Deep Dive on Amazon Aurora with PostgreSQL Compatibility

Jim Mlodgenski, Principal Database Engineer Amazon RDS

May, 2019

Amazon RDS is . . .

Cloud native engine

Amazon Aurora

Open source engines

MySQL

MariaDB

PostgreSQL

Commercial engines

Microsoft SQL Server

ORACLE

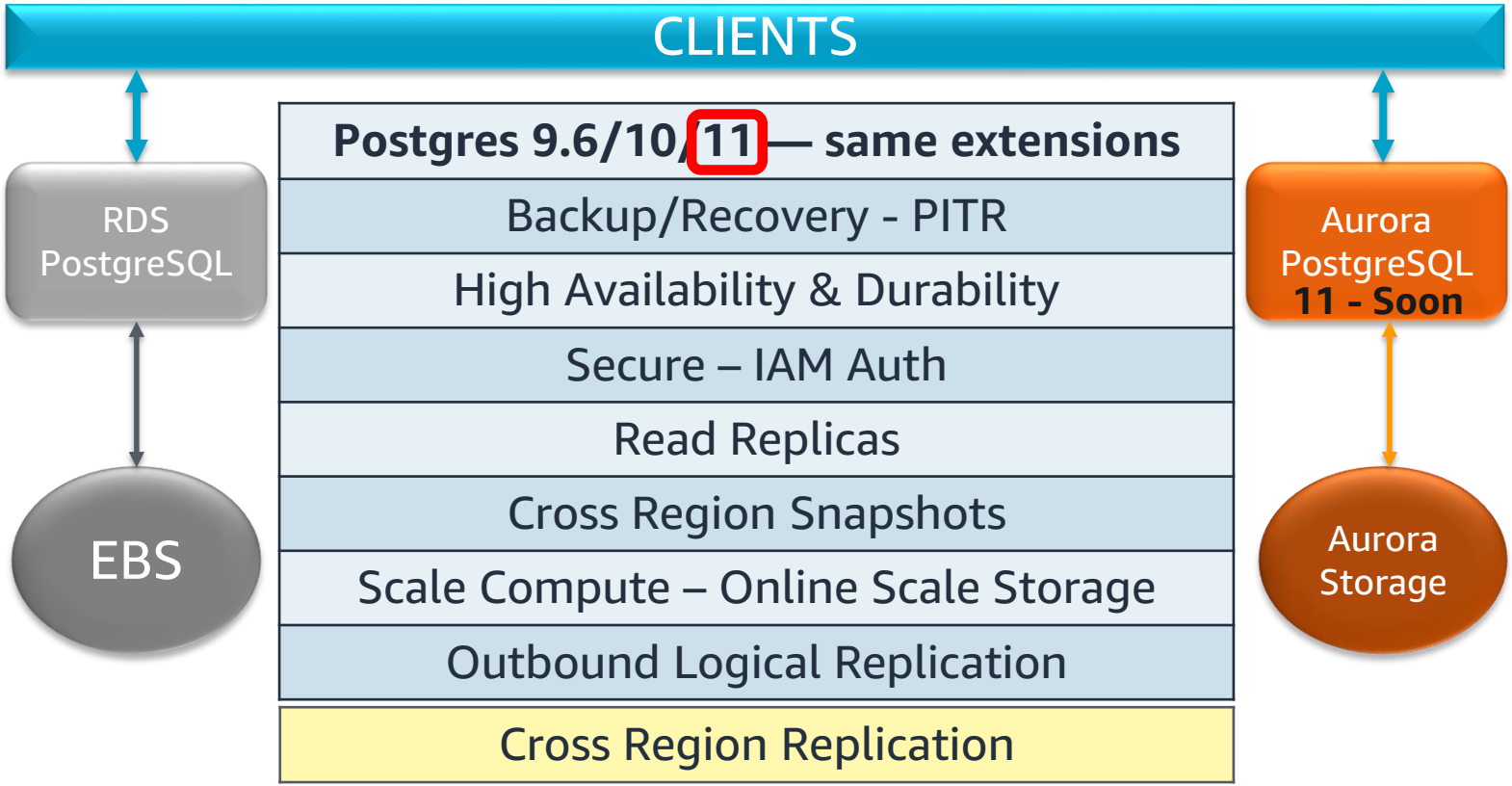
RDS platform

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

Image credit: By Mackphillips - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=55946550

