Amazon Aurora

Relational database reimagined for the cloud

What is Amazon Aurora?

MySQL-compatible relational database

Performance and **availability** of commercial databases

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

Delivered as a managed service

Customers have been frustrated by the proprietary nature, high cost, and licensing terms of traditional, commercial-grade database providers. And while many companies have started moving toward more open engines like MySQL and Postgres, they often struggle to get the performance they need. Customers asked us if we could eliminate that inconvenient trade-off, and **that’s why we built Aurora.**

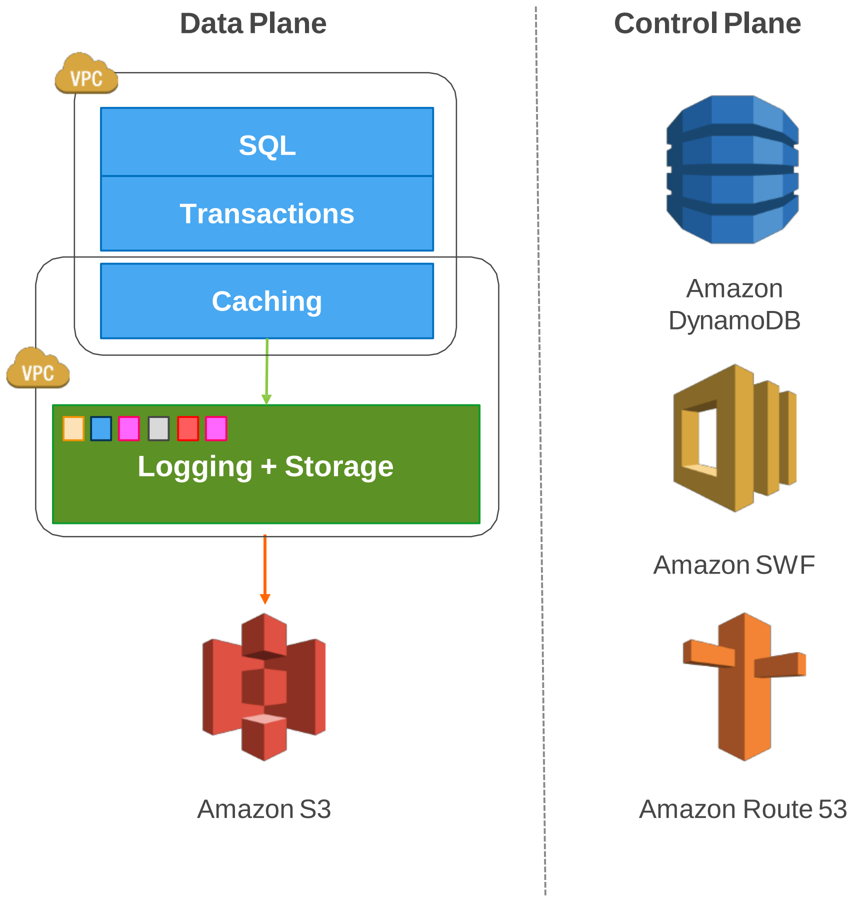
Jeff Bezos, Founder and CEO, Amazon.com
Annual letter to the share holders, 2016

Re-imagined for the cloud

1 Architected for the cloud – e.g. moved the logging and storage layer into a multitenant, scale-out database-optimized storage service

2 Leverages existing AWS services: Amazon EC2, Amazon VPC, Amazon DynamoDB, Amazon SWF, and Amazon S3

3 Maintain compatibility with MySQL – customers can migrate their MySQL applications as-is, use all MySQL tools.



Aurora customer adoption

Fastest growing service in AWS history

Expedia: Online travel marketplace

Migration from SQL Server

World's leading online travel company, with a portfolio that includes more than 150 travel sites in 70 countries.

- Real-time business intelligence and analytics on a growing corpus of online travel market place data.
- Current SQL server based architecture is too expensive. Performance degrades as data volume grows.
- Cassandra with Solr index requires large memory footprint and hundreds of nodes, adding cost.

Aurora benefits:

- Aurora meets scale and performance requirements with much lower cost.
- 25,000 inserts/sec with peak up to 70,000. 30 ms average response time for write and 17 ms for read.

Pearson Education: Publishing and testing

Migration from MySQL

A leading firm in educational publishing, testing, and certification, as well as in-class and online learning tools. Pearson operates in over 70 countries and serves millions of students.

- Pearson's applications enable student registration, instruction, and testing
- Database reliability is critical when dealing with millions of students' data. Data cannot be lost.

Aurora benefits:

- Allowed Pearson to stop self-managing their database while still achieving their performance and availability goals.
- "No more babysitting MySQL traditional async replication"

