Amazon Aurora with PostgreSQL Compatibility

Kevin Jernigan, Senior Product Manager, Amazon Aurora PostgreSQL

January 24, 2018

aws

Agenda

- Why did we build Amazon Aurora?
- PostgreSQL compatibility
- Durability and availability architecture
- Performance architecture
- Performance results vs. PostgreSQL
- Announcing *Performance Insights*
- Getting data In
- Feature roadmap
- Questions

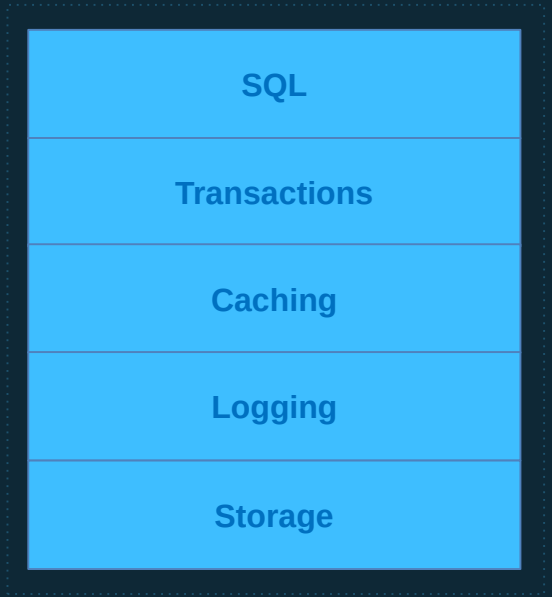
aws

A Brief History of Amazon RDS

- 2006: Amazon launches Amazon Web Services (AWS) with Simple Storage Service (S3)
- 2009: Amazon Relational Database Service (RDS) launches with support for MySQL
- 2012: Amazon RDS adds support for Oracle and SQL Server
- 2013: Amazon RDS adds support for PostgreSQL
- 2014: Amazon RDS announces Amazon Aurora
- July 2015: Amazon Aurora with MySQL compatibility goes GA
- October 2017: Amazon Aurora with PostgreSQL compatibility goes GA

aws

Traditional relational databases are hard to scale

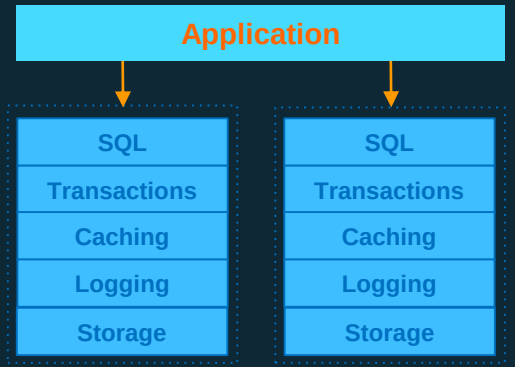


Multiple layers of functionality all in a monolithic stack

Traditional approaches to scale databases

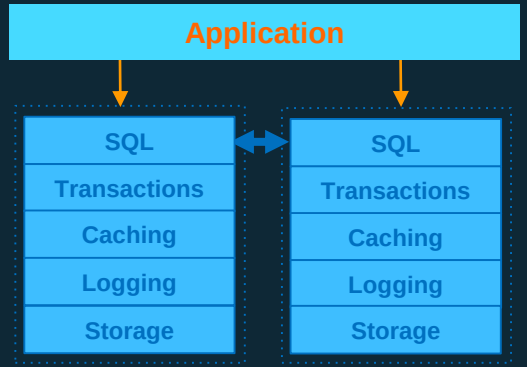
Sharding

Coupled at the application layer



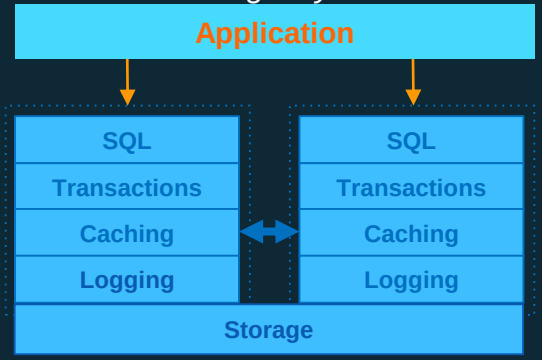
Shared Nothing

Coupled at the SQL layer



Shared Disk

Coupled at the caching and storage layer



Each architecture is limited by the monolithic mindset

aws

Reimagining the relational database

What if you were inventing the database today?

- You would break apart the stack
- You would build something that:
  - ✓ Can scale out…
  - ✓ Is self-healing…
  - ✓ Leverages distributed services…

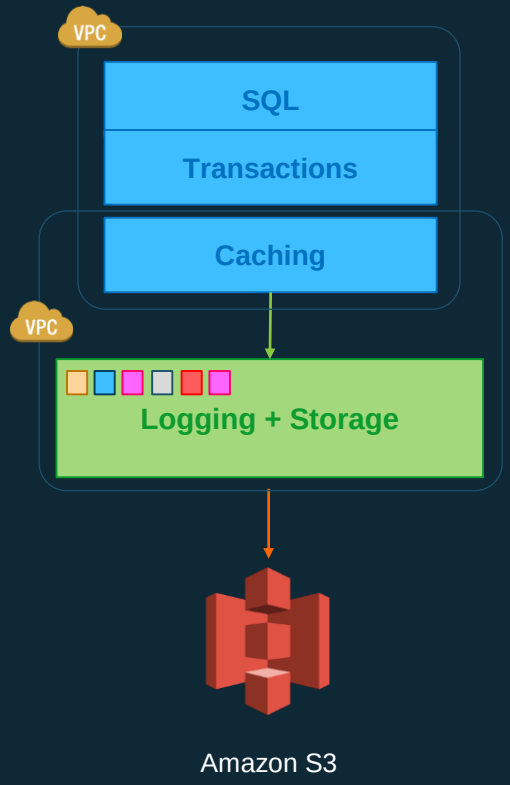
aws

A service-oriented architecture applied to the database

1. Move the logging and storage layer into a multitenant, scale-out, database-optimized storage service

2. Integrate with other AWS services like Amazon EC2, Amazon VPC, Amazon DynamoDB, Amazon SWF, and Amazon Route 53 for control & monitoring

3. Make it a managed service – using Amazon RDS. Takes care of management and administrative functions.




Amazon RDS


Amazon DynamoDB


Amazon SWF


Amazon Route 53

Making Amazon Aurora better

In 2014, we launched Amazon Aurora with MySQL compatibility.

Now we have added PostgreSQL compatibility.

Customers can now choose how to use Amazon's cloud-optimized relational database, with the performance and availability of commercial databases and the simplicity and cost-effectiveness of open source databases.

aws

PostgreSQL fast facts

Open Source Initiative

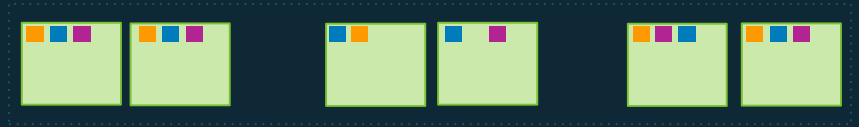
- Open source database
- In active development for 20 years
- Owned by a foundation, not a single company
- Permissive innovation-friendly open source license
- High performance out of the box
- Object-oriented and ANSI-SQL:2008 compatible
- Most geospatial features of any open source database
- Supports stored procedures in 12 languages (Java, Perl, Python, Ruby, Tcl, C/C++, its own Oracle-like PL/pgSQL, etc.)
- Most Oracle-compatible open source databases
- Highest AWS Schema Conversion Tool automatic conversion rates are from Oracle to PostgreSQL

aws

PostgreSQL compatibility

PostgreSQL 9.6 + Amazon Aurora cloud-optimized storage

- Performance: 2x-3x higher throughput than PostgreSQL alone
- Availability: failover time of < 30 seconds
- Durability: 6 copies across 3 Availability Zones
- Read Replicas: single-digit millisecond lag times on up to 15 replicas
- Fully compatible with PostgreSQL
  - Not a compatibility layer
  - Native PostgreSQL implementation