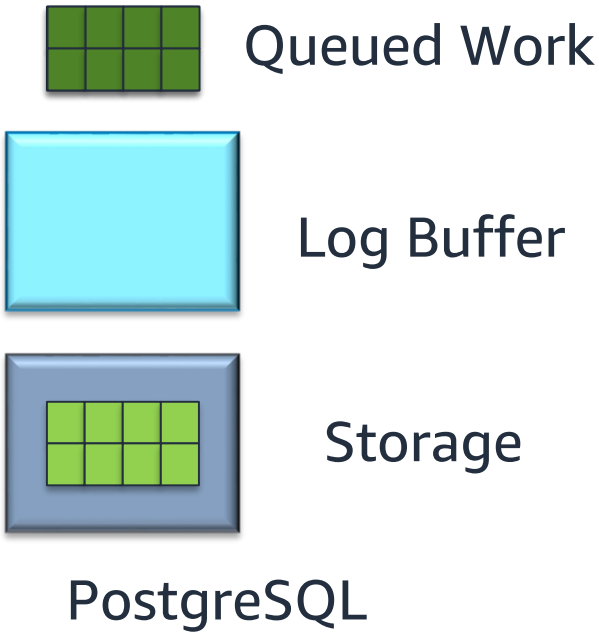
aws

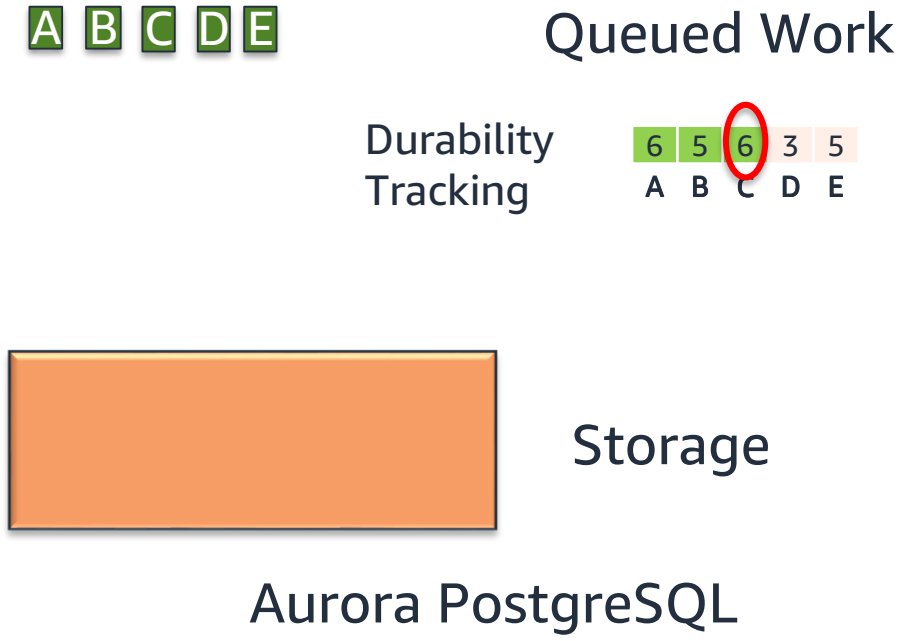
RDS PostgreSQL Universe



Log-Based Storage

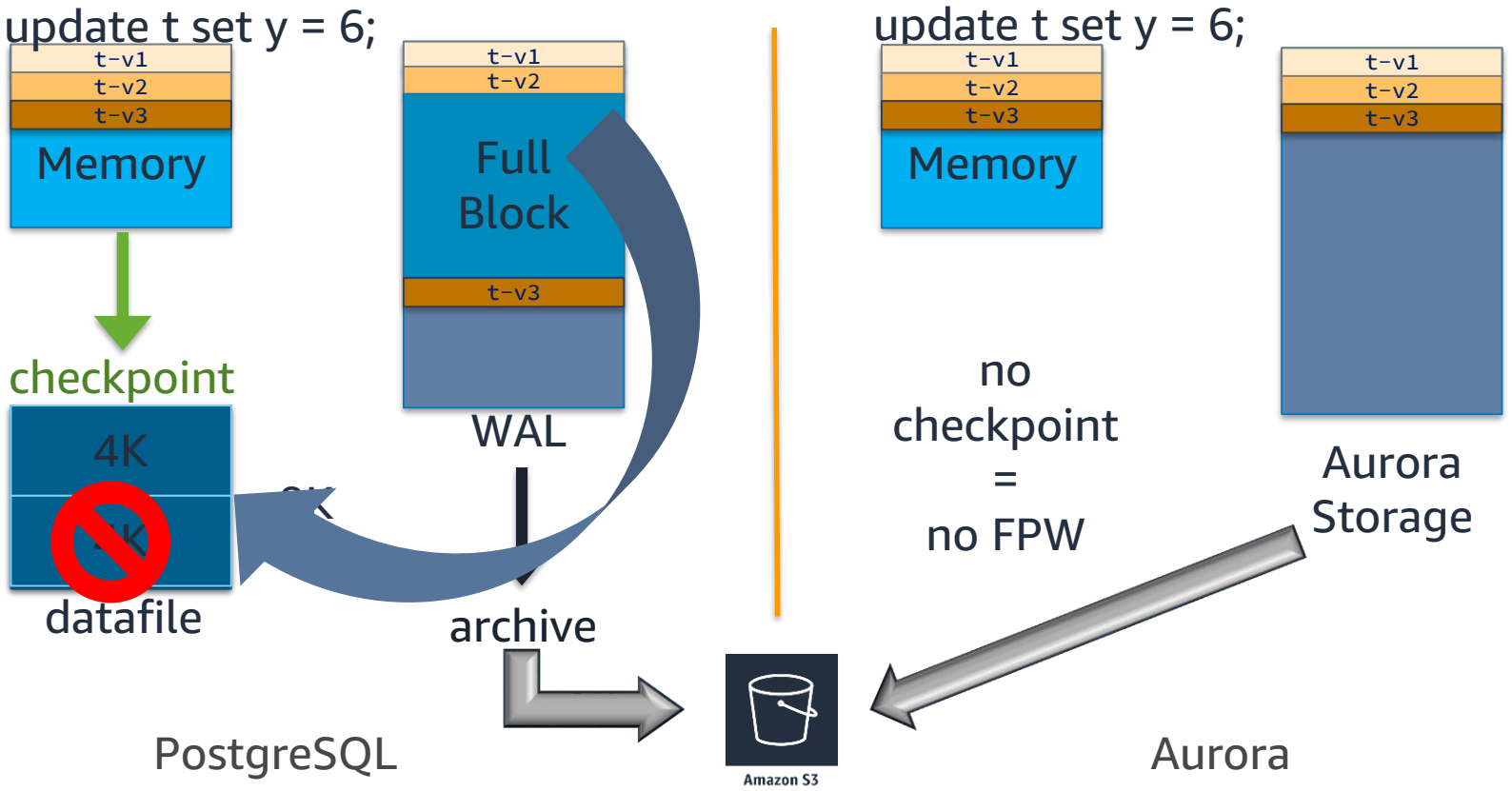
Concurrency—Remove Log Buffer





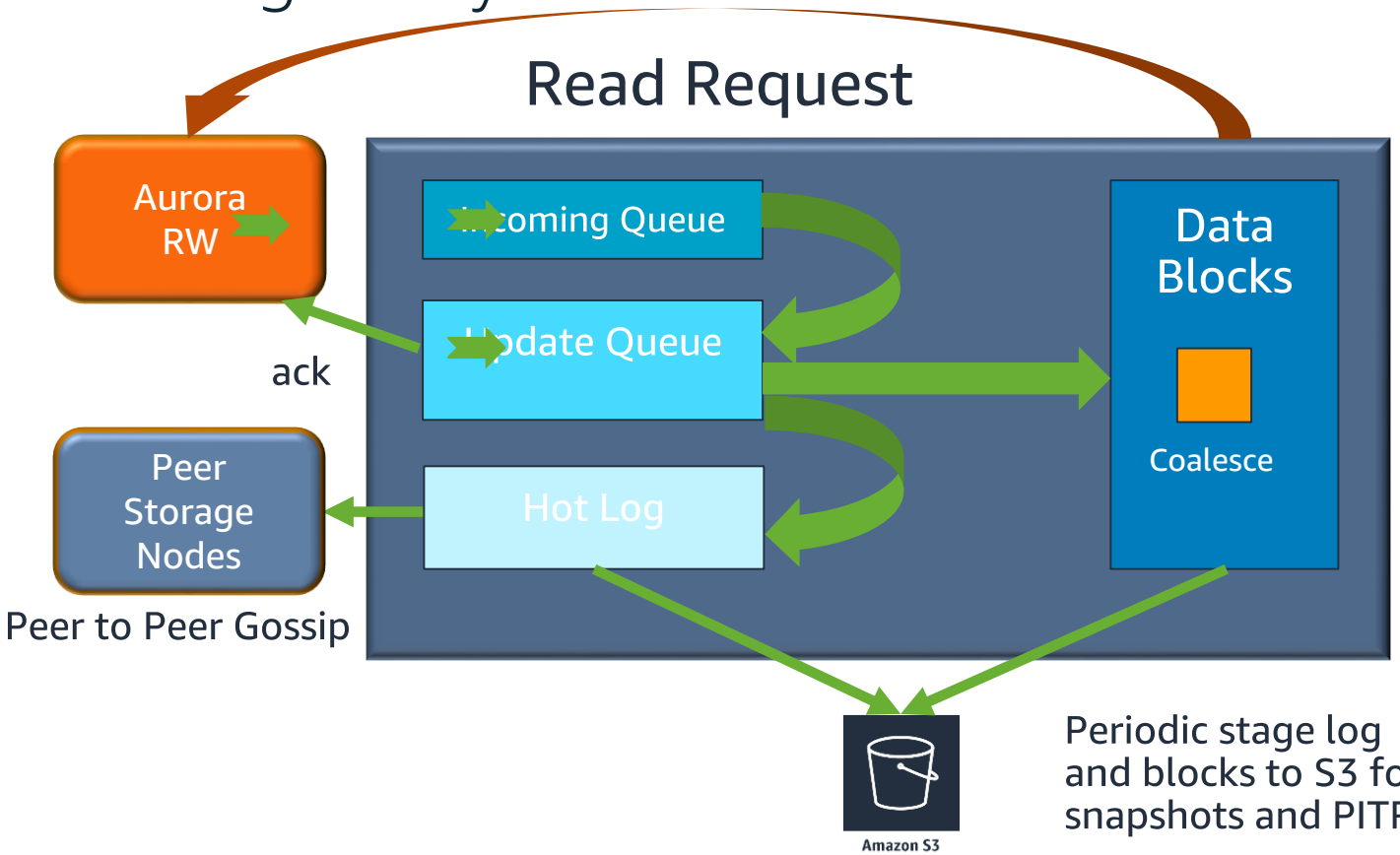
aws

Aurora PostgreSQL—Writing Less



aws

Storage Layer



Insert Test

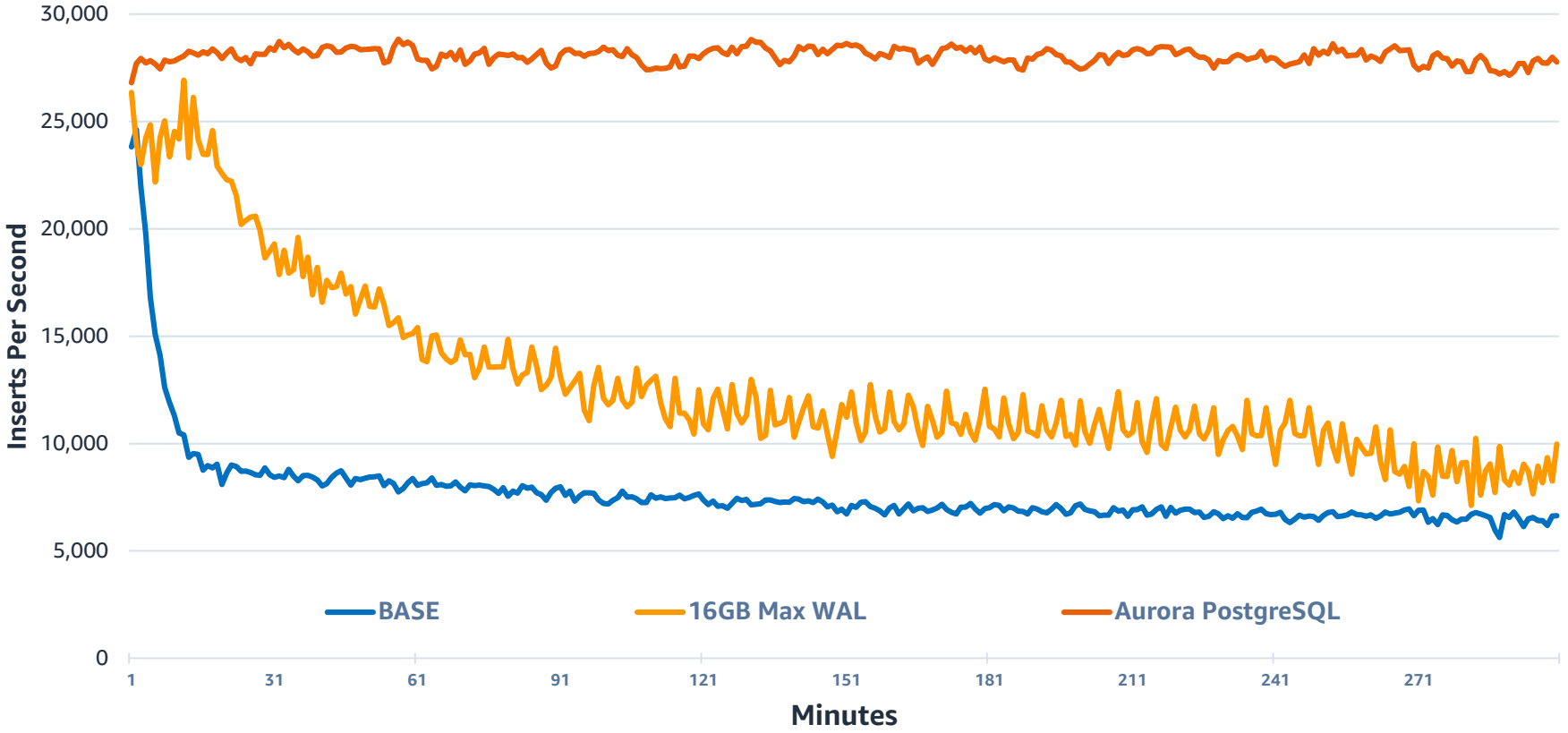
Test Table

