog replication.
 	EC2, on premise, RDS	Schema conversion using SCT and data migration via DMS.

Leverage MySQL and AWS ecosystems

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

Source: Amazon

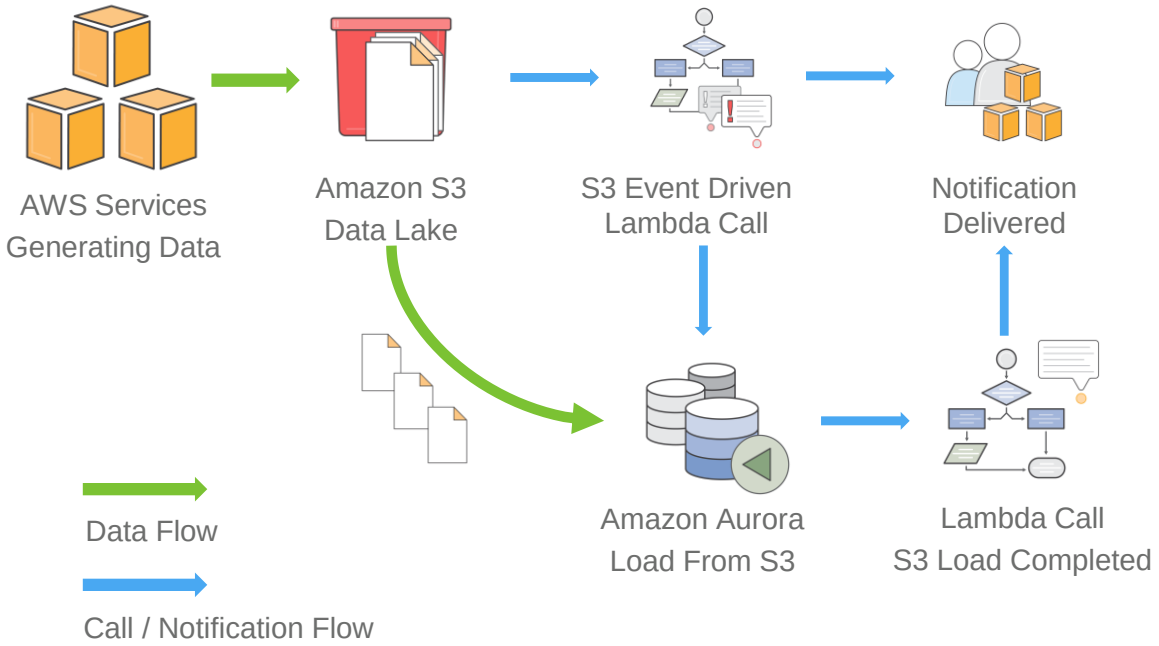


tableau

"We ran our compatibility test suites against Amazon Aurora and everything just worked." - Dan Jewett, Vice President of Product Management at Tableau

Use case: Event driven data pipeline

- Simplify custom database logic by moving it from functions, stored procedures etc. to cloud-based code repository.
  - Enable database developers to create rich software features accessible from SQL layer.
- Run code in response to ad-hoc requests, triggers or scheduled database events in any language supported by AWS Lambda (Java, Node.js, Python).
- Accelerate the migration from any programmable database



