Amazon Aurora Deep Dive

Kevin Jernigan, Sr. Product Manager
Amazon Aurora PostgreSQL
Amazon RDS for PostgreSQL

May 18, 2017

amazon web services | Webinars

Agenda

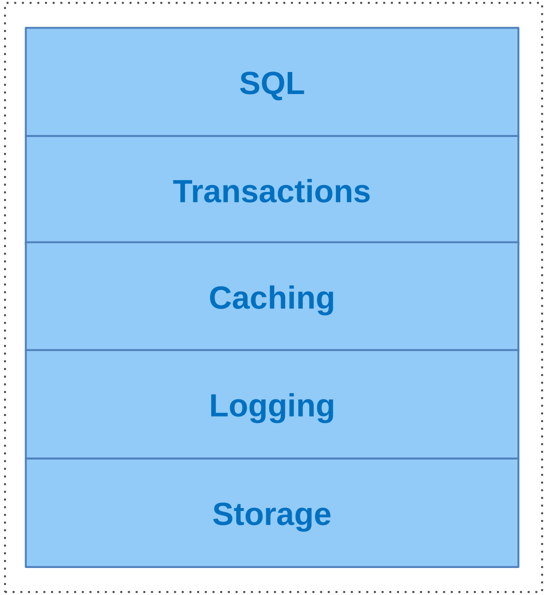
- Why did we build Amazon Aurora?
    - Why add PostgreSQL compatibility?
- Durability and Availability Architecture
- Performance Results
- Performance Architecture
- Announcing *Performance Insights*
- Getting Data In
- Feature Roadmap
- Preview Information & Questions

Amazon Aurora

+

PostgreSQL

Traditional relational databases are hard to scale

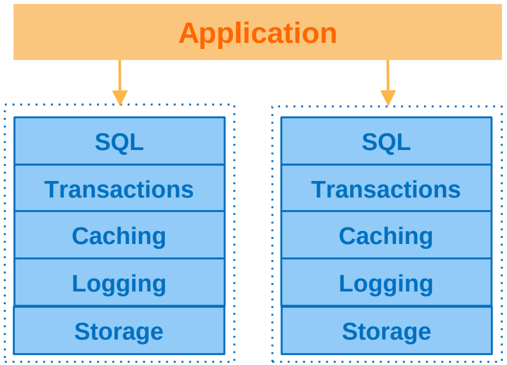


Multiple layers of functionality all in a monolithic stack

Traditional approaches to scale databases

Sharding

Coupled at the application layer



Application

SQL | SQL

Transactions | Transactions

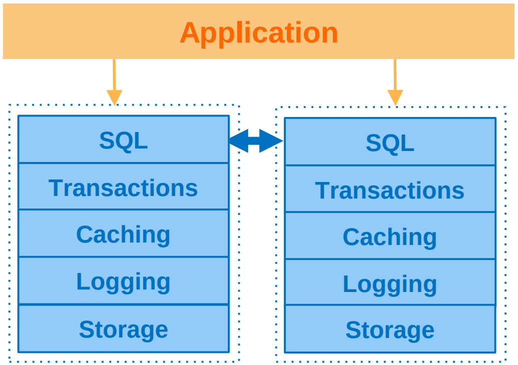
Caching | Caching

Logging | Logging

Storage | Storage

Shared Nothing

Coupled at the SQL layer



Application

SQL | SQL

Transactions | Transactions

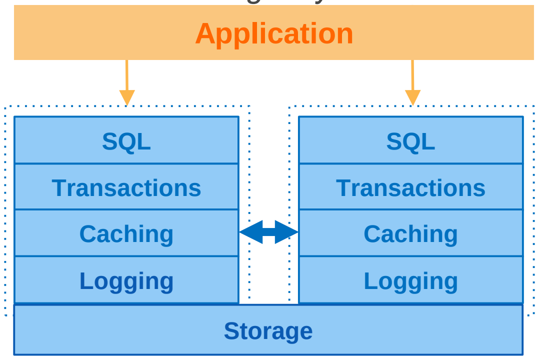
Caching | Caching

Logging | Logging

Storage | Storage

Shared Disk

Coupled at the caching and storage layer



Application

SQL | SQL

Transactions | Transactions

Caching | Caching

Logging | Logging

Storage

Each architecture is limited by the monolithic mindset

Reimagining the relational database

What if you were inventing the database today?

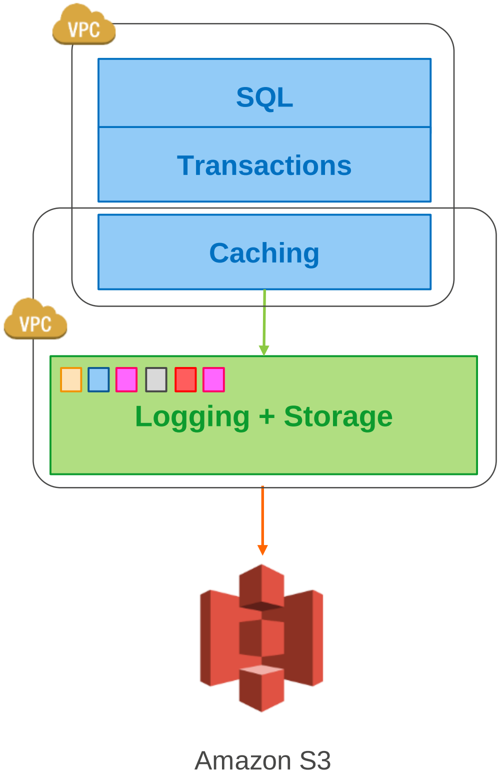
- You would break apart the stack
- You would build something that:
  - ✓ Can scale out…
  - ✓ Is self-healing…
  - ✓ Leverages distributed services…

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

What is Amazon Aurora?

Cloud-optimized relational database

Performance and availability of
commercial databases

Simplicity and cost-effectiveness of
open source databases,
with MySQL compatibility

So what’s next?

Making Amazon Aurora Better

In 2014, we launched Amazon Aurora with MySQL compatibility.

Now, we are adding 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.

Start With the Customer – Why Add PostgreSQL?

Amazon Aurora + MySQL

Start With the Customer – Why Add PostgreSQL?

Amazon Aurora + PostgreSQL

PostgreSQL Fast Facts

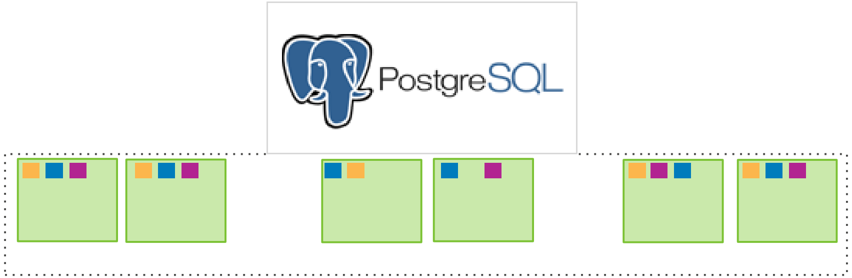
- Open source database
- In active development for 20 years
- Owned by a foundation, not a single company
- Permissive innovation-friendly open source license
- High performance out of the box
- Object-oriented and ANSI-SQL:2008 compatible
- Most geospatial features of any open-source database
- Supports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

Open Source Initiative

What does PostgreSQL compatibility mean?

PostgreSQL 9.6 + Amazon Aurora cloud-optimized storage

- Performance: Up to 2x+ better performance 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



Amazon Aurora Storage

What does PostgreSQL compatibility mean?

- Cloud-native security and encryption
  - AWS Key Management Service (KMS) and AWS Identity and Access Management (IAM)
- Easy to manage with Amazon RDS
- Easy to load and unload
  - AWS Database Migration Service and AWS Schema Conversion Tool
- Fully compatible with PostgreSQL, now and for the foreseeable future
  - Not a compatibility layer – native PostgreSQL implementation

PostgreSQL

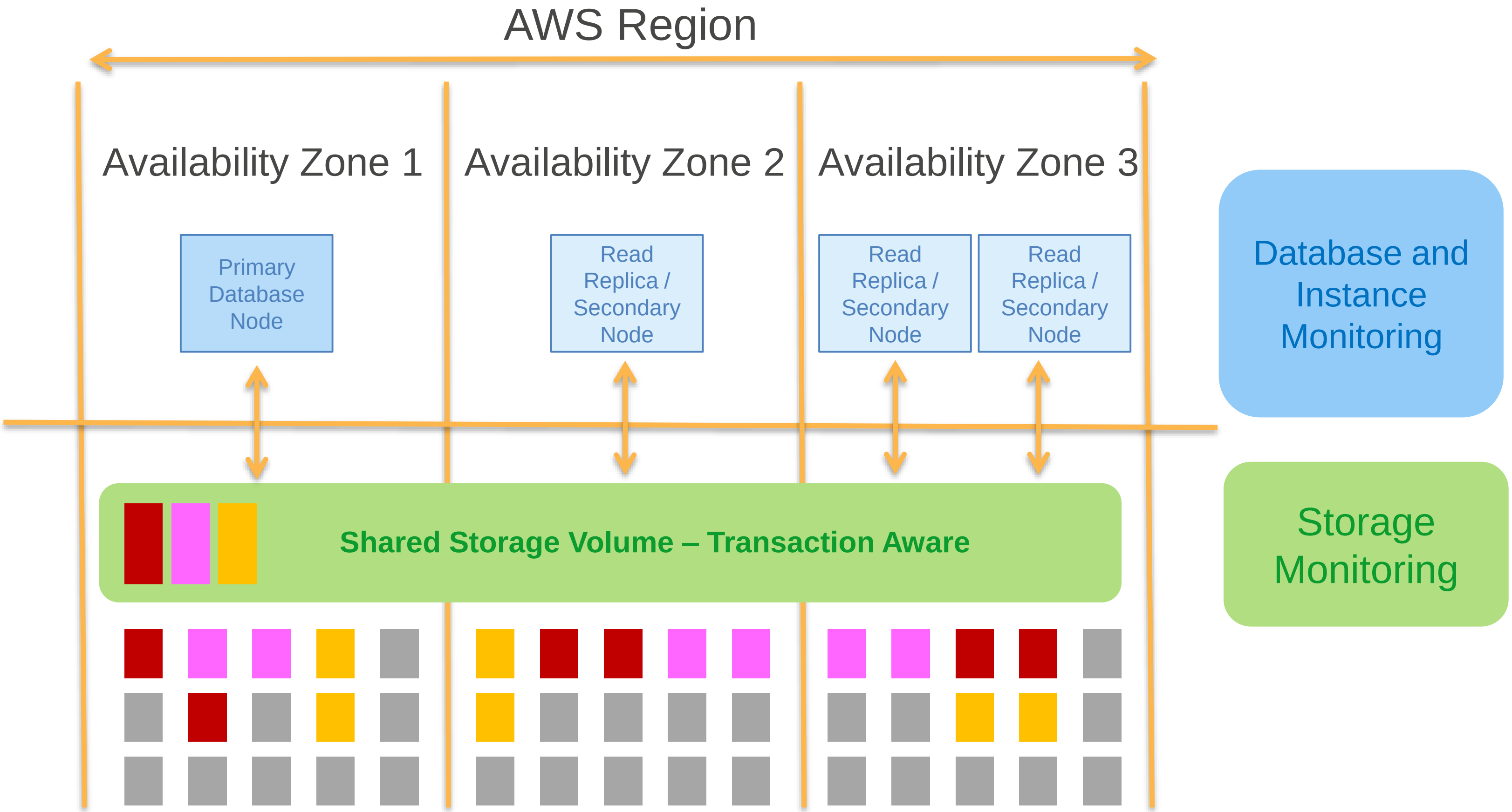
Amazon RDS

AWS DMS

Amazon Aurora

Durability & Availability

Scale-out, distributed, log structured storage



AWS Region

Availability Zone 1

Availability Zone 2

Availability Zone 3

Primary Database Node

Read Replica / Secondary Node

Read Replica / Secondary Node

Read Replica / Secondary Node

Shared Storage Volume – Transaction Aware

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)

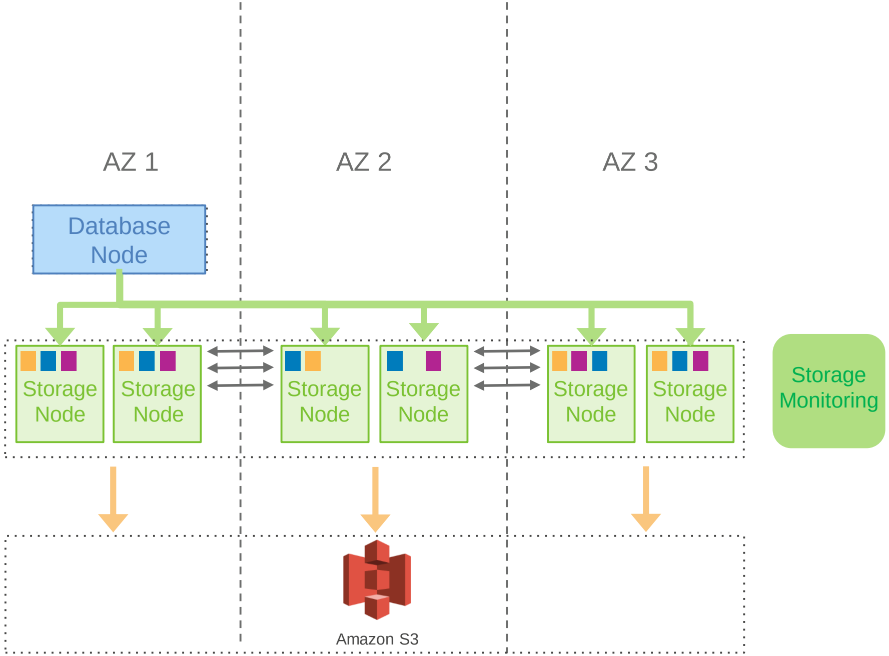
Continuous monitoring of nodes and disks for repair