- 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

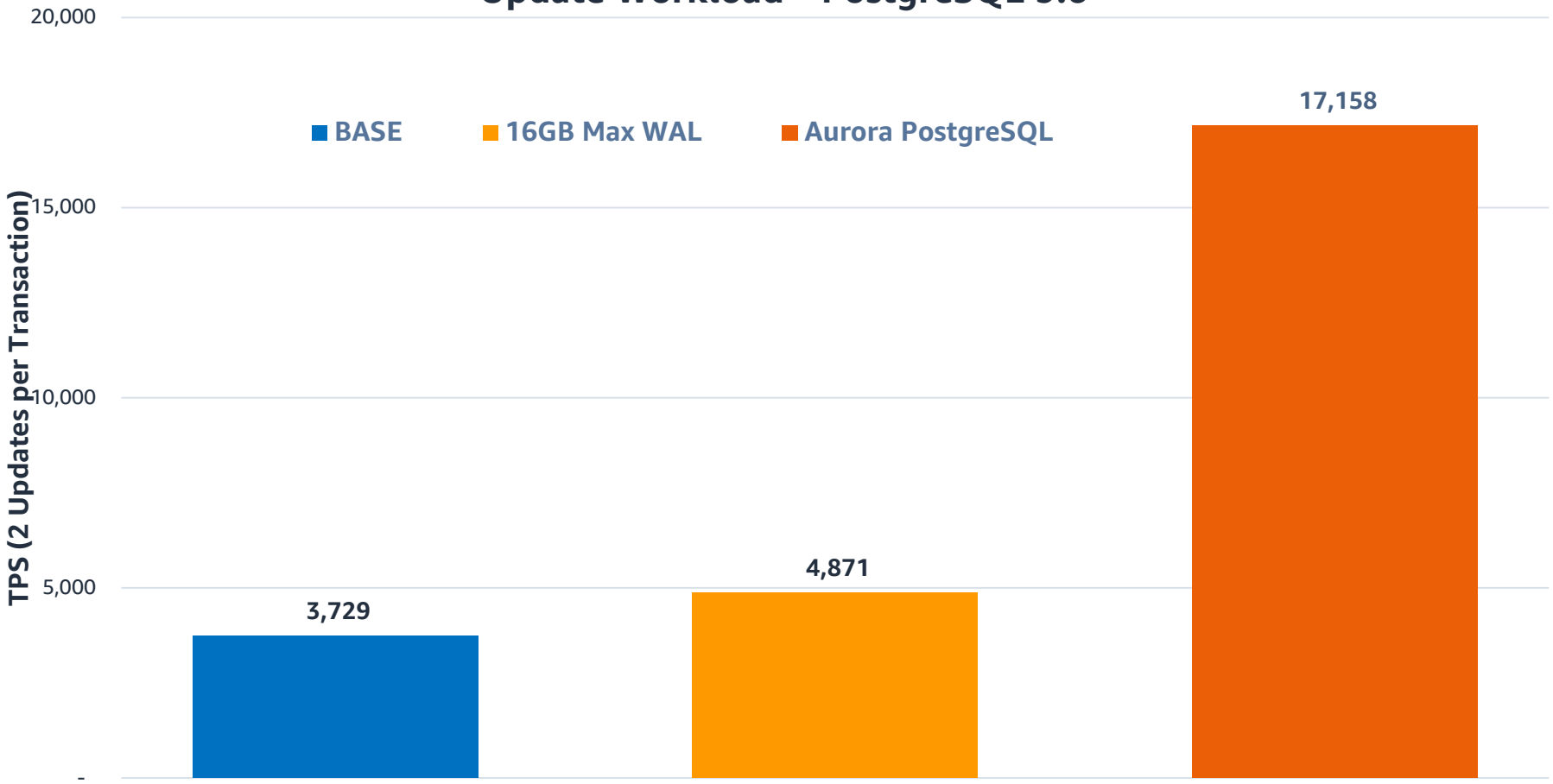
Index Every Column

aws

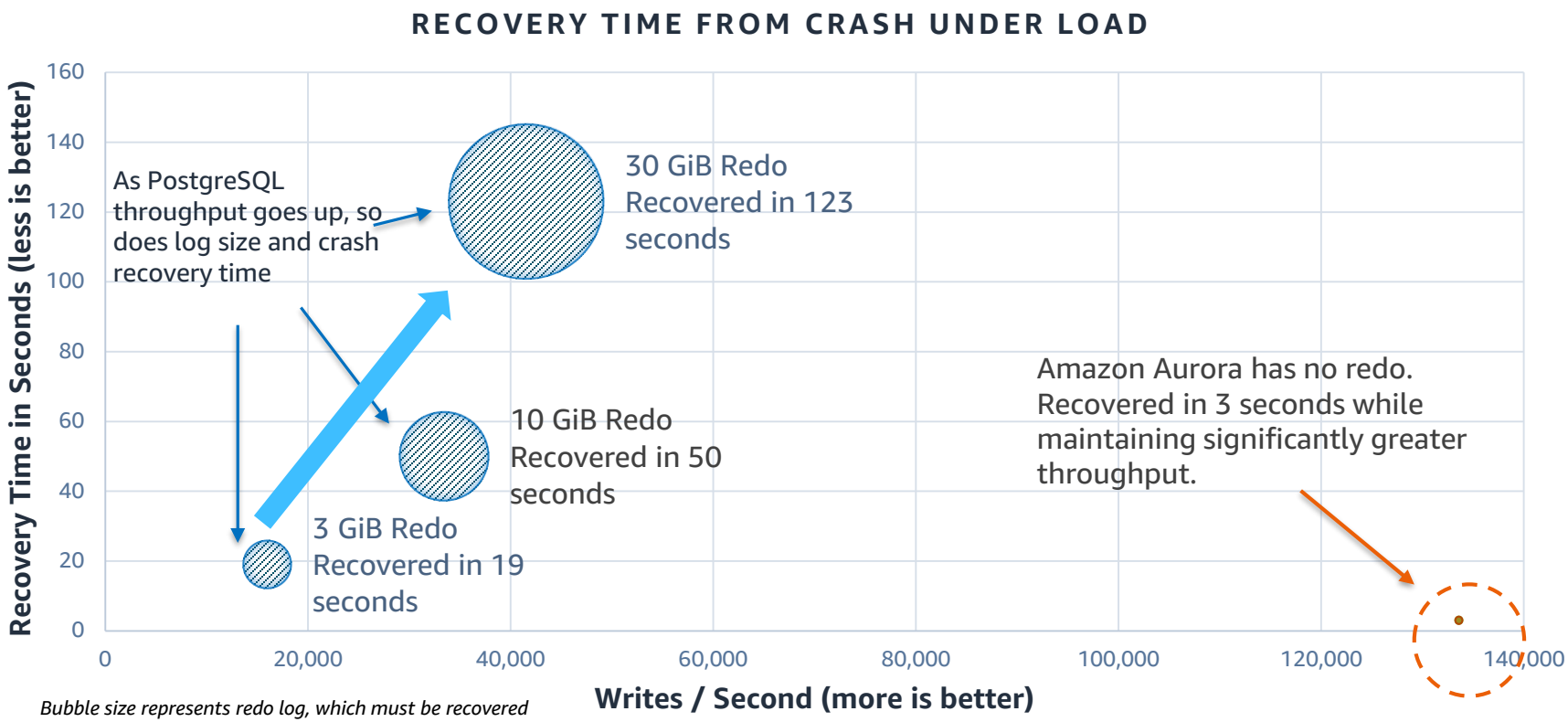
Insert Workload—PostgreSQL 9.6



aws



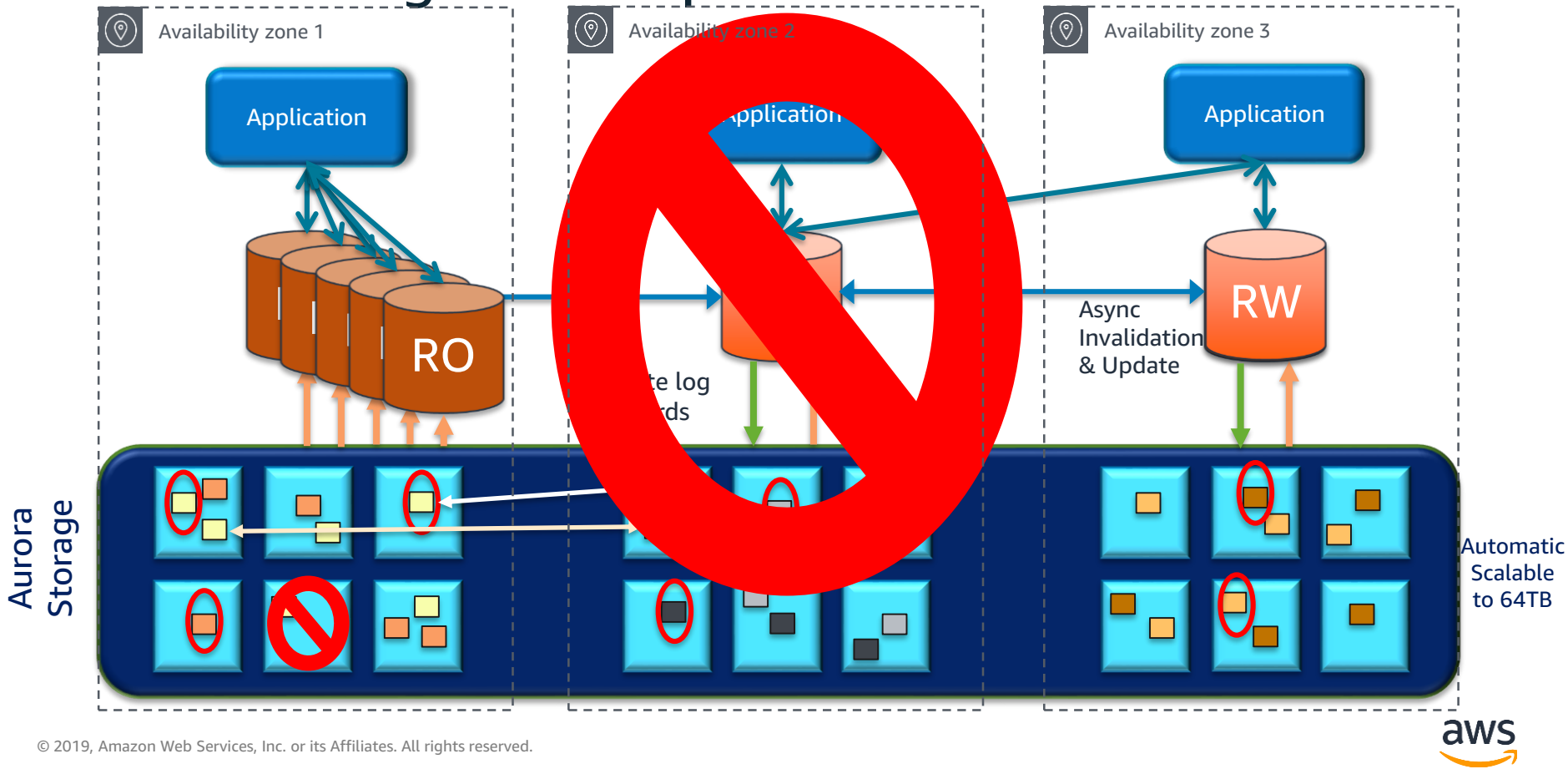
Amazon Aurora Recovers Up to 97%Faster



Bubble size represents redo log, which must be recovered

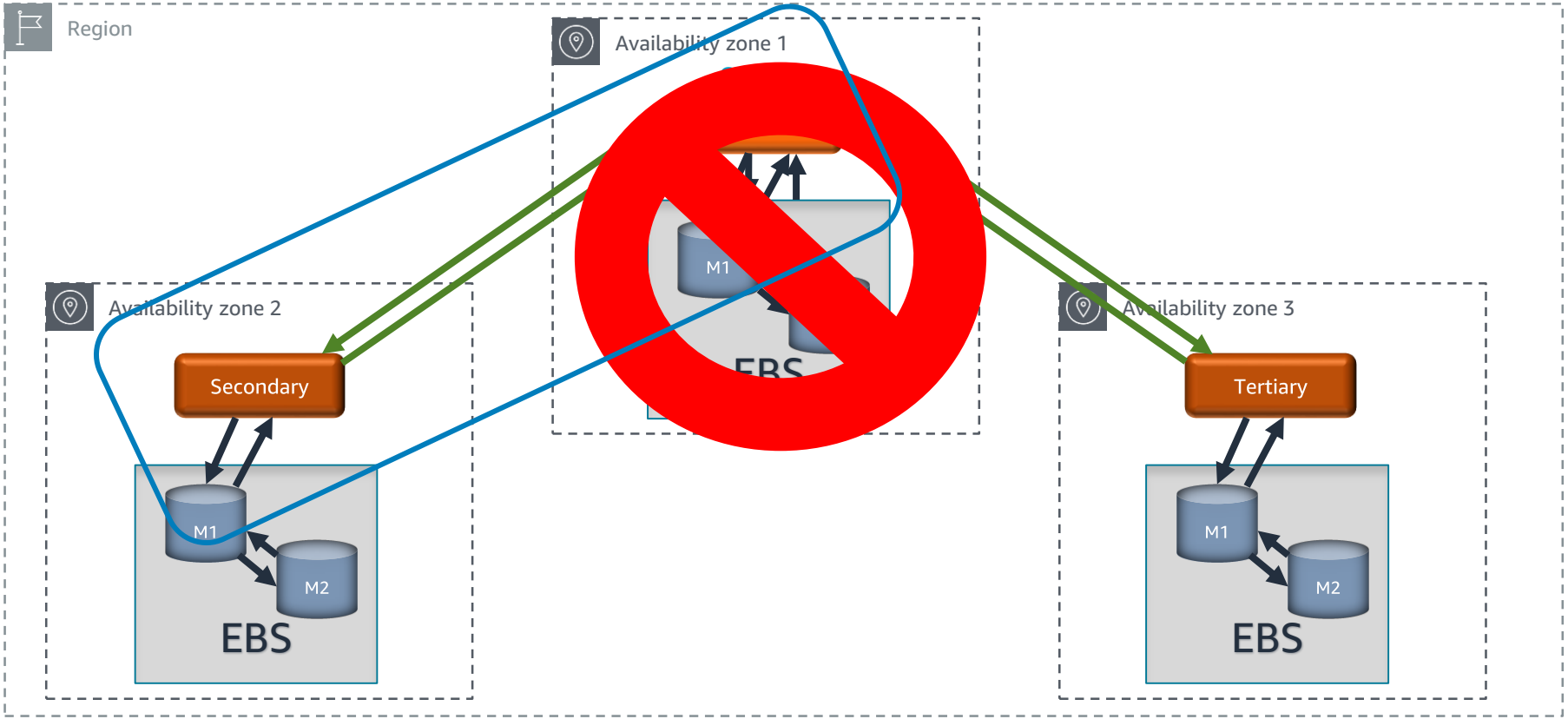