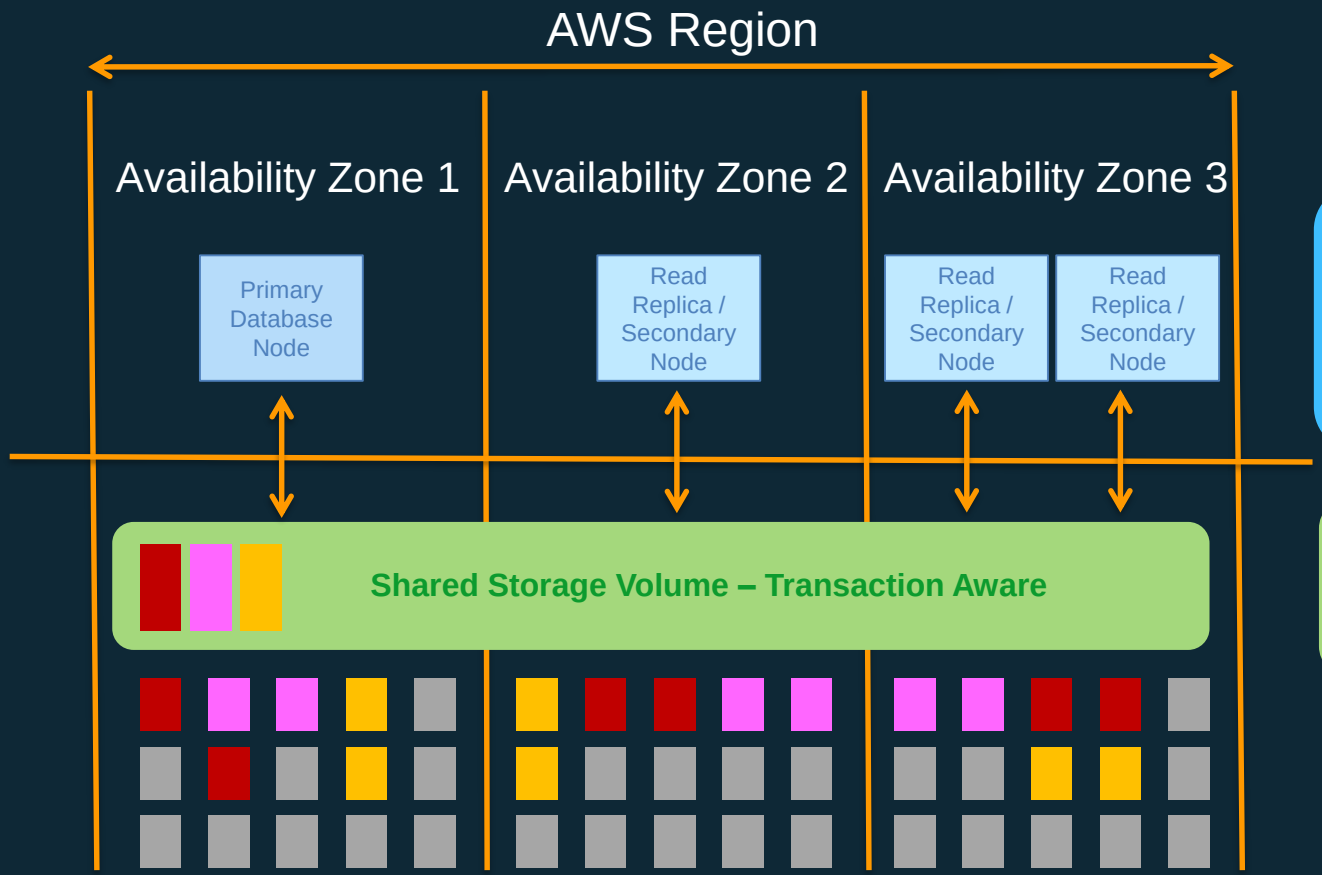
Amazon Aurora Storage

aws

Amazon Aurora with PostgreSQL Compatibility

Durability & Availability

Scale-out, distributed, log structured storage



Database and Instance Monitoring

Storage Monitoring

Amazon Aurora storage engine overview

Data is replicated 6 times across 3 Availability Zones

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

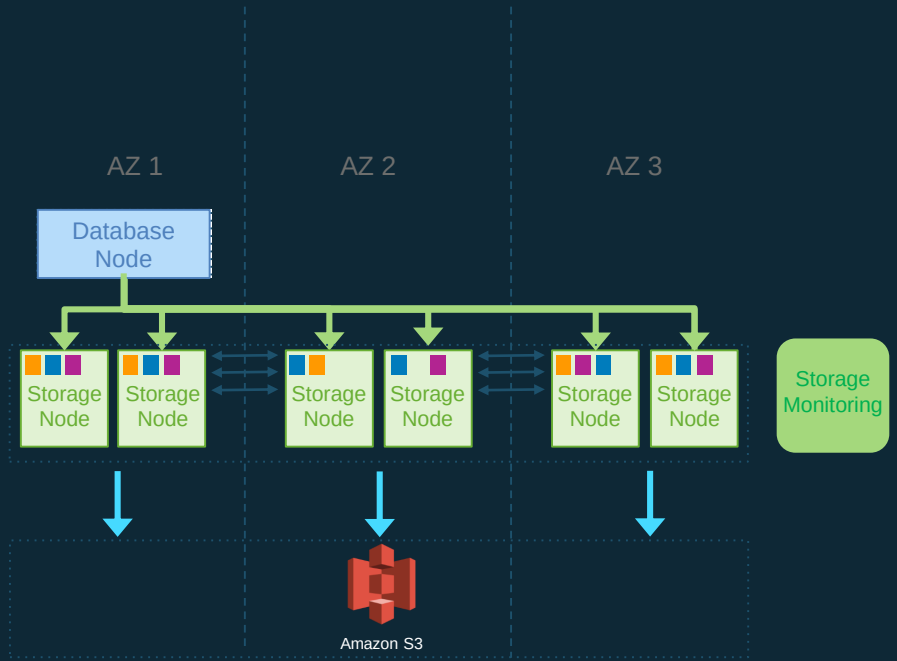
Continuous monitoring of nodes and disks for repair

10GB segments as unit of repair or hotspot rebalance

Quorum system for read/write; latency tolerant

Quorum membership changes do not stall writes

Storage volume automatically grows up to 64 TB



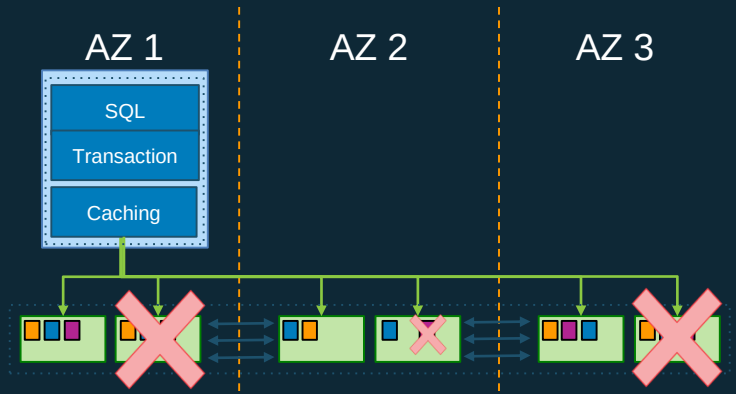
Amazon Aurora storage engine fault tolerance

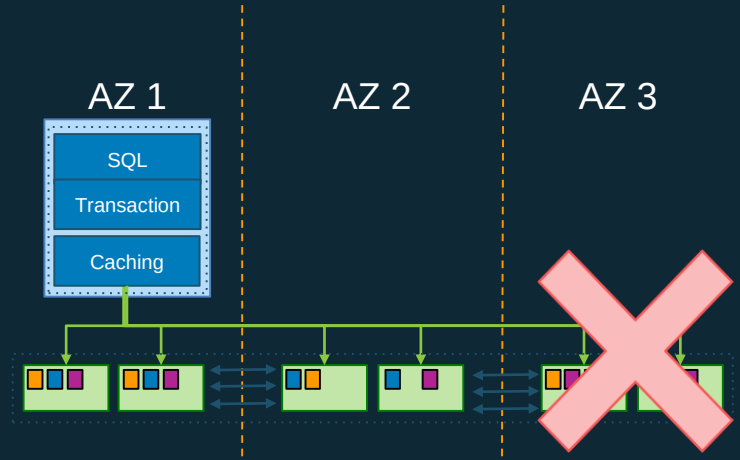
What can fail?

- Segment failures (disks)
- Node failures (machines)
- AZ failures (network or datacenter)

Optimizations

- 4 out of 6 write quorum
- 3 out of 6 read quorum
- Peer-to-peer replication for repairs





Amazon Aurora Read Nodes

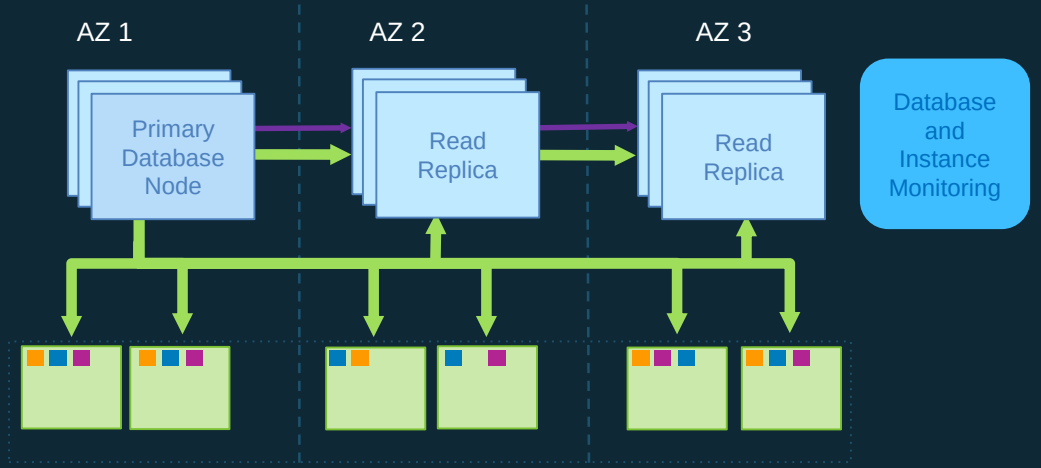
Availability

Failing database nodes are automatically detected and replaced

Failing database processes are automatically detected and recycled

Replicas are automatically promoted to primary if needed (failover)

Customer specifiable fail-over order

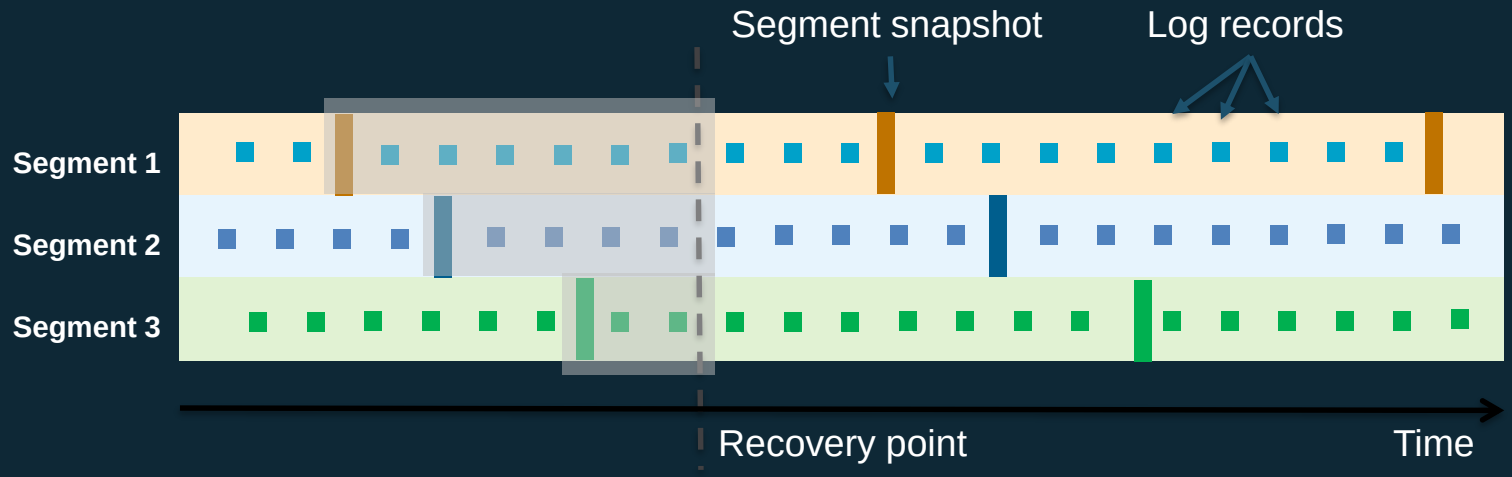


Performance

Customer applications can scale out read traffic across read replicas

Read balancing across read replicas

Amazon Aurora continuous backup



- Take periodic snapshot of each segment in parallel; stream the 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