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



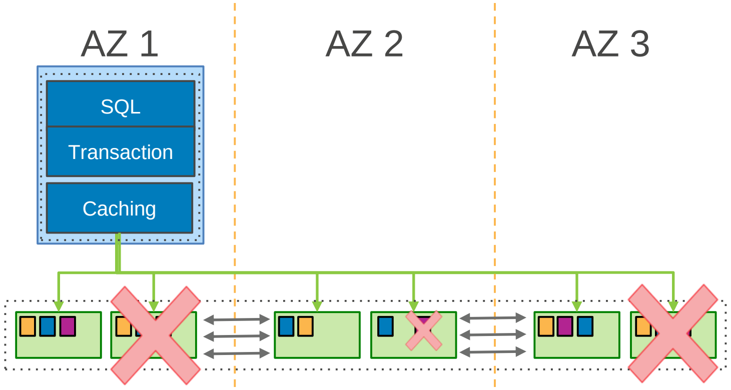
Amazon Aurora Storage Engine Fault-tolerance

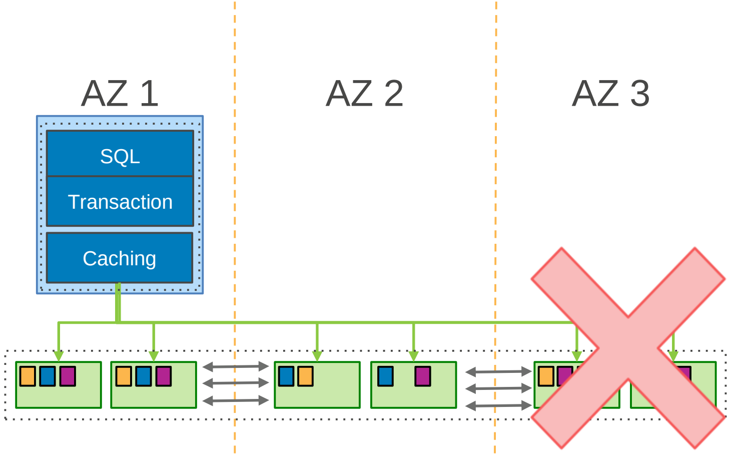
What can fail?

- 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 Replicas

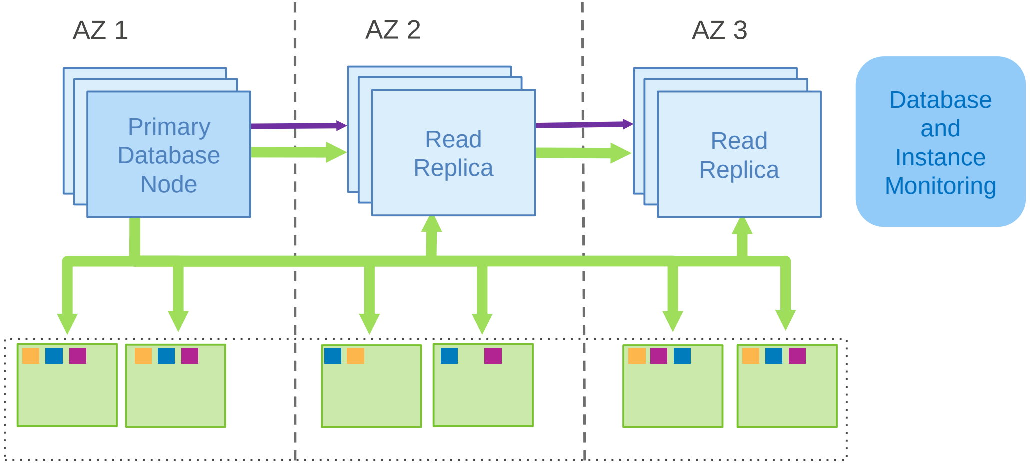
Availability

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

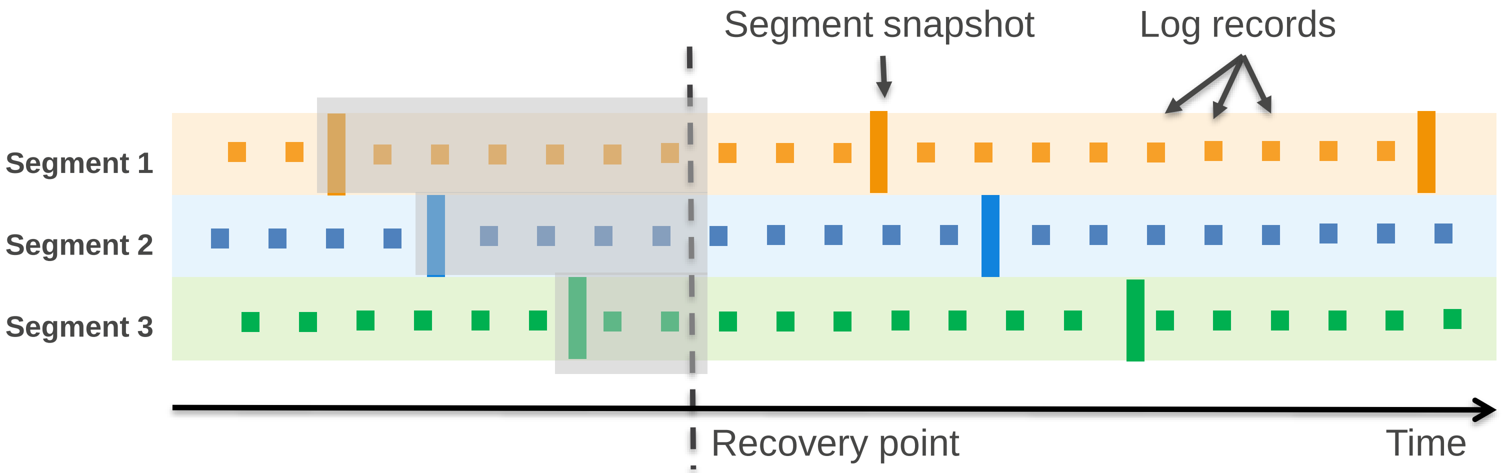


Performance

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

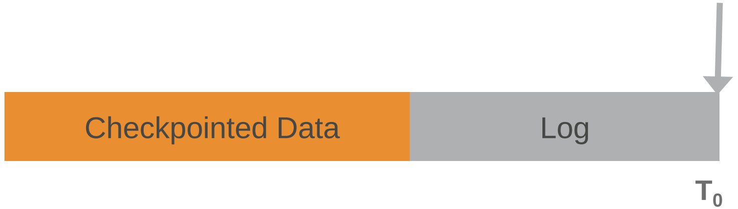
Traditional databases

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



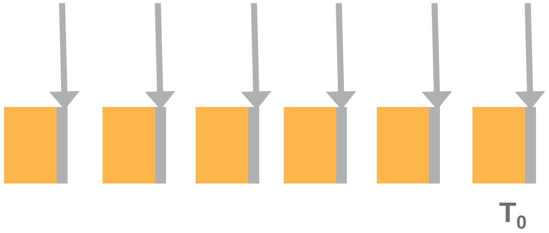
Amazon Aurora

No replay at startup because storage system is transaction-aware

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

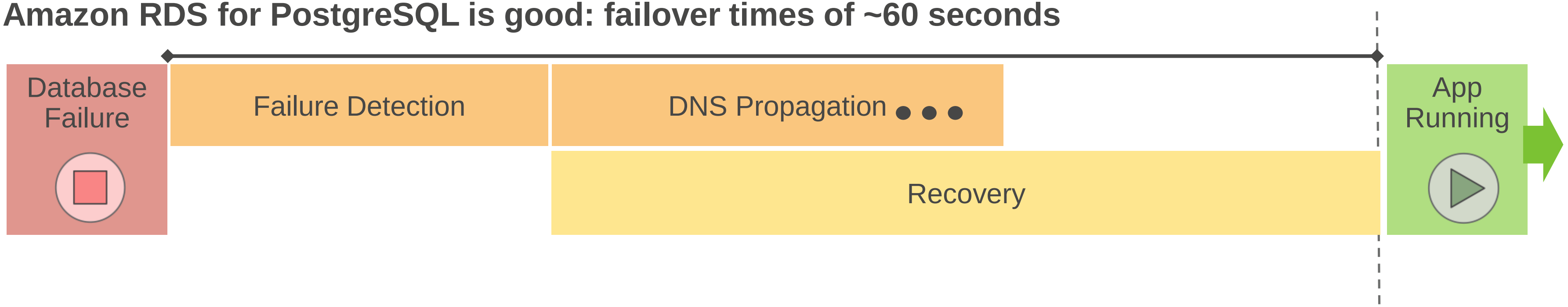
Coalescing is parallel, distributed, and asynchronous