Base Architecture

Aurora Storage and Replicas

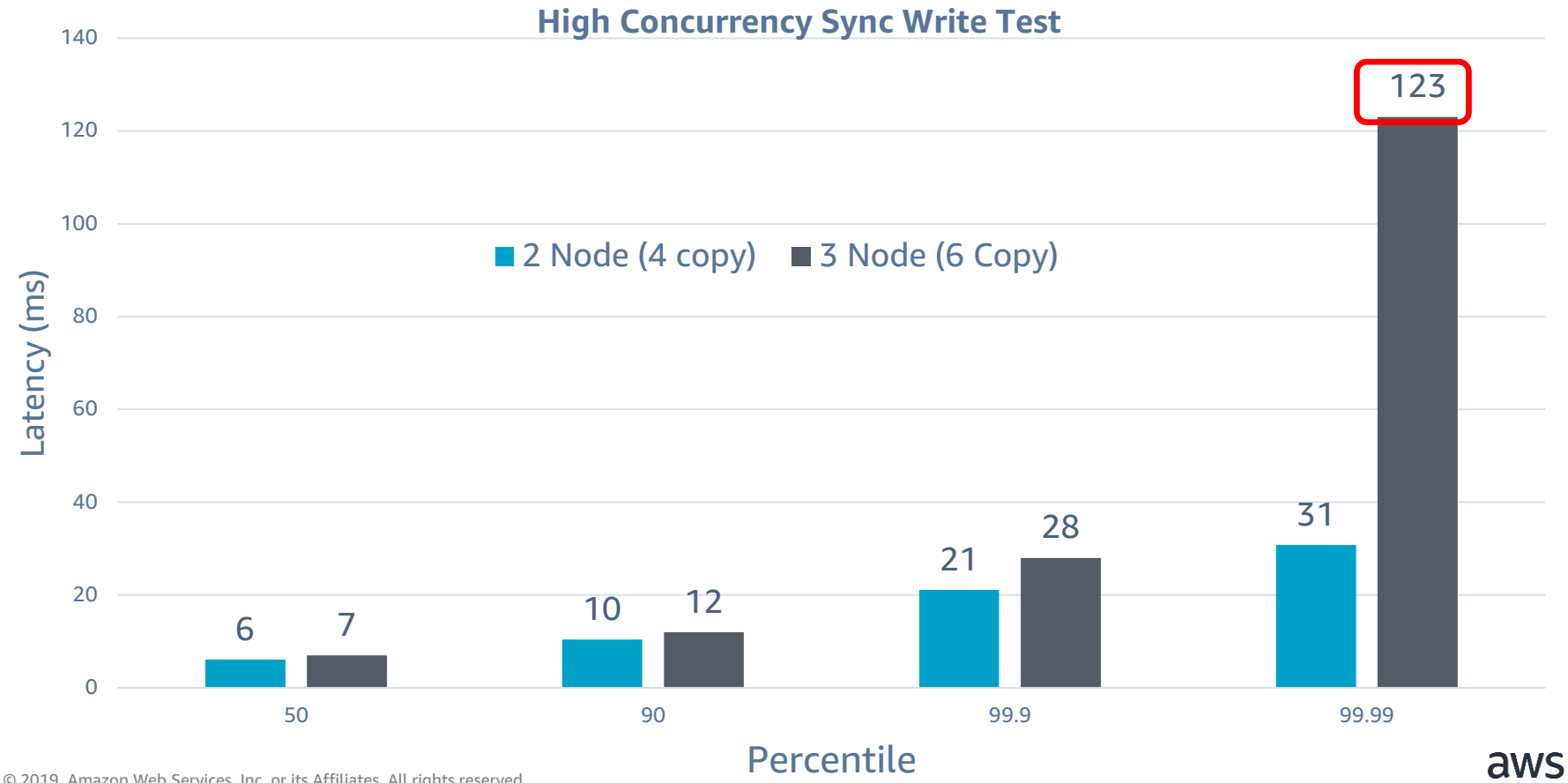


Durability—4 of 6 Quorum

Typical synchronous replication – 3 locations

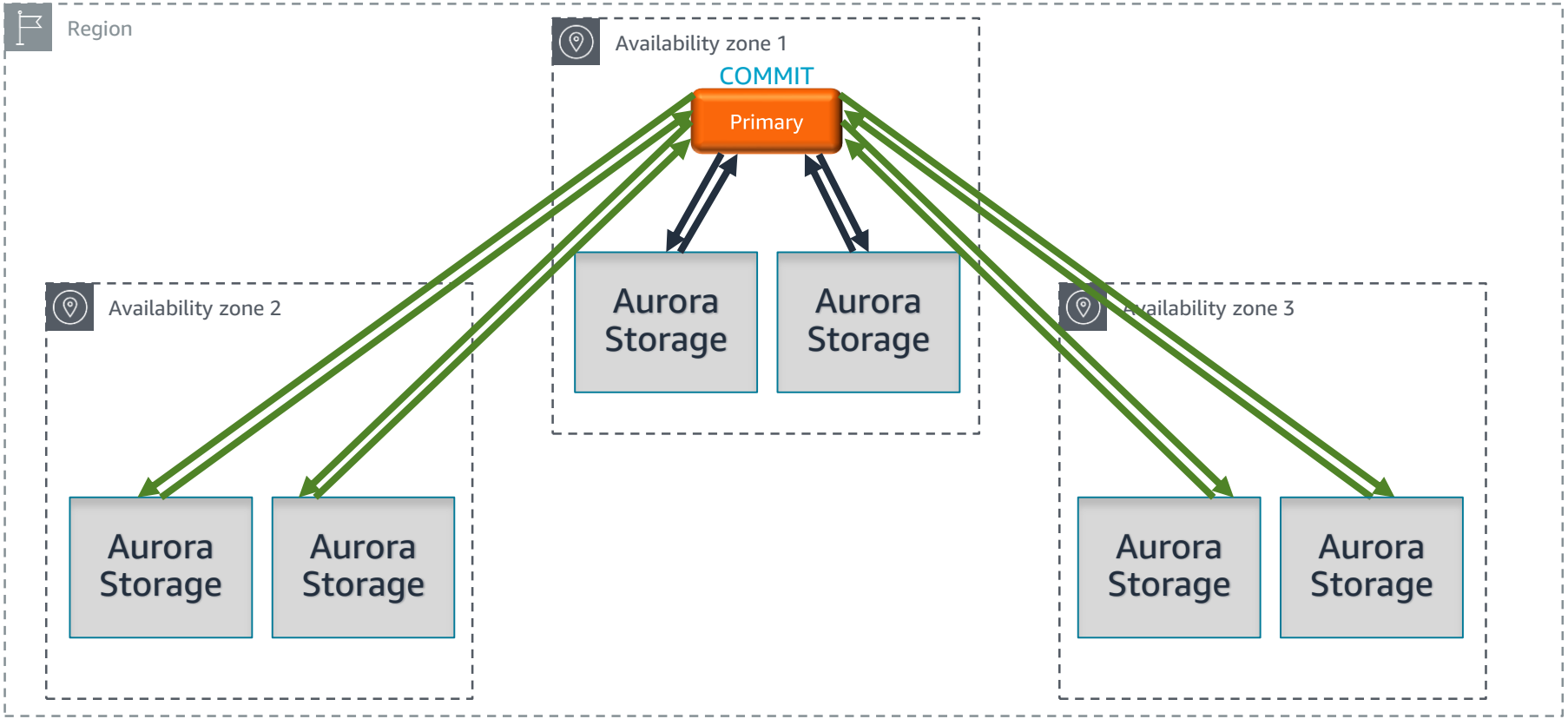


Cost of Additional Synchronous Replicas

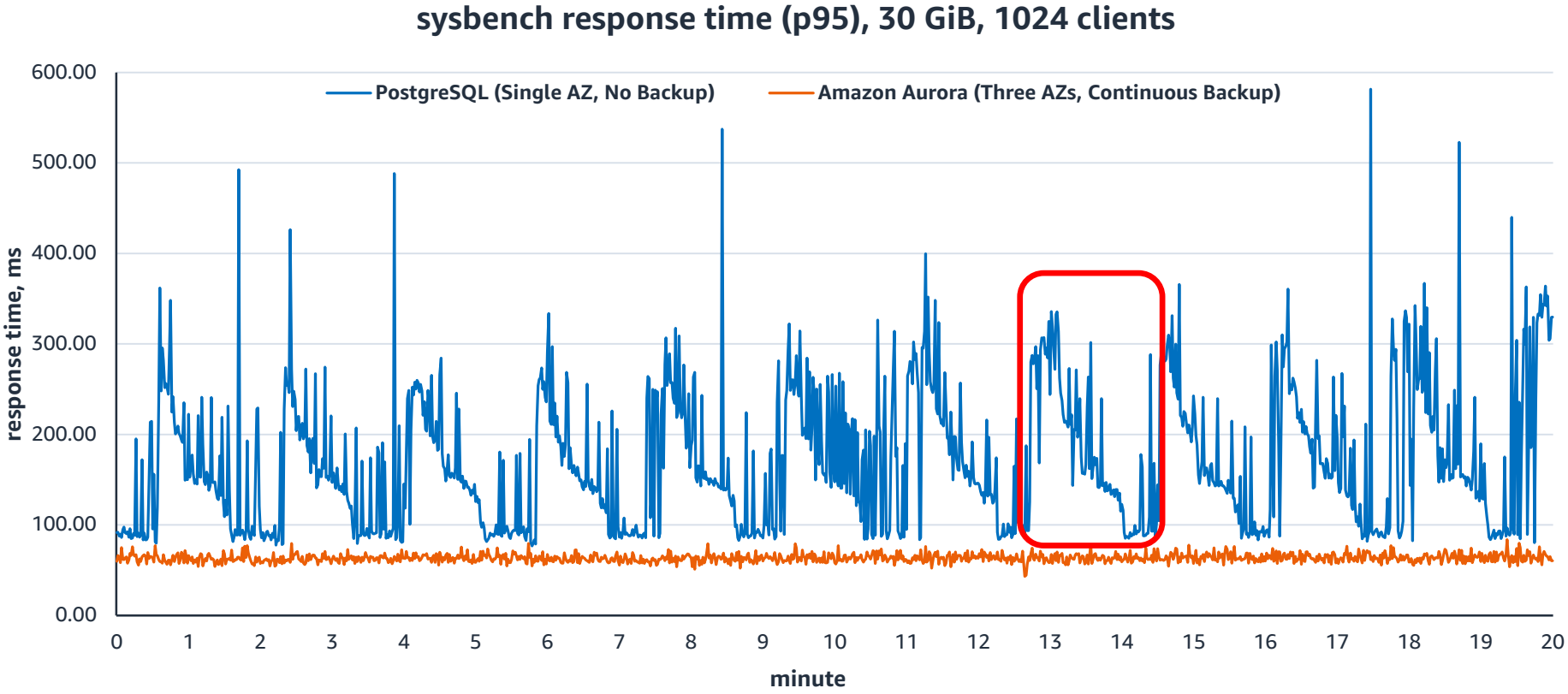


aws

Aurora – 3 AZ's – 6 copies

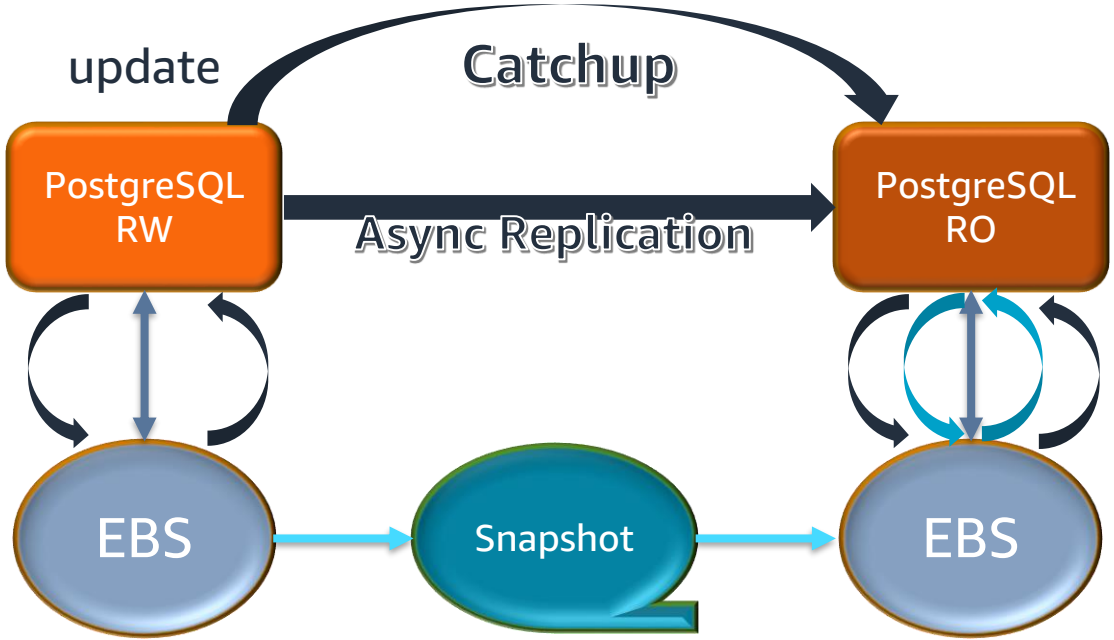