Amazon Aurora instant crash recovery

Traditional databases

Have to replay logs since the last checkpoint

Typically 5 minutes between checkpoints

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

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

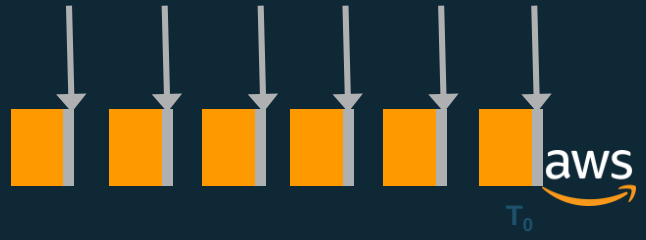
Amazon Aurora

No replay at startup because storage system is transaction-aware

Underlying storage replays log records continuously, whether in recovery or not

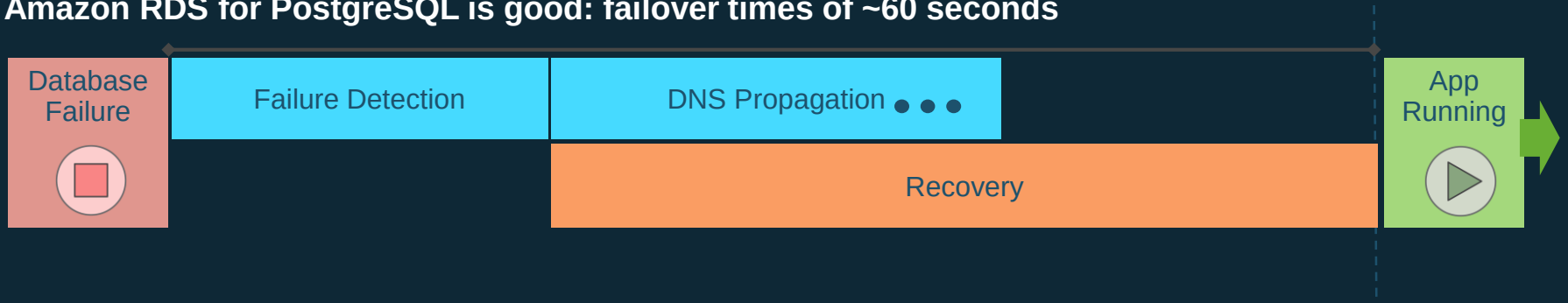
Coalescing is parallel, distributed, and asynchronous

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

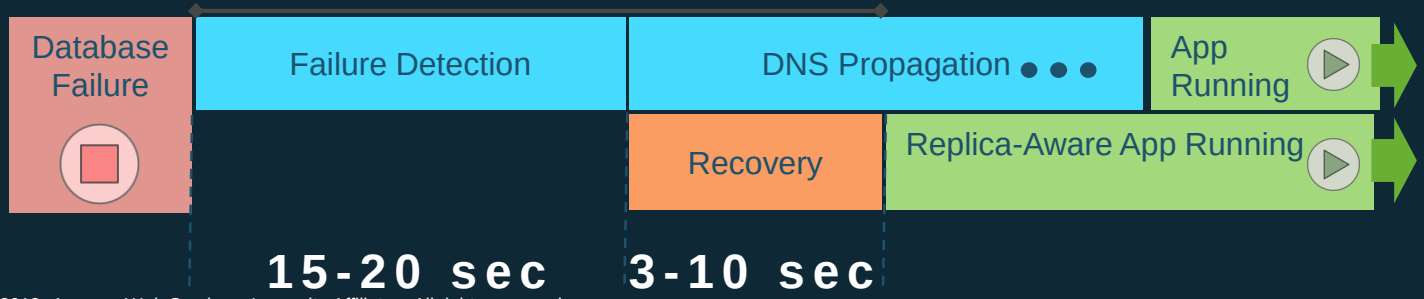


Faster, more predictable failover with Amazon Aurora

Amazon RDS for PostgreSQL is good: failover times of ~60 seconds



Amazon Aurora is better: failover times < 30 seconds



Amazon Aurora with PostgreSQL Compatibility

Performance Architecture

How does Amazon Aurora achieve high performance?

DO LESS WORK

Do fewer IOs

Minimize network packets

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

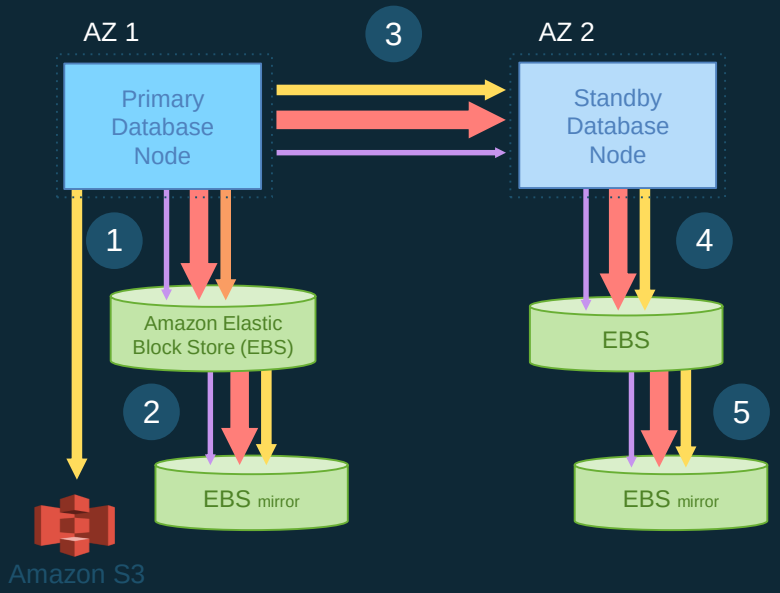
NETWORK-ATTACHED STORAGE IS ALL ABOUT PACKETS/SECOND

HIGH-THROUGHPUT PROCESSING NEEDS CPU AND MEMORY OPTIMIZATIONS

aws

Write IO Traffic in Amazon RDS for PostgreSQL

RDS FOR POSTGRESQL WITH MULTI-AZ



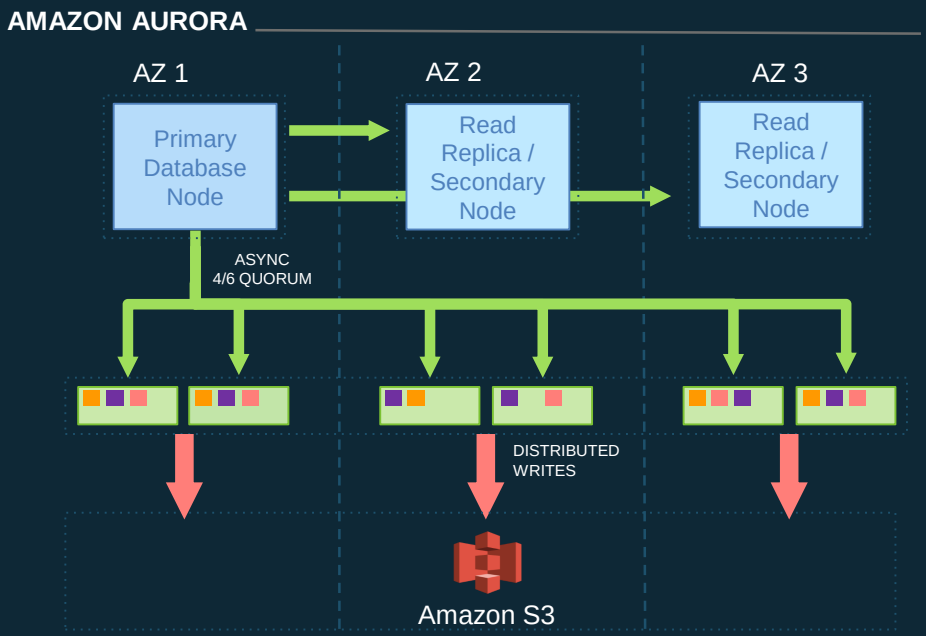
IO FLOW

- Issue write to Amazon EBS, EBS issues to mirror, acknowledge when both done
- Stage write to standby instance
- Issue write to EBS on standby instance

OBSERVATIONS

- Steps 1, 3, 5 are sequential and synchronous
- This amplifies both latency and jitter
- Many types of writes for each user operation

Write IO Traffic in an Amazon Aurora database node



IO FLOW

- Boxcar log records – fully ordered by LSN
- Shuffle to appropriate segments – partially ordered
- Boxcar to storage nodes and issue writes

OBSERVATIONS

- Only write WAL 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

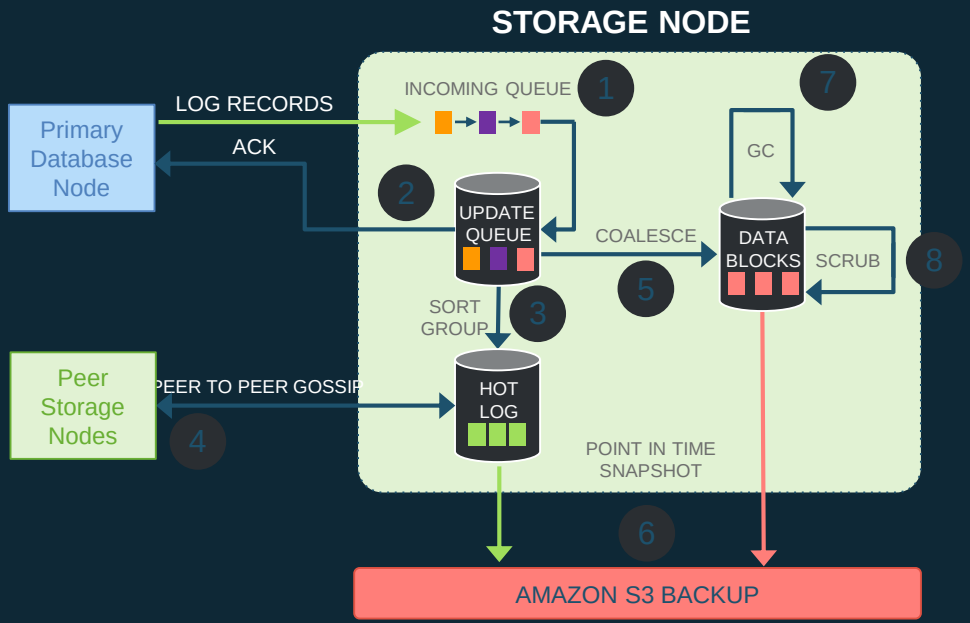
PERFORMANCE

- 2x or better PostgreSQL Community Edition performance on write-only or mixed read-write workloads

