Deep Dive on Amazon Aurora with PostgreSQL Compatibility

Jim Mlodgenski, Principal Database Engineer Amazon RDS

May, 2019
Amazon RDS is . . .

<table>
<thead>
<tr>
<th>Cloud native engine</th>
<th>Open source engines</th>
<th>Commercial engines</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="https://example.com" alt="Amazon Aurora" /></td>
<td><img src="https://example.com" alt="MySQL" /></td>
<td><img src="https://example.com" alt="SQL Server" /></td>
</tr>
<tr>
<td><img src="https://example.com" alt="PostgreSQL" /></td>
<td><img src="https://example.com" alt="MariaDB" /></td>
<td><img src="https://example.com" alt="Oracle" /></td>
</tr>
</tbody>
</table>

**RDS platform**

- Automatic fail-over
- Backup & recovery
- X-region replication
- Isolation & security
- Industry compliance
- Automated patching
- Advanced monitoring
- Routine maintenance
- Push-button scaling

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Image credit: By Mackphillips - Own work, CC BY-SA 4.0, [https://commons.wikimedia.org/w/index.php?curid=55946550](https://commons.wikimedia.org/w/index.php?curid=55946550)
# RDS PostgreSQL Universe

## Clients

<table>
<thead>
<tr>
<th>Postgres 9.6/10/11 — same extensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backup/Recovery - PITR</td>
</tr>
<tr>
<td>High Availability &amp; Durability</td>
</tr>
<tr>
<td>Secure – IAM Auth</td>
</tr>
<tr>
<td>Read Replicas</td>
</tr>
<tr>
<td>Cross Region Snapshots</td>
</tr>
<tr>
<td>Scale Compute – Online Scale Storage</td>
</tr>
<tr>
<td>Outbound Logical Replication</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cross Region Replication</th>
</tr>
</thead>
</table>

## RDS PostgreSQL

## EBS

## Aurora PostgreSQL

## Aurora Storage

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
Log-Based Storage
Concurrency—Remove Log Buffer

Queued Work

Log Buffer

Storage

PostgreSQL

Queued Work

Durability Tracking

Aurora PostgreSQL

Storage
Aurora PostgreSQL—Writing Less

update t set y = 6;

Memory

Full Block

Memory

update t set y = 6;

PostgreSQL

Aurora Storage

checkpoint

4K

datafile

no checkpoint = no FPW

WAL

archive

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
## Insert Test

### Test Table

- **UUID PK**—Random
- **ID int**—Right Lean Sequence
- **VARCHAR(100)**—Random
- **VARCHAR(50)**—Small Set of Words
- **INT**—Random
- **INT**—Random (smaller set)
- **BOOLEAN**—Random (50/50)
- **BOOLEAN**—Somewhat Random (75/25)
- **Timestamp**—Right Lean
Update Workload—PostgreSQL 9.6

- **BASE**: 3,729
- **16GB Max WAL**: 4,871
- **Aurora PostgreSQL**: 17,158

TPS (2 Updates per Transaction)
Amazon Aurora Recovers Up to 97% Faster

Recovery Time from Crash Under Load

- As PostgreSQL throughput goes up, so does log size and crash recovery time.
- Amazon Aurora has no redo. Recovered in 3 seconds while maintaining significantly greater throughput.

Bubble size represents redo log, which must be recovered.
Base Architecture
Aurora Storage and Replicas

1. Application
2. RO
3. Write log records
4. Read blocks
5. Async Invalidation & Update
6. RW
7. Automatic Scalable to 64TB
8. Availability zone 1
9. Availability zone 2
10. Availability zone 3

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
Durability—4 of 6 Quorum
Typical synchronous replication – 3 locations
Cost of Additional Synchronous Replicas

High Concurrency Sync Write Test

- 2 Node (4 copy)
- 3 Node (6 Copy)

Latency (ms)

Percentile

50  90  99.9  99.99

6  7  10  12  21  28  31

Latency:
- 50th Percentile:
  - 2 Node: 6 ms
  - 3 Node: 7 ms
- 90th Percentile:
  - 2 Node: 10 ms
  - 3 Node: 12 ms
- 99.9th Percentile:
  - 2 Node: 21 ms
  - 3 Node: 28 ms
- 99.99th Percentile:
  - 2 Node: 31 ms
  - 3 Node: 123 ms

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
Aurora – 3 AZ’s – 6 copies
Amazon Aurora Gives >2x Lower Response Times

sysbench response time (p95), 30 GiB, 1024 clients

- PostgreSQL (Single AZ, No Backup)
- Amazon Aurora (Three AZs, Continuous Backup)
Replicas and Clones
Replicas—PostgreSQL

Async Replication

update

Catchup
Replicas—Amazon Aurora

Aurora Storage

Async Replication

update
Aurora RW

update in memory
Aurora RO
Async Replication and apply

Async Replication and memory update
Replicas—Backfill on PostgreSQL

pgbench RW 8K tps on Primary – RO 200k tps on replica

backfill
pgbench_history
Replicas—Backfill on Amazon Aurora

pgbench RW 8K tps on Primary – RO 200k tps on replica

backfill

pgbench_history
Fast Clones

Availability zone 1

Application

RO

Availability zone 2

Application

RW

Write log records

Read blocks

Clone

Availability zone 3

Reporting
Application

RW

Aurora Storage

Primary Storage

Clone Storage
Fast clone example

PGBench RW Scale 10K - Target Rate 20K TPS

Main Database

Clone Database

request clone
Replication
Logical Replication Support

Converts Physical Changes (WAL to SQL Statements (i.e. logical))

- Logical Decoding Plugin
  (test_decoder, decoder_raw, wal2jason)