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



Database Failure

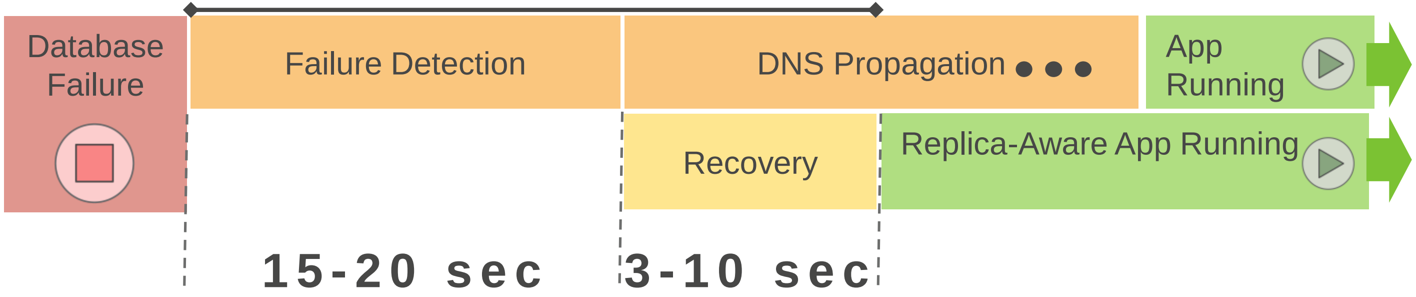
Failure Detection

DNS Propagation • • •

Recovery

App Running

Amazon Aurora is better: failover times < 30 seconds



Database Failure

Failure Detection

DNS Propagation • • •

App Running

Recovery

Replica-Aware App Running

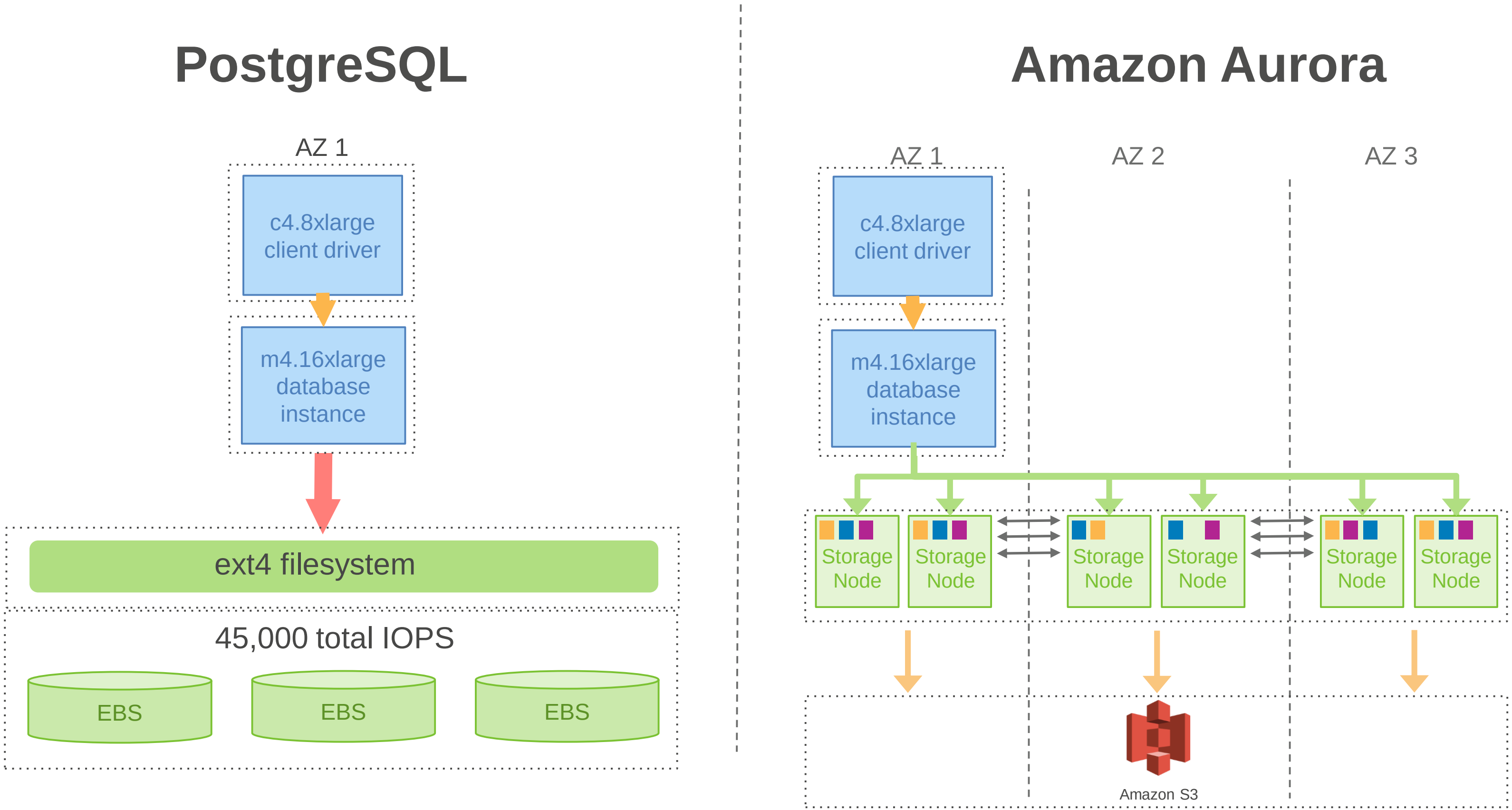
15-20 sec

3-10 sec

Amazon Aurora

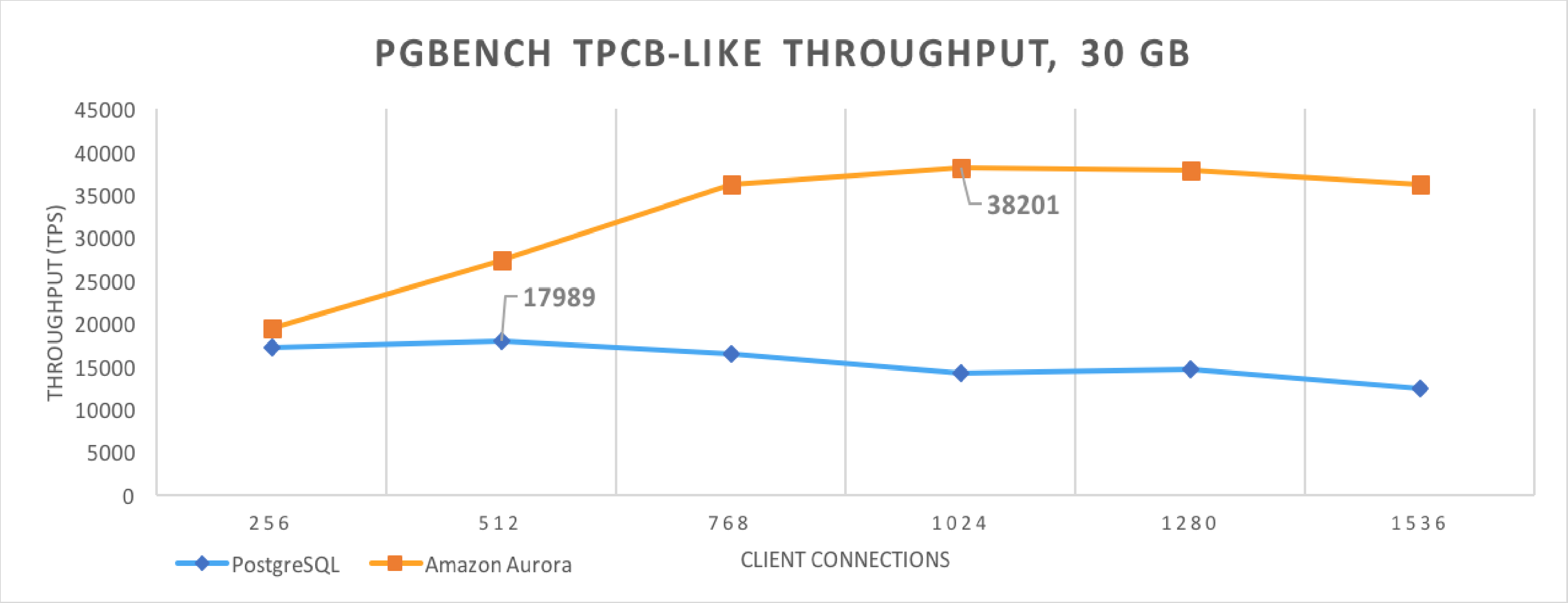
Performance vs. PostgreSQL

Benchmark System Configurations



m4.16xlarge (64 VCPU, 256GiB), c4.8xlarge (36 VCPU, 60GiB)

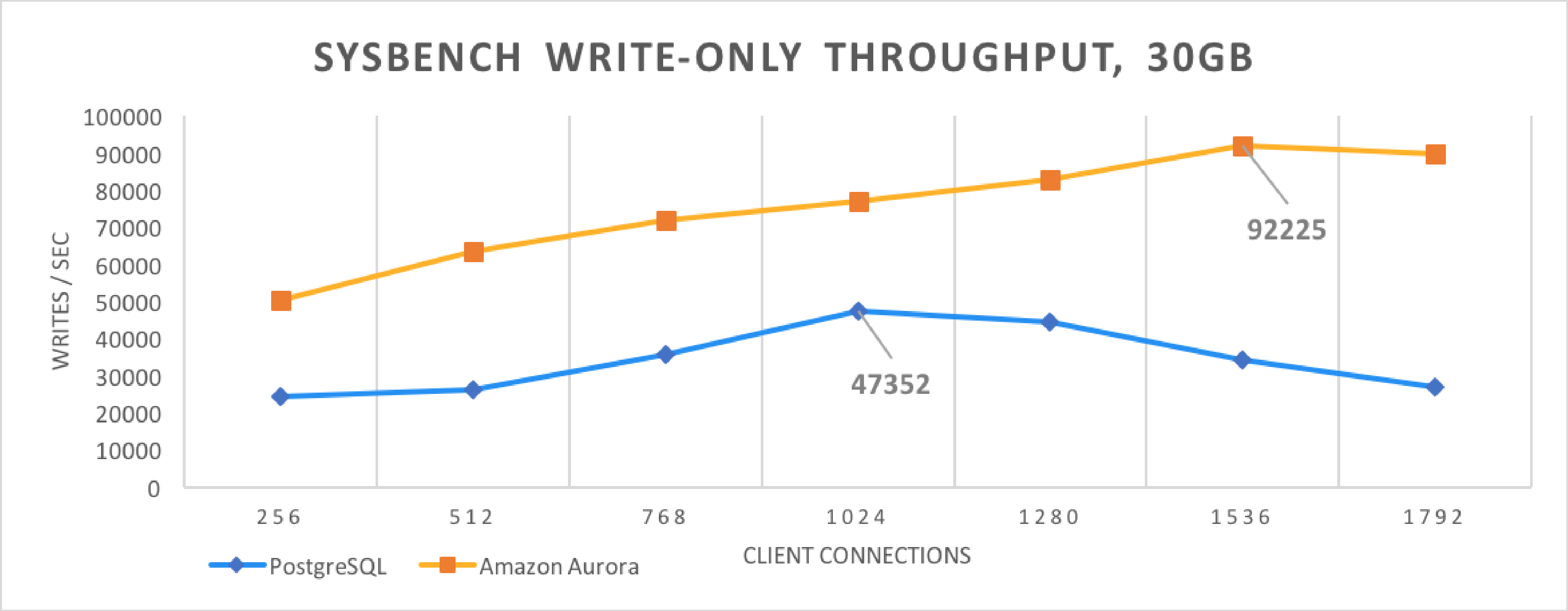
Amazon Aurora is >=2x Faster on PgBench



pgbench “tpcb-like” workload, scale 2000 (30GiB). All configurations run for 60 minutes

Amazon Aurora is 2x-3x Faster on SysBench

Amazon Aurora delivers 2x the absolute peak of PostgreSQL and 3x PostgreSQL performance at high client counts



SysBench oltp(write-only) workload with 30 GB database with 250 tables and 400,000 initial rows per table

Amazon Aurora: Over 120,000 Writes/Sec

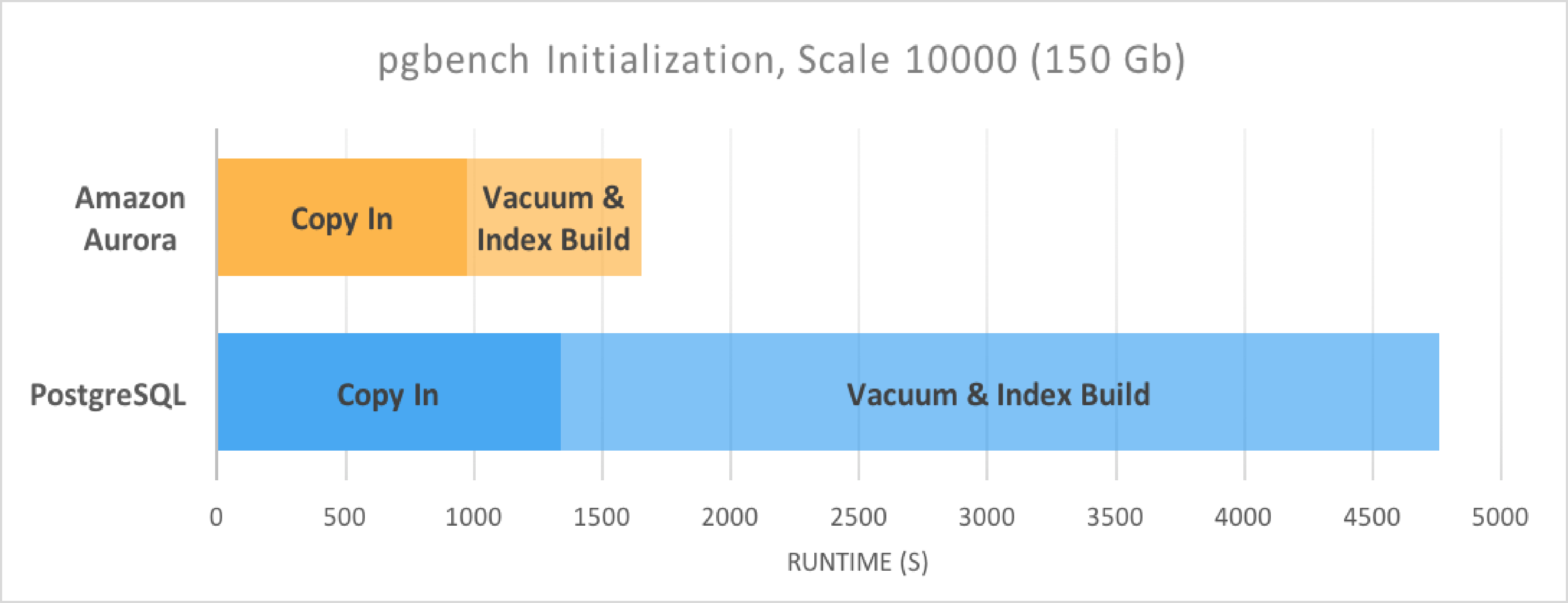
Sustained sysbench throughput over 120K writes/sec

```
OLTP test statistics:
    queries performed:
        read:                            0
        write:                           432772903
        other:(begin + commit)           216366749
        total:                           649139652
    transactions:                       108163671 (30044.73 per sec.)
    read/write requests:                432772903 (120211.75 per sec.)
    other operations:                   216366749 (60100.40 per sec.)
    ignored errors:                     39407  (10.95 per sec.)
    reconnects:                         0      (0.00 per sec.)
```

sysbench write-only 10GB workload with 250 tables and 25,000 initial rows per table. 10-minute warmup, 3,076 clients
Ignored errors are key constraint errors, designed into sysbench

Amazon Aurora Loads Data 3x Faster

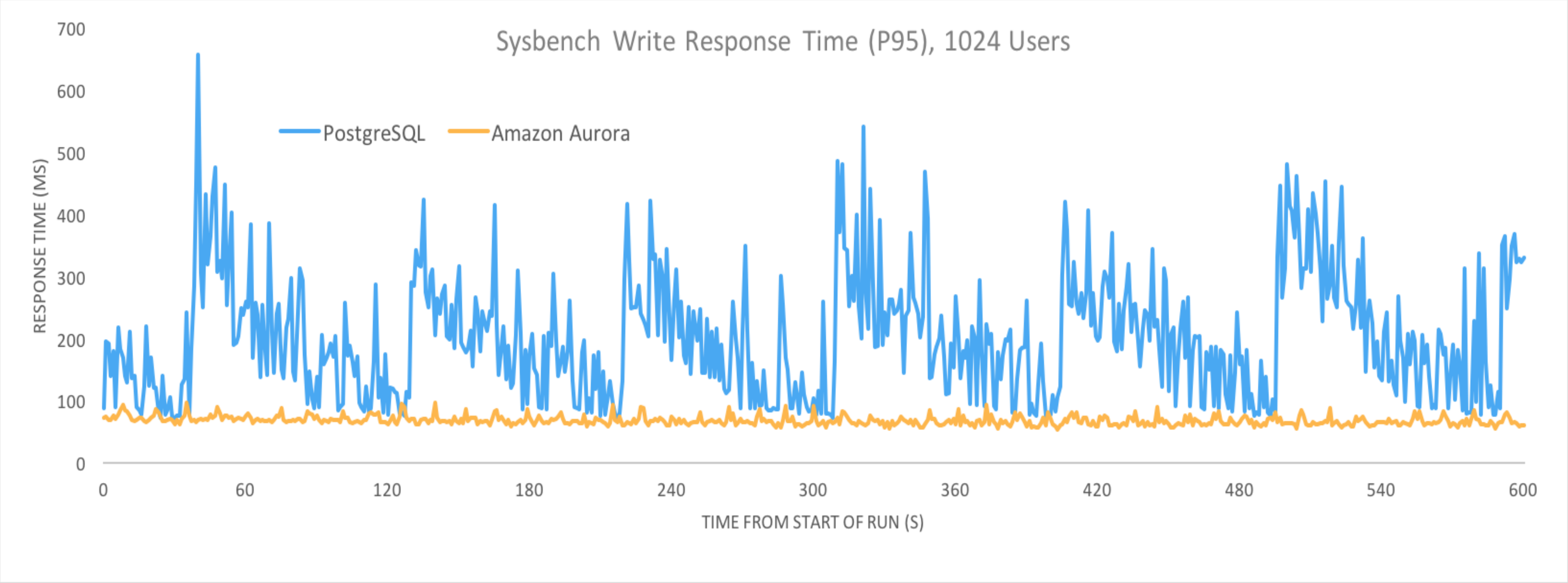
Database initialization is three times faster than PostgreSQL using the standard PgBench benchmark