TYPE OF WRITE

AMAZON AURORA + WAL LOG | WAL | DATA | COMMIT LOG & FILES

Write IO Traffic in an Amazon Aurora storage node



IO FLOW

1. Receive record and add to in-memory queue
2. Persist record and acknowledge
3. Organize records and identify gaps in log
4. Gossip with peers to fill in holes
5. Coalesce log records into new data block versions
6. Periodically stage log and new block versions to Amazon S3
7. Periodically garbage collect old versions
8. Periodically validate CRC codes on blocks

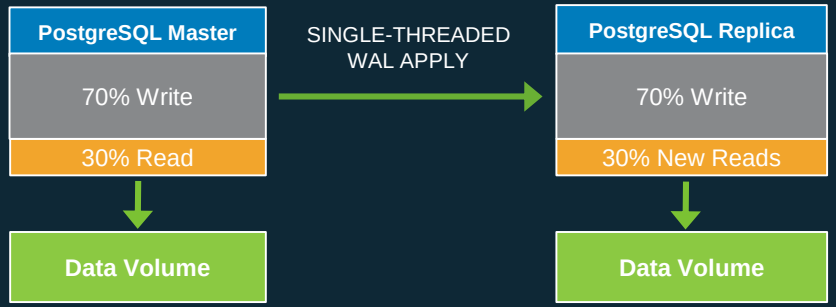
OBSERVATIONS

- All steps are asynchronous
- Only steps 1 and 2 are in foreground latency path
- Input queue is **far smaller** than PostgreSQL
- Favors latency-sensitive operations
- Uses disk space to buffer against spikes in activity

aws

IO Traffic in Aurora Replicas

POSTGRESQL READ SCALING

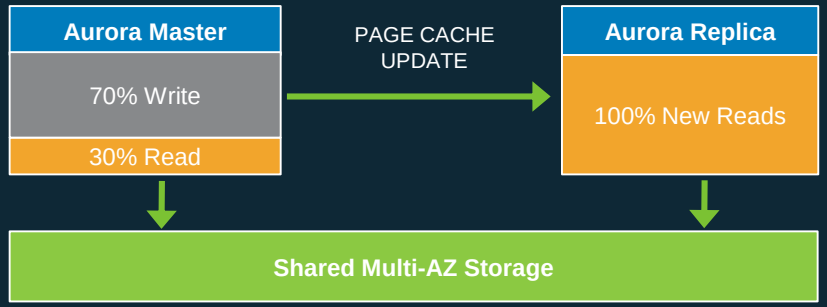


Physical: Ship redo (WAL) to Replica

Write workload similar on both instances

Independent storage

AMAZON AURORA READ SCALING



Physical: Ship redo (WAL) from Master to Replica

Replica shares storage: No writes performed

Cached pages have redo applied

Advance read view when all commits seen

Applications restart faster with survivable caches

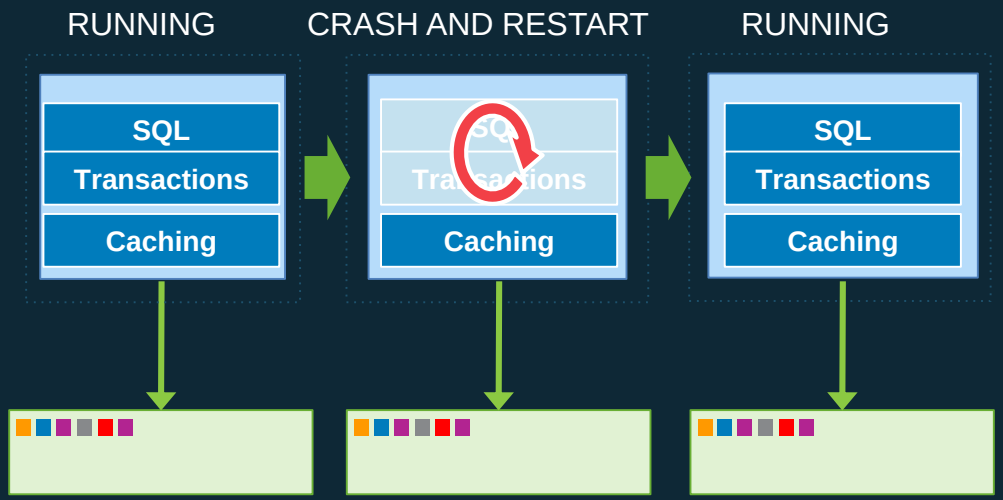
Cache normally lives inside the operating system database process– and goes away when/if that database dies

Aurora moves the cache out of the database process

Cache remains warm in the event of a database restart

Lets the database resume fully loaded operations much faster

Cache lives outside the database process and remains warm across database restarts