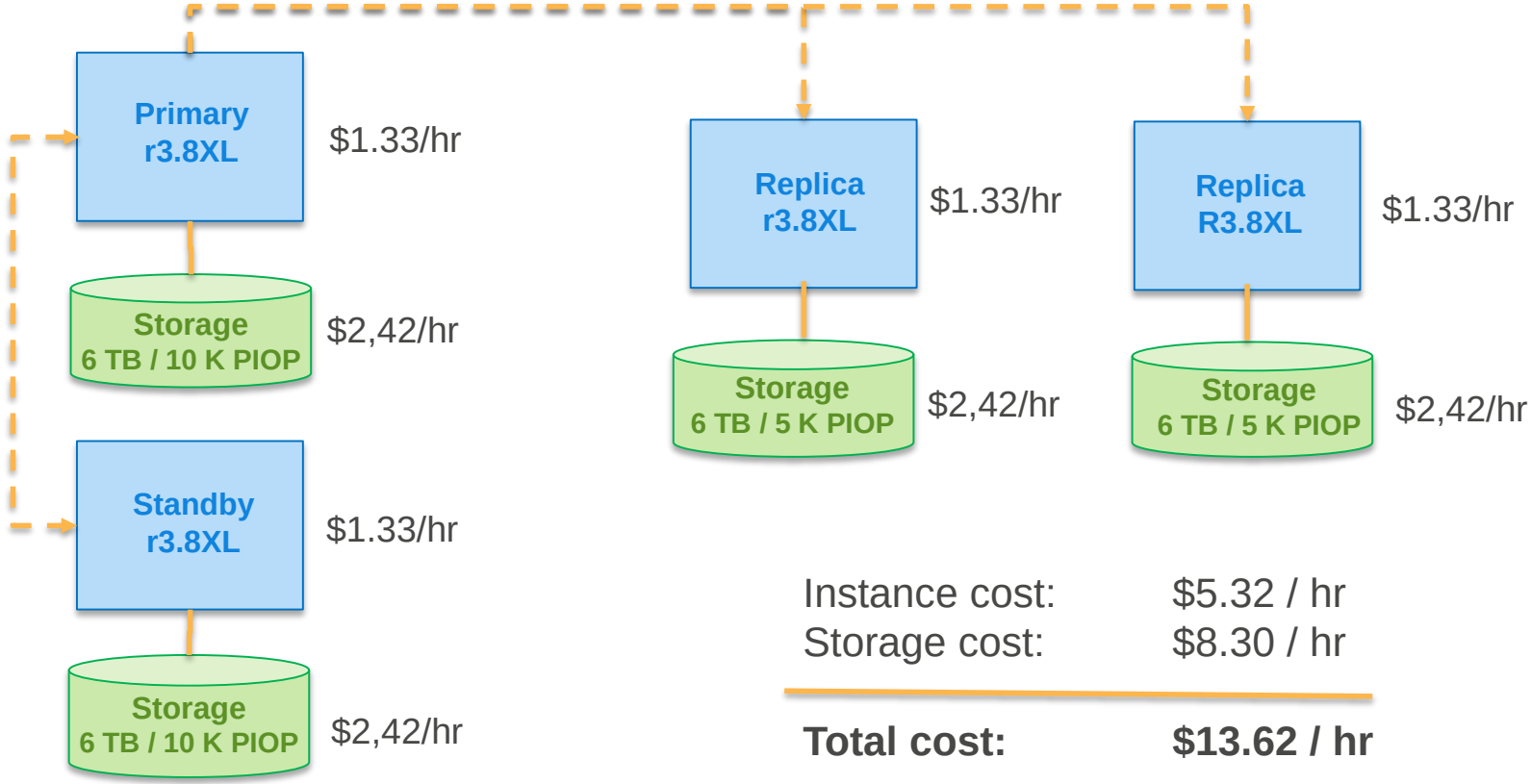
Amazon Aurora saves you money

1/10th of the cost of commercial databases

Cheaper than even MySQL

Cost of ownership: Aurora vs. MySQL

MySQL configuration hourly cost

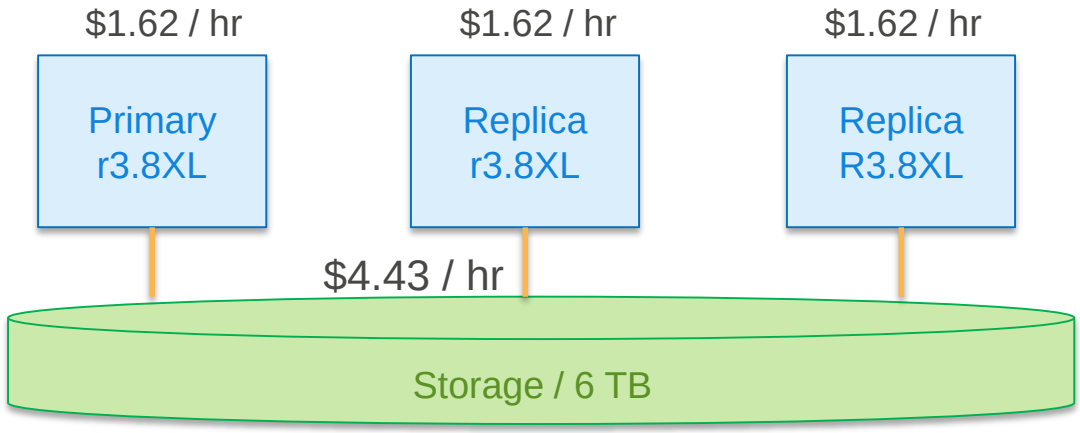


Instance cost:	$5.32 / hr
Storage cost:	$8.30 / hr
Total cost:	**$13.62 / hr**

Cost of ownership: Aurora vs. MySQL

Aurora configuration hourly cost

- No idle standby instance
- Single shared storage volume
- No PIOPs – pay for use I/O
- Reduction in overall IOP



Instance cost:	$4.86 / hr
Storage cost:	$4.43 / hr
Total cost:	**$9.29 / hr**

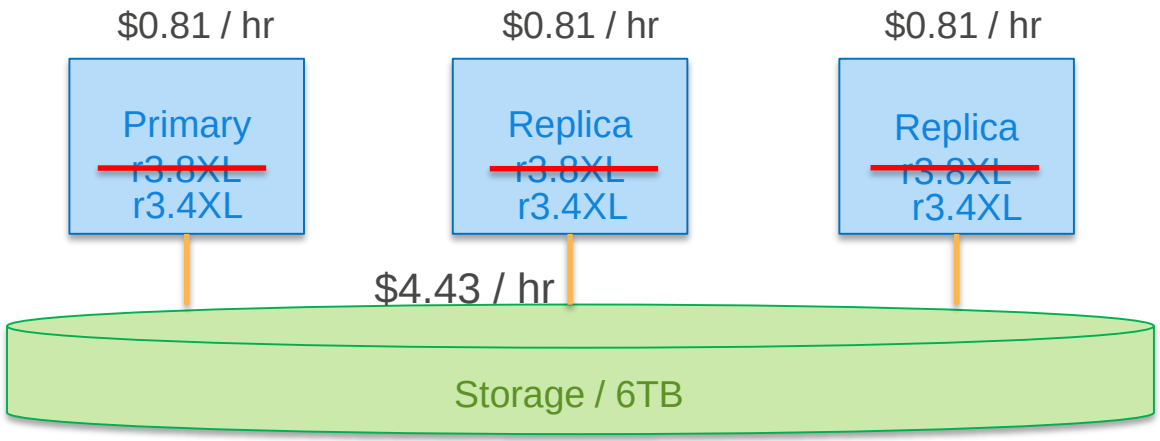
31.8% Savings

*At a macro level Aurora saves over 50% in storage cost compared to RDS MySQL.

Cost of ownership: Aurora vs. MySQL

Further opportunity for saving

- Use smaller instance size
- Pay-as-you-go storage



Instance cost:	$2.43 / hr
Storage cost:	$4.43 / hr
Total cost:	**$6.86 / hr**

49.6% Savings

Storage IOPs assumptions:

1. Average IOPs is 50% of Max IOPs
2. 50% savings from shipping logs vs. full pages

Higher performance, lower Cost

Safe.com lowered their bill by **40%** by switching from sharded MySQL to a single Aurora instance.

Double Down Interactive (gaming) lowered their bill by **67%** while also achieving better latencies (most queries ran faster) and lower CPU utilization.

- Fewer instances needed
- Smaller instances can be used
- No need to pre-provision storage
- No additional storage for read replicas

R3.large is too expensive for dev/test?