Amazon Aurora Gives >2x Lower Response Times



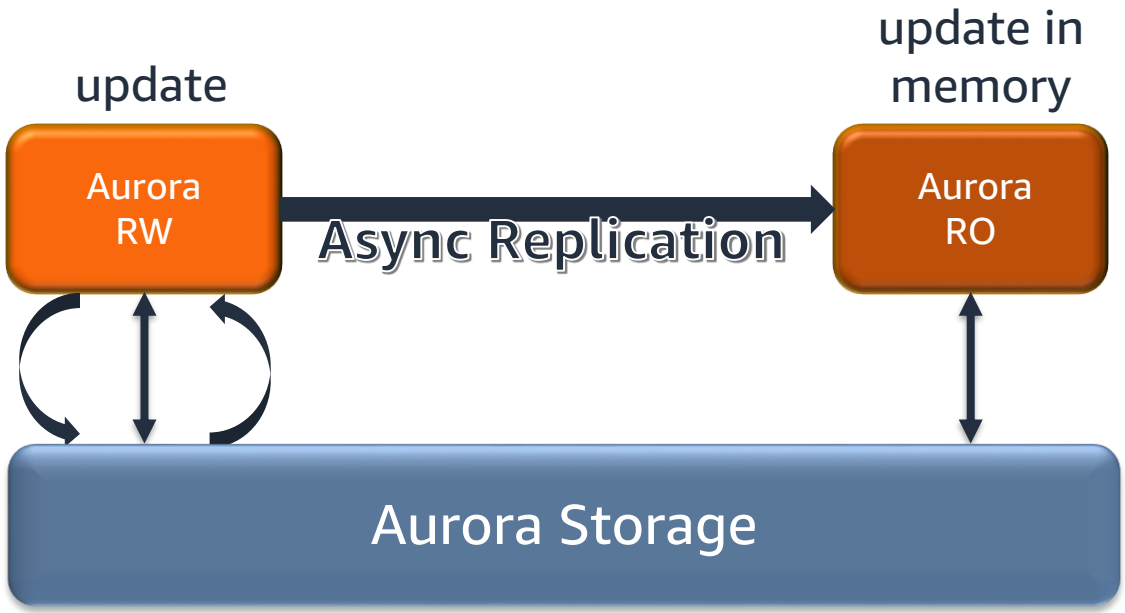
Replicas and Clones

Replicas—PostgreSQL

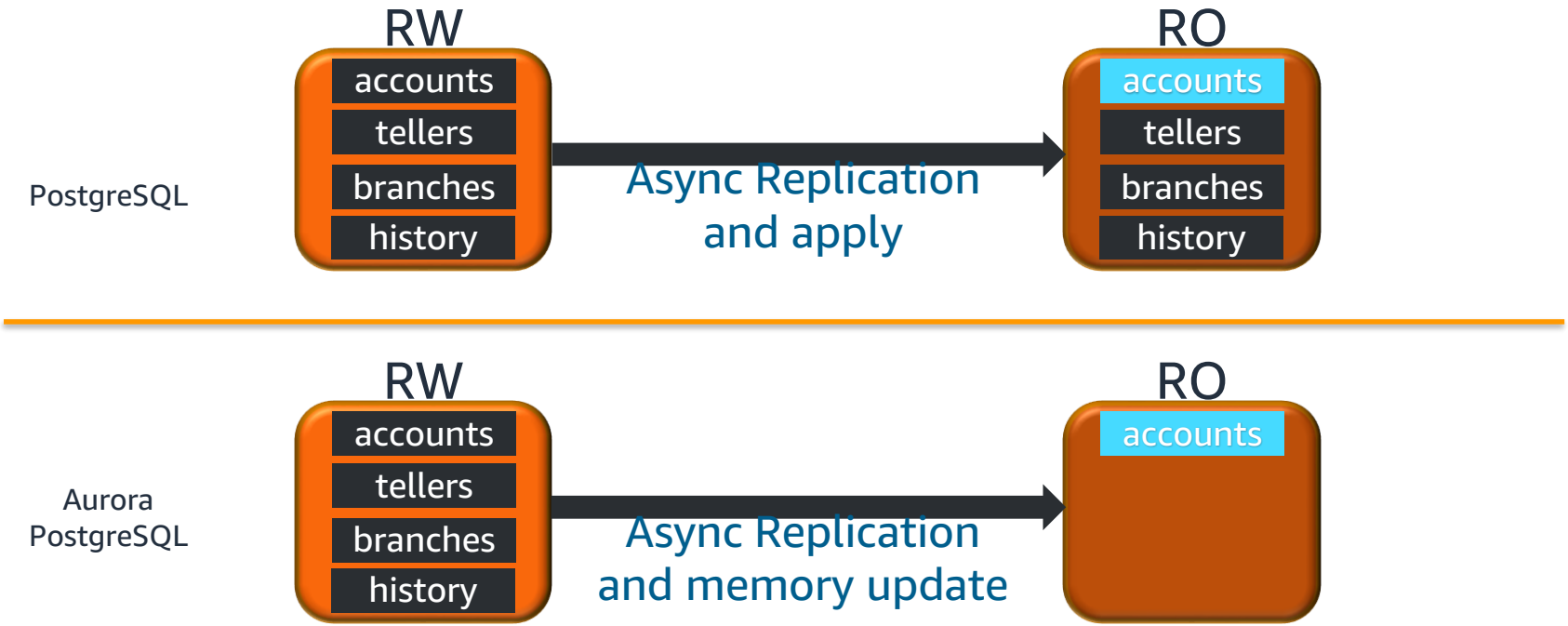


aws

Replicas—Amazon Aurora



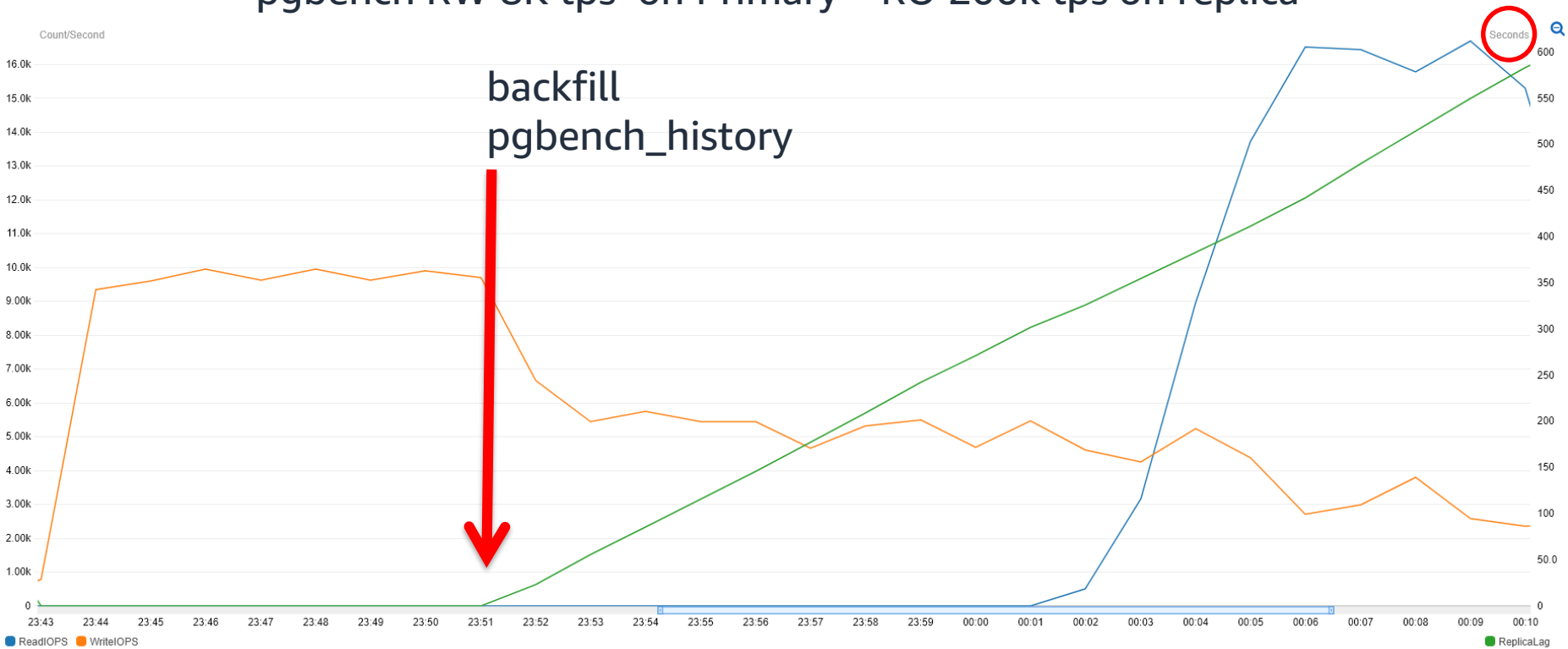
pgbench Benchmark



aws

Replicas—Backfill on PostgreSQL

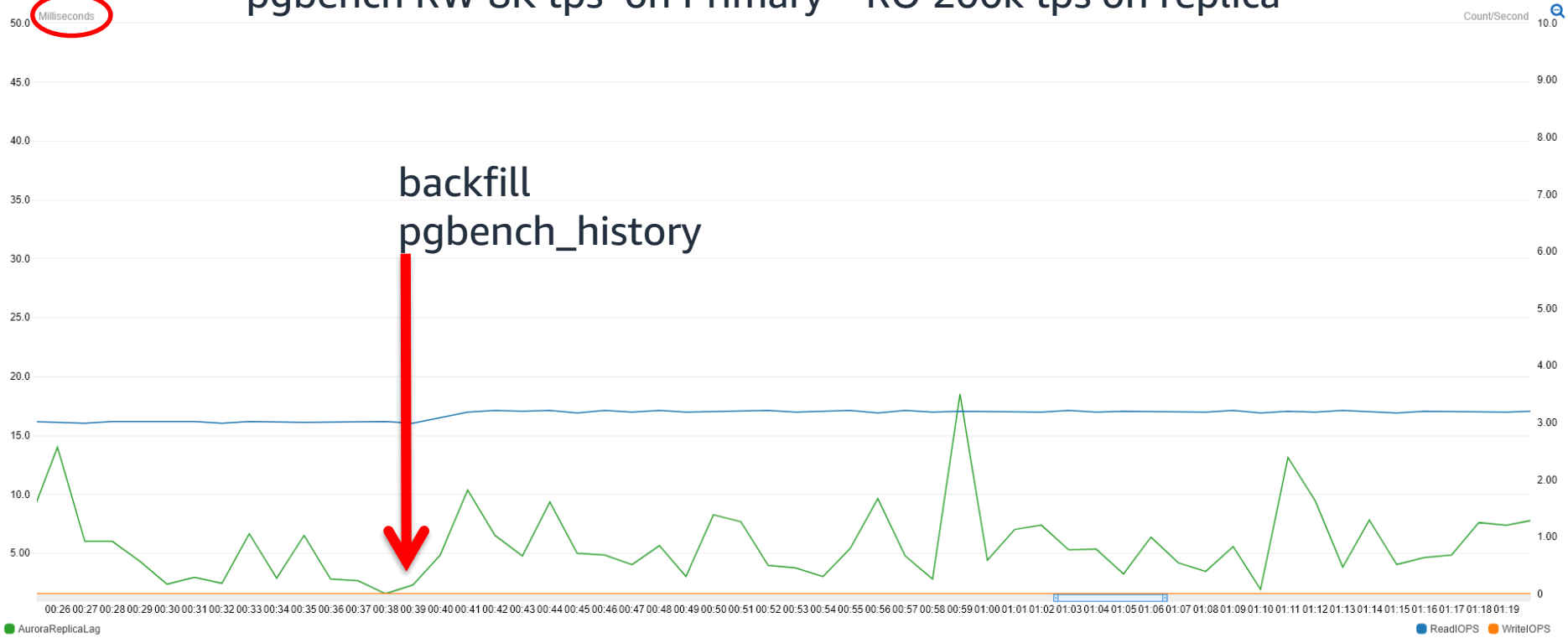
pgbench RW 8K tps on Primary – RO 200k tps on replica