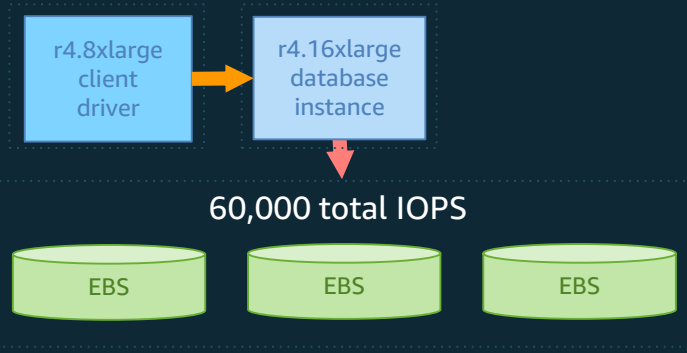


Amazon Aurora with PostgreSQL Compatibility

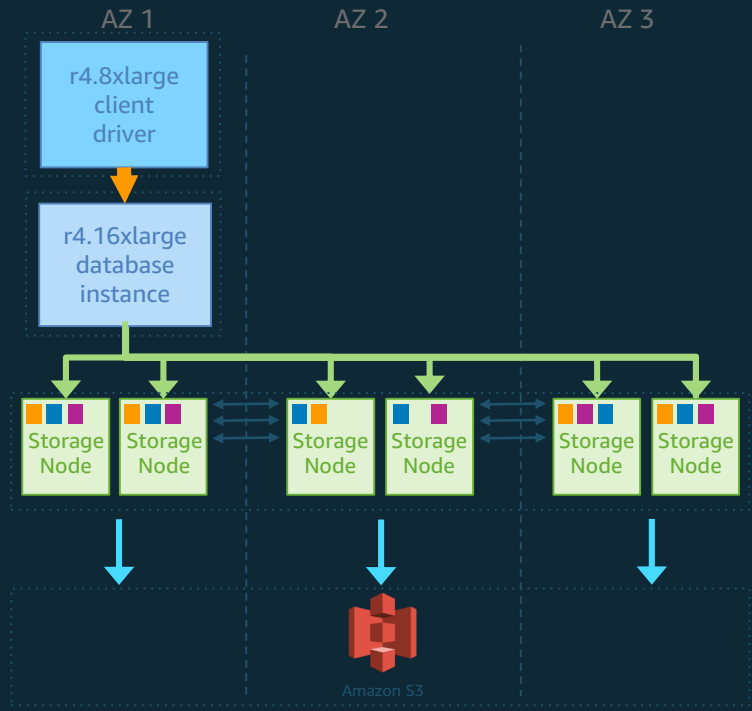
Performance vs. PostgreSQL

System Configuration

PostgreSQL – Single AZ, no backup

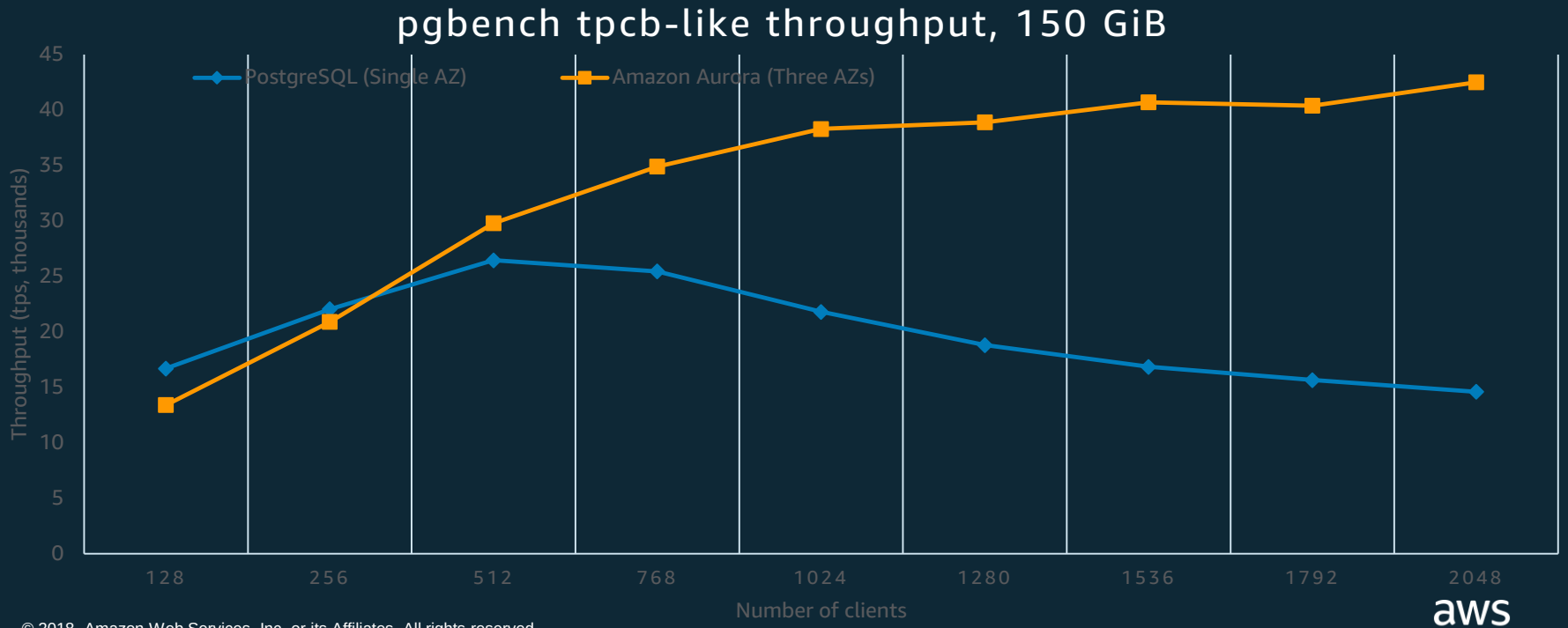


Amazon Aurora



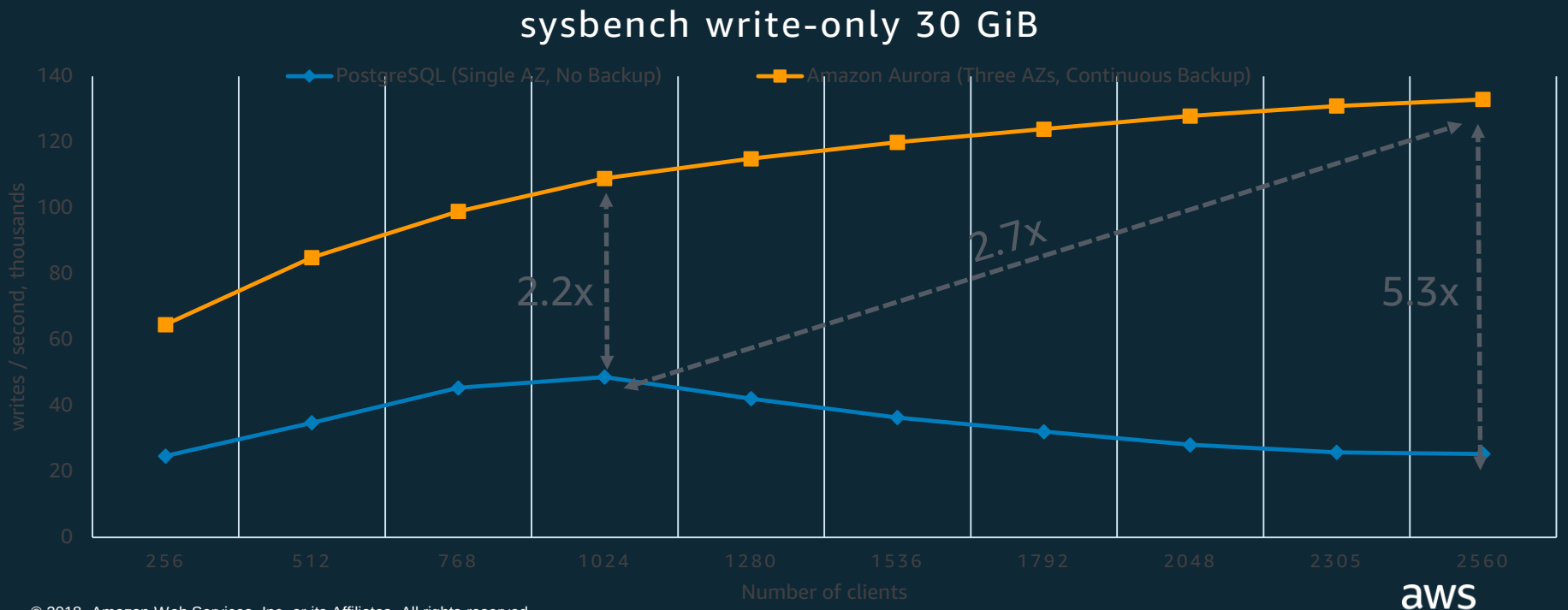
Amazon Aurora is Faster on PgBench

Running the standard pgbench benchmark, Amazon Aurora delivers 1.6x the peak throughput of PostgreSQL and 2.9x at high client counts



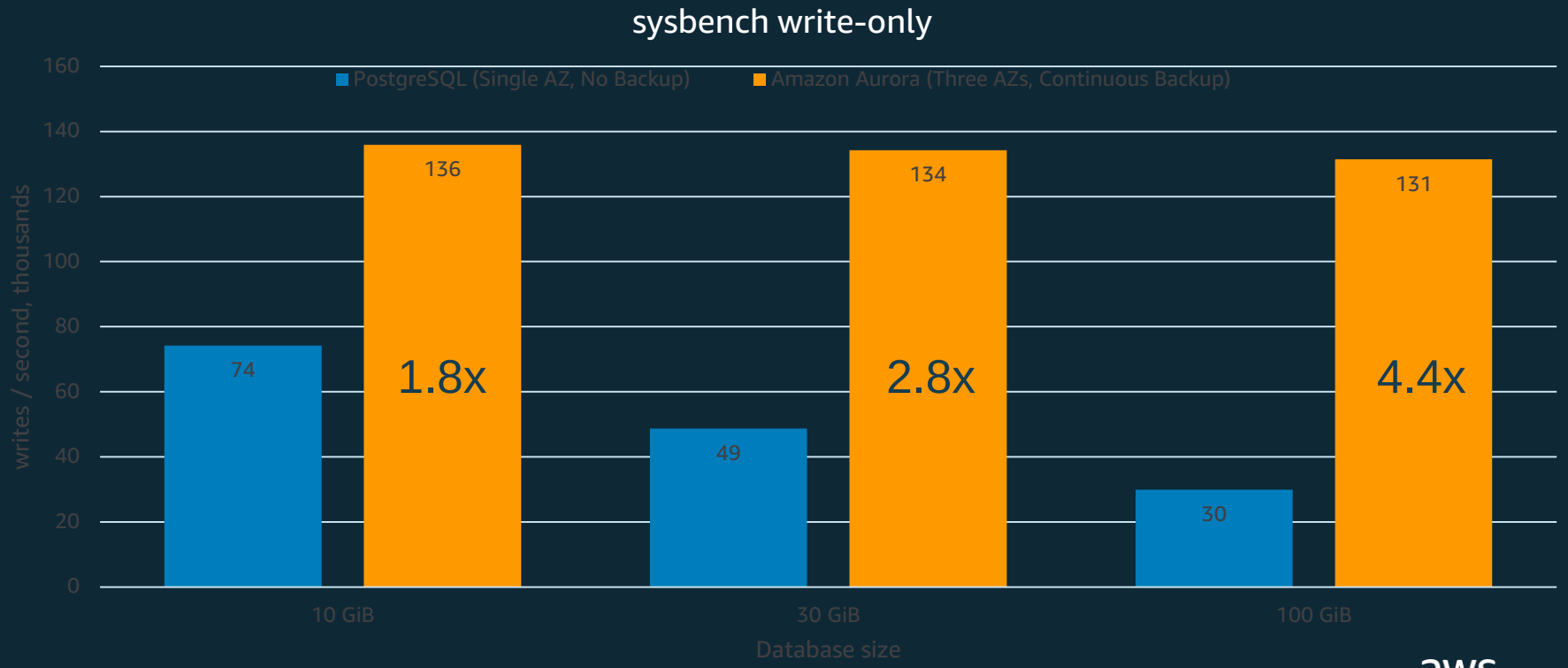
Amazon Aurora is 2x–5x Faster on Sysbench

Running the standard sysbench benchmark, Amazon Aurora delivers >2x the absolute peak of PostgreSQL and 5x at high client counts.



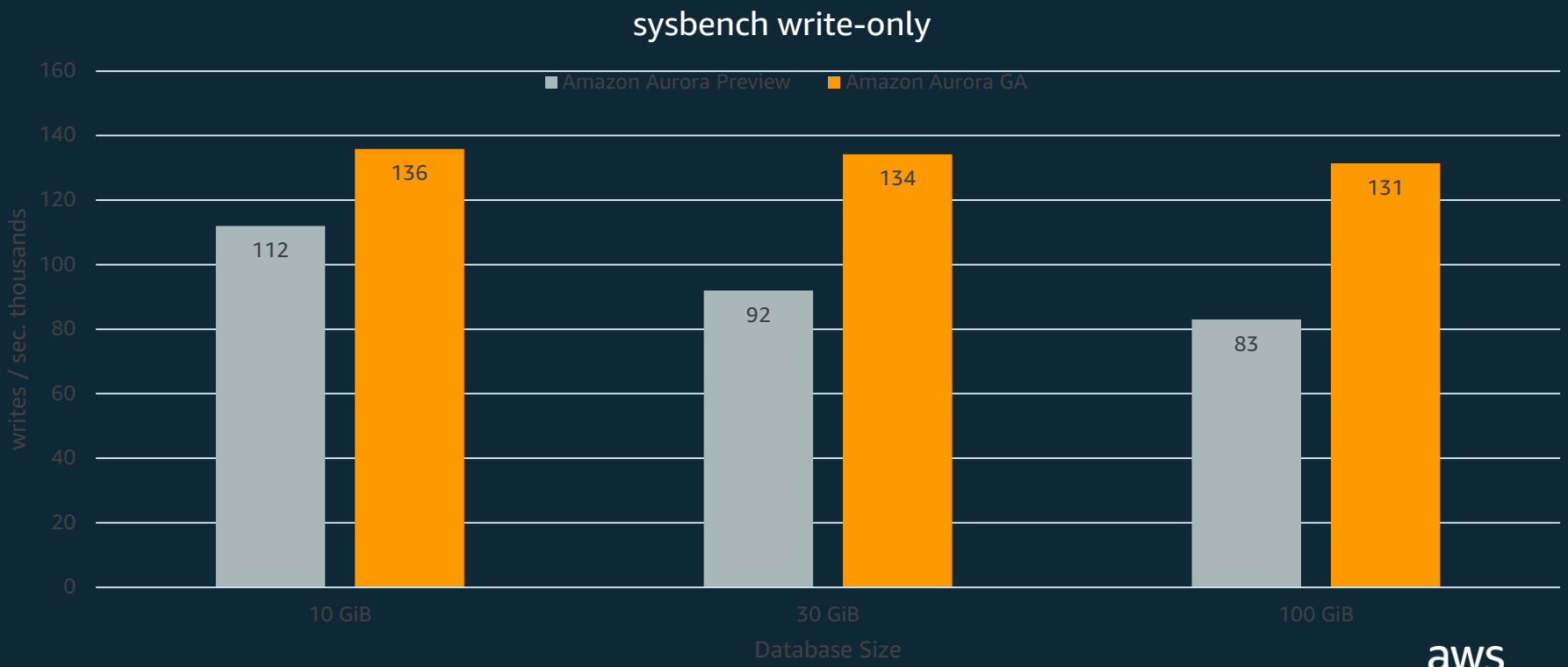
Amazon Aurora is 4x Faster at Large Scale