Higher Performance, Lower Cost

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

SAFE SOFTWARE

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

igt DOUBLE DOWN

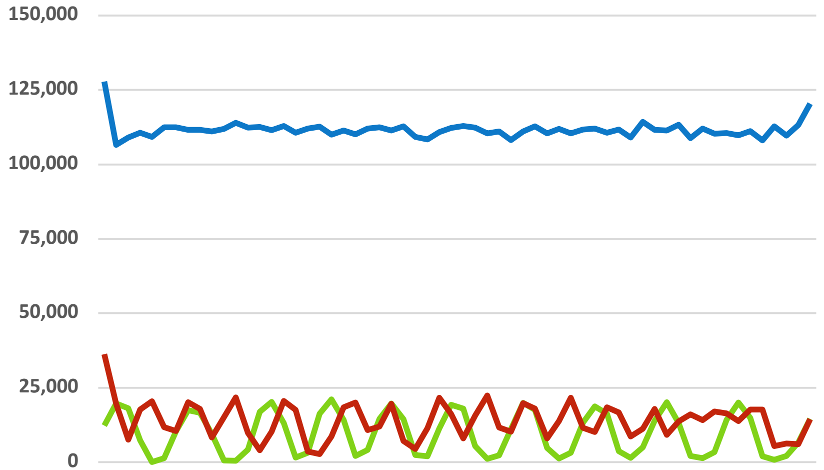
Double Down Interactive (gaming) lowered their bill by 67% while also achieving better latencies (most queries ran faster) and lower CPU utilization.

Amazon Aurora is fast …

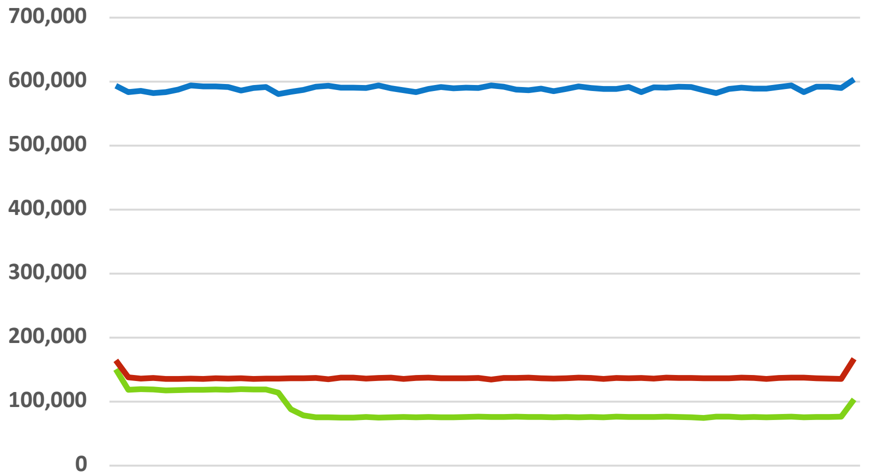
5x faster than MySQL on SYSBENCH

5X faster than RDS MySQL 5.6 & 5.7

WRITE PERFORMANCE



READ PERFORMANCE



MySQL SysBench results

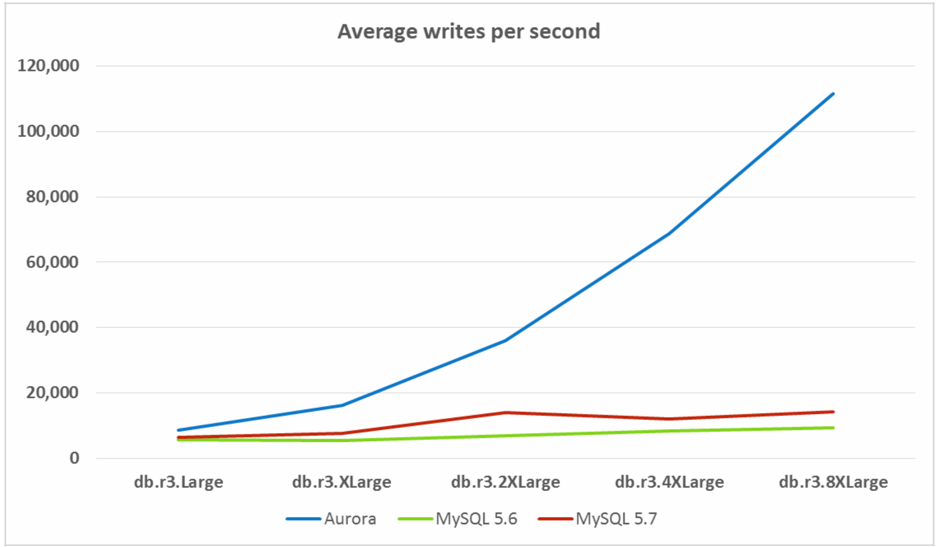
R3.8XL: 32 cores / 244 GB RAM

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

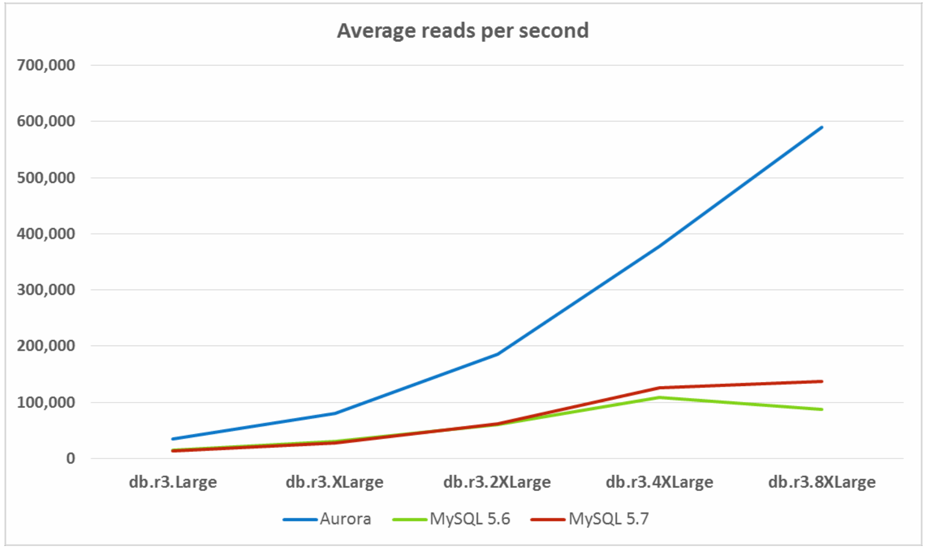
Five times higher throughput than stock MySQL based on industry standard benchmarks.

