



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



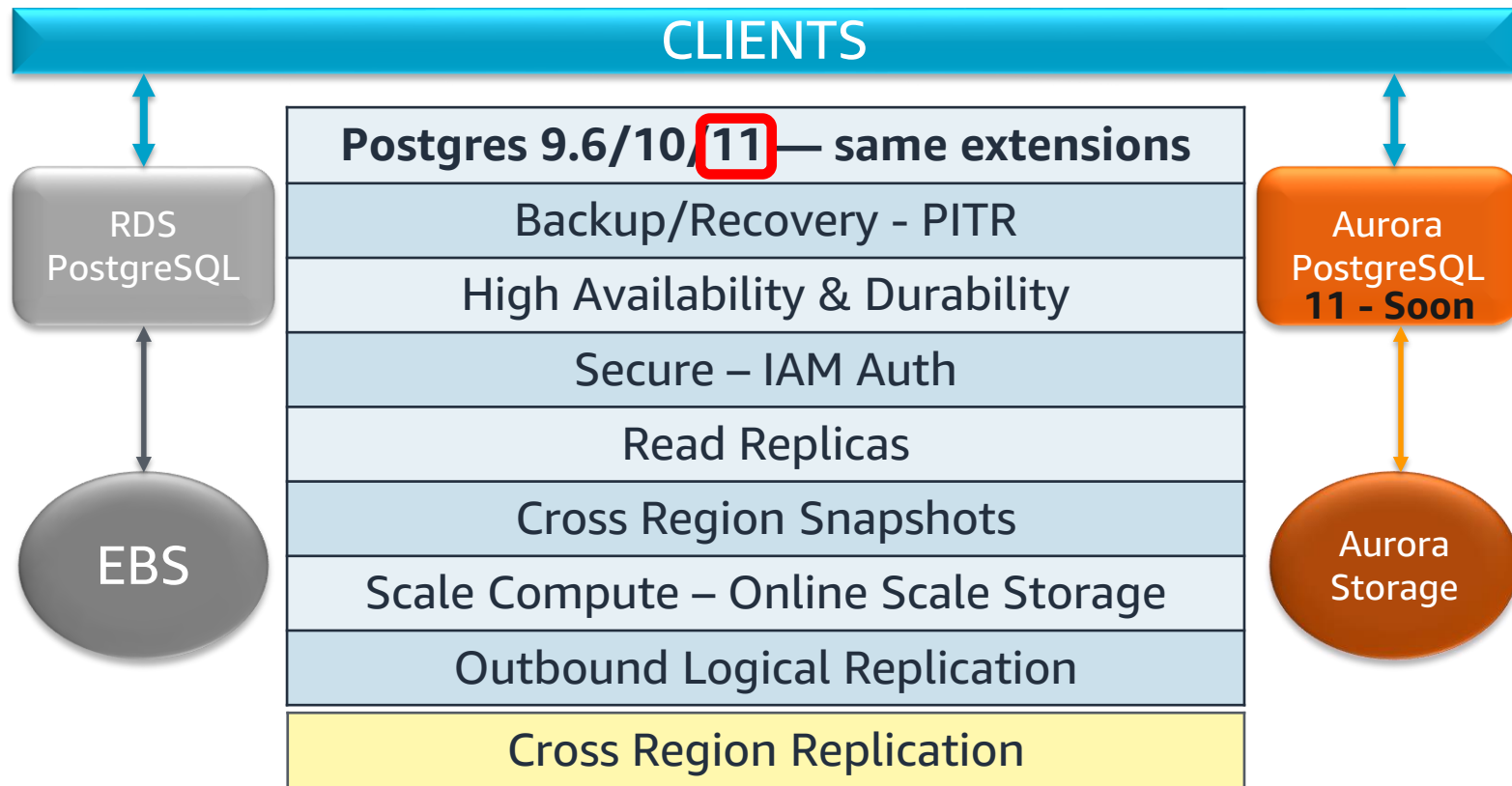
Commercial engines



RDS platform

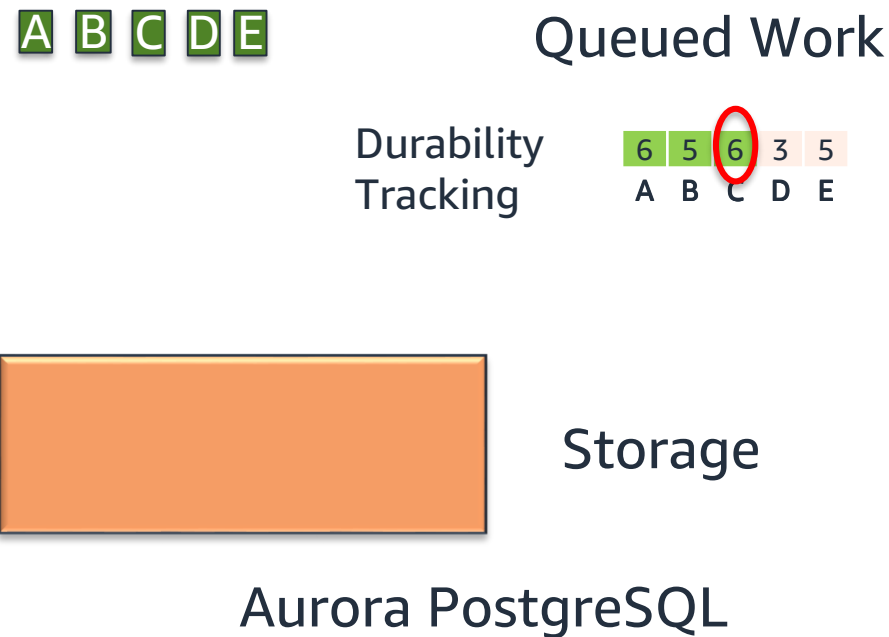
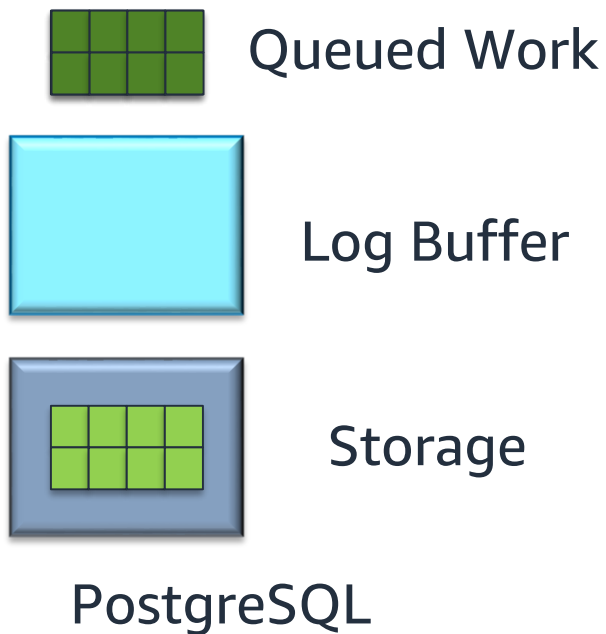
- Automatic fail-over
- Backup & recovery
- X-region replication
- Isolation & security
- Industry compliance
- Automated patching
- Advanced monitoring
- Routine maintenance
- Push-button scaling

RDS PostgreSQL Universe



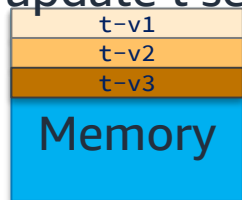
Log-Based Storage

Concurrency—Remove Log Buffer

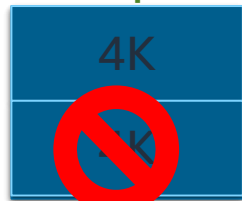


Aurora PostgreSQL—Writing Less

update t set y = 6;



checkpoint



datafile



WAL

archive

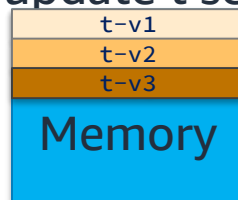


PostgreSQL



Amazon S3

update t set y = 6;