We just introduced t2.medium

	vCPU	Mem	Hourly Price
db.t2.medium	2	4	$0.082
db.r3.large	2	15.25	$0.29
db.r3.xlarge	4	30.5	$0.58
db.r3.2xlarge	8	61	$1.16
db.r3.4xlarge	16	122	$2.32
db.r3.8xlarge	32	244	$4.64

NEW

t2 RI discounts

Up to 34% with a 1-year RI

Up to 57% with a 3-year RI

*Prices are for Virginia

Other Aurora sessions at re:Invent

DAT303 - Deep Dive on Amazon Aurora

Thu 11:30-12:30, Venetian, Level 4, Delfino 4004

DAT301 - Amazon Aurora Best Practices: Getting the Best Out of Your Databases

Wed 5:30-6:30, Venetian, Level 4, Lando 4205

DAT322 - Workshop: Stretching Scalability: Doing more with Amazon Aurora

Option 1: Wed 2:00-4:30, Mirage, Trinidad B,

Option 2: Thu 2:30-5:00, Mirage, Antigua B

Have more questions

Rory Richardson, BD
roryrich@amazon.com

Linda Wu, PM
wulind@amazon.com

David Lang, PM
davelang@amazon.com

Debanjan Saha, GM
deban@amazon.com

Reach out to your super friendly Aurora team

ticketmaster®

Linda Xu

Principal Architect, Data @ Ticketmaster

20 years of database experience

https://www.linkedin.com/in/linda-yun-xu-8a53034

ABOUT US

HISTORY

- 1976 - Founded at Arizona State University
- 1996 - Ticketmaster.com launched
- 2010 - Live Nation and Ticketmaster join forces to power live experiences
- 2013 - Transformation journey begins...

- Publicly Traded Company (LYV)
- $7.6B Revenue
- $25B in GTV (Gross Transaction Value)
- Top 5 eCommerce site

ticketmaster®

We power unforgettable moments of joy!

What we do …

ticketmaster®

LIVE NATION BY THE NUMBERS

EVERY 20 MINUTES
Somewhere In The World, There Is A Live Nation Event

70M
Unique Monthly Visitors On The Live Nation Family Of Websites

530M
Fans Touched By Our Platform

25,500
Concert Events Worldwide

465M
Tickets Sold Globally By Ticketmaster

37
Countries Host Live Nation Events

350+
Global Icons Under Artist Nation Management

145M
Consumer Database Profile

21M
Mobile App Downloads

70+
Global Festivals

900
Brand Partners

100
Artist Nation Managers

Our IT infrastructure Today

ticketmaster®

Our private cloud is made up of:

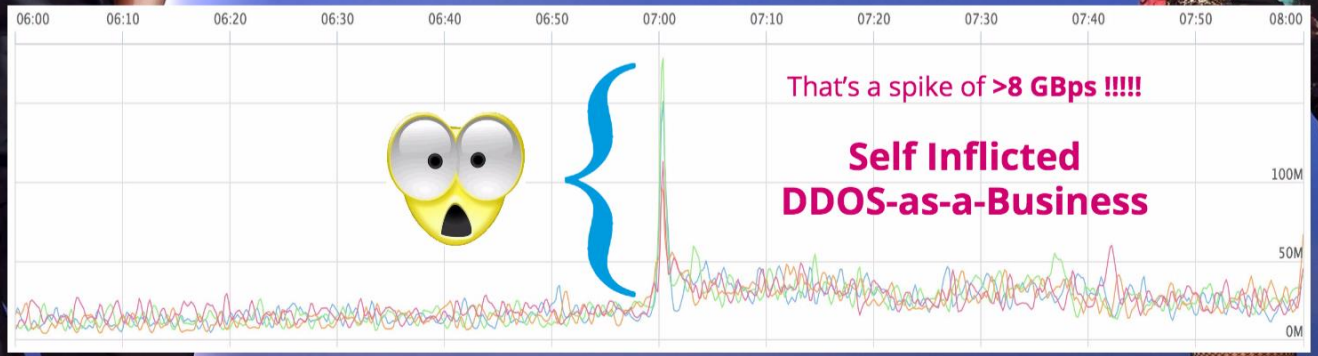
- Hundreds of Ticketmaster products
- Tens of thousands of servers
- Multiple Data Centers, globally
- More than 1,000 databases
  - Oracle, MySQL, MongoDB, Teradata, Microsoft SQL Server and PostgreSQL

BIG SCALE, BIG CHALLENGES

Onsales = Black Friday every day!