Command: `pgbench -i -s 2000 –F 90`

Amazon Aurora Gives >2x Faster Response Times

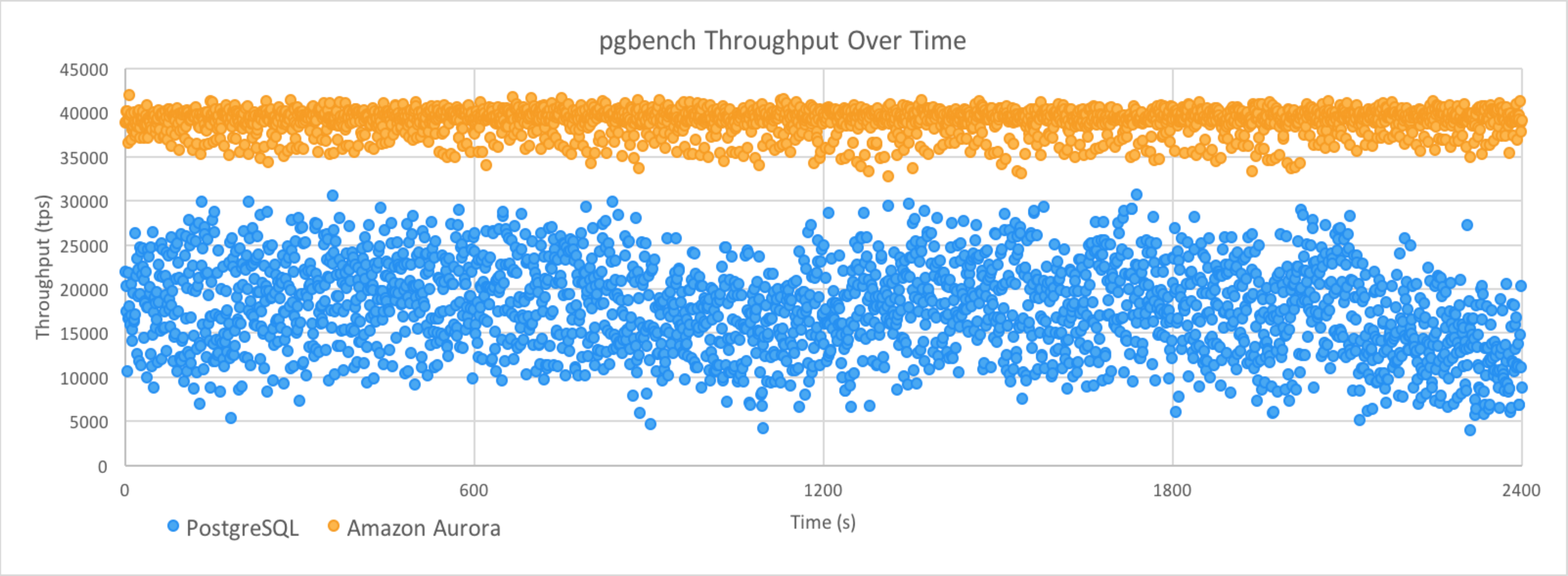
Response time under heavy write load >2x faster than PostgreSQL (and >10x more consistent)



SysBench oltp(write-only) 23GiB workload with 250 tables and 300,000 initial rows per table. 10-minute warmup.

Amazon Aurora Has More Consistent Throughput

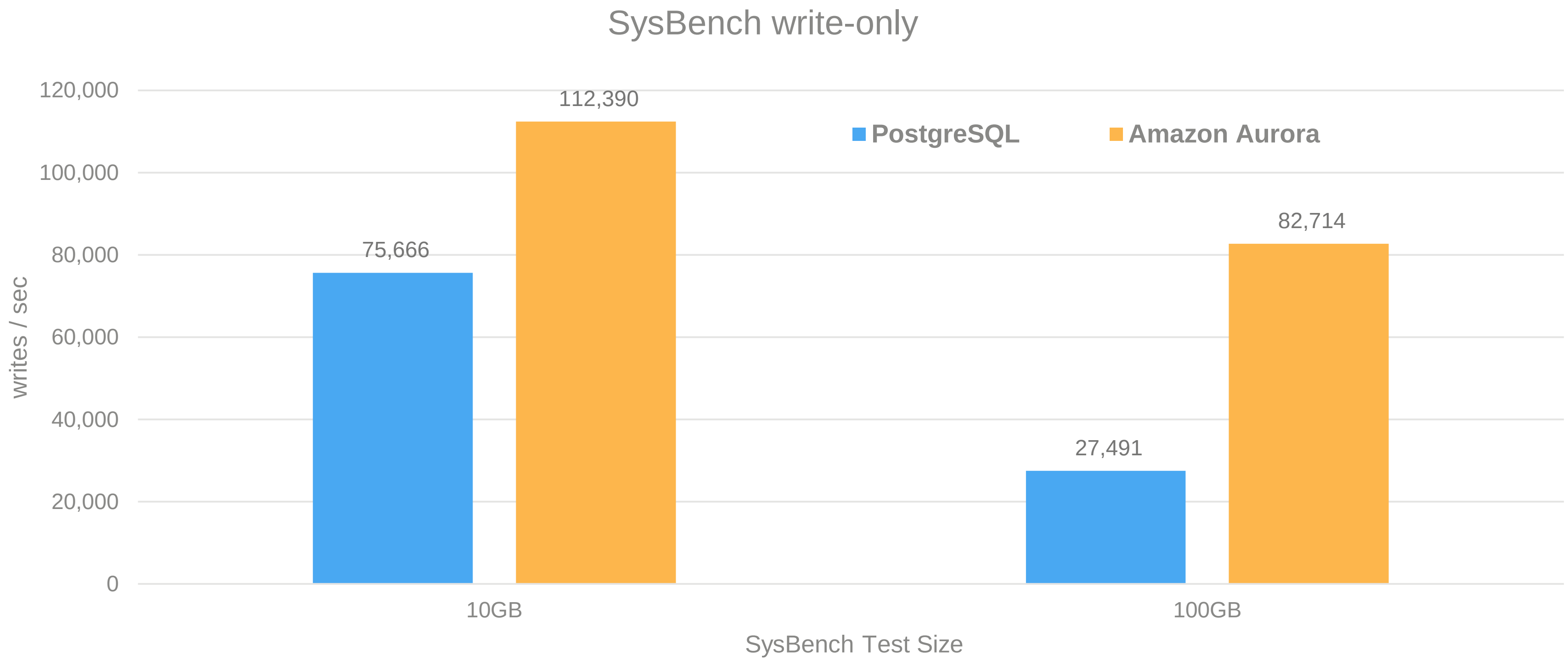
While running at load, performance is more than three times more consistent than PostgreSQL



PgBench "tpcb-like" workload at scale 2000. Amazon Aurora was run with 1280 clients. PostgreSQL was run with 512 clients (the concurrency at which it delivered the best overall throughput)

Amazon Aurora is 3x Faster at Large Scale

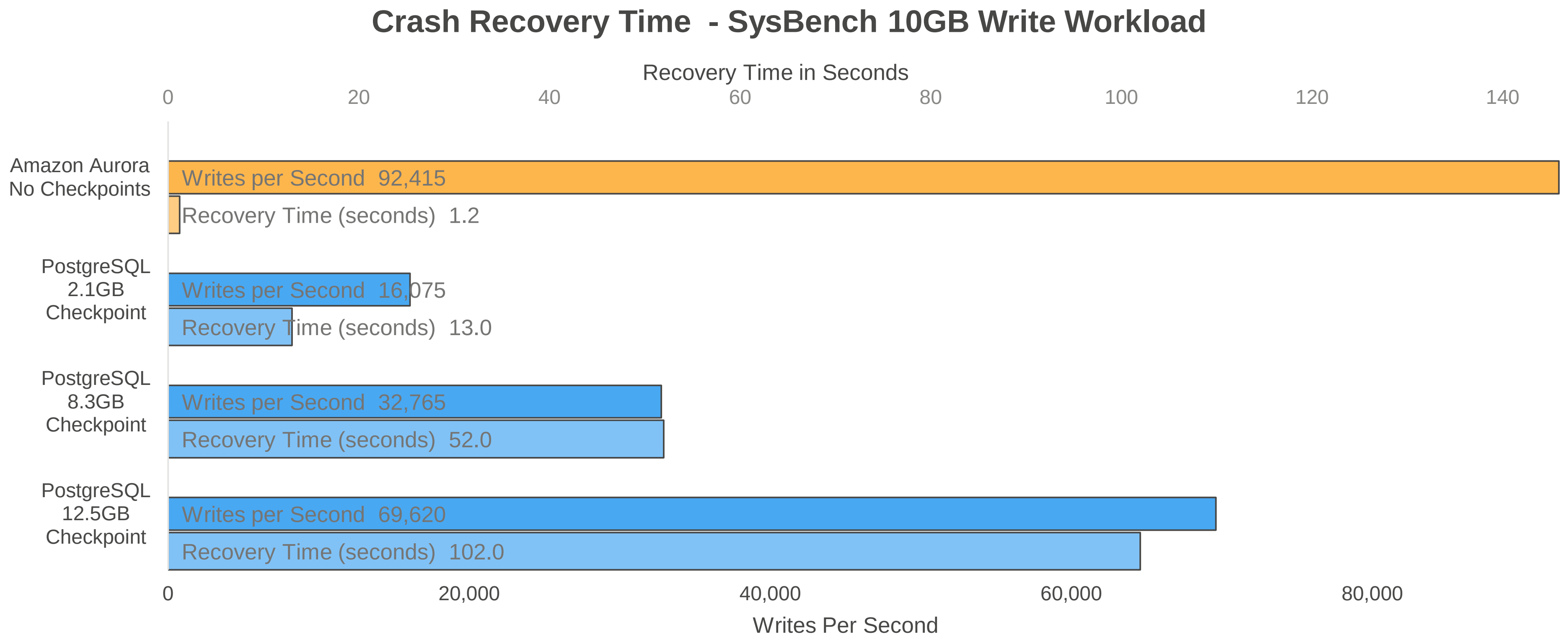
Scales from 1.5x to 3x faster as database grows from 10 GiB to 100 GiB



SysBench oltp(write-only) – 10GiB with 250 tables & 150,000 rows and 100GiB with 250 tables & 1,500,000 rows

Amazon Aurora Delivers up to 85x Faster Recovery

Transaction-aware storage system **recovers almost instantly**



SysBench oltp(write-only) 10GiB workload with 250 tables & 150,000 rows

Amazon Aurora with PostgreSQL Compatibility

Performance By The Numbers

Measurement	Result
PgBench	>= 2x faster
SysBench	2x-3x faster
Data Loading	3x faster
Response Time	>2x faster
Throughput Jitter	>3x more consistent
Throughput at Scale	3x faster
Recovery Speed	Up to 85x faster