Scaling with instance sizes

WRITE PERFORMANCE



READ PERFORMANCE

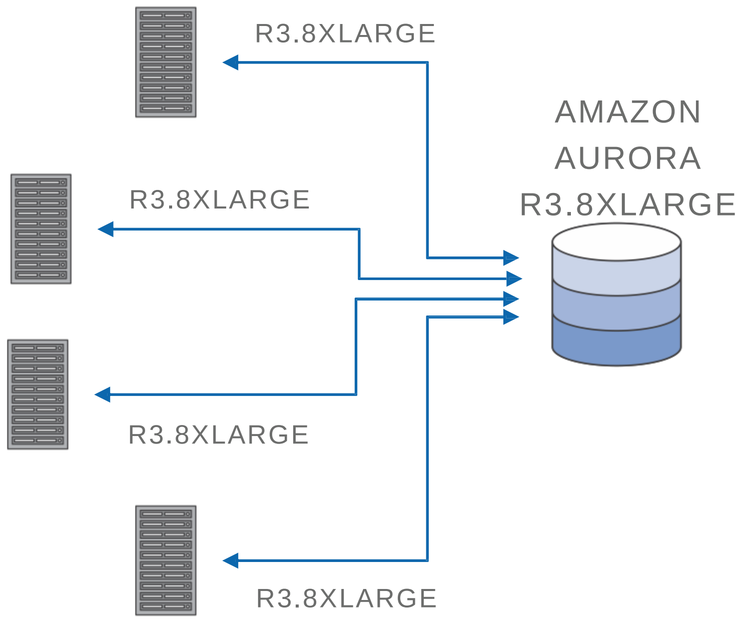


Aurora 🟦	MySQL 5.6 🟩	MySQL 5.7 🟥

Aurora scales with instance size for both read and write.

Reproducing benchmark results

1. Create an Amazon VPC (or use an existing one).
2. Create four EC2 R3.8XL client instances to run the SysBench client. All four should be in the same AZ.
3. Enable enhanced networking on your clients
4. Tune your Linux settings (see whitepaper)
5. Install Sysbench version 0.5
6. Launch a r3.8xlarge Amazon Aurora DB Instance in the same VPC and AZ as your clients
7. Start your benchmark!



https://d0.awsstatic.com/product-marketing/Aurora/RDS_Aurora_Performance_Assessment_Benchmarking_v1-2.pdf

Beyond benchmarks

If only real world applications saw benchmark performance

POSSIBLE DISTORTIONS

Real world requests contend with each other

Real world metadata rarely fits in data dictionary cache

Real world data rarely fits in buffer cache

Real world production databases need to run with HA enabled

How did we achieve this?

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 DOES NOT ALLOW CONTEXT SWITCHES

What about availability

“Performance only matters if your database is up”

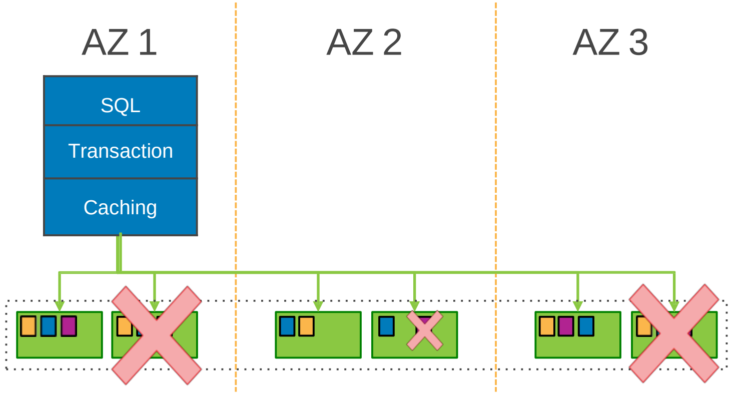
Fault-tolerant storage

Six copies across three availability zones

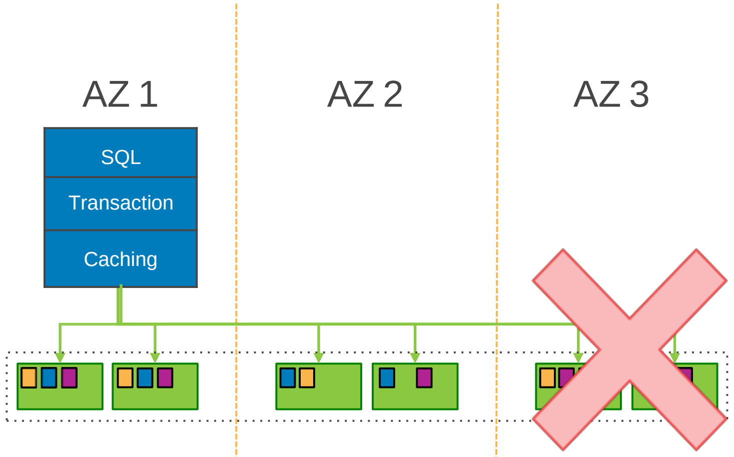
4 out 6 write quorum; 3 out of 6 read quorum