aws

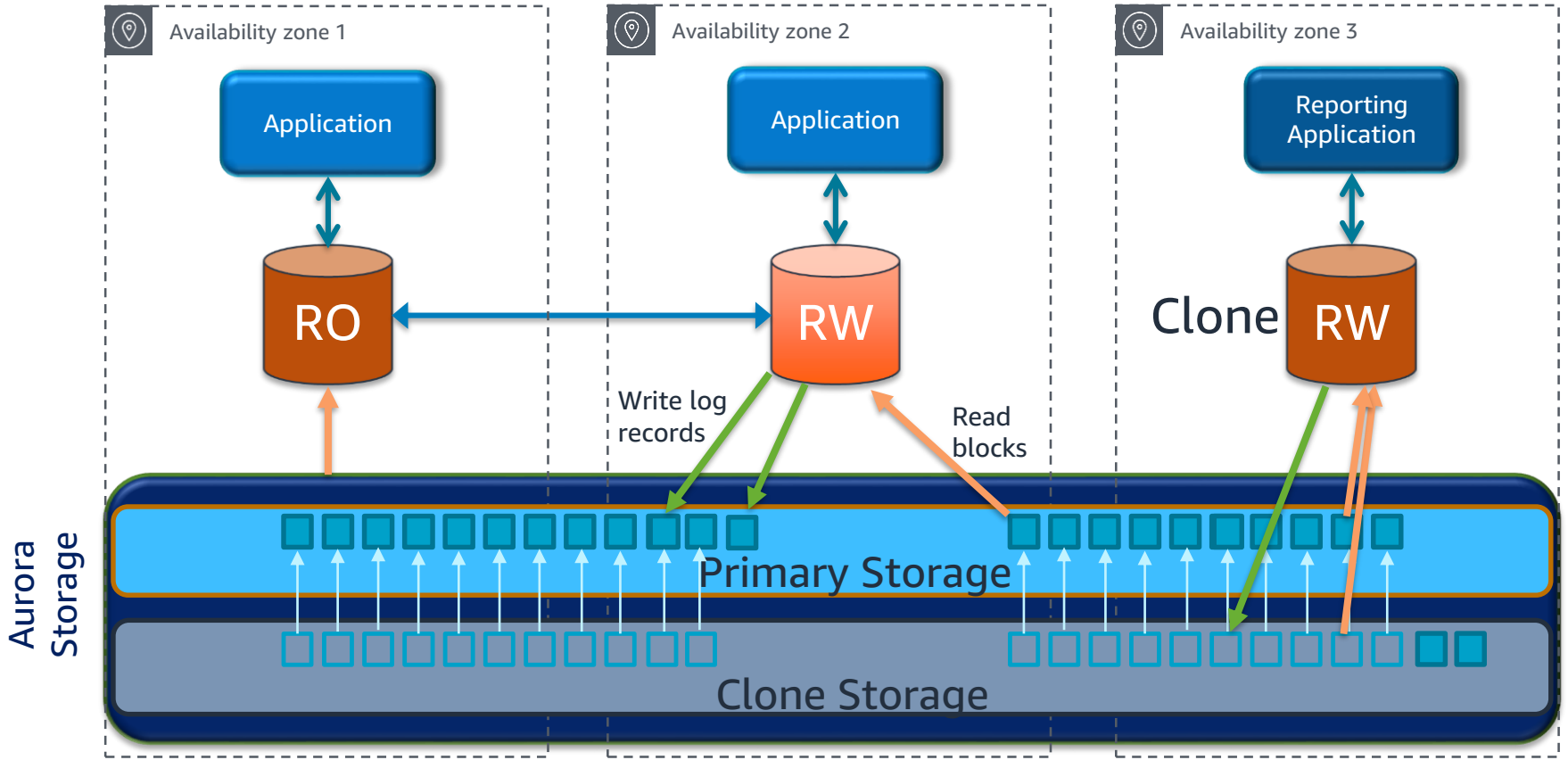
Replicas—Backfill on Amazon Aurora

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

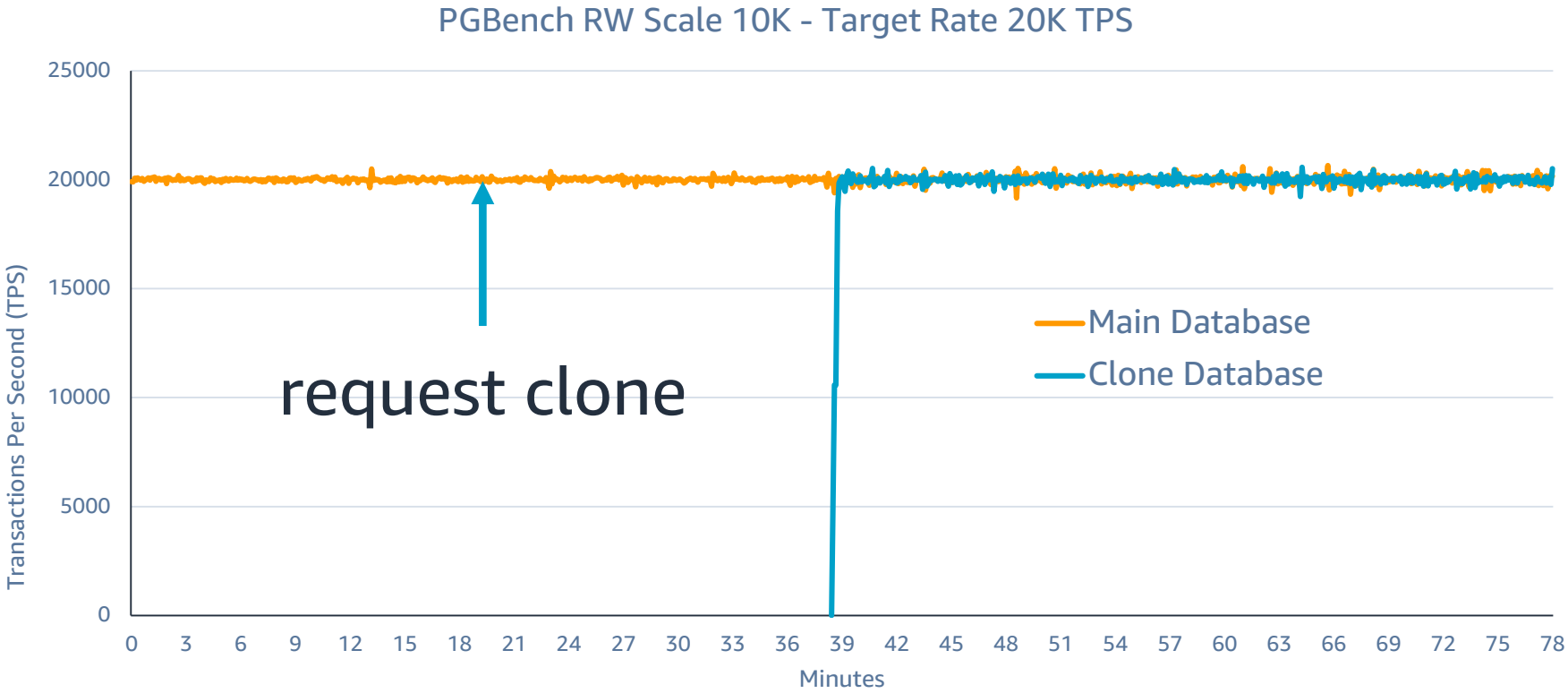


aws

Fast Clones



Fast clone example



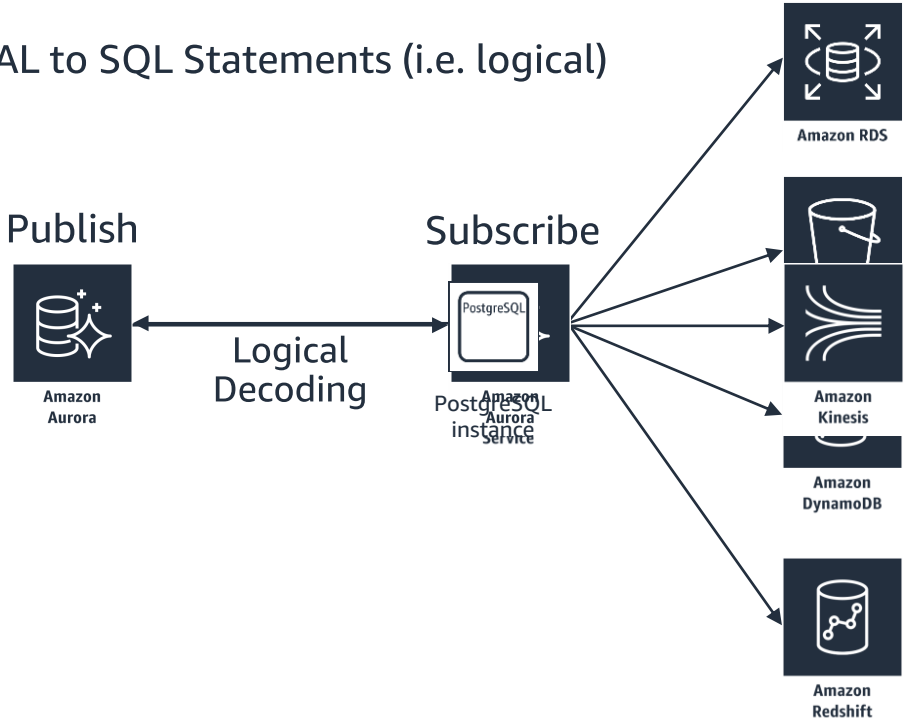
aws

Replication

Logical Replication Support

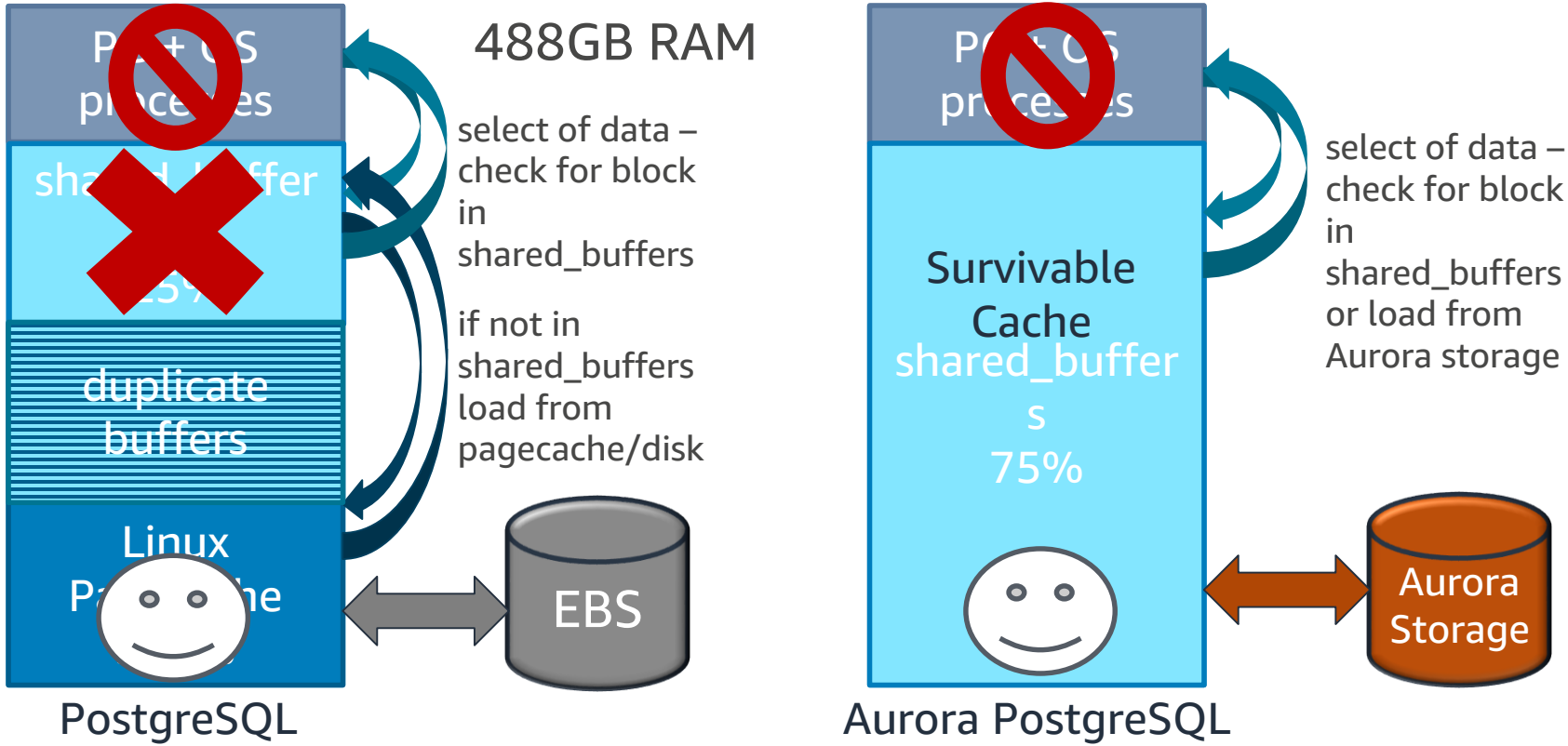
Converts Physical Changes (WAL to SQL Statements (i.e. logical)

- Logical Decoding Plugin
  (test_decoder, decoder_raw, wal2jason)
- V10 – Publish / Subscribe to another PostgreSQL Instance