- V10 – Publish / Subscribe to another PostgreSQL Instance
Caching
Caching Changes—No Double Buffering

PostgreSQL

488GB RAM

select of data – check for block in shared_buffers
if not in shared_buffers load from pagecache/disk

Linux Pagecache

duplicate buffers

EBS

Aurora PostgreSQL

Survivable Cache

shared_buffers 75%

select of data – check for block in shared_buffers or load from Aurora storage

Aurora Storage

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
Caching Changes—No Double Buffering

pgbench read only - scale 22,000 - r4.16xlarge

Aurora 75% Cache

PostgreSQL 25% Cache

- no reads
- heavy double buffering
- no survivable cache

18K read iops

Approx 350GB working set
Cluster Cache Management - Failover

PGBench 20X RO / 1X RW 160GB Cached - Failover at 600 Seconds

- Baseline
- Failure at 600 Seconds
- 340 seconds
- 32 seconds
Cluster Cache Management (CCM) Feature

Availability zone 1

Application

Failover Priority 1

RO

Availability zone 2

Application

RW

Failover Priority 0

bloom filter - replica cache

Availability zone 3

Application

RO

Failover Priority 0

block addresses to load

Aurora Storage

apg_ccm_enabled=on

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
Cluster Cache Management

PGBench 20X RO / 1X RW 160GB Cached - Failover at 600 Seconds

- Baseline
- CCM Enabled

Failover at 600 Seconds:
- 340 seconds
- 32 seconds
Performance
Performance Insights
Performance Insights

Database load
Average active sessions (AAS)

Lead By Waits (AAS)

<table>
<thead>
<tr>
<th>SQL</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>select sum(delta), sum(balance) from pgbench_history h, pgbench_branches b where b.bid = h.bid and b.bid in (?,?,?,?) and runtime between now() - interval ? and now() - interval ?</td>
<td>29.81</td>
</tr>
<tr>
<td>UPDATE pgbench_accounts SET balance = balance + ? WHERE aid = ?;</td>
<td>3.7</td>
</tr>
<tr>
<td>SELECT balance FROM pgbench_accounts WHERE aid = ?;</td>
<td>2.3</td>
</tr>
</tbody>
</table>
Plan Change

Before

Aggregate (cost=3804.15..3804.16 rows=1 width=16)
  └─ Nested Loop (cost=12.67..3802.61 rows=307 width=8)
      └─ Index Scan using pgbench_branches_pkey on pgbench_branches b (cost=0.29..16.60 rows=2 width=8)
          Index Cond: (bid = ANY ('\{1,4\}':integer[]))
      └─ Bitmap Heap Scan on pgbench_history h (cost=12.39..1891.47 rows=154 width=8)
          Recheck Cond: (bid = b.bid)
          Filter: ((mtime > (now() - '01:00:00':interval)) AND (mtime <= (now() - '00:30:00':interval)))
      └─ Bitmap Index Scan on i_p_bid (cost=0.00..12.35 rows=522 width=8)
          Index Cond: (bid = b.bid)

After

Aggregate (cost=171092.96..171092.97 rows=1 width=16)
  └─ Hash Join (cost=329.02..171091.42 rows=307 width=8)
      Hash Cond: (h.bid = b.bid)
      └─ Seq Scan on pgbench_history h (cost=0.00..166712.20 rows=1542280 width=8)
          Filter: ((mtime >= (now() - '01:00:00':interval)) AND (mtime <= (now() - '00:30:00':interval)))
      └─ Hash (cost=329.00..329.00 rows=2 width=8)
          └─ Seq Scan on pgbench_branches b (cost=0.00..329.00 rows=2 width=8)
              Filter: (bid = ANY ('\{1,4\}':integer[]))

• stats change?
• config change?
• index change?

• enable_bitmapscan=off
• enable_indexscan=off
Query Plan Management - QPM

- Capture statements
- Approve statements
- Evolve better plans
QPM – Use Plan Baselines

Database load
Average active sessions (AAS)

Waits SQL Hosts Users

<table>
<thead>
<tr>
<th>SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>select sum(delta), sum(balance) from pgbench_history h, pgbench_branches b where b.bid = h.bid and b.bid in (7, 7, 7, 7) and mtime between now() - interval ? and now() - interval ?;</td>
</tr>
<tr>
<td>UPDATE pgbench_accounts SET abalance = abalance + ? WHERE aid = ?;</td>
</tr>
<tr>
<td>SELECT abalance FROM pgbench_accounts WHERE aid = ?;</td>
</tr>
<tr>
<td>UPDATE pgbench_branches SET bbalance = bbalance + ? WHERE bid = ?;</td>
</tr>
<tr>
<td>UPDATE pgbench_tellers SET tbalance = tbalance + ? WHERE tid = ?;</td>
</tr>
<tr>
<td>select sum(delta), sum(balance) from pgbench_history h, pgbench_branches b where b.bid = h.bid and b.bid in (7, 7, 7, 7) and mtime between now() - interval ? and now() - interval ?;</td>
</tr>
<tr>
<td>UPDATE pgbench_accounts SET abalance = abalance + ? WHERE aid = ?;</td>
</tr>
<tr>
<td>UPDATE pgbench_branches SET bbalance = bbalance + ? WHERE bid = ?;</td>
</tr>
<tr>
<td>UPDATE pgbench_tellers SET tbalance = tbalance + ? WHERE tid = ?;</td>
</tr>
</tbody>
</table>
Vacuuming
Updates—No Vacuum Running

transaction id wrap around
Intelligent Vacuum Prefetch

PostgreSQL 402 seconds

visibility & frozen map

Aurora PostgreSQL 163 seconds

Submit Batch I/O up to 256 blocks

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
Thank you!