no
checkpoint
=
no FPW



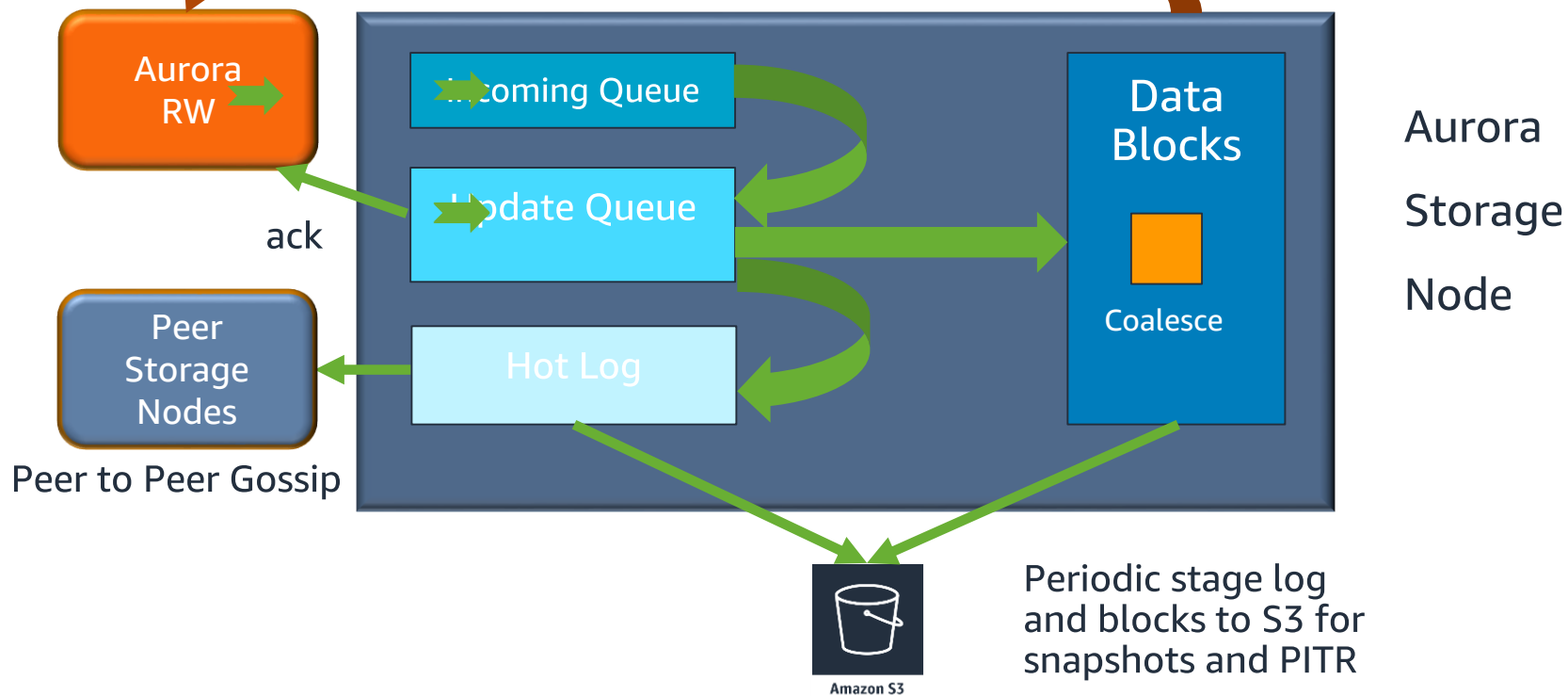
Aurora
Storage



Aurora

Storage Layer

Read Request



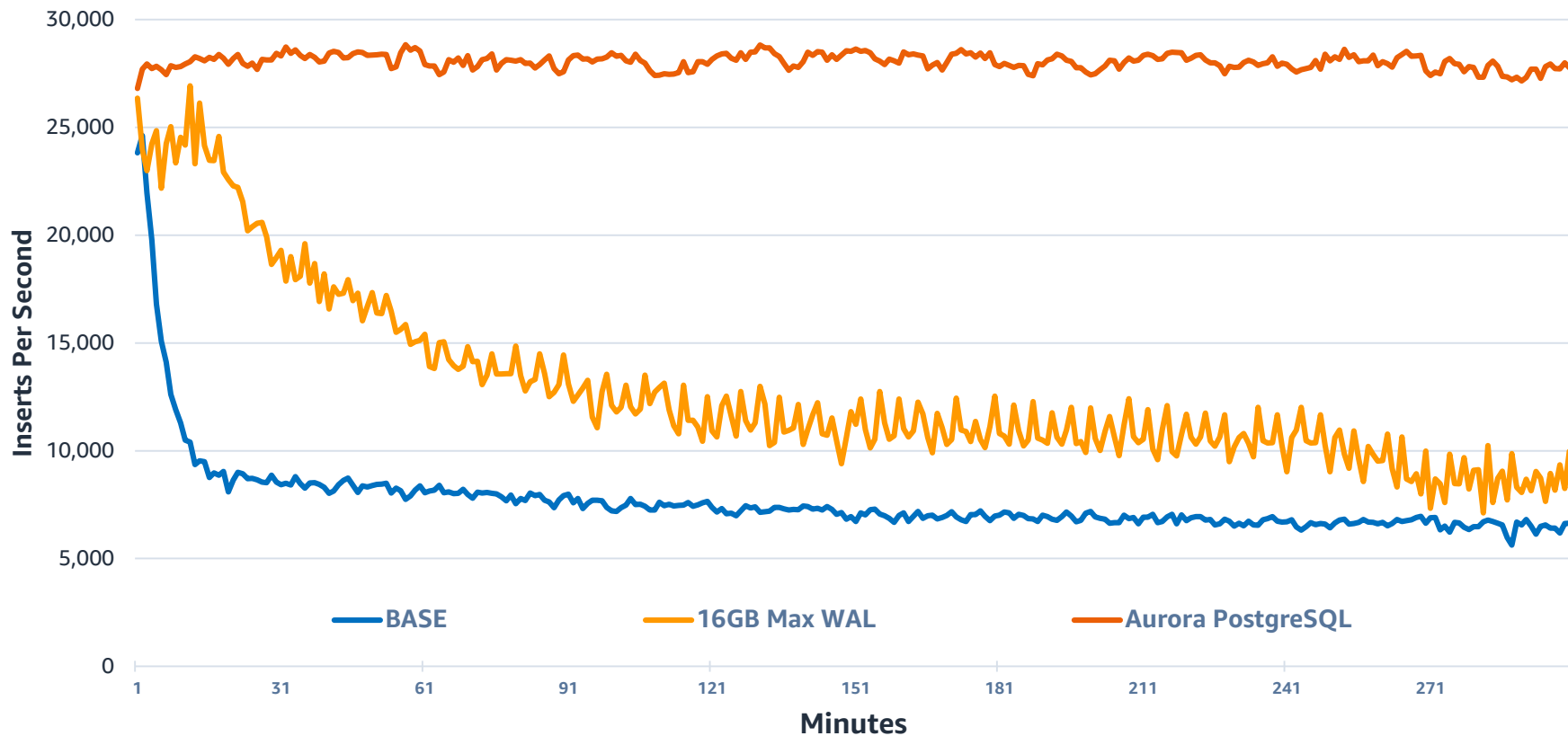
Insert Test

Test Table

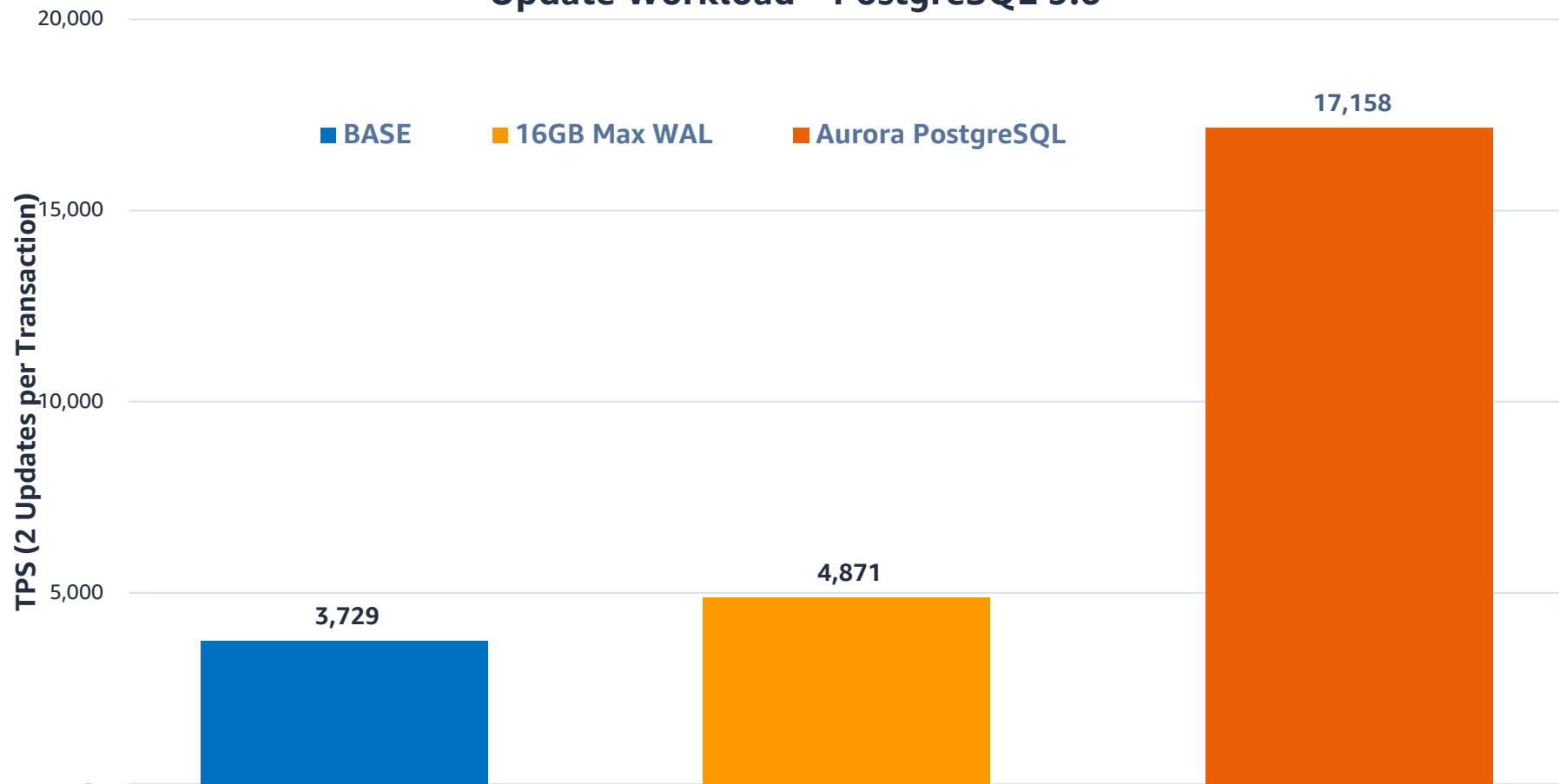
- UUID PK—Random
- ID int—Right Lean Sequence
- VARCHAR(100)—Random
- VARCHAR(50)—Small Set of Words
- INT—Random
- INT—Random (smaller set)
- BOOLEAN—Random (50/50)
- BOOLEAN—Somewhat Random (75/25)
- Timestamp—Right Lean