Scales from 1.8x to 4.4x better as database grows from 10GiB to 100GiB



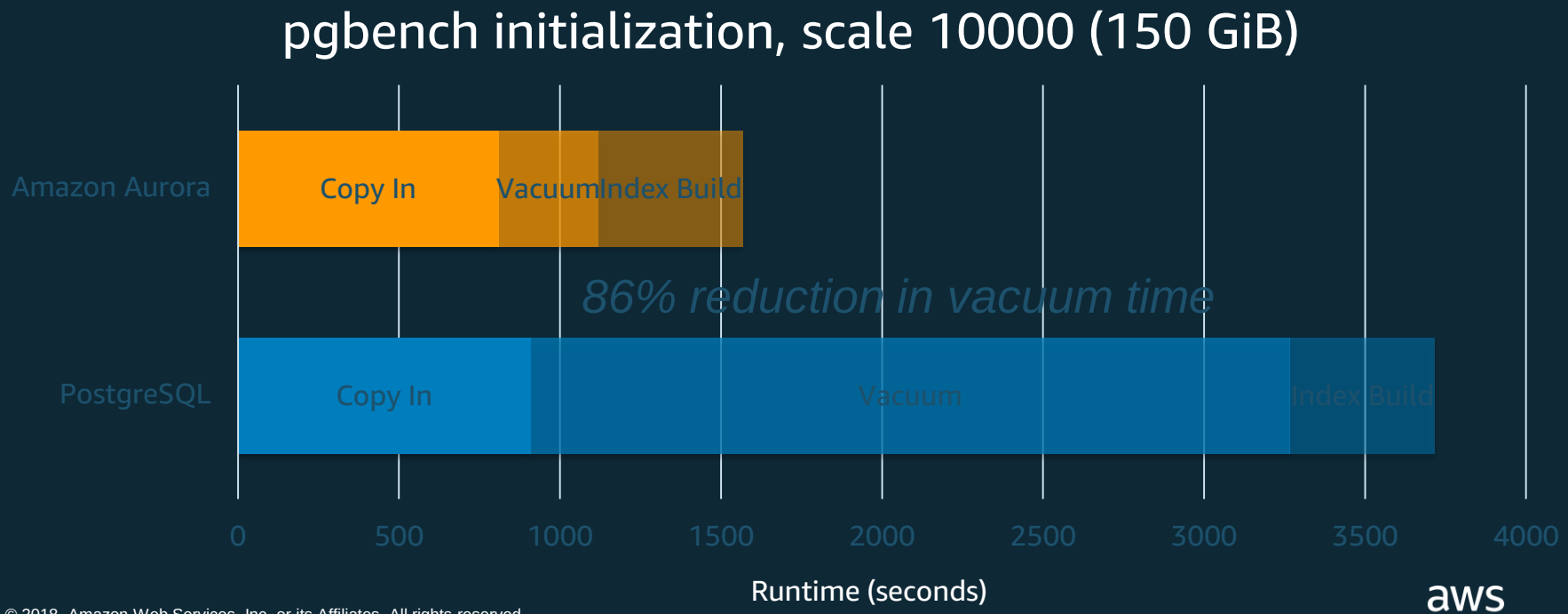
Amazon Aurora Improvements For GA

GA release improves performance on larger working sets



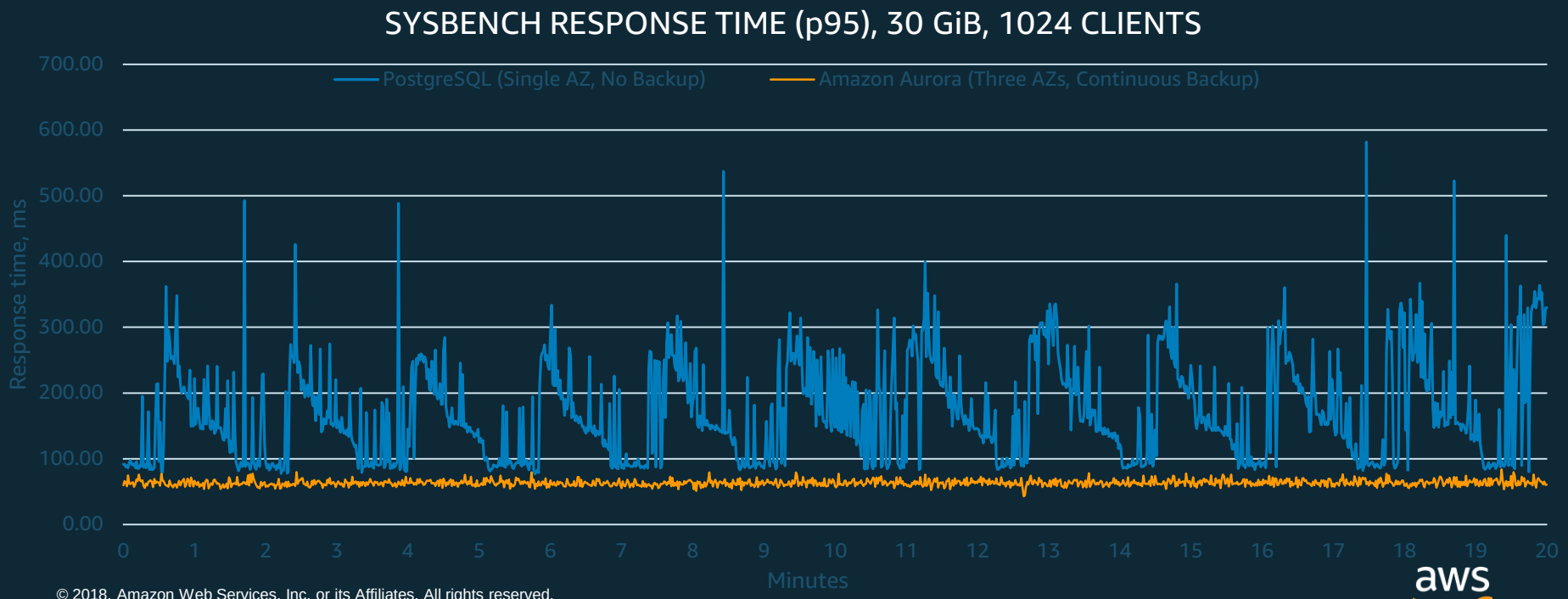
Amazon Aurora Loads Data Faster

Database initialization is >2x faster than PostgreSQL using the standard pgbench benchmark



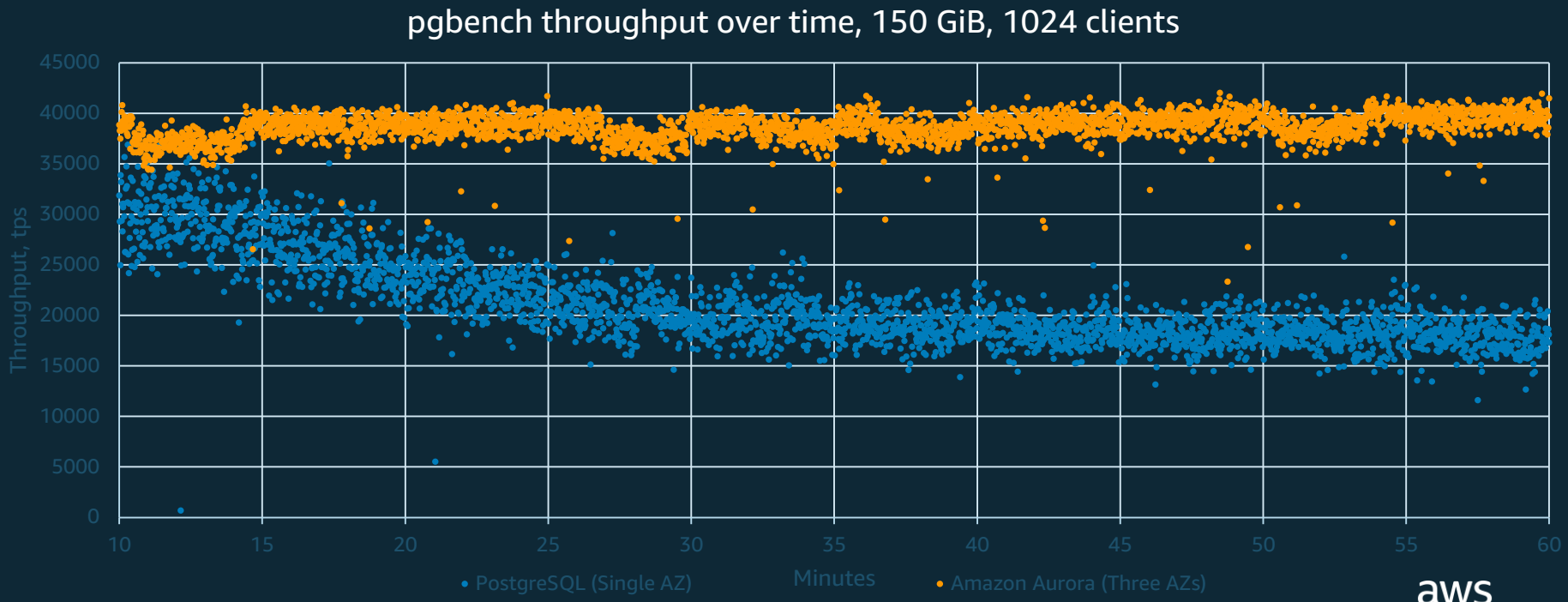
Amazon Aurora gives >2x Lower Response Times

Response time under heavy write load >2x lower than PostgreSQL, variance reduced by 99%



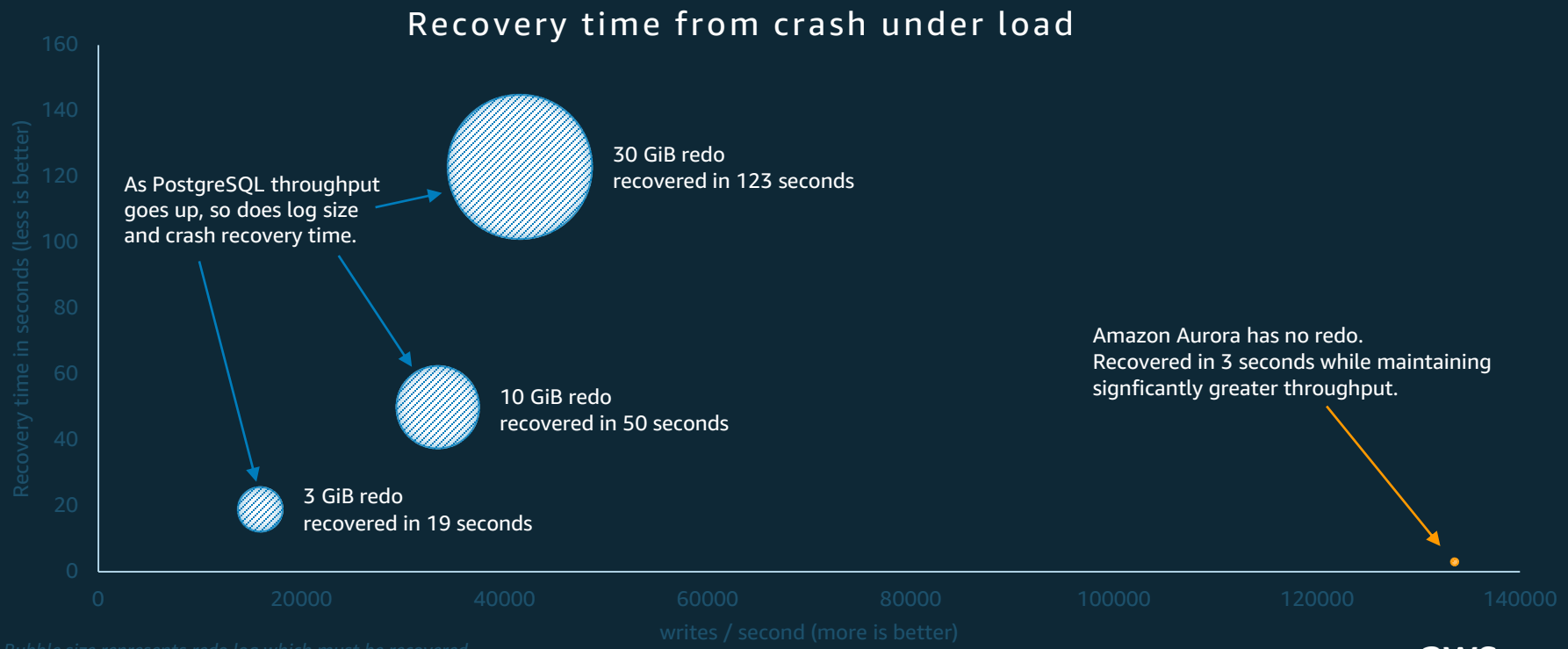
Amazon Aurora has more Consistent Throughput