Peer-to-peer replication for repairs

Volume striped across hundreds of storage nodes



Read availability



Read and write availability

Read replicas are failover targets

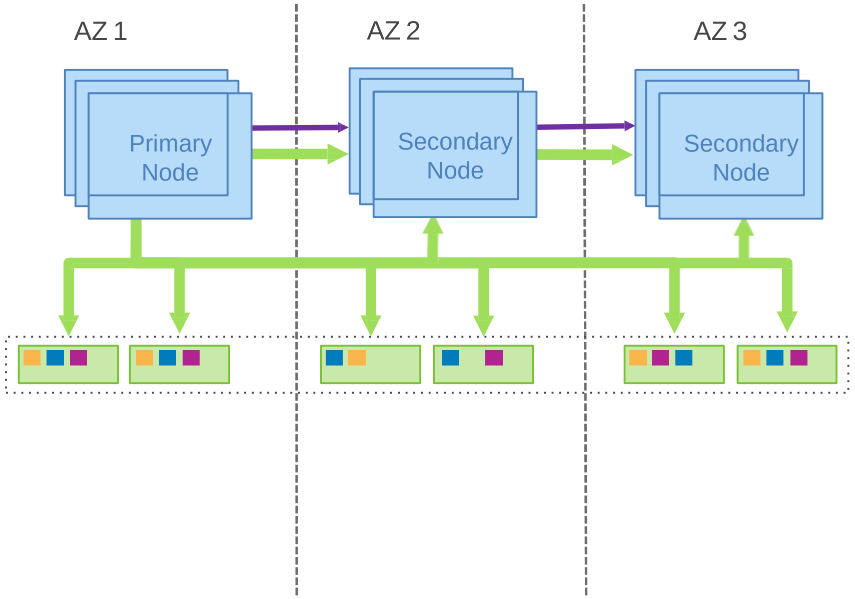
Aurora cluster contains primary node and up to fifteen secondary nodes

Failing database nodes are automatically detected and replaced

Failing database processes are automatically detected and recycled

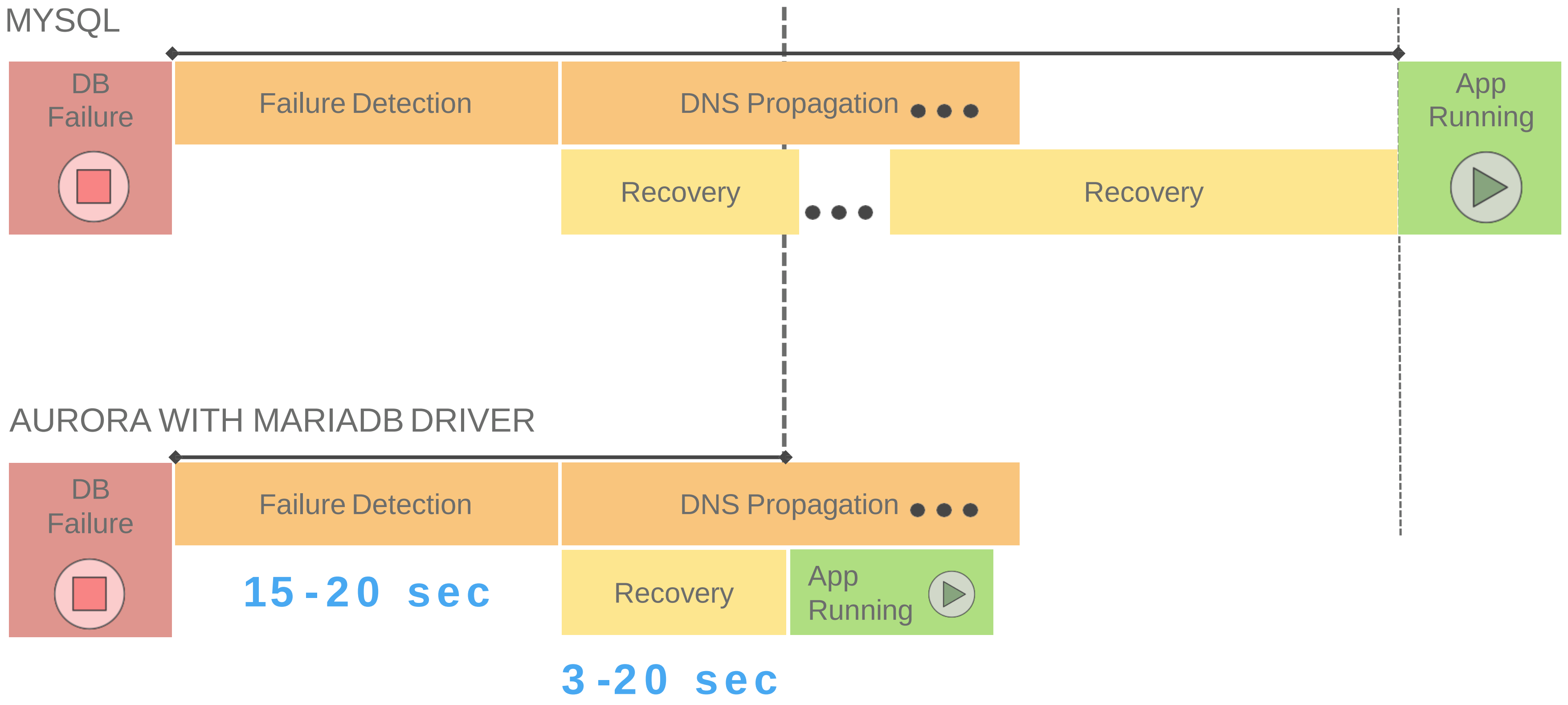
Secondary nodes automatically promoted on persistent outage, no single point of failure