Index Every Column

Insert Workload—PostgreSQL 9.6

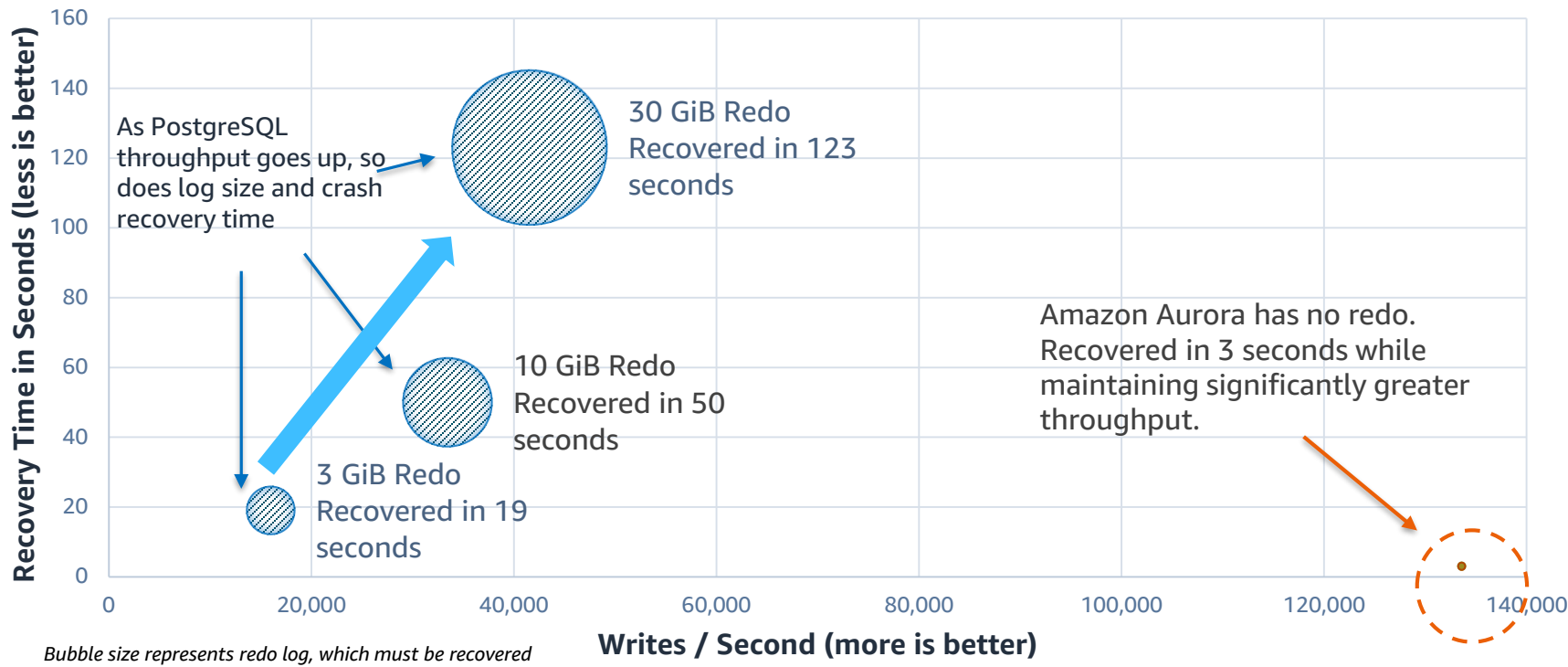


Update Workload—PostgreSQL 9.6



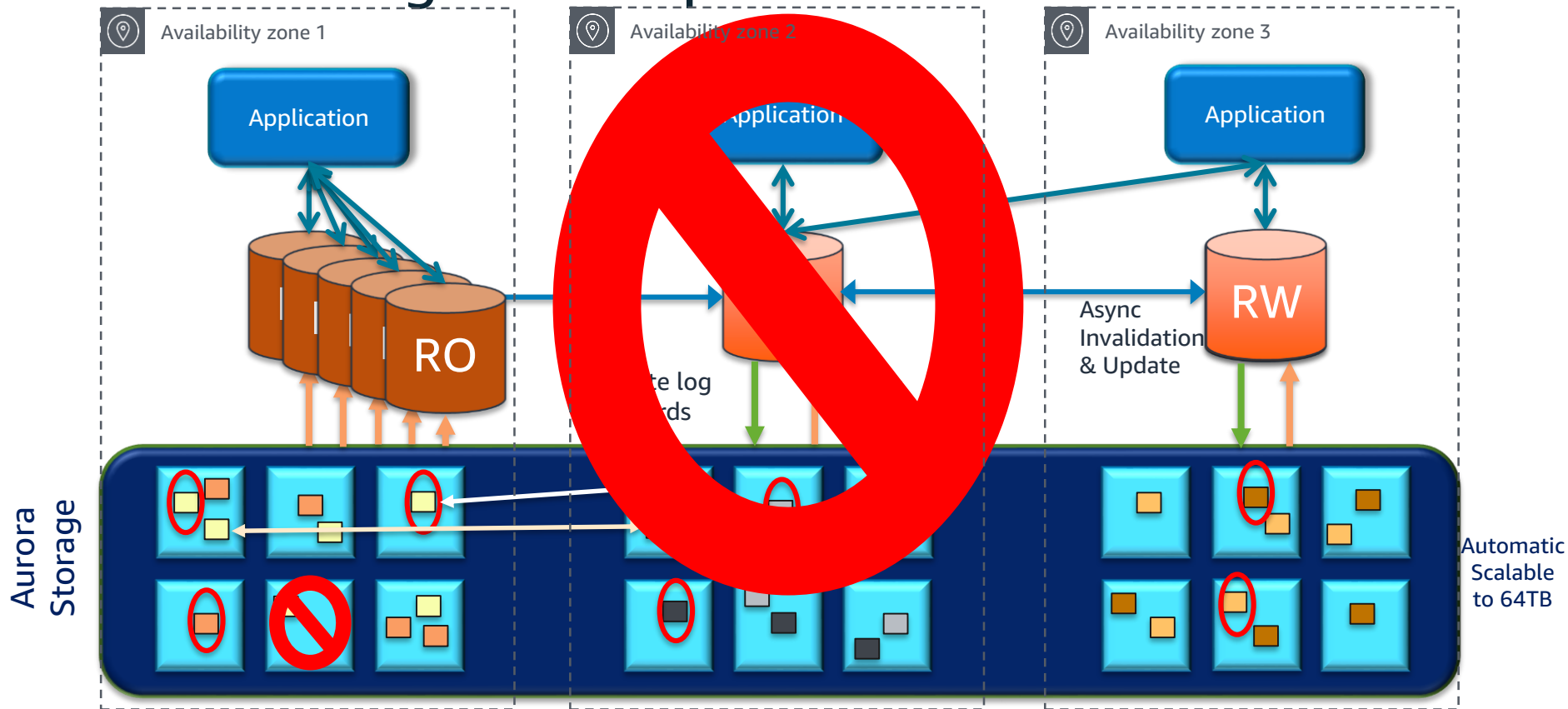