While running pgbench at load, throughput is 3x more consistent than PostgreSQL



Amazon Aurora Reduced Recovery Time Up to 97%

Transaction-aware storage system recovers almost instantly



Bubble size represents redo log which must be recovered

Amazon Aurora with PostgreSQL Compatibility Performance By The Numbers

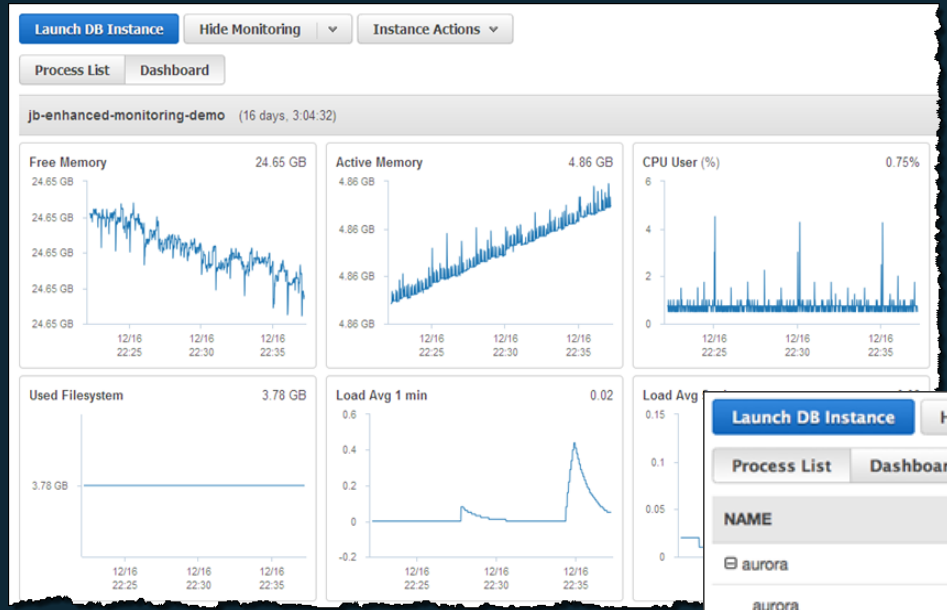
Measurement	Result
PgBench	Up to 2.9x faster
Sysbench	2x-5x faster
Vacuum	Time reduced up to 86%
Response time	>2x lower, 99% < variance
Throughput jitter	3x more consistent
Throughput at scale	4x faster
Recovery speed	Time reduced up to 97%

aws

Amazon Aurora with PostgreSQL Compatibility

Performance Monitoring & Management

First Step: Enhanced Monitoring

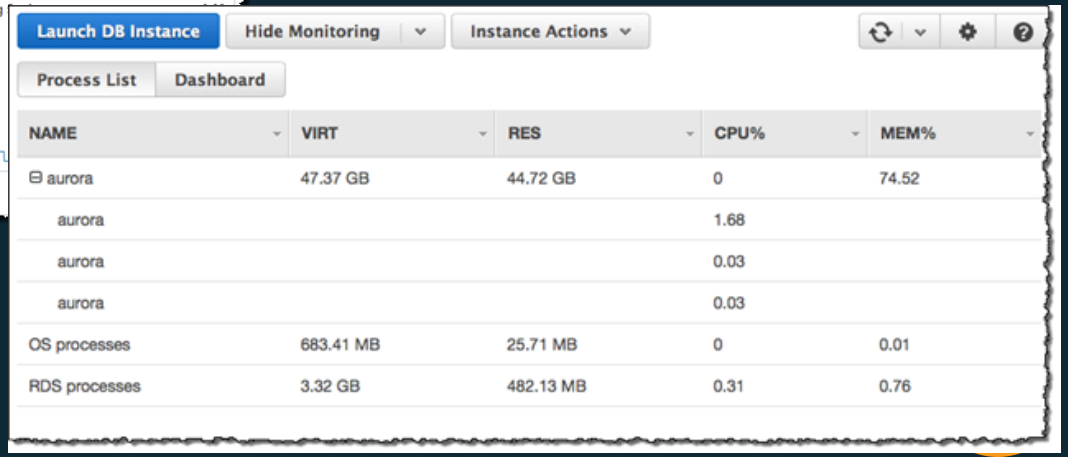


Released 2016

O/S Metrics

Process & thread List

Up to 1 second granularity



Next Step: Performance Insights

Why database tuning?

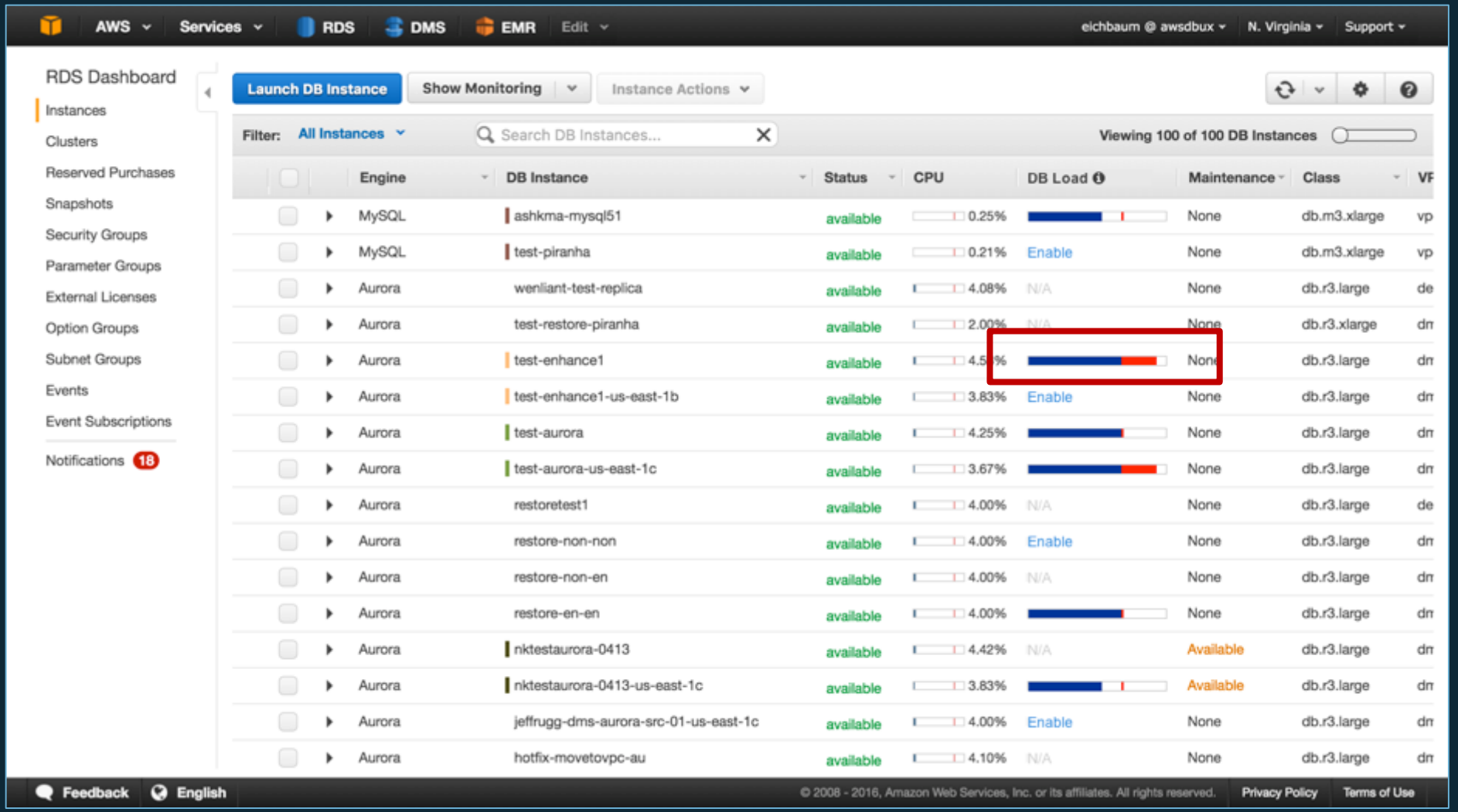
Amazon RDS is all about managed databases

Customers want performance managed too:

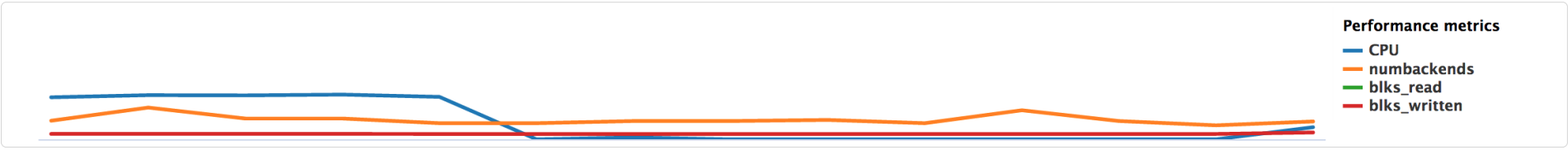
- ❑ **Want easy tool for optimizing cloud database workloads**
- ❑ **May not have deep tuning expertise**