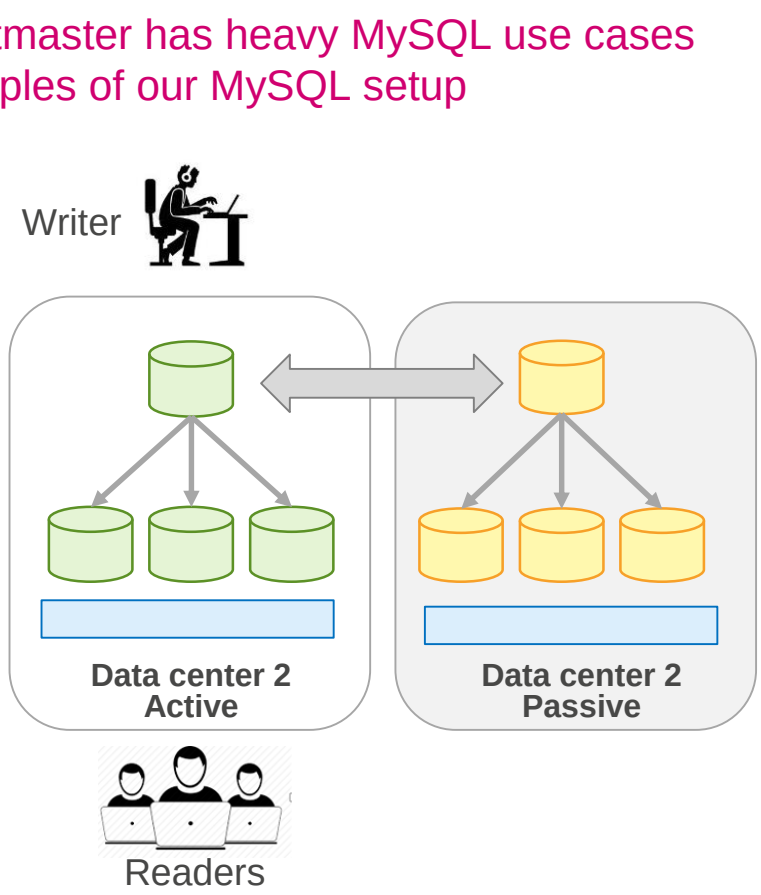
- Huge spikes / demand for tickets
- Global company = across time zones
- Limited inventory (Beyonce Tickets!)
- Multiple sales channels

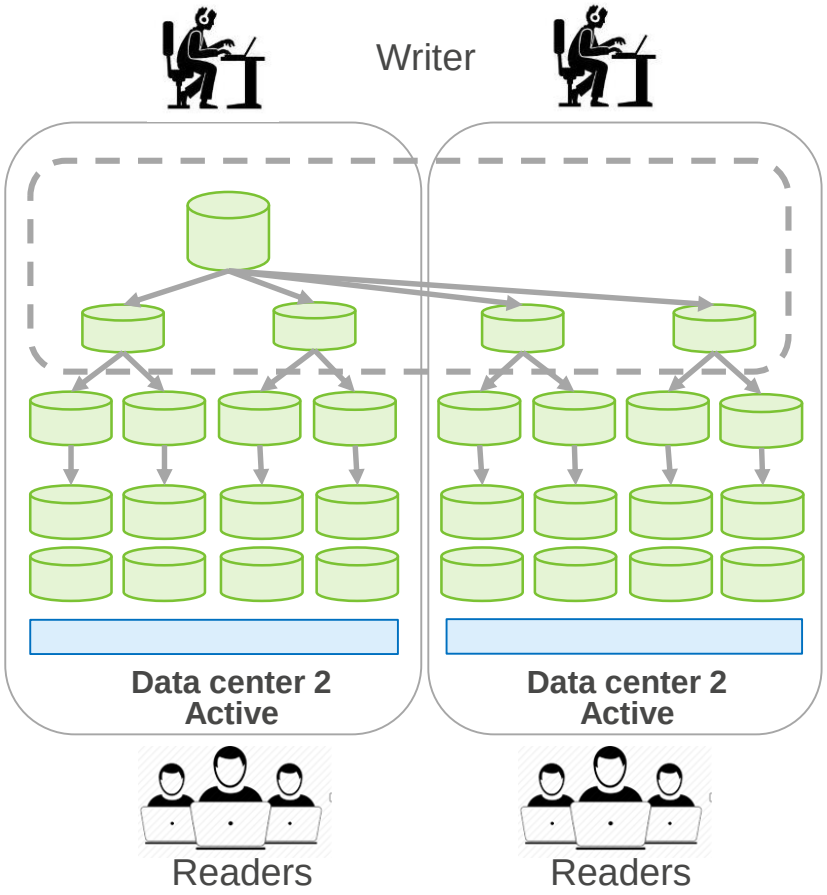
0 to 150M transactions <u>in minutes!</u>