Amazon Aurora Recovers Up to 97% Faster

RECOVERY TIME FROM CRASH UNDER LOAD



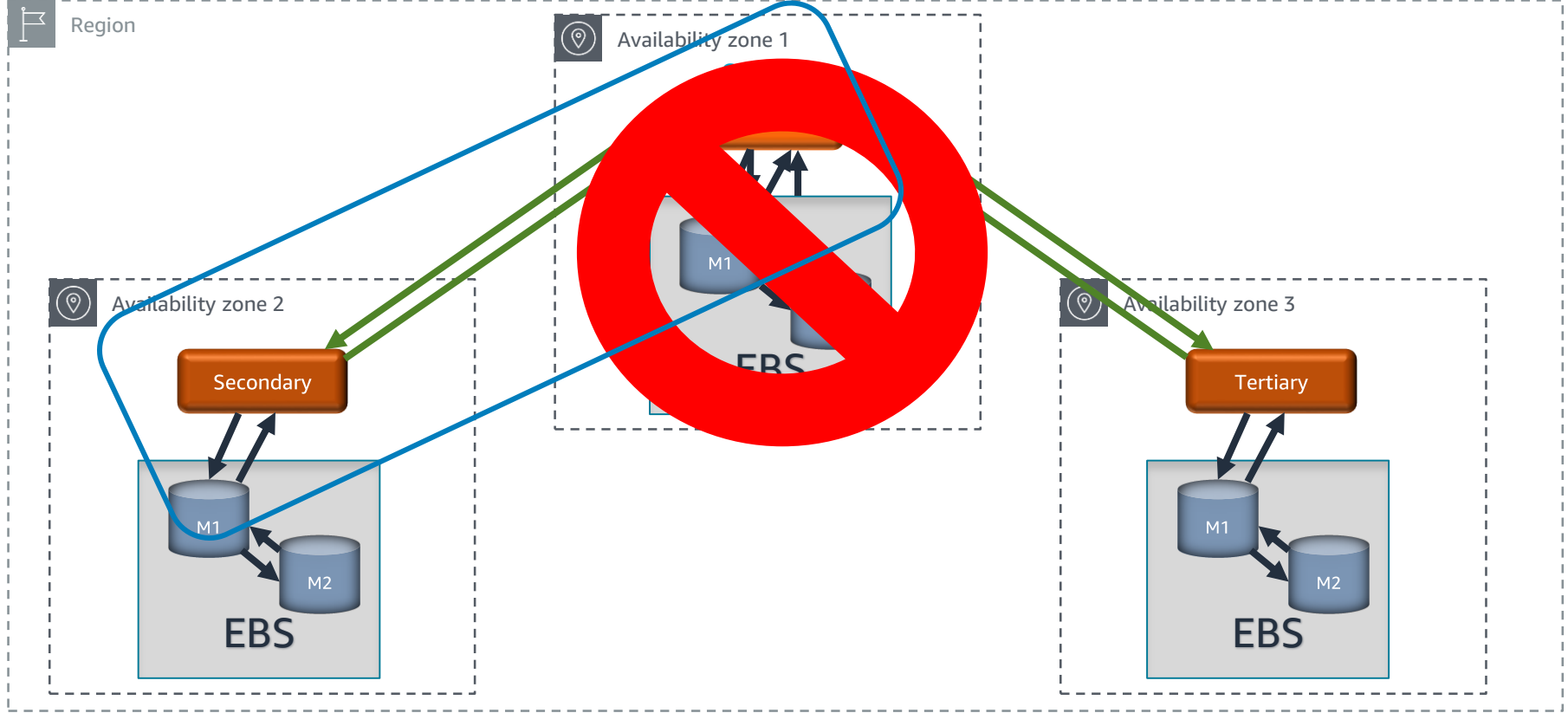
Base Architecture