→ Want a single pane of glass to achieve this

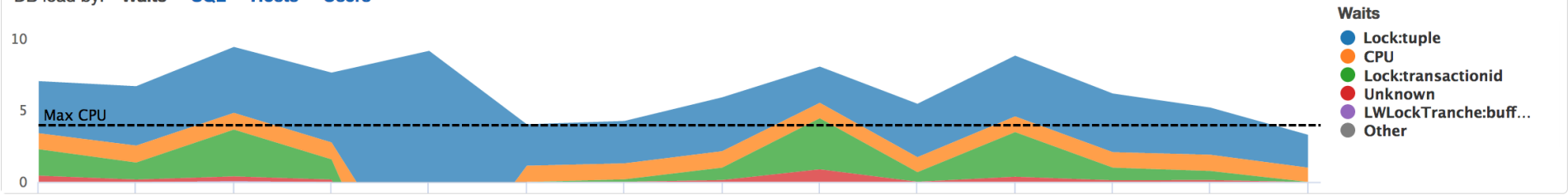
aws


AWS
Services
RDS
DMS
EMR
Edit
eichbaum @ awsdbux
N. Virginia
Support
RDS Dashboard
Instances
Clusters
Reserved Purchases
Snapshots
Security Groups
Parameter Groups
External Licenses
Option Groups
Subnet Groups
Events
Event Subscriptions
Notifications 18
Launch DB Instance
Show Monitoring
Instance Actions
Filter: All Instances
Search DB Instances...
Viewing 100 of 100 DB Instances
Engine
DB Instance
Status
CPU
DB Load
Maintenance
Class
MySQL ashkma-mysql51 available 0.25% None db.m3.xlarge
MySQL test-piranha available 0.21% Enable None db.m3.xlarge
Aurora wenliant-test-replica available 4.08% N/A None db.r3.large
Aurora test-restore-piranha available 2.00% N/A None db.r3.xlarge
Aurora test-enhance1 available 4.5% None db.r3.large
Aurora test-enhance1-us-east-1b available 3.83% Enable None db.r3.large
Aurora test-aurora available 4.25% None db.r3.large
Aurora test-aurora-us-east-1c available 3.67% None db.r3.large
Aurora restoretest1 available 4.00% N/A None db.r3.large
Aurora restore-non-non available 4.00% Enable None db.r3.large
Aurora restore-non-en available 4.00% N/A None db.r3.large
Aurora restore-en-en available 4.00% None db.r3.large
Aurora nktestaurora-0413 available 4.42% N/A Available db.r3.large
Aurora nktestaurora-0413-us-east-1c available 3.83% Available db.r3.large
Aurora jeffrugg-dms-aurora-src-01-us-east-1c available 4.00% Enable None db.r3.large
Aurora hotfix-movetovpc-au available 4.10% N/A None db.r3.large
Feedback
English
© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

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 = ?;

Amazon Aurora with PostgreSQL Compatibility

Getting Your Data in

AWS Database Migration Service

Start your first migration in 10 minutes or less

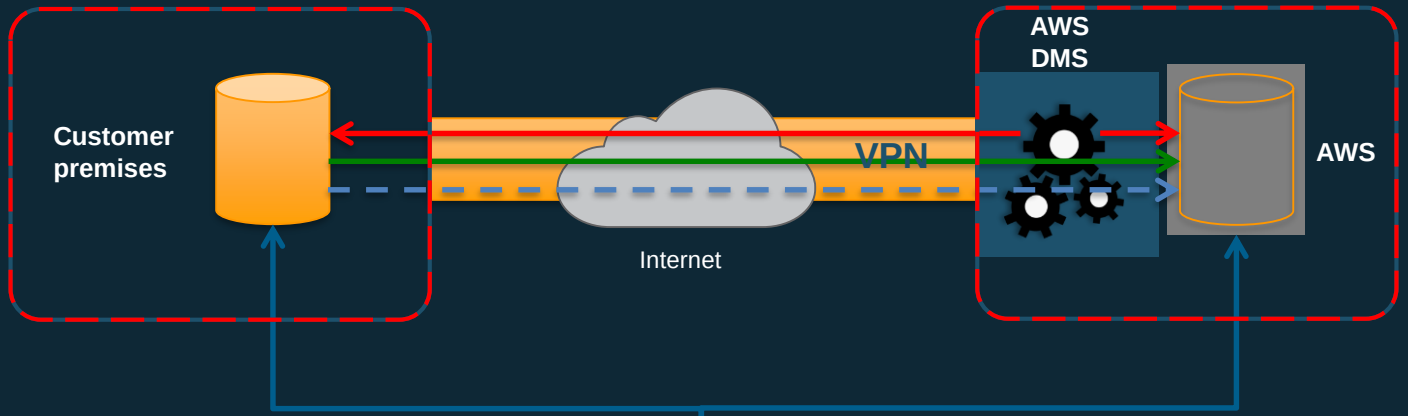
Keep your apps running during the migration

Replicate within, to, or from Amazon EC2 or Amazon RDS

Move data to the same or a different database engine

aws

Keep your apps running during the migration

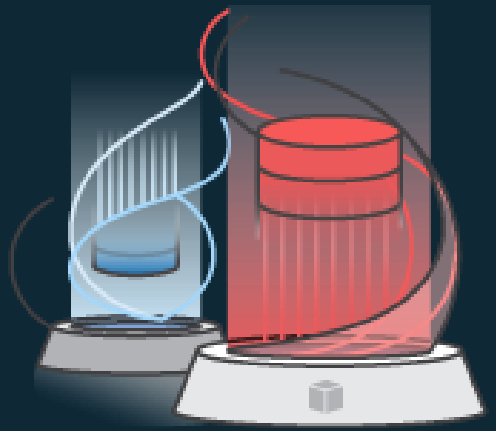


- Start a replication instance
- Connect to source and target databases
- Select tables, schemas, or databases

- Let AWS DMS create tables, load data, and keep them in sync
- Switch applications over to the target at your convenience

AWS Schema Conversion Tool

The AWS Schema Conversion Tool helps automate many database schema and code conversion tasks when migrating between database engines or data warehouse engines



Features

Oracle and Microsoft SQL Server schema conversion to MySQL, Amazon Aurora, MariaDB, and PostgreSQL

Or convert your schema between PostgreSQL and any MySQL engine

Database Migration Assessment report for choosing the best target engine

Code browser that highlights places where manual edits are required

Secure connections to your databases with SSL

Cloud native code optimization

AWS Schema Conversion Tool

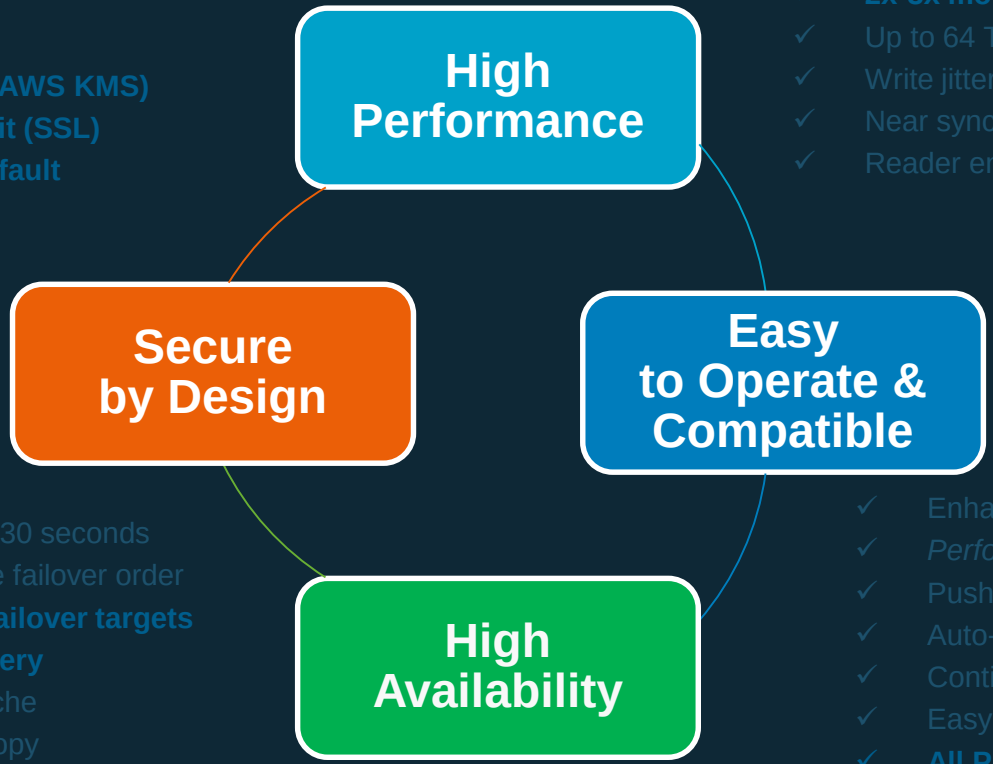
Converts relational databases

Converts warehouses

Amazon Aurora with PostgreSQL Compatibility

Recent Launch

Amazon Aurora with PostgreSQL compatibility – 10/24/2017 launch



- ✓ **2x-3x more throughput than PostgreSQL**
- ✓ Up to 64 TB of storage per instance
- ✓ Write jitter reduction
- ✓ Near synchronous replicas
- ✓ Reader endpoint

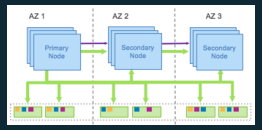
- ✓ **Encryption at rest (AWS KMS)**
- ✓ **Encryption in transit (SSL)**
- ✓ **Amazon VPC by default**
- ✓ Row Level Security

- ✓ Failover in less than 30 seconds
- ✓ Customer specifiable failover order
- ✓ **Up to 15 readable failover targets**
- ✓ **Instant crash recovery**
- ✓ Survivable buffer cache
- ✓ X-region snapshot copy

- ✓ Enhanced OS monitoring
- ✓ *Performance Insights*
- ✓ Push button migration
- ✓ Auto-scaling storage
- ✓ Continuous backup and PITR
- ✓ Easy provisioning / patching
- ✓ **All PostgreSQL features**
- ✓ **All RDS for PostgreSQL extensions**
- ✓ **AWS DMS supported inbound**

The Amazon Aurora Database Family

Encryption at rest and in transit



6 Copies

Amazon S3

Automatic Failover

Read Replicas

AWS DMS

Secure

Durable

High Performance & Scale

Amazon Aurora

Available

Compatible

Convenient

Amazon RDS

AWS IAM, KMS & VPC

Enterprise Performance

64TB Storage

MySQL

PostgreSQL

aws

Questions

Kevin Jernigan, Senior Product Manager: kmj@amazon.com

Amazon Aurora documentation
http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Aurora.html

Amazon Aurora PostgreSQL forum
https://forums.aws.amazon.com/forum.jspa?forumID=227

Thank you!