Customer application may scale-out read traffic across secondary nodes



- Customer specifiable fail-over order
- Read balancing across read replicas

Faster failover



Simulate failures using SQL

To cause the failure of a component at the database node:

```
ALTER SYSTEM CRASH [{INSTANCE | DISPATCHER | NODE}]
```

To simulate the failure of disks:

```
ALTER SYSTEM SIMULATE percent_failure DISK failure_type IN
[DISK index | NODE index] FOR INTERVAL interval
```

To simulate the failure of networking:

```
ALTER SYSTEM SIMULATE percent_failure NETWORK failure_type
[TO {ALL | read_replica | availability_zone}] FOR INTERVAL interval
```

Compatible with the MySQL ecosystem

Use all your existing tools and applications

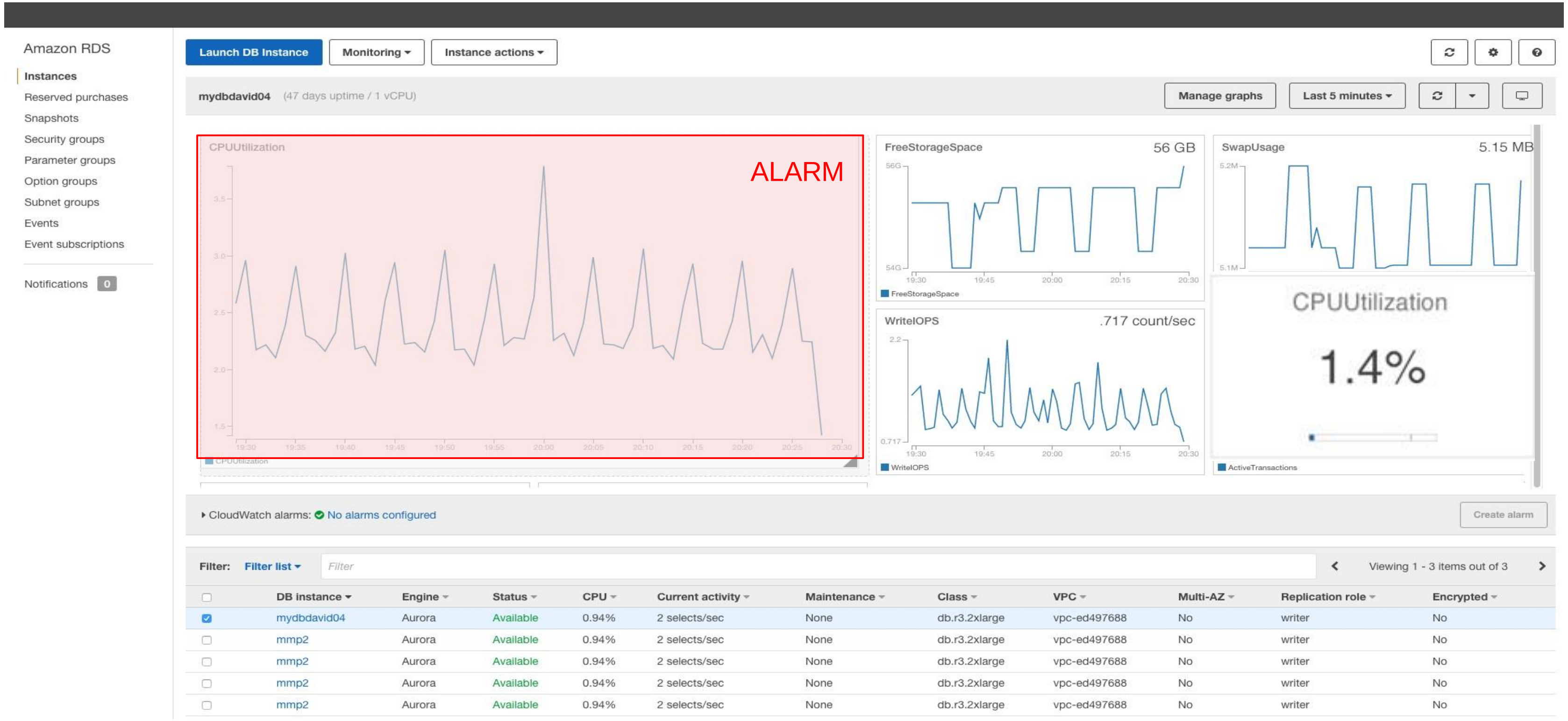
Well established MySQL ecosystem

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

Business Intelligence	Data Integration	Query and Monitoring	SI and Consulting
tableau	talend	Webyog	8K Miles
ZOOMDATA	ATTUNITY	DATADOG	2ND WATCH
looker	informatica	Navicat Premium for Aurora	NORDCLOUD
			slalom
			Pythian love your data
			apps associates extreme expertise