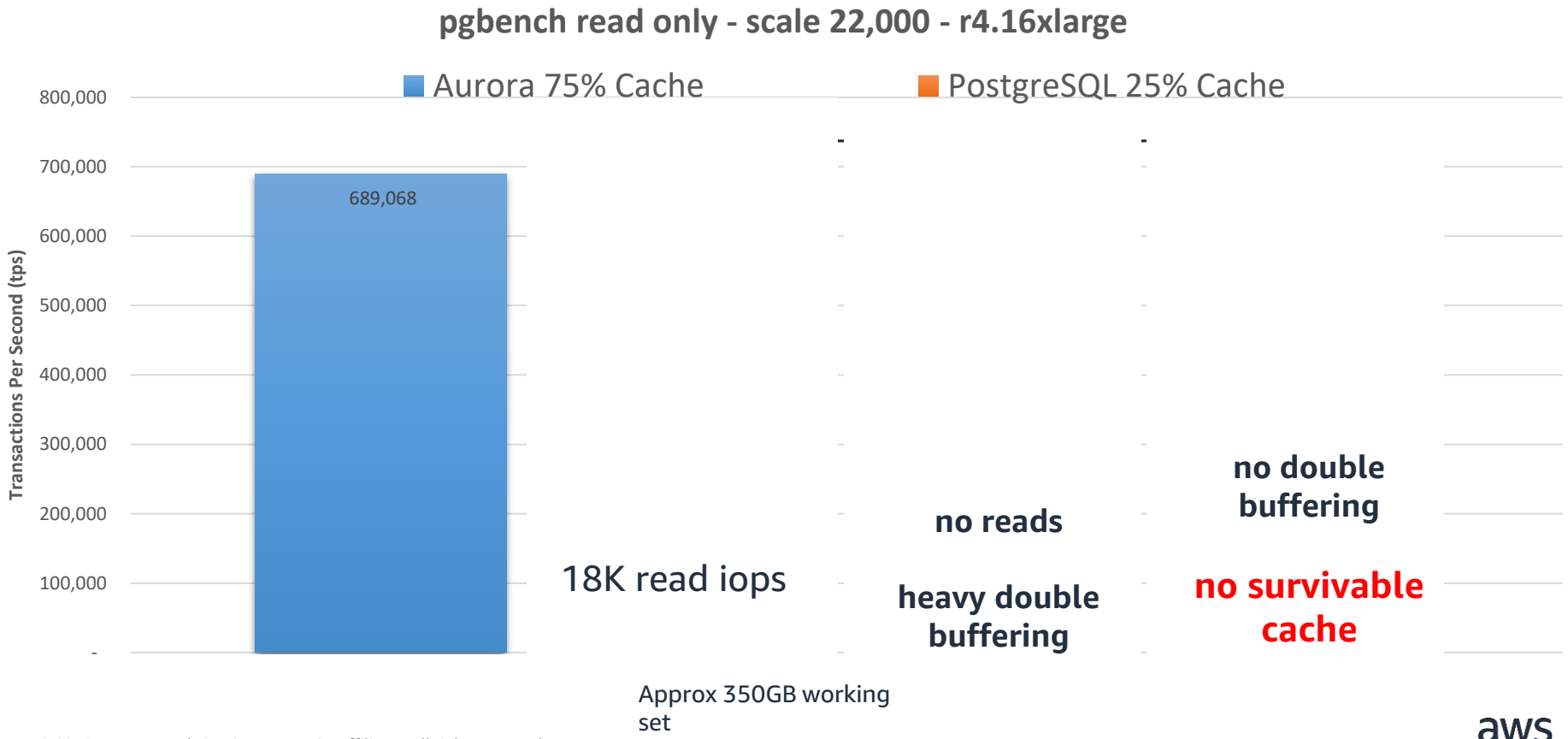


Caching

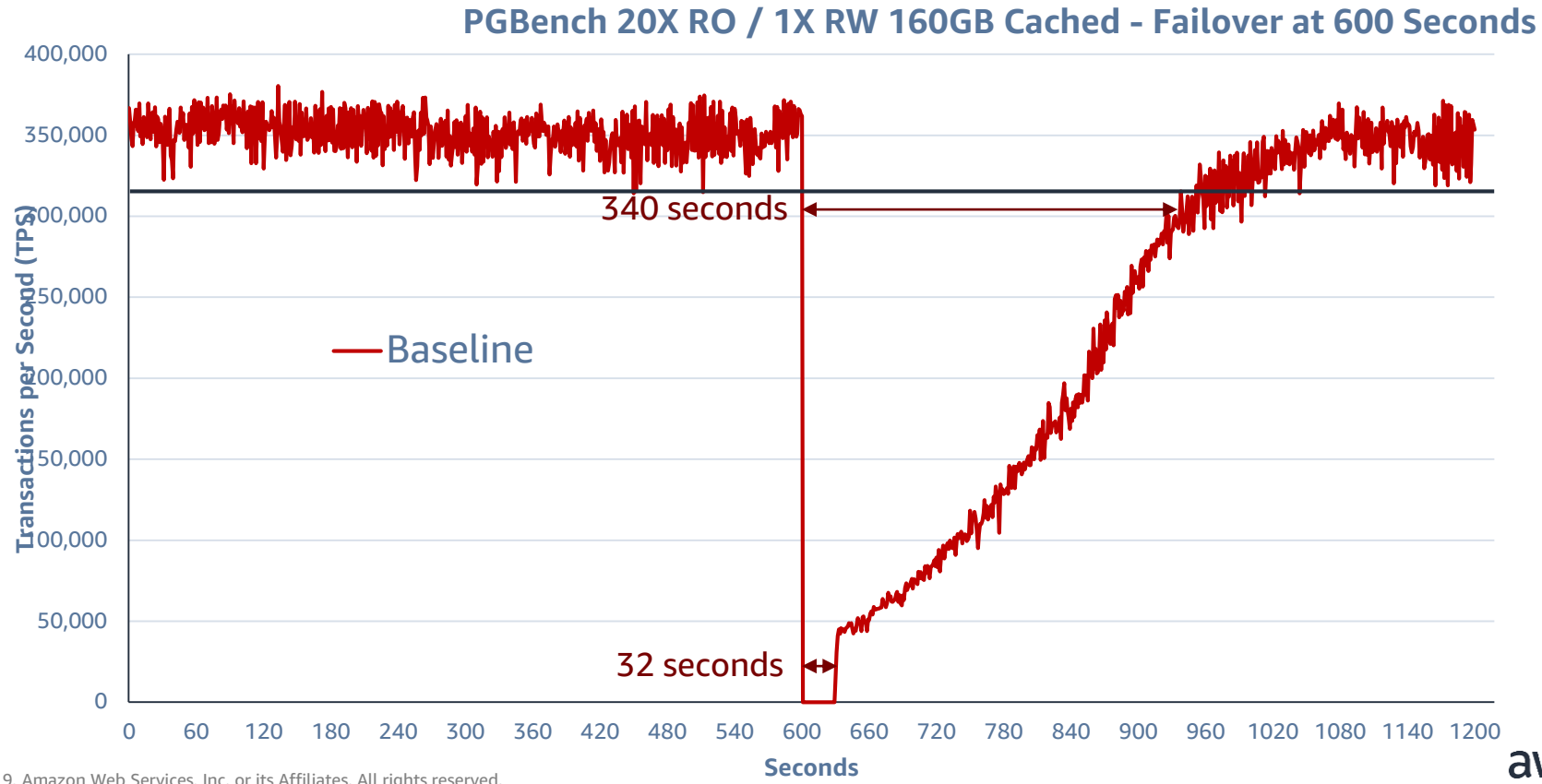
Caching Changes—No Double Buffering



Caching Changes—No Double Buffering

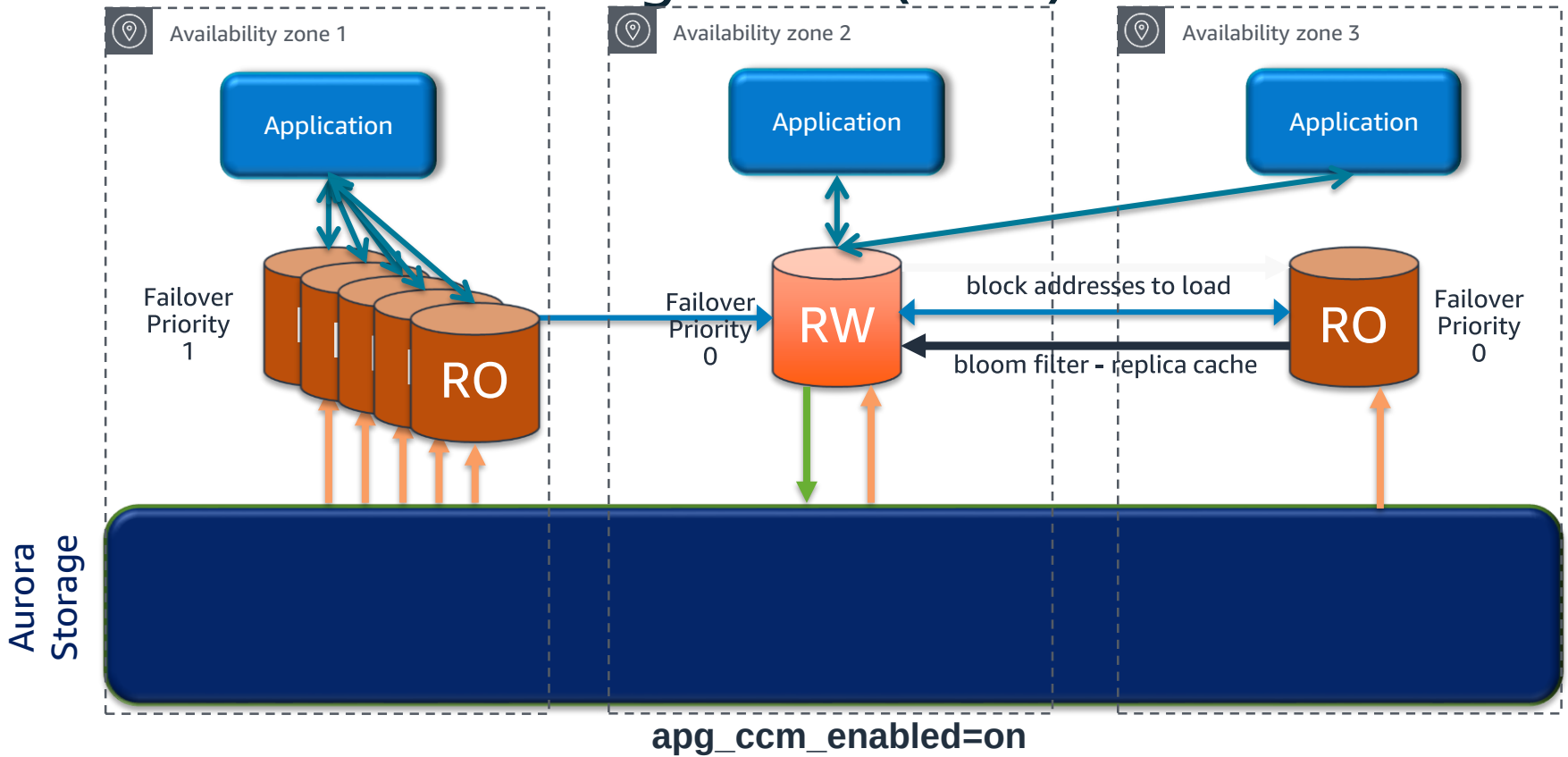


Cluster Cache Management - Failover

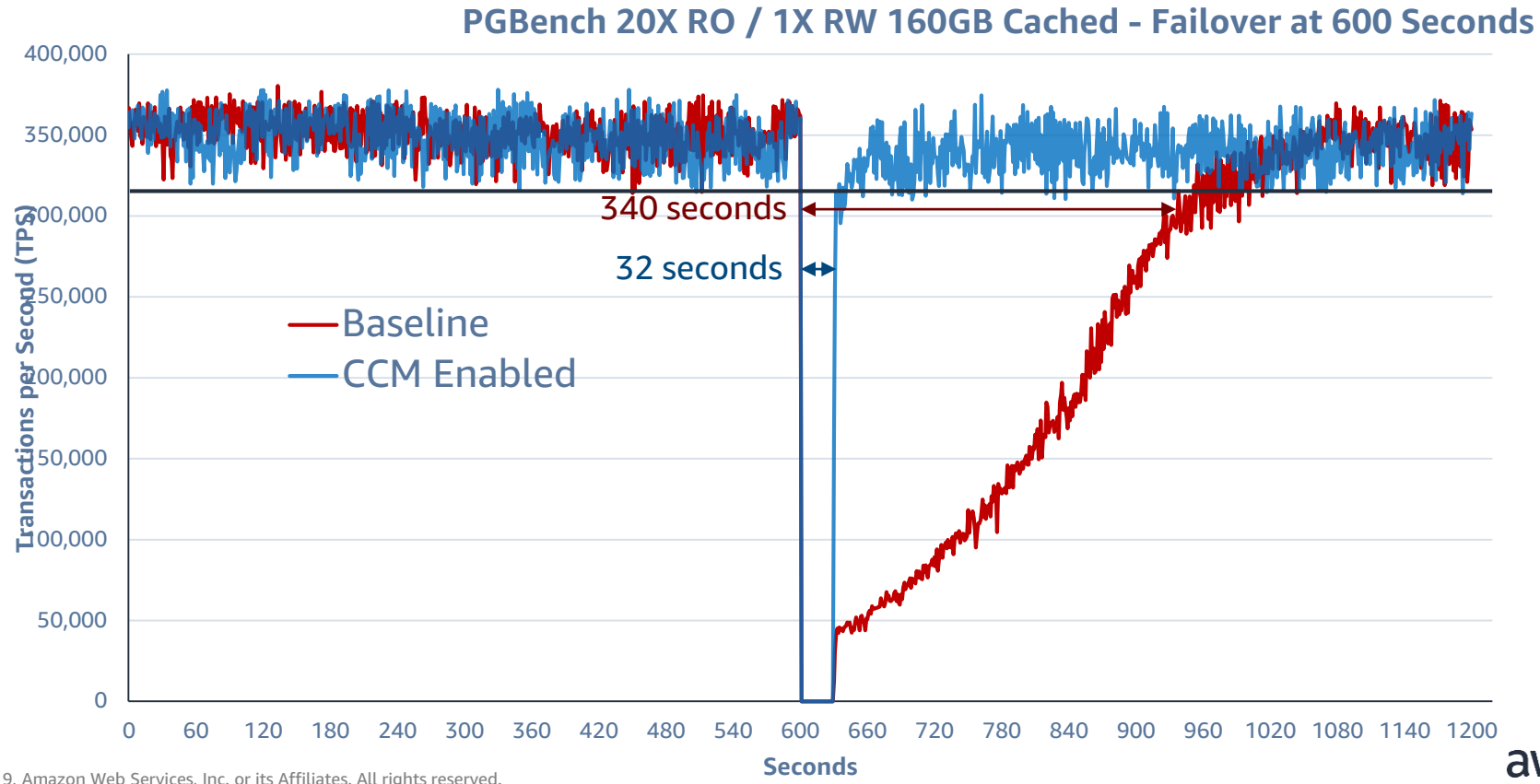


aws

Cluster Cache Management (CCM) Feature

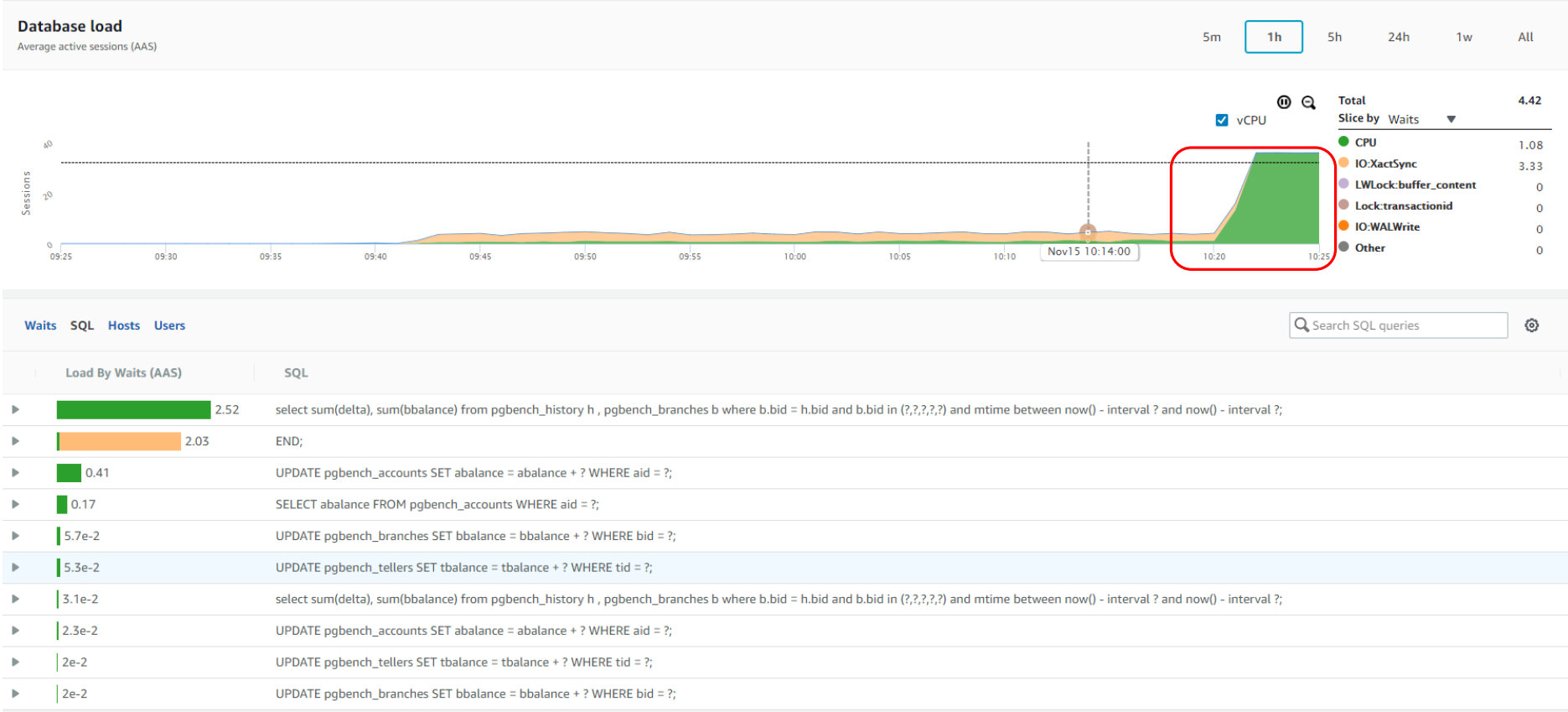


Cluster Cache Management

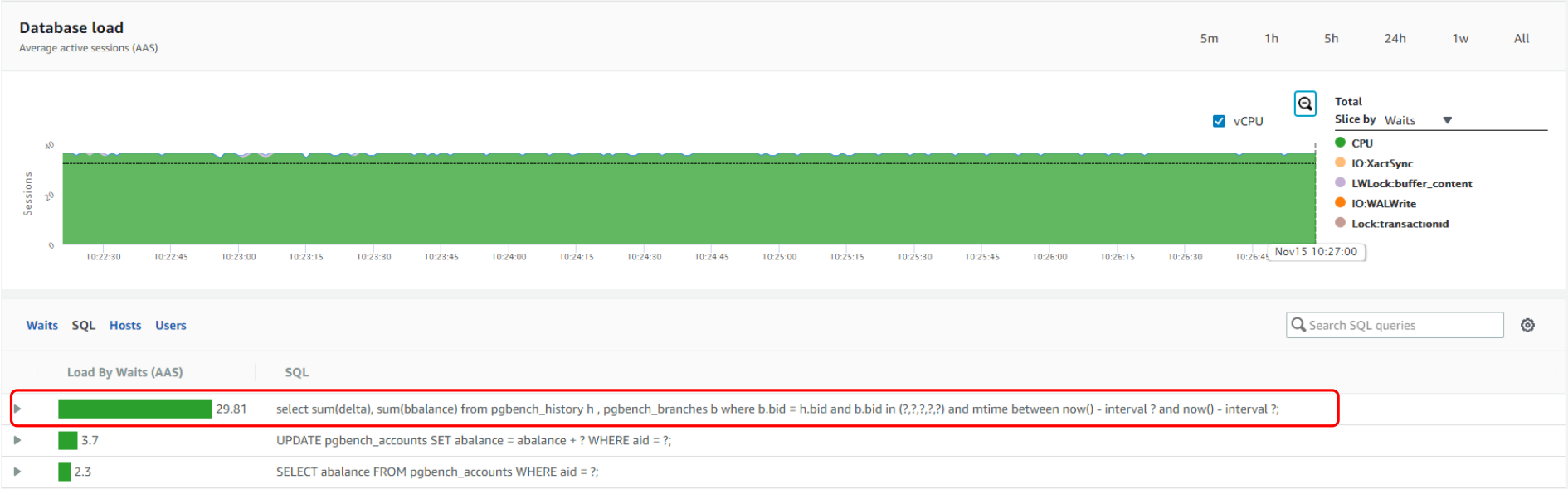


Performance

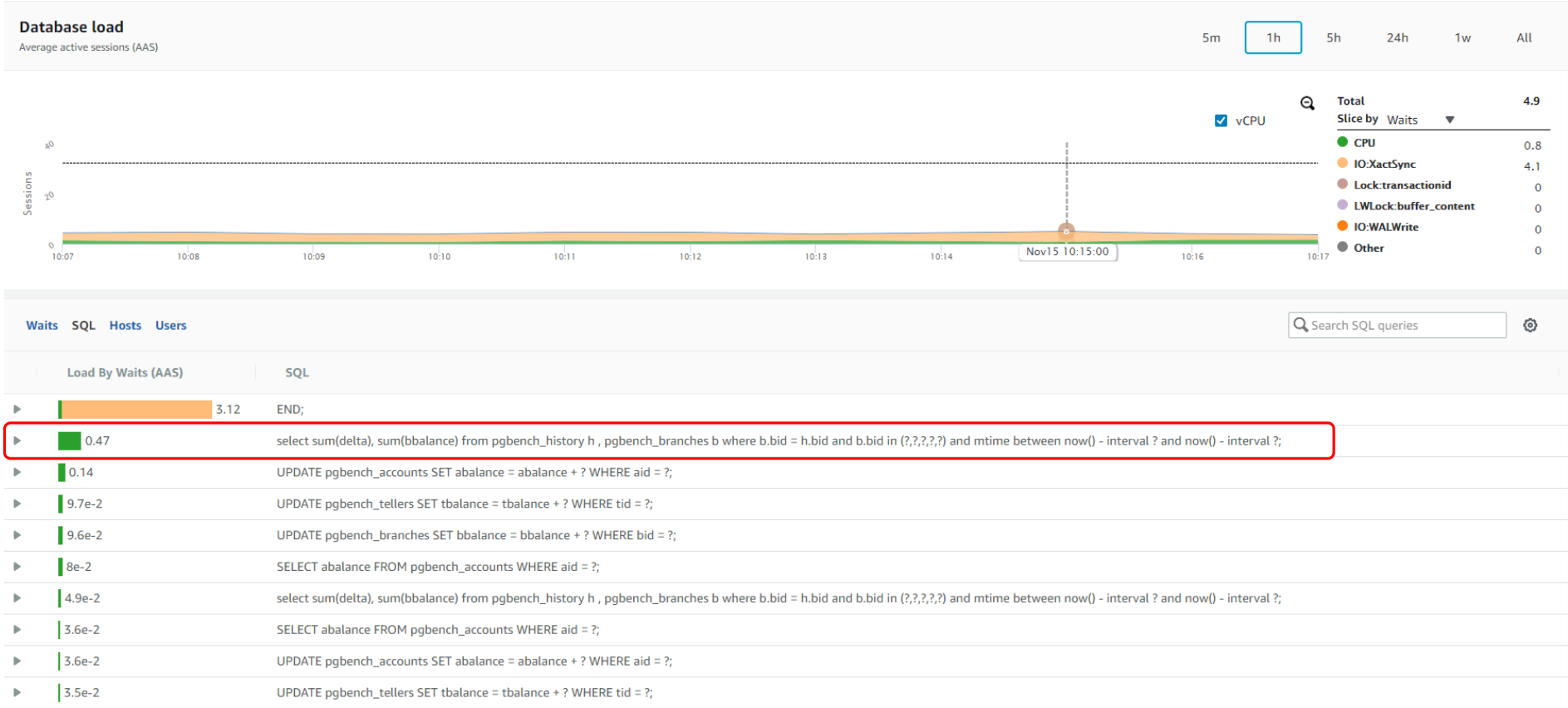
Performance Insights



Performance Insights



Performance Insights



Plan Change

- stats change?
- config change?
- index 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=0)