Challenges we face with MySQL

Ticketmaster has heavy MySQL use cases
Examples of our MySQL setup





Challenges we face with MySQL

ticketmaster®

Scalability

- Scaled for max capacity at all times

Replication Lag

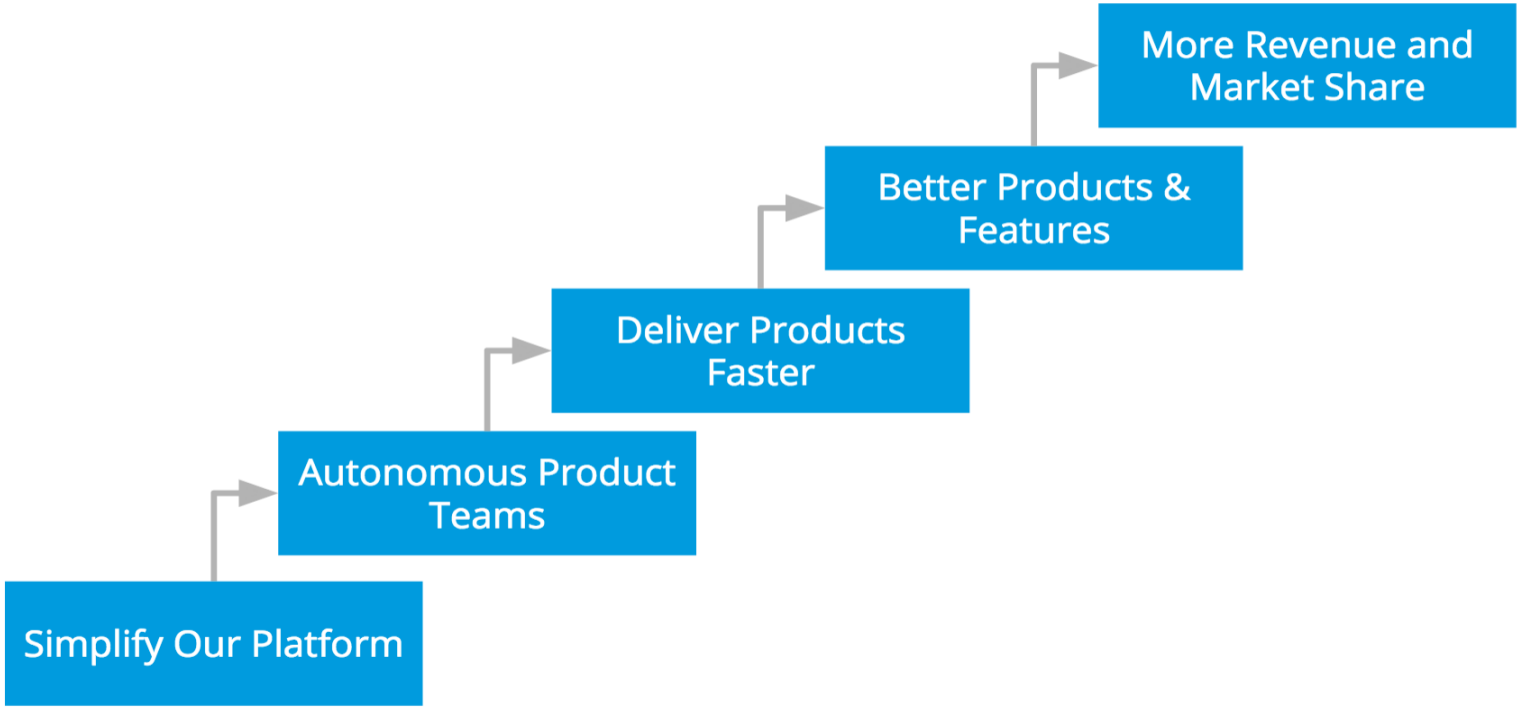
- Read consistency cross slaves
- Single thread replication
- Application can cause high replication lags

Operations

- Manual failover
- Maintenance on multi-master setup
- Version updates

Where are we headed..