Source: Amazon

Advanced monitoring

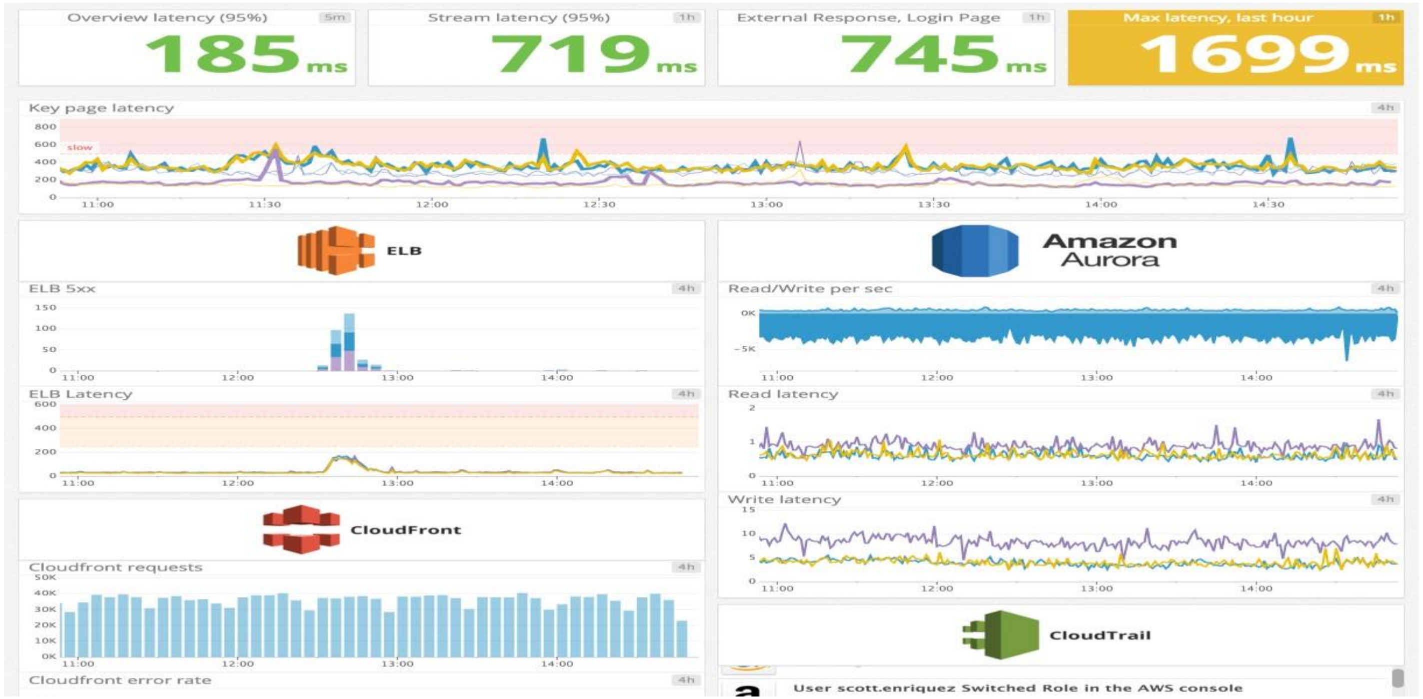


	DB instance	Engine	Status	CPU	Current activity	Maintenance	Class	VPC	Multi-AZ	Replication role	Encrypted
✓	mydbdavid04	Aurora	Available	0.94%	2 selects/sec	None	db.r3.2xlarge	vpc-ed497688	No	writer	No
	mmp2	Aurora	Available	0.94%	2 selects/sec	None	db.r3.2xlarge	vpc-ed497688	No	writer	No
	mmp2	Aurora	Available	0.94%	2 selects/sec	None	db.r3.2xlarge	vpc-ed497688	No	writer	No
	mmp2	Aurora	Available	0.94%	2 selects/sec	None	db.r3.2xlarge	vpc-ed497688	No	writer	No
	mmp2	Aurora	Available	0.94%	2 selects/sec	None	db.r3.2xlarge	vpc-ed497688	No	writer	No

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

alarms on specific metrics | egress to Amazon CloudWatch Logs | integration with third-party tools