Amazon Aurora

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

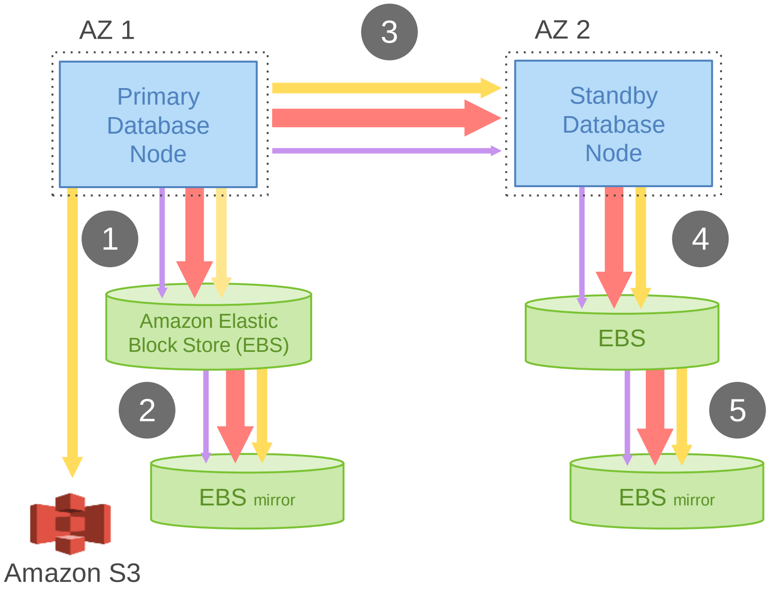
DATABASES ARE ALL ABOUT **I/O**

NETWORK-ATTACHED STORAGE IS ALL ABOUT **PACKETS/SECOND**

HIGH-THROUGHPUT PROCESSING NEEDS **CPU AND MEMORY OPTIMIZATIONS**

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

TYPE OF WRITE

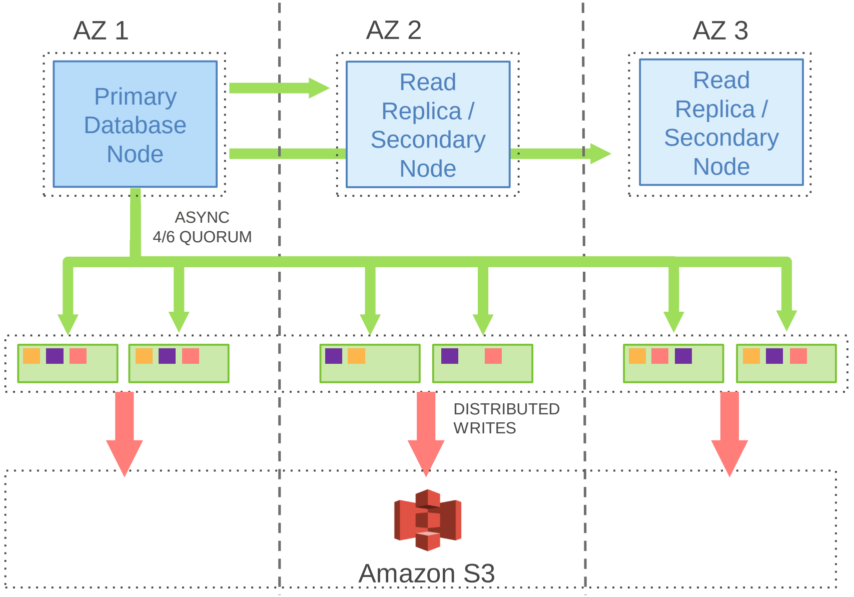
→ WAL

→ DATA

→ COMMIT LOG & FILES

Write IO Traffic in an Amazon Aurora Database Node

AMAZON AURORA



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

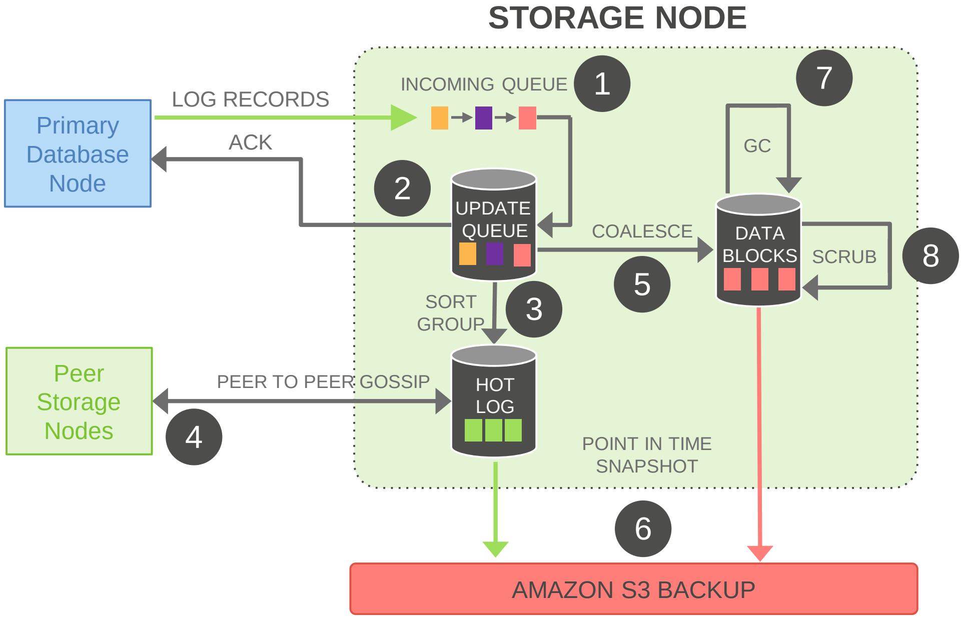
→ AMAZON AURORA + WAL LOG

🟧 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

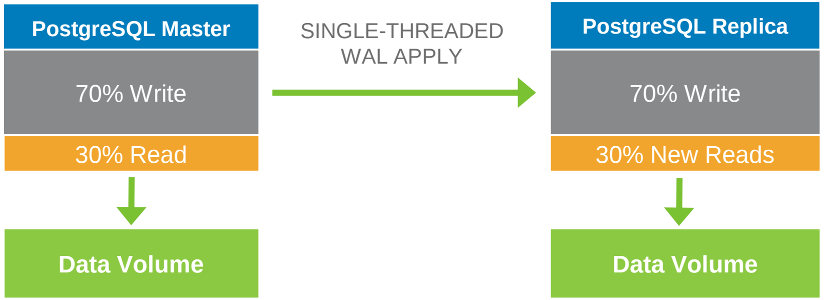
Input queue is **far smaller** than PostgreSQL

Favors latency-sensitive operations

Uses disk space to buffer against spikes in activity

IO traffic in Aurora Replicas

POSTGRESQL READ SCALING

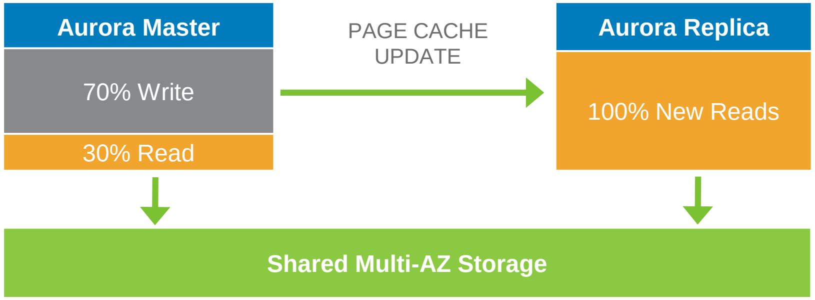


Physical: Ship redo (WAL) to Replica

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

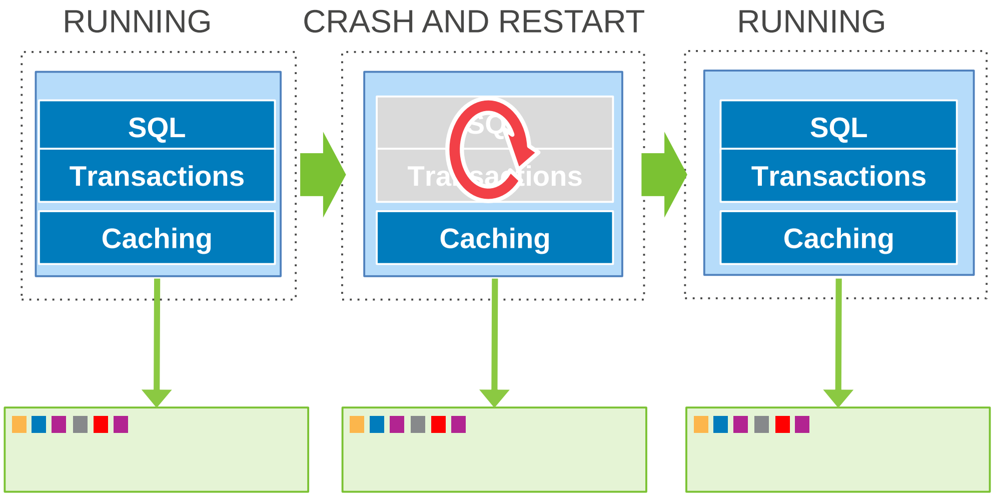
Cache normally lives inside the operating system database process– and goes away when/if that database dies

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 monitoring and management

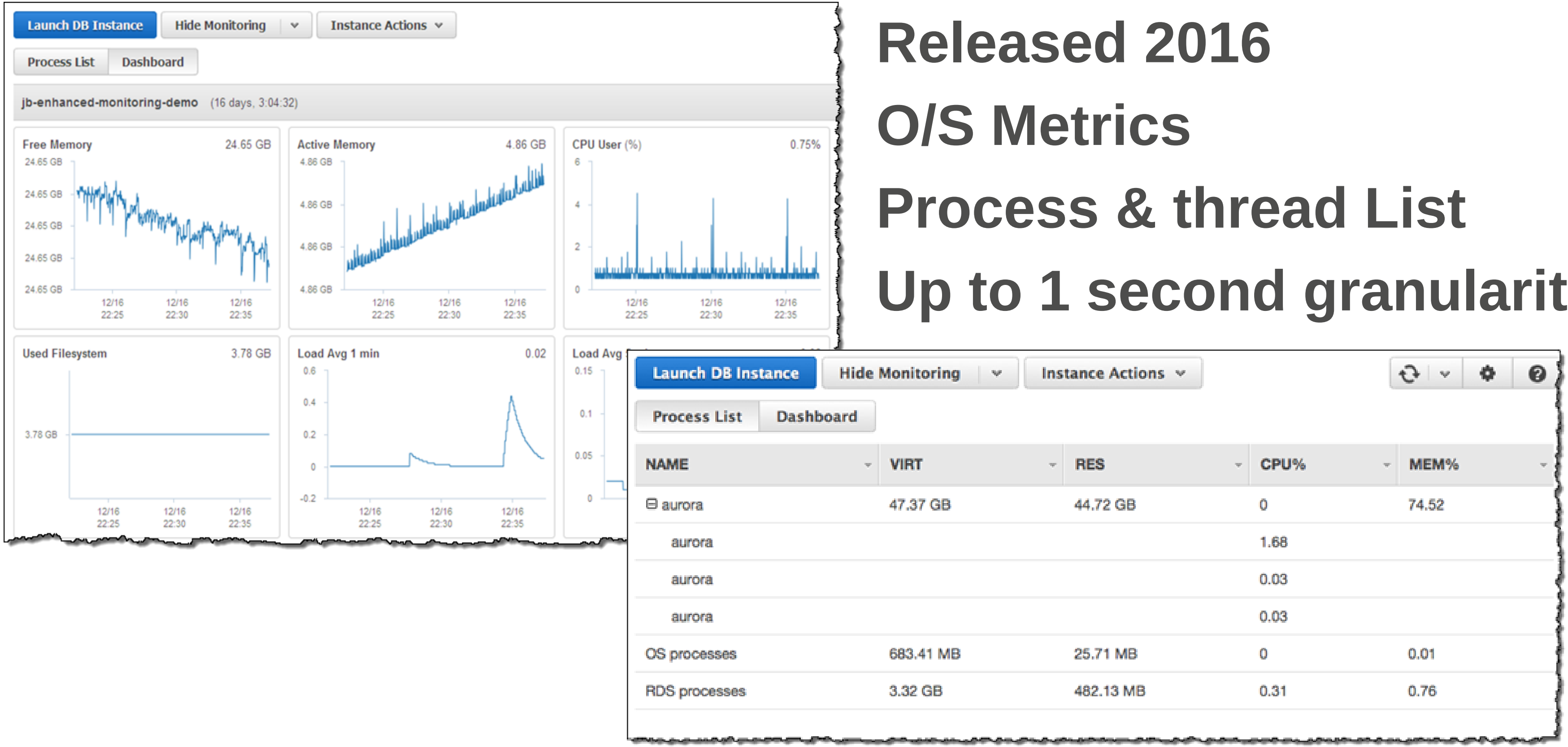
First Step: Enhanced Monitoring

Released 2016

O/S Metrics

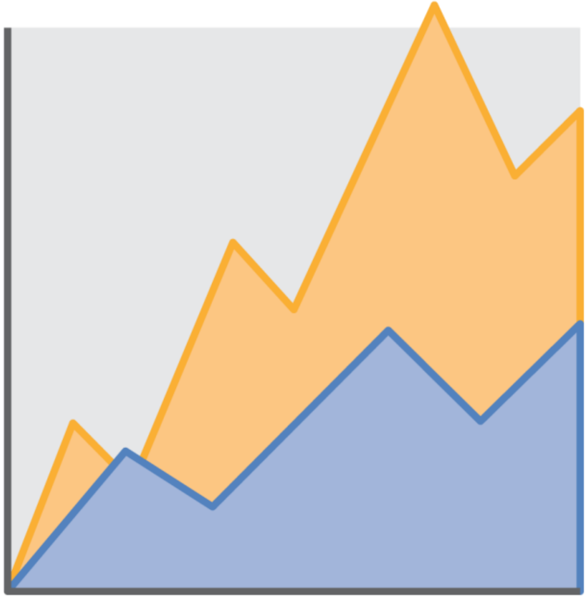
Process & thread List

Up to 1 second granularity



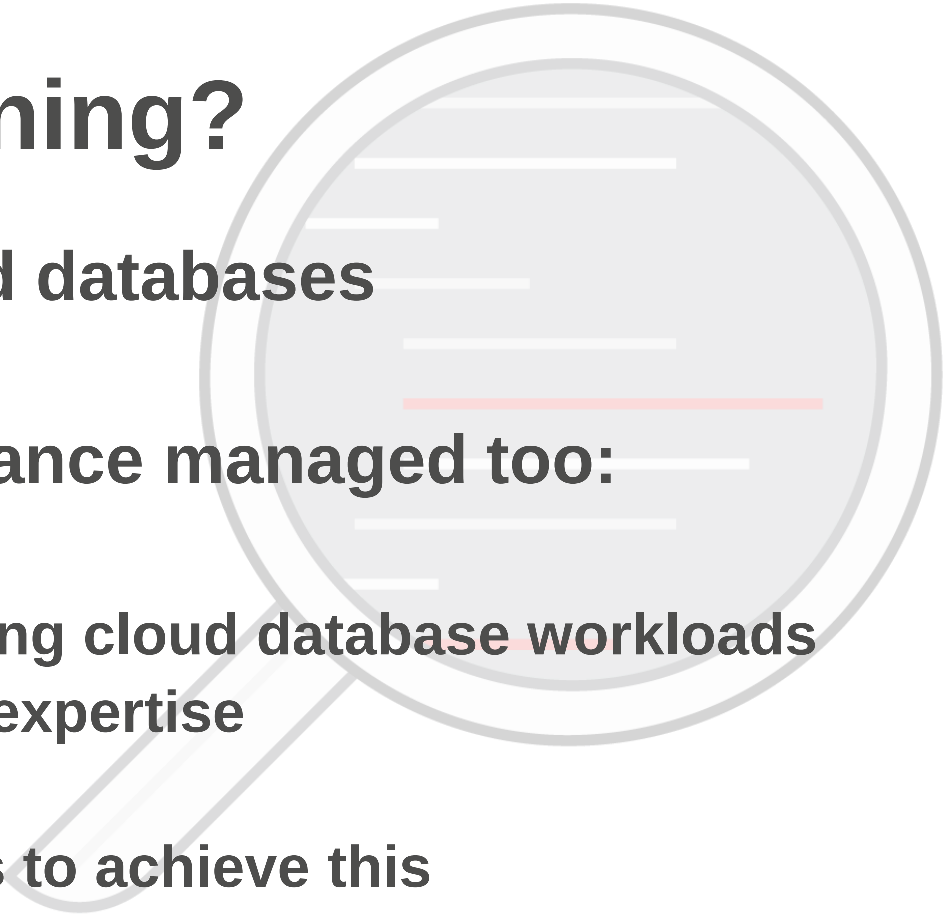
NAME	VIRT	RES	CPU%	MEM%
⊟ aurora	47.37 GB	44.72 GB	0	74.52
aurora			1.68	
aurora			0.03	
aurora			0.03	
OS processes	683.41 MB	25.71 MB	0	0.01
RDS processes	3.32 GB	482.13 MB	0.31	0.76

Next Step: Performance Insights



Database Engine
Performance Tuning

Why Database Tuning?