                     Index Cond: (bid = b.bid)
```

- enable_bitmapscan=off
- enable_indexscan=off

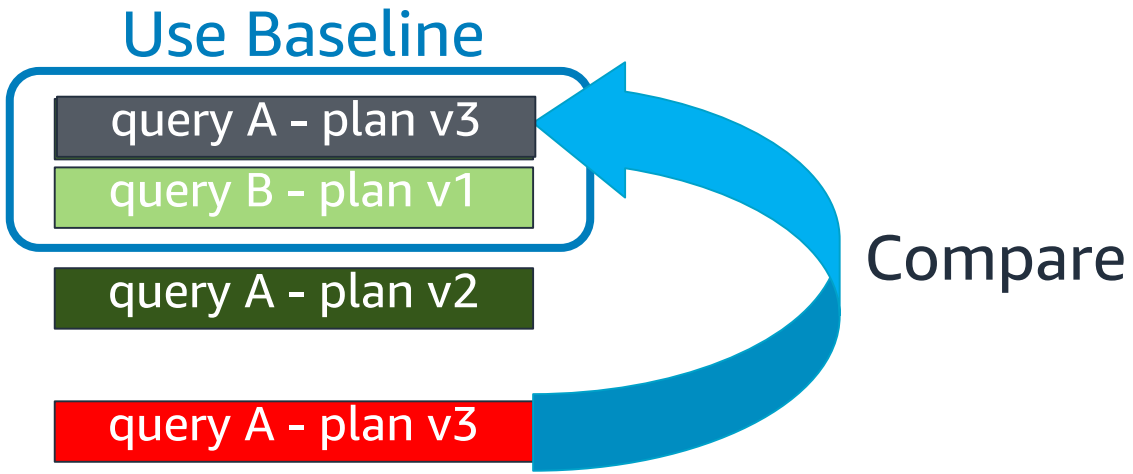
After

```
 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[]))
```


aws

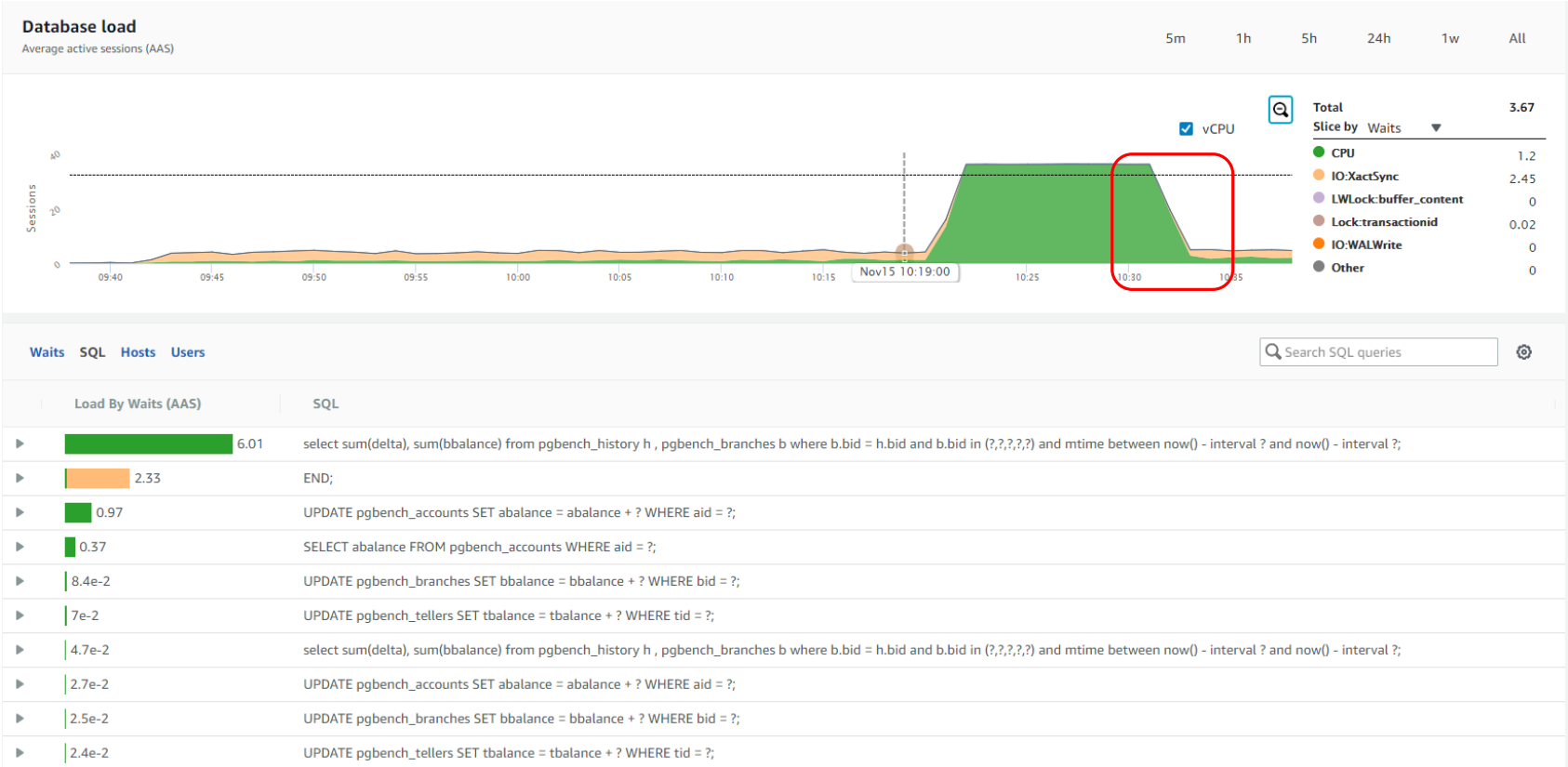
Query Plan Management - QPM

- Capture statements
- Approve statements
- Evolve better plans



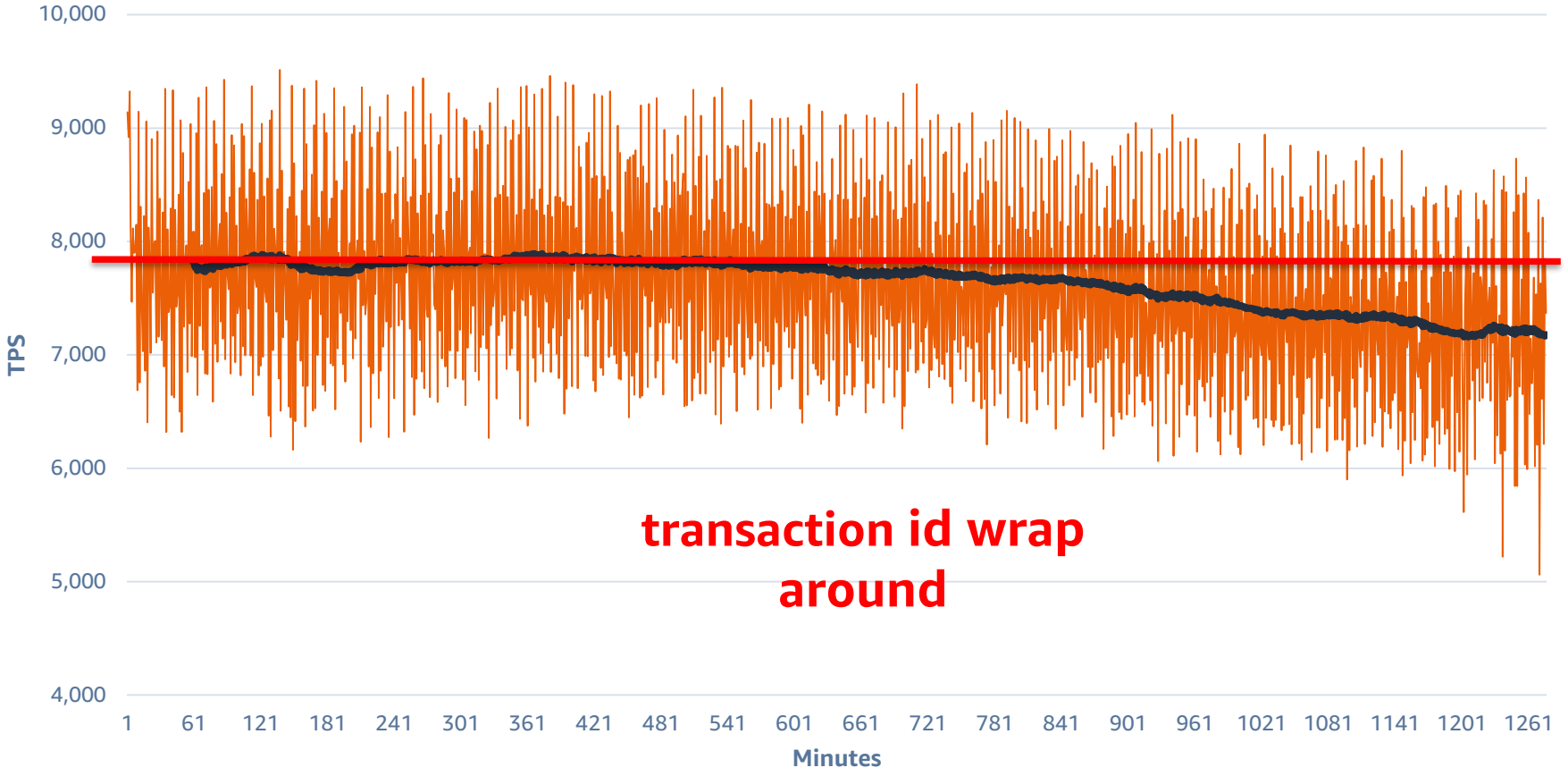
aws

QPM – Use Plan Baselines



Vacuuming

Updates—No Vacuum Running

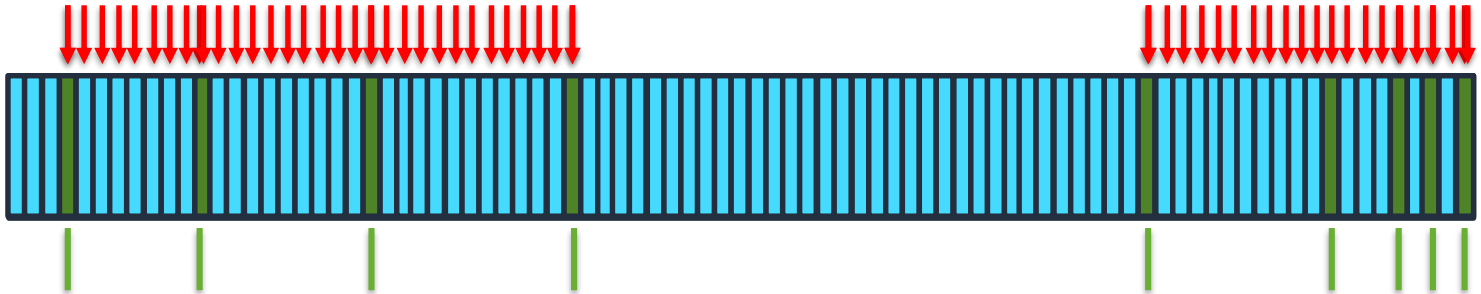


aws

Intelligent Vacuum Prefetch

PostgreSQL 402 seconds

visibility & frozen map



Aurora PostgreSQL 163 seconds

Submit Batch I/O up to 256 blocks

Thank you!