Monitoring the whole stack



User scott.enriquez Switched Role in the AWS console

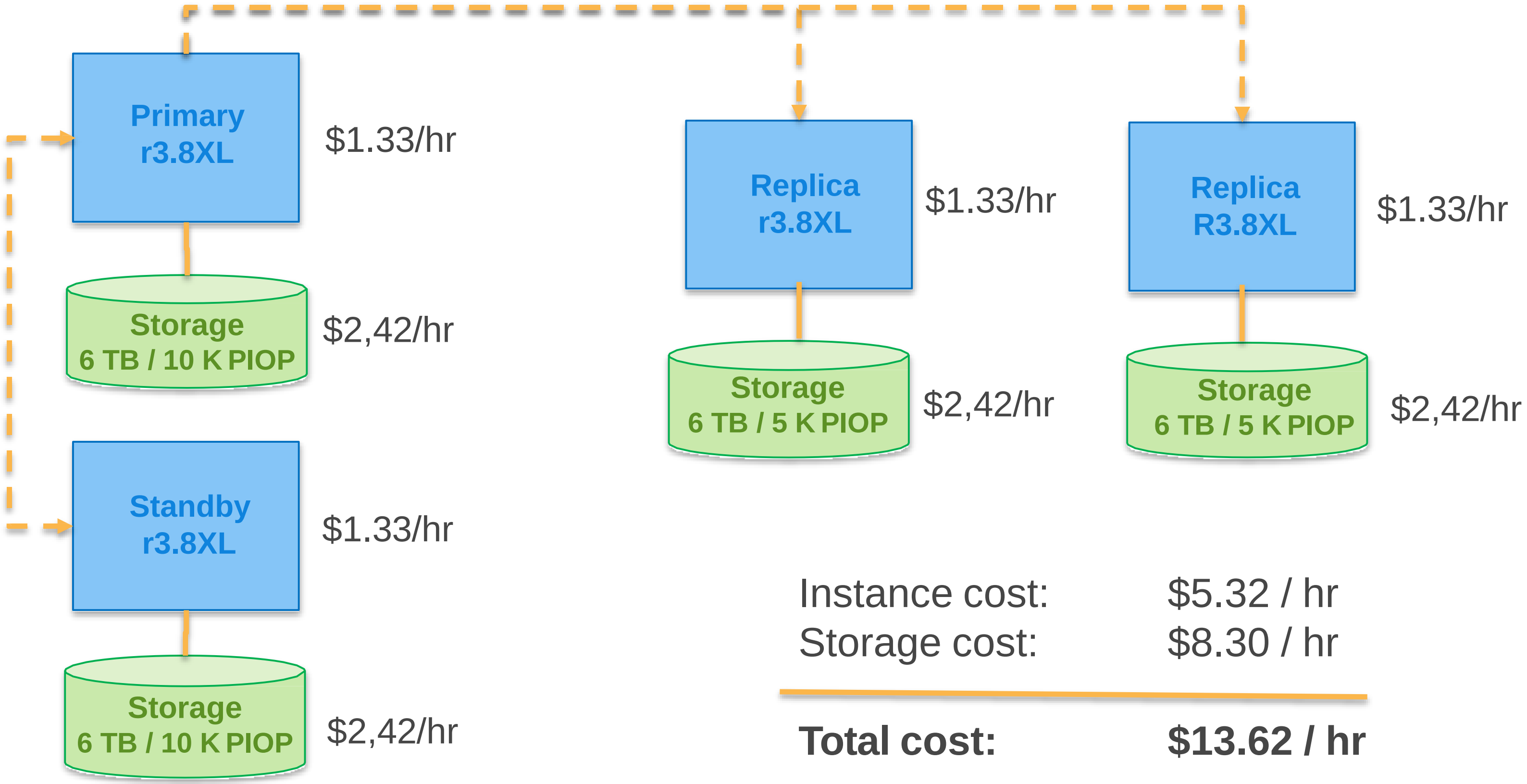
Correlate Aurora metrics with metrics and events from the rest of your infrastructure

How much does it cost?