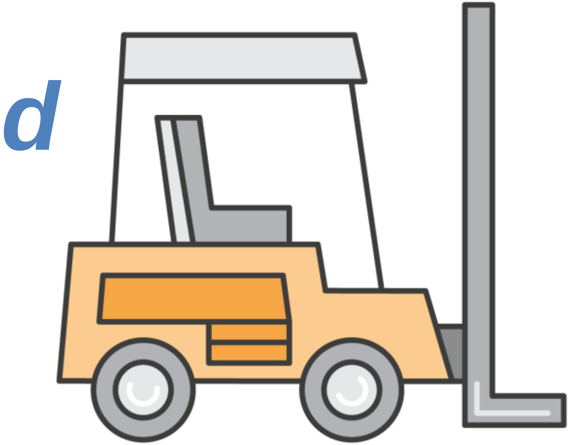
RDS is all about managed databases

Customers want performance managed too:

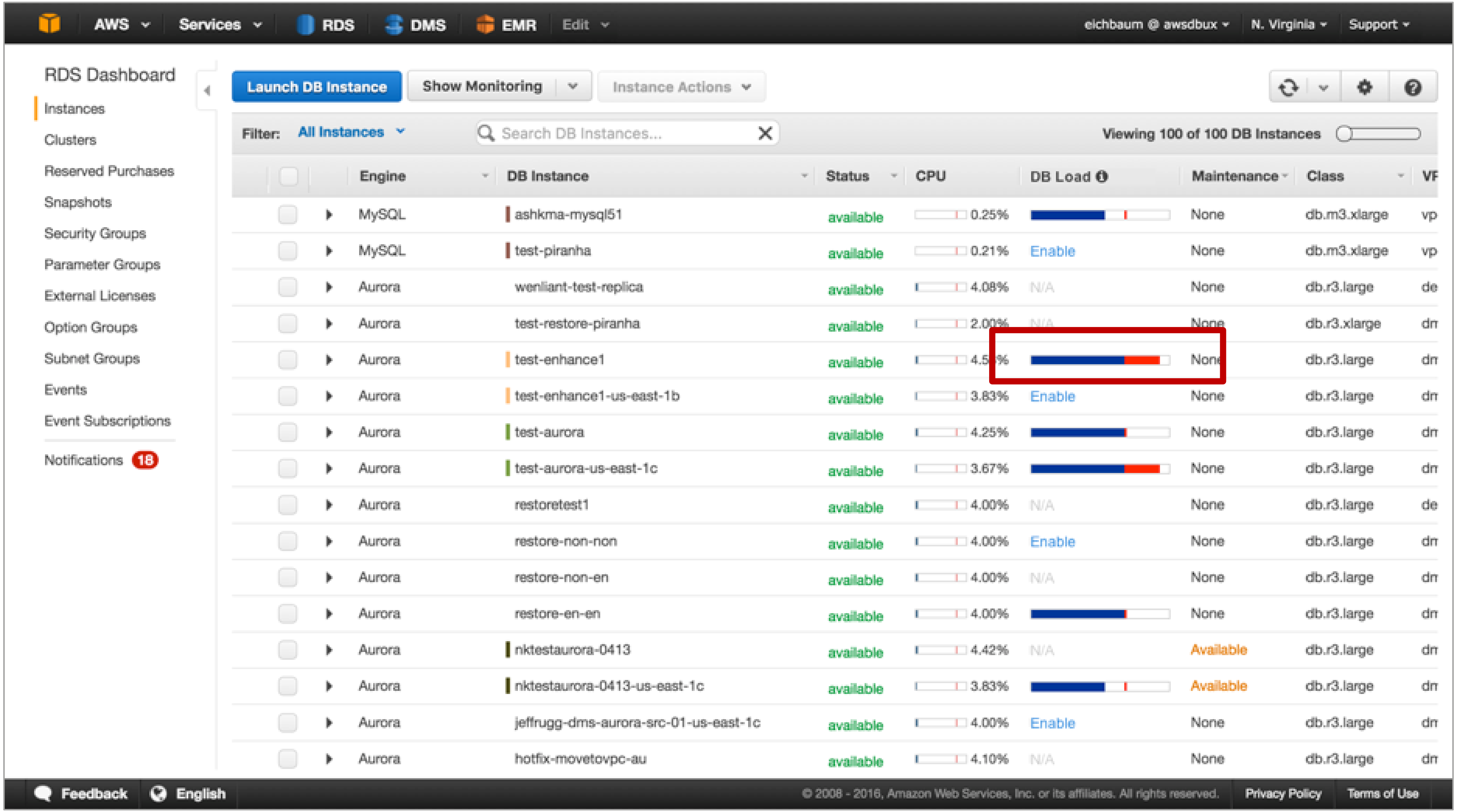
- ❑ Want easy tool for optimizing cloud database workloads
- ❑ May not have deep tuning expertise

→ Want a single pane of glass to achieve this

What makes *Database Load* such a useful metric?



- Based on sampling active database requests
- Frequent sampling builds a time model of usage
- Visualizations illuminate the time model in one chart



RDS Dashboard

- Instances
- Clusters
- Reserved Purchases
- Snapshots
- Security Groups
- Parameter Groups
- External Licenses
- Option Groups
- Subnet Groups
- Events
- Event Subscriptions
- Notifications 18

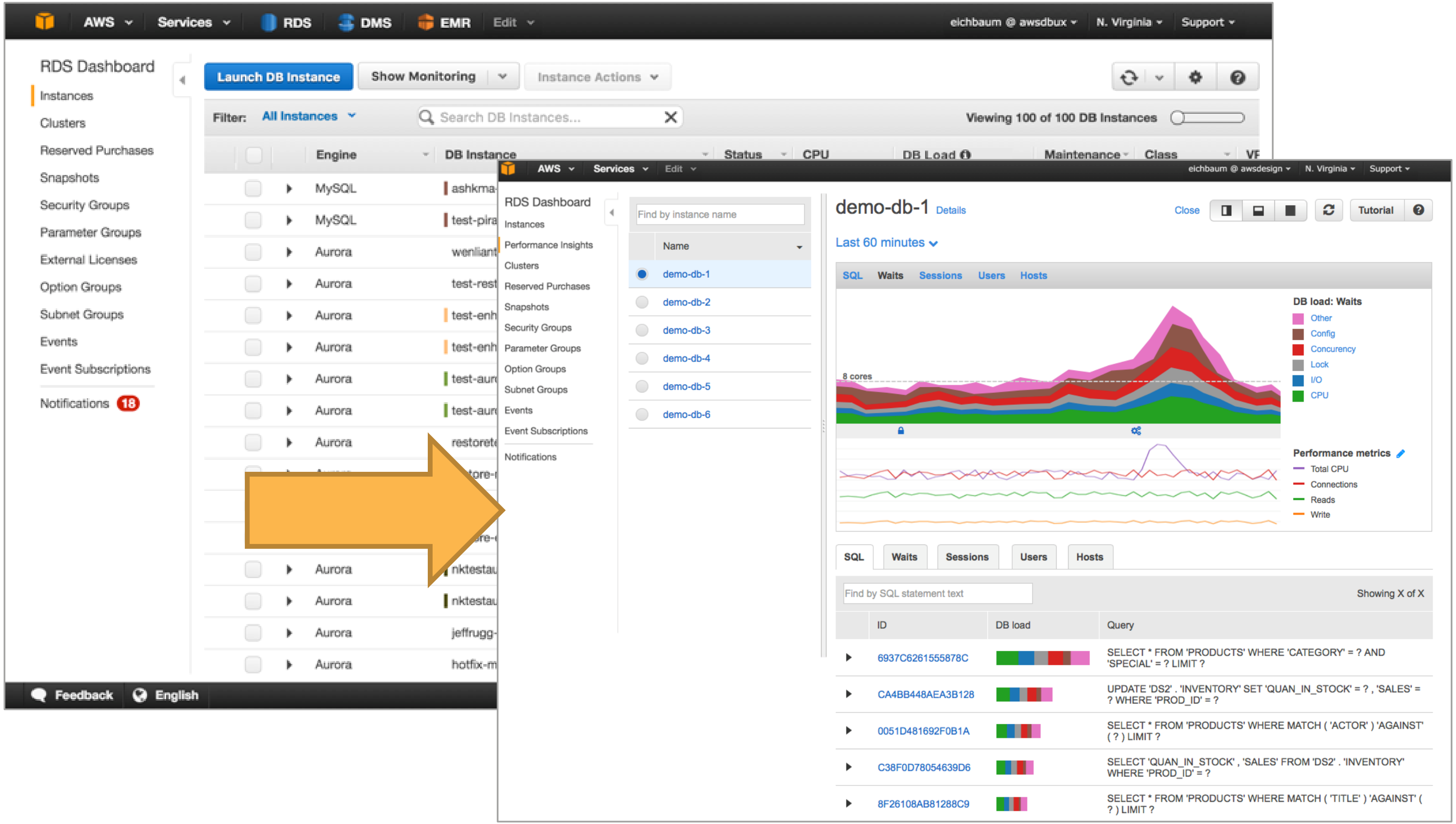
Filter: All Instances

Search DB Instances...

Viewing 100 of 100 DB Instances

□		Engine	DB Instance	Status	CPU	DB Load	Maintenance	Class	VP
□	▸	MySQL	ashkma-mysql51	available	0.25%		None	db.m3.xlarge	vp
□	▸	MySQL	test-piranha	available	0.21%	Enable	None	db.m3.xlarge	vp
□	▸	Aurora	wenliant-test-replica	available	4.08%	N/A	None	db.r3.large	de
□	▸	Aurora	test-restore-piranha	available	2.00%	N/A	None	db.r3.xlarge	dn
□	▸	Aurora	test-enhance1	available	4.5[illegible]%		None	db.r3.large	dn
□	▸	Aurora	test-enhance1-us-east-1b	available	3.83%	Enable	None	db.r3.large	dn
□	▸	Aurora	test-aurora	available	4.25%		None	db.r3.large	dn
□	▸	Aurora	test-aurora-us-east-1c	available	3.67%		None	db.r3.large	dn
□	▸	Aurora	restoretest1	available	4.00%	N/A	None	db.r3.large	de
□	▸	Aurora	restore-non-non	available	4.00%	Enable	None	db.r3.large	dn
□	▸	Aurora	restore-non-en	available	4.00%	N/A	None	db.r3.large	dn
□	▸	Aurora	restore-en-en	available	4.00%		None	db.r3.large	dn
□	▸	Aurora	nktestaurora-0413	available	4.42%	N/A	Available	db.r3.large	dn
□	▸	Aurora	nktestaurora-0413-us-east-1c	available	3.83%		Available	db.r3.large	dn
□	▸	Aurora	jeffrugg-dms-aurora-src-01-us-east-1c	available	4.00%	Enable	None	db.r3.large	dn
□	▸	Aurora	hotfix-movetovpc-au	available	4.10%	N/A	None	db.r3.large	dn

© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Engine	DB Instance	Status	CPU	DB Load	Maintenance	Class	VF
MySQL	ashkma						
MySQL	test-pira						
Aurora	wenliant						
Aurora	test-rest						
Aurora	test-enh						
Aurora	test-enh						
Aurora	test-aur						
Aurora	test-aur						
Aurora	restorete						
	tore-						
	re-						
Aurora	nktestau						
Aurora	nktestau						
Aurora	jeffrugg-						
Aurora	hotfix-m						

ID	DB load	Query
6937C6261555878C		SELECT * FROM 'PRODUCTS' WHERE 'CATEGORY' = ? AND 'SPECIAL' = ? LIMIT ?
CA4BB448AEA3B128		UPDATE 'DS2' . 'INVENTORY' SET 'QUAN_IN_STOCK' = ? , 'SALES' = ? WHERE 'PROD_ID' = ?
0051D481692F0B1A		SELECT * FROM 'PRODUCTS' WHERE MATCH ('ACTOR') 'AGAINST' (?) LIMIT ?
C38F0D78054639D6		SELECT 'QUAN_IN_STOCK' , 'SALES' FROM 'DS2' . 'INVENTORY' WHERE 'PROD_ID' = ?
8F26108AB81288C9		SELECT * FROM 'PRODUCTS' WHERE MATCH ('TITLE') 'AGAINST' (?) LIMIT ?