Aurora Storage and Replicas

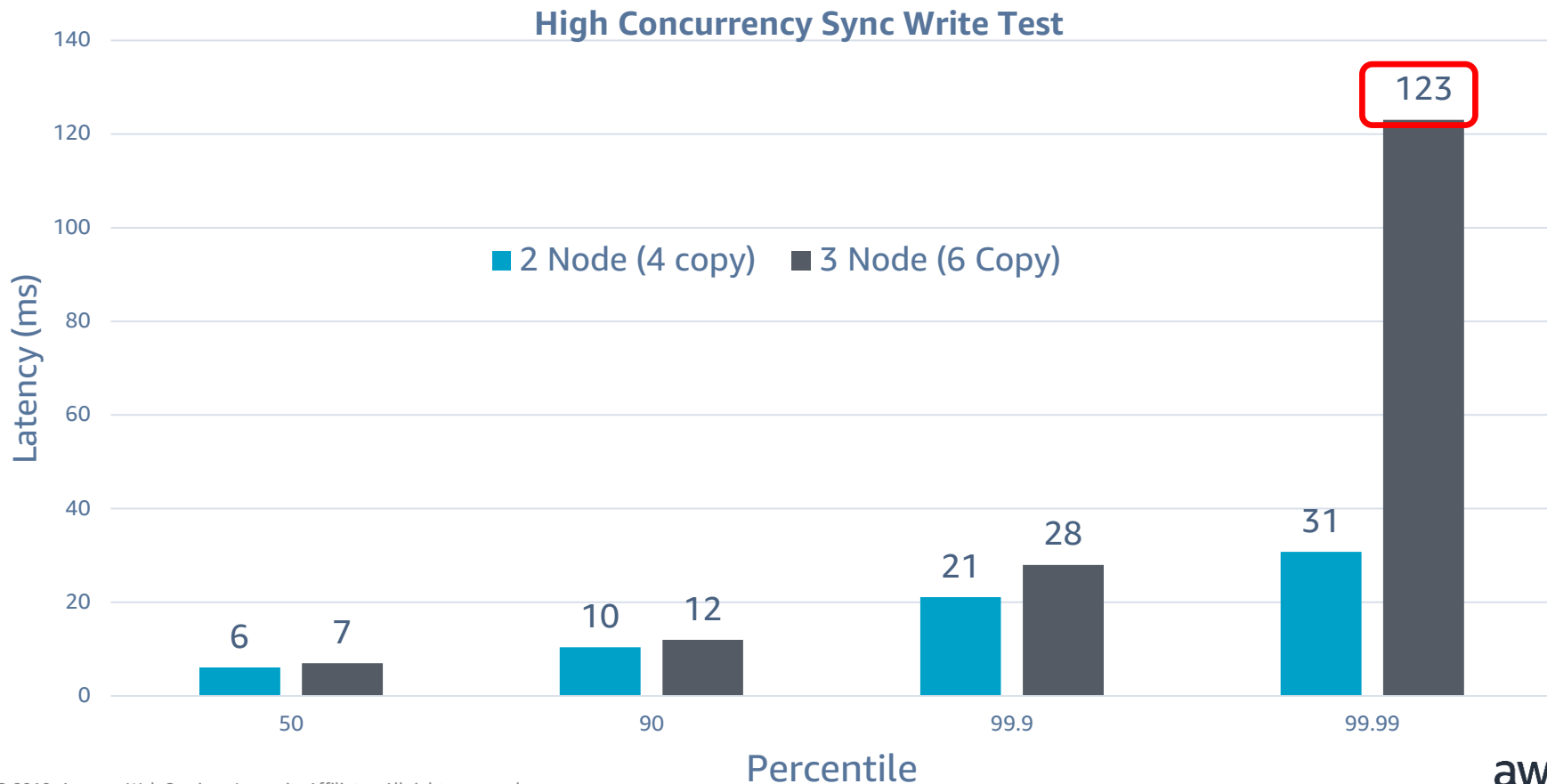


Durability—4 of 6 Quorum

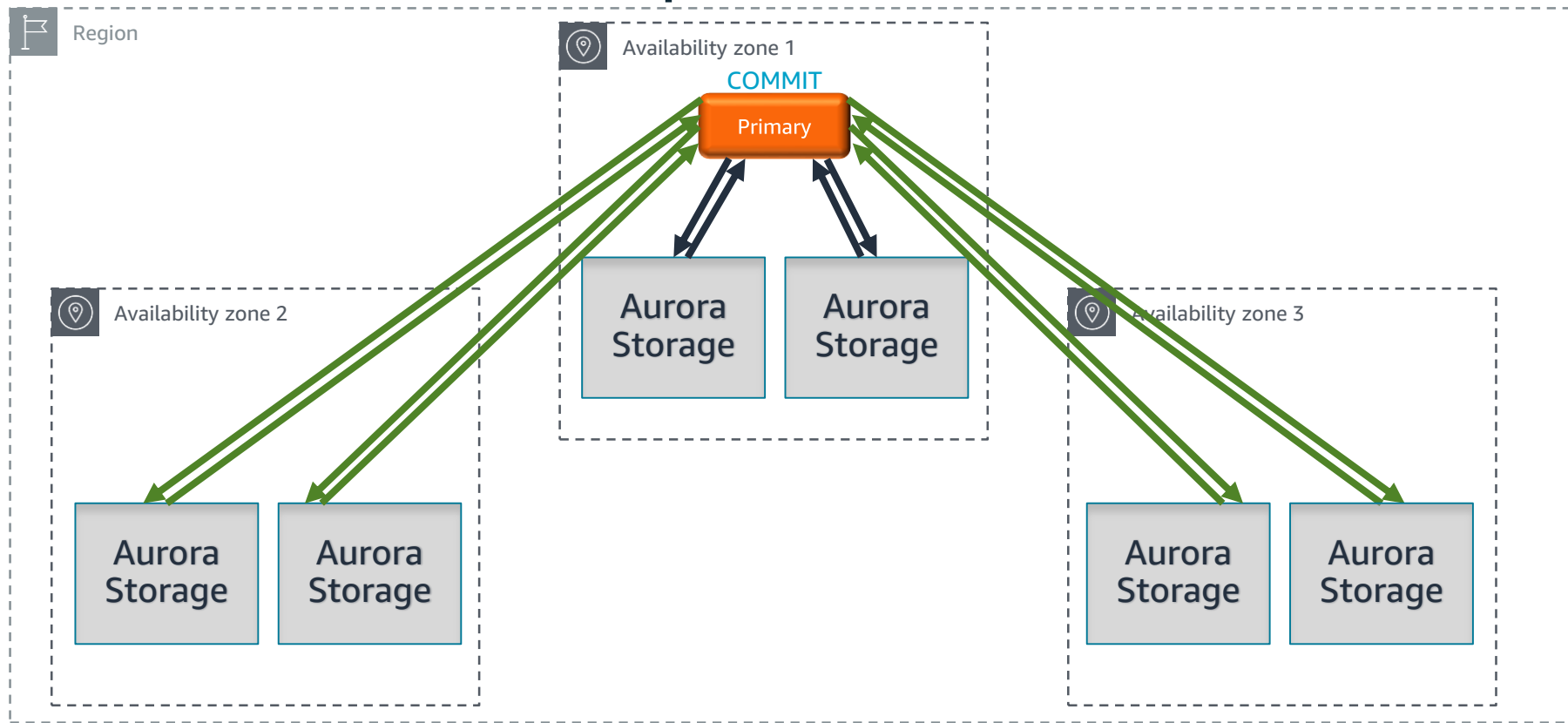