Less expensive than MySQL
1/10 the cost of commercial databases

Cost of ownership: Aurora vs. MySQL

MySQL configuration hourly cost

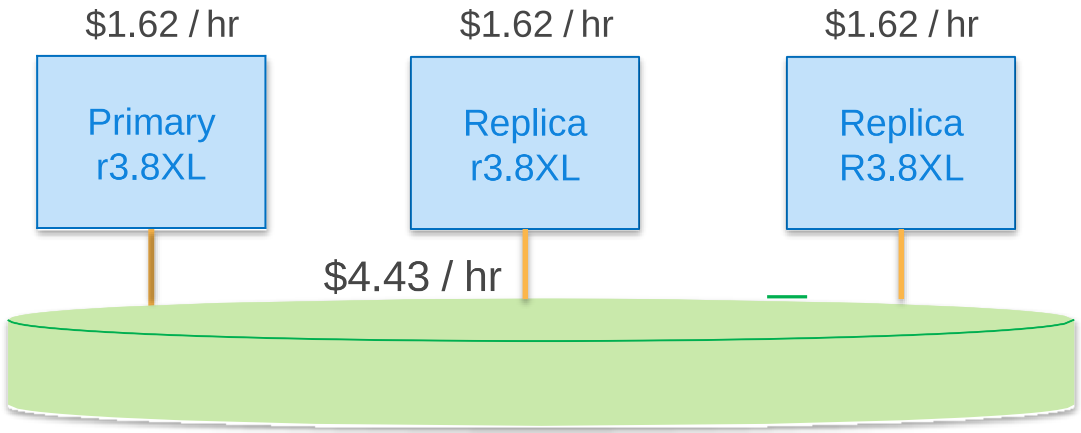


Instance cost:	$5.32 / hr
Storage cost:	$8.30 / hr
Total cost:	**$13.62 / hr**

Cost of ownership: Aurora vs. MySQL

Aurora configuration hourly cost

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



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

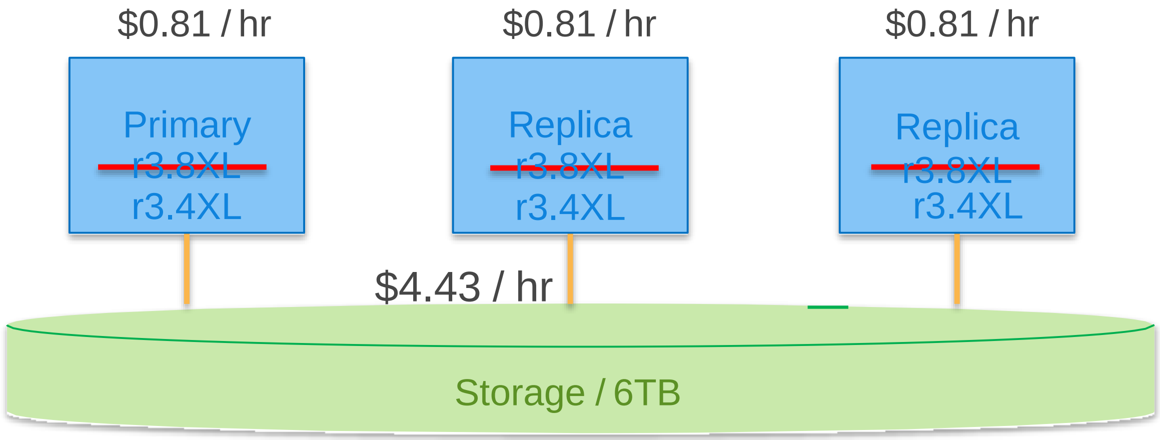
31.8% Savings

*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**

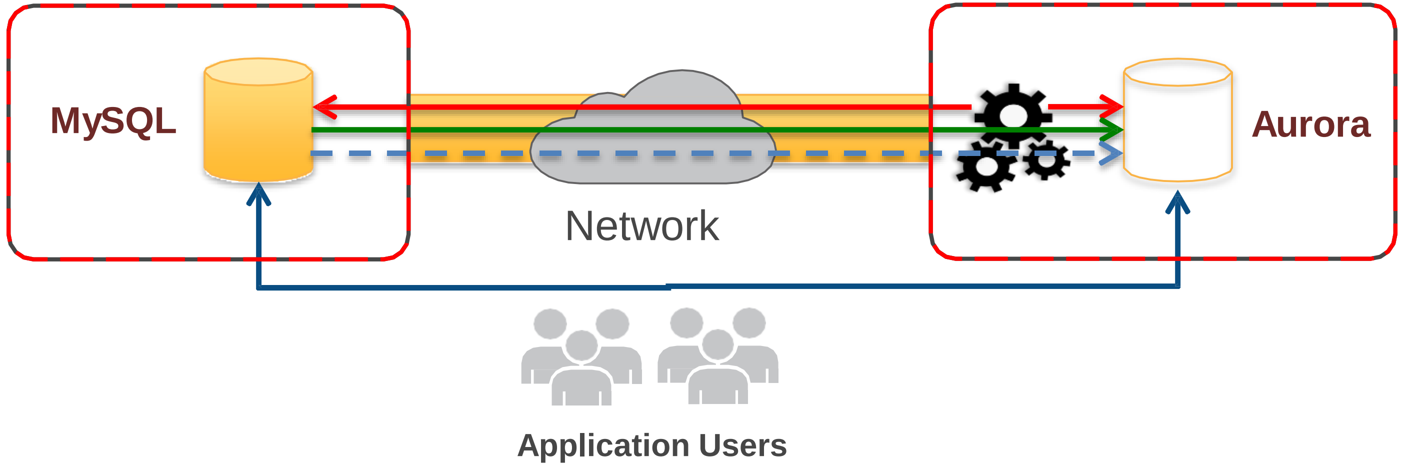
49.6% Savings

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

Ready to move?

We made it easy to migrate ..

Simplify migration from RDS MySQL



1. Establish baseline

a. RDS MySQL to Aurora DB snapshot migration

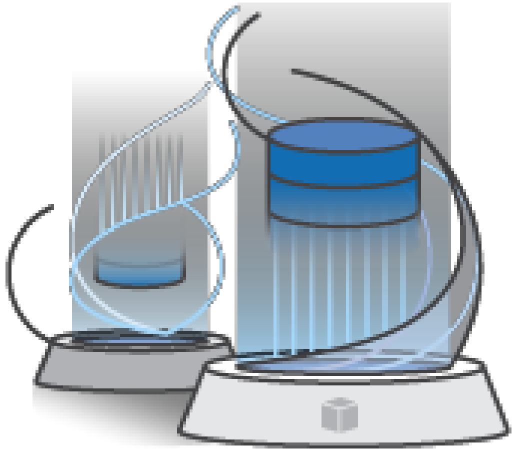
b. MySQL dump/import

2. Catch-up changes

a. Binlog replication

b. Tungsten replicator

Migration from EC2 & on-premise MySQL



Data migration service

- Logical data replication from on-premise or EC2
- Code & schema conversion across engines

S3 integration

- Load partial datasets directly from / to S3
- Ingest large database snapshots (>2TB)

- Snowball integration
  - Ingest huge database snapshots (>10TB)
  - Send us your data in a suitcase!

Migration non-MySQL databases

AWS Database Migration Service

ORACLE

Microsoft SQL Server

PostgreSQL

MariaDB

- ✓ Move data to the same or different database engine
- ✓ Keep your apps running during the migration
- ✓ Start your first migration in 10 minutes or less
- ✓ Replicate within, to, or from Amazon EC2 or RDS