SIMPLIFY OUR PLATFORM



Where are we headed..

ticketmaster®

…. We are moving to AWS and to Aurora!

Why did we choose Aurora?

ticketmaster®

Compatibility

- Fully compatible with MySQL 5.6
- Migration from MySQL using MySQL replication
- Migration from Oracle leverages DMS and SCT

Ease of Operation

- Replication Setup - Auto Failover
- Reader and Writer Endpoint
- Tuned Parameters
- Auto minor version upgrade

Scalability

- Vertically: Add more CPU and memory per instance
- Horizontally: Add more database instances
- Storage: Automatically add space and IOPs

Our success story

ticketmaster®

Account Manager, our first product migrated to AWS.

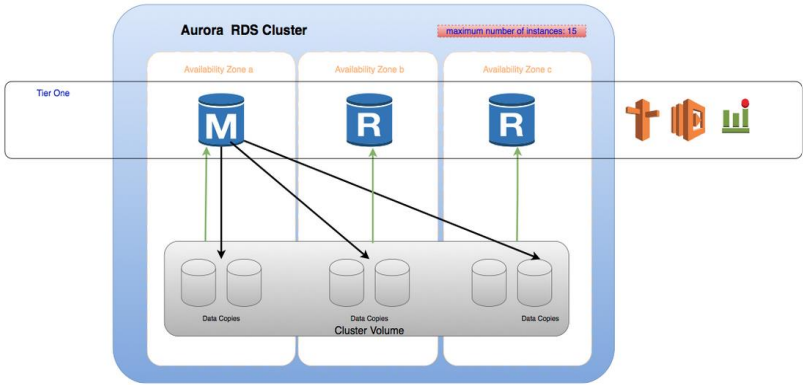
- Migrated from MySQL 5.6 to Aurora
- 0 issue be reported since launch
- Replication lag is only 20ms
- 12 MySQL servers to 5 Aurora nodes
- Deployment time took from 1~2 hours to 20 minutes in AWS
- Build a new test environment took from 1 week to 30 minutes in AWS

Our MySQL application worked on Aurora without any code changes

Our Tools: Terraformer

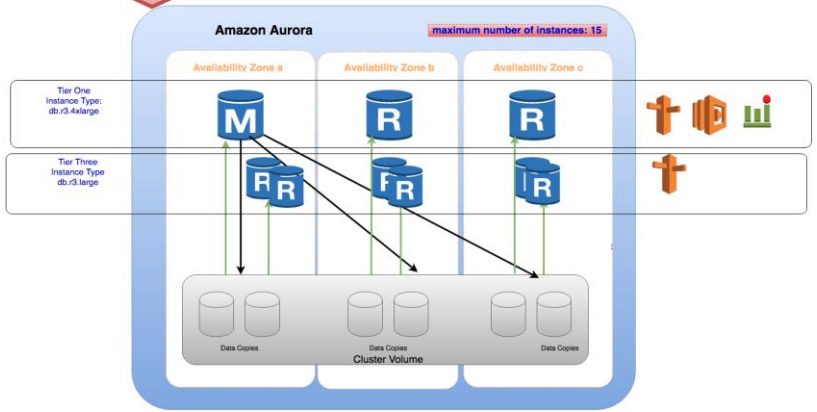
Terraform ticketmaster®

Regular Business Traffic



OnSale Traffic

Peak Traffic



- Infrastructure in code including database!
- We use terraformer to manage Aurora
  - Scale in/out horizontally within **10 minutes**, without downtime
  - Scale up/down vertically with **30s** of failover downtime

Thank you!