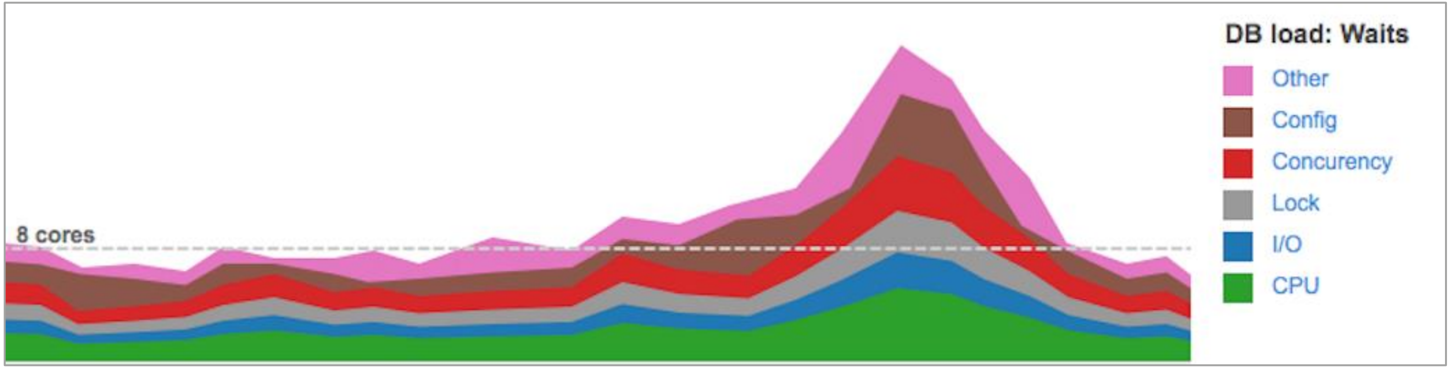
Performance Insights at a glance

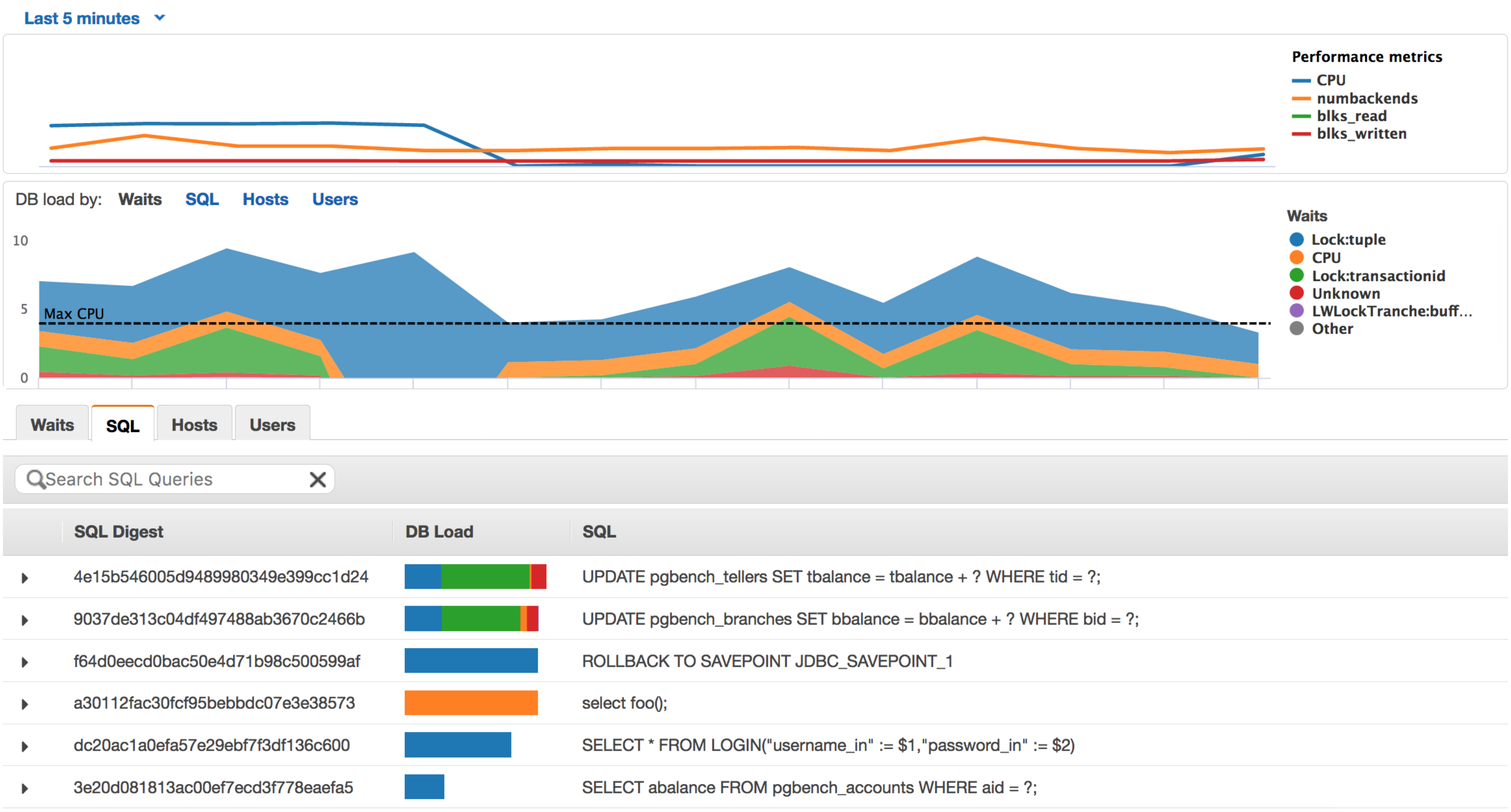
Automates sampling of data

Exposes data via API

Provides UI to show Database Load

Database Load:





	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 = ?;

Beyond Database Load

- Lock detection
- Execution plans
- API access
- Included with RDS
- 35 days data retention
- Support for all RDS database engines in 2017

Amazon Aurora with PostgreSQL Compatibility

Getting Your Data In

AWS Database Migration Service

PostgreSQL ORACLE Amazon Aurora MariaDB

AMAZON REDSHIFT MySQL Microsoft SQL Server SAP ASE

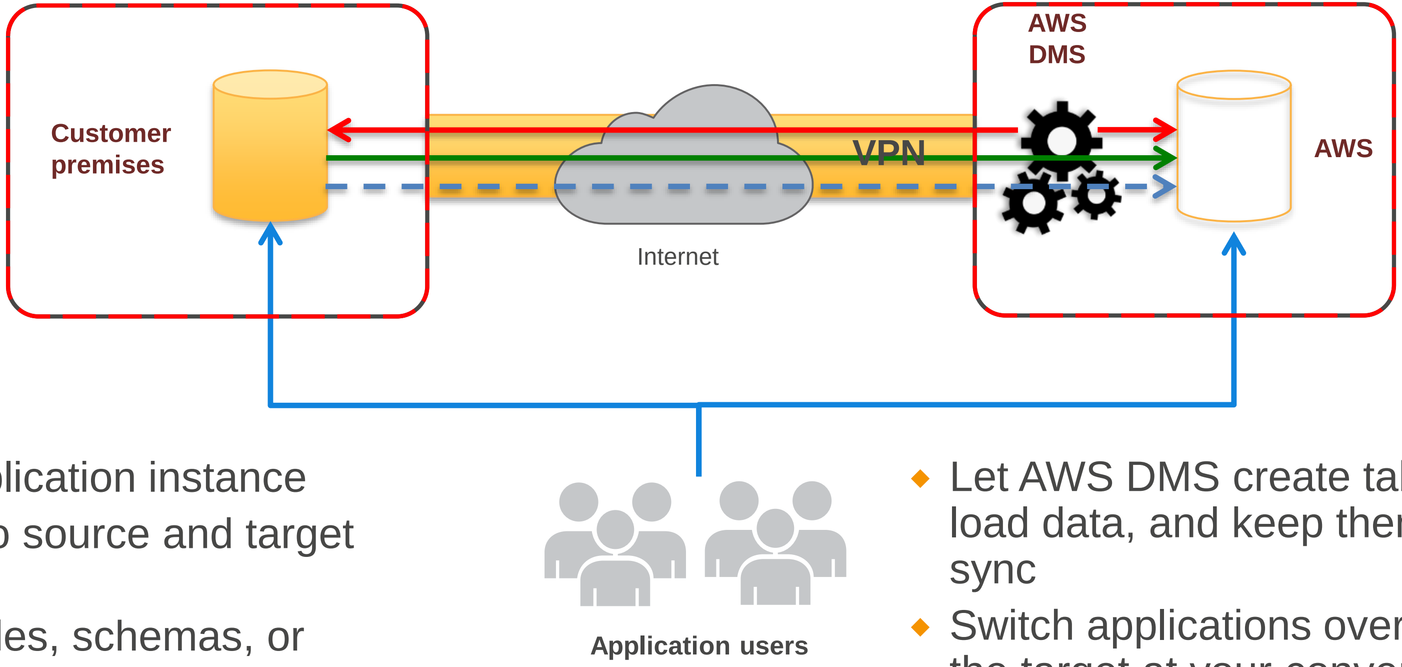
Start your first migration in 10 minutes or less

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

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 Database Migration Partners

2ND WATCH

apps associates
extreme expertise

CLOUDNEXA

DATAPIPE

DB
DB-BEST
TECHNOLOGIES

logicworks

Pythian
love your data®

slalom

UST Global®

APN Consulting Partners and Amazon Aurora

Experienced APN Partners, validated by AWS service teams and AWS customers

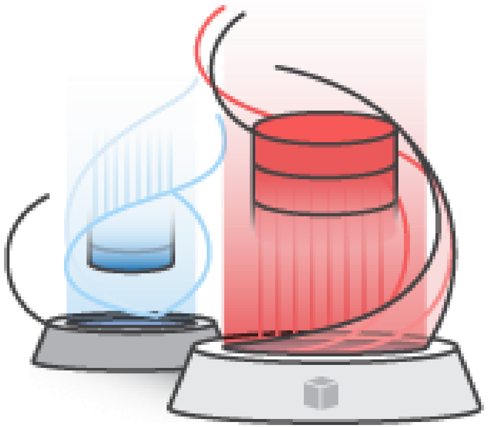
Amazon Aurora, Amazon RDS PostgreSQL, AWS Database Migration Service

Assessments, Proof of Concept, Migrations, Net New Implementations

slalom
apps associates
extreme expertise
CORPINFO
2ND WATCH
clearscale
DATAPIPE
amazon web services
Partner Network
CONSULTING PARTNER

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

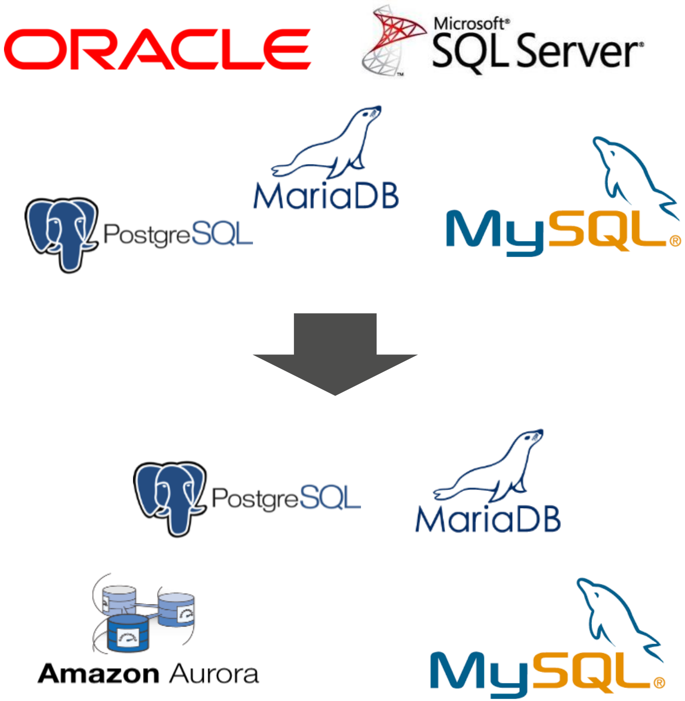
Code browser that highlights places where manual edits are required

Secure connections to your databases with SSL

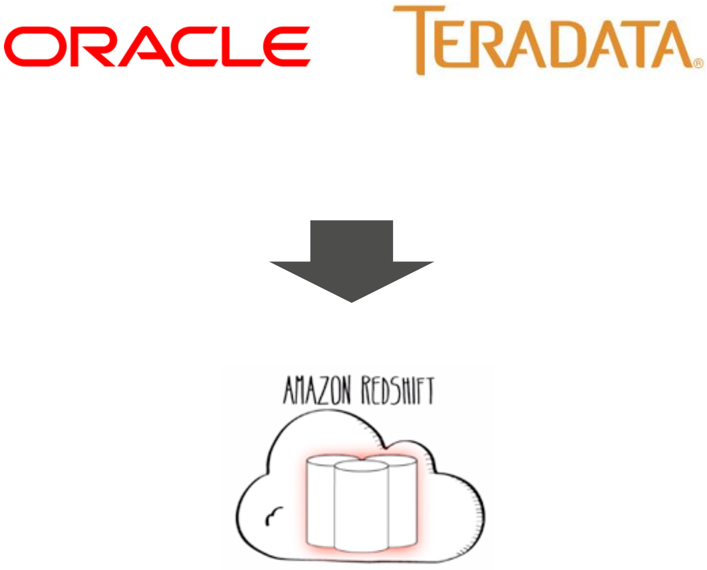
Cloud native code optimization

AWS Schema Conversion Tool

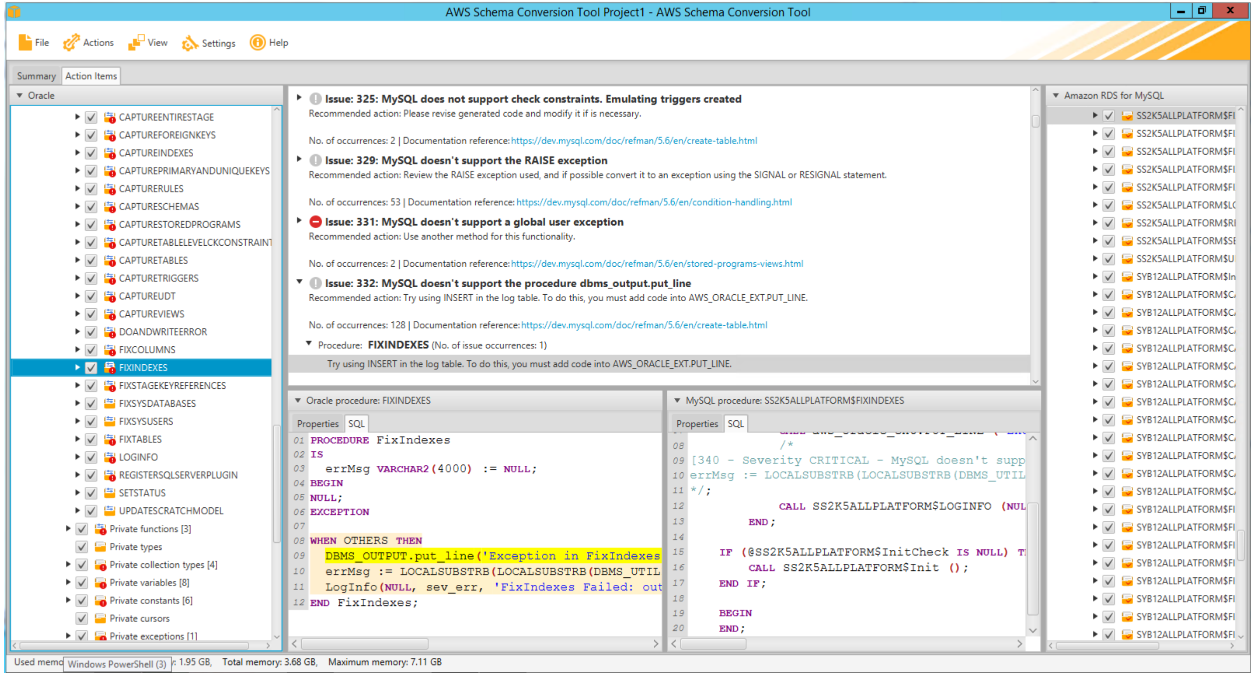
Converts relational databases



Converts warehouses



SCT helps with converting tables, views, and code


AWS Schema Conversion Tool Project1 - AWS Schema Conversion Tool
File
Actions
View
Settings
Help
Summary
Action Items