Typical synchronous replication – 3 locations



Cost of Additional Synchronous Replicas

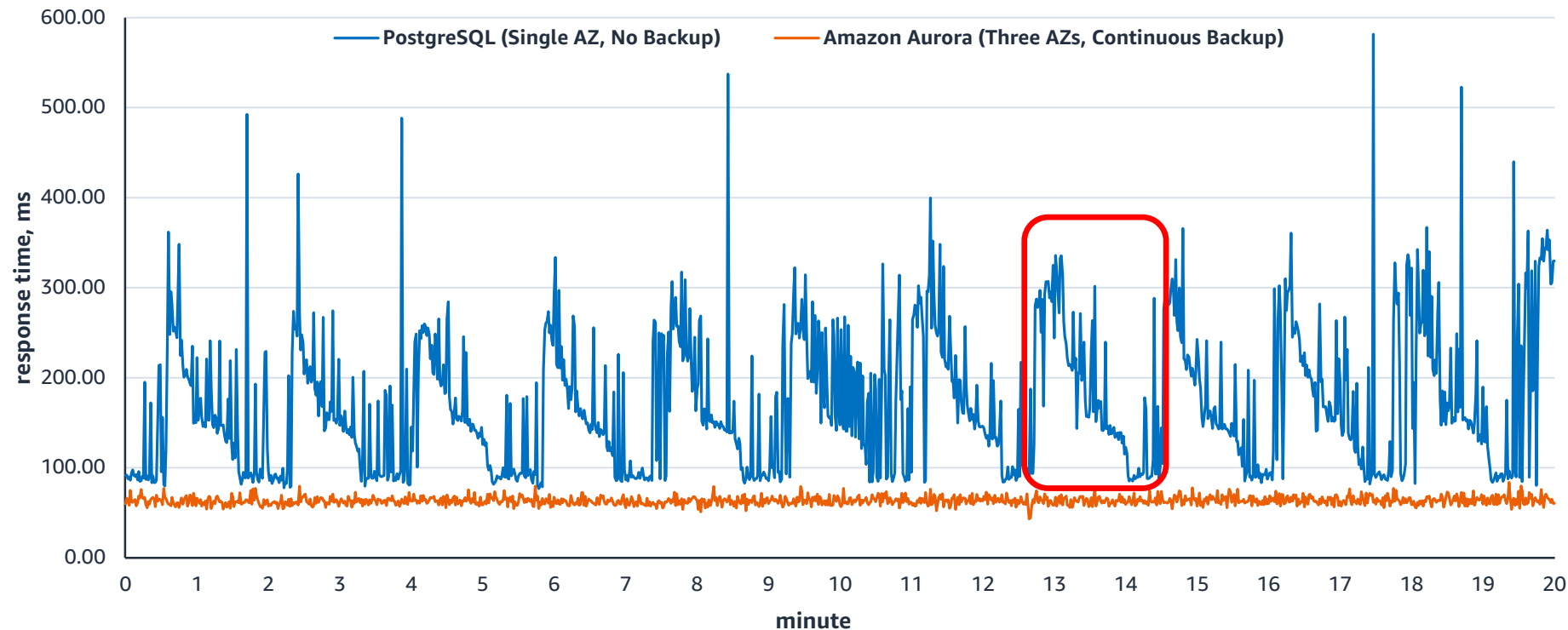


Aurora – 3 AZ's – 6 copies