▼ Oracle

- ☑ CAPTUREENTIRESTAGE
- ☑ CAPTUREFOREIGNKEYS
- ☑ CAPTUREINDEXES
- ☑ CAPTUREPRIMARYANDUNIQUEKEYS
- ☑ CAPTURERULES
- ☑ CAPTURESCHEMAS
- ☑ CAPTURESTOREDPROGRAMS
- ☑ CAPTURETABLELEVELCKCONSTRAINT
- ☑ CAPTURETABLES
- ☑ CAPTURETRIGGERS
- ☑ CAPTUREUDT
- ☑ CAPTUREVIEWS
- ☑ DOANDWRITEERROR
- ☑ FIXCOLUMNS
- ☑ FIXINDEXES
- ☑ FIXSTAGEKEYREFERENCES
- ☑ FIXSYSDATABASES
- ☑ FIXSYSUSERS
- ☑ FIXTABLES
- ☑ LOGINFO
- ☑ REGISTERSQLSERVERPLUGIN
- ☑ SETSTATUS
- ☑ UPDATESCRATCHMODEL
- ☑ Private functions [3]
- ☑ Private types
- ☑ Private collection types [4]
- ☑ Private variables [8]
- ☑ Private constants [6]
- ☑ Private cursors
- ☑ Private exceptions [1]

Issue: 325: MySQL does not support check constraints. Emulating triggers created
Recommended action: Please revise generated code and modify it if is necessary.

No. of occurrences: 2 | Documentation reference: https://dev.mysql.com/doc/refman/5.6/en/create-table.html

Issue: 329: MySQL doesn't support the RAISE exception
Recommended action: Review the RAISE exception used, and if possible convert it to an exception using the SIGNAL or RESIGNAL statement.

No. of occurrences: 53 | Documentation reference: https://dev.mysql.com/doc/refman/5.6/en/condition-handling.html

Issue: 331: MySQL doesn't support a global user exception
Recommended action: Use another method for this functionality.

No. of occurrences: 2 | Documentation reference: https://dev.mysql.com/doc/refman/5.6/en/stored-programs-views.html

Issue: 332: MySQL doesn't support the procedure dbms_output.put_line
Recommended action: Try using INSERT in the log table. To do this, you must add code into AWS_ORACLE_EXT.PUT_LINE.

No. of occurrences: 128 | Documentation reference: https://dev.mysql.com/doc/refman/5.6/en/create-table.html

▼ Procedure: **FIXINDEXES** (No. of issue occurrences: 1)

Try using INSERT in the log table. To do this, you must add code into AWS_ORACLE_EXT.PUT_LINE.

▼ Oracle procedure: FIXINDEXES

Properties | SQL

```
PROCEDURE FixIndexes
IS
  errMsg VARCHAR2(4000)  := NULL;
BEGIN
NULL;
EXCEPTION

WHEN OTHERS THEN
  DBMS_OUTPUT.put_line('Exception in FixIndexes
  errMsg := LOCALSUBSTRB(LOCALSUBSTRB(DBMS_UTIL
  LogInfo(NULL, sev_err, 'FixIndexes Failed: out
END FixIndexes;
```

▼ MySQL procedure: SS2K5ALLPLATFORM$FIXINDEXES

Properties | SQL

```
                /*
[340 - Severity CRITICAL - MySQL doesn't supp
errMsg := LOCALSUBSTRB(LOCALSUBSTRB(DBMS_UTIL
*/;
                CALL SS2K5ALLPLATFORM$LOGINFO (NUL
            END;

        IF (@SS2K5ALLPLATFORM$InitCheck IS NULL) T
            CALL SS2K5ALLPLATFORM$Init ();
        END IF;

        BEGIN
        END;
```

▼ Amazon RDS for MySQL

- ☑ SS2K5ALLPLATFORM$FI
- ☑ SS2K5ALLPLATFORM$FI
- ☑ SS2K5ALLPLATFORM$FI
- ☑ SS2K5ALLPLATFORM$FI
- ☑ SS2K5ALLPLATFORM$FI
- ☑ SS2K5ALLPLATFORM$LC
- ☑ SS2K5ALLPLATFORM$RI
- ☑ SS2K5ALLPLATFORM$SE
- ☑ SS2K5ALLPLATFORM$U
- ☑ SYB12ALLPLATFORM$In
- ☑ SYB12ALLPLATFORM$C
- ☑ SYB12ALLPLATFORM$C
- ☑ SYB12ALLPLATFORM$C
- ☑ SYB12ALLPLATFORM$C
- ☑ SYB12ALLPLATFORM$C
- ☑ SYB12ALLPLATFORM$C
- ☑ SYB12ALLPLATFORM$C
- ☑ SYB12ALLPLATFORM$C
- ☑ SYB12ALLPLATFORM$C
- ☑ SYB12ALLPLATFORM$C
- ☑ SYB12ALLPLATFORM$C
- ☑ SYB12ALLPLATFORM$C
- ☑ SYB12ALLPLATFORM$FI
- ☑ SYB12ALLPLATFORM$FI
- ☑ SYB12ALLPLATFORM$FI
- ☑ SYB12ALLPLATFORM$FI
- ☑ SYB12ALLPLATFORM$FI
- ☑ SYB12ALLPLATFORM$FI
- ☑ SYB12ALLPLATFORM$FI
- ☑ SYB12ALLPLATFORM$FI

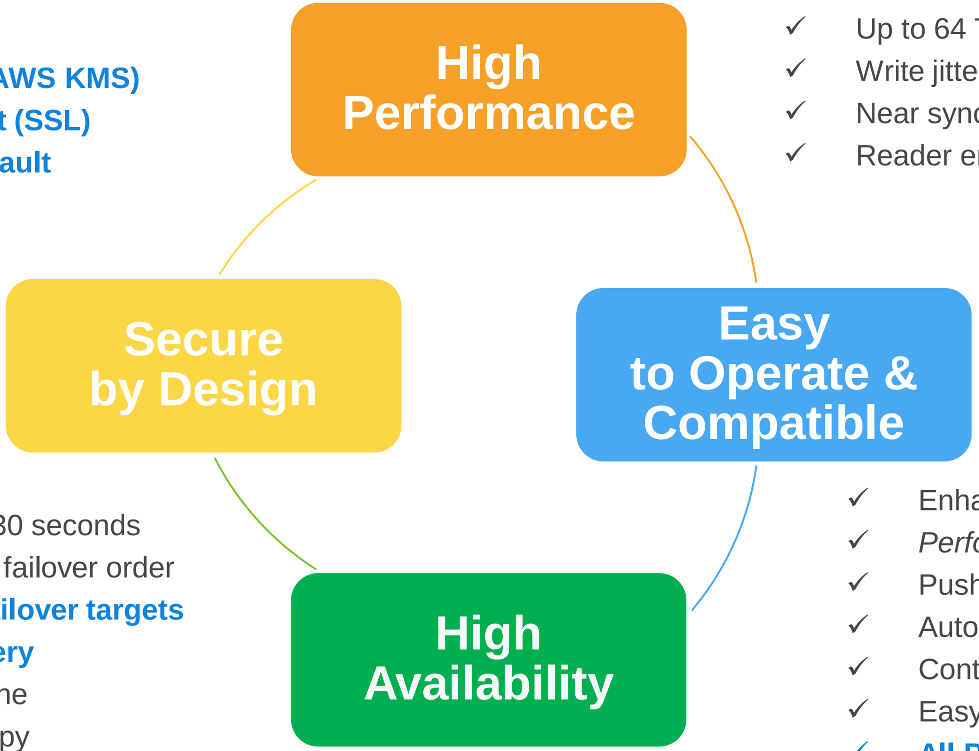
Used memo Windows PowerShell (3) r: 1.95 GB, Total memory: 3.68 GB, Maximum memory: 7.11 GB

Sequences
User-defined types
Synonyms
Packages
Stored procedures
Functions
Triggers
Schemas
Tables
Indexes
Views
Sort and distribution keys

Amazon Aurora with PostgreSQL Compatibility

Roadmap

Amazon Aurora with PostgreSQL Compatibility – Launch Roadmap



High Performance

Secure by Design

Easy to Operate & Compatible

High Availability

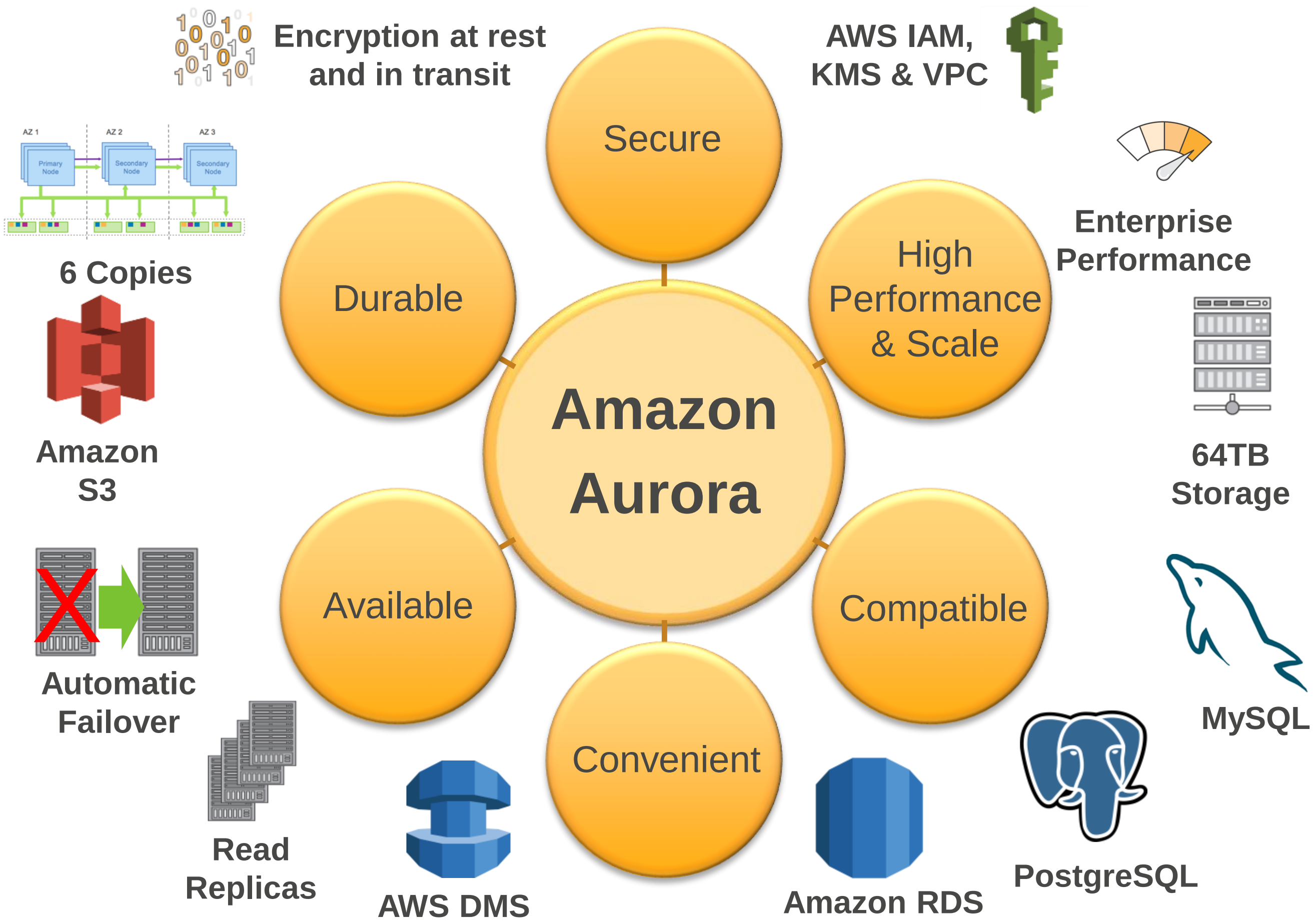
- ✓ **2x or faster than PostgreSQL**
- ✓ Up to 64 TB of storage per instance
- ✓ Write jitter reduction
- ✓ Near synchronous replicas
- ✓ Reader endpoint

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

- ✓ 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**

- ✓ 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

The Amazon Aurora Database Family



Amazon Aurora

Secure

High Performance & Scale

Compatible

Convenient

Available

Durable

Encryption at rest and in transit

AWS IAM, KMS & VPC

Enterprise Performance

64TB Storage

MySQL

PostgreSQL

Amazon RDS

AWS DMS

Read Replicas

Automatic Failover

Amazon S3

6 Copies

Questions

Timeline

We are taking signups for the open preview now

We plan to release in general availability in 2017

How do I sign up for the preview?

https://pages.awscloud.com/amazon-aurora-with-postgresql-compatibility-preview-form.html

FAQs

https://aws.amazon.com/rds/aurora/faqs/#postgresql

Kevin Jernigan, Senior Product Manager (kmj@amazon.com)

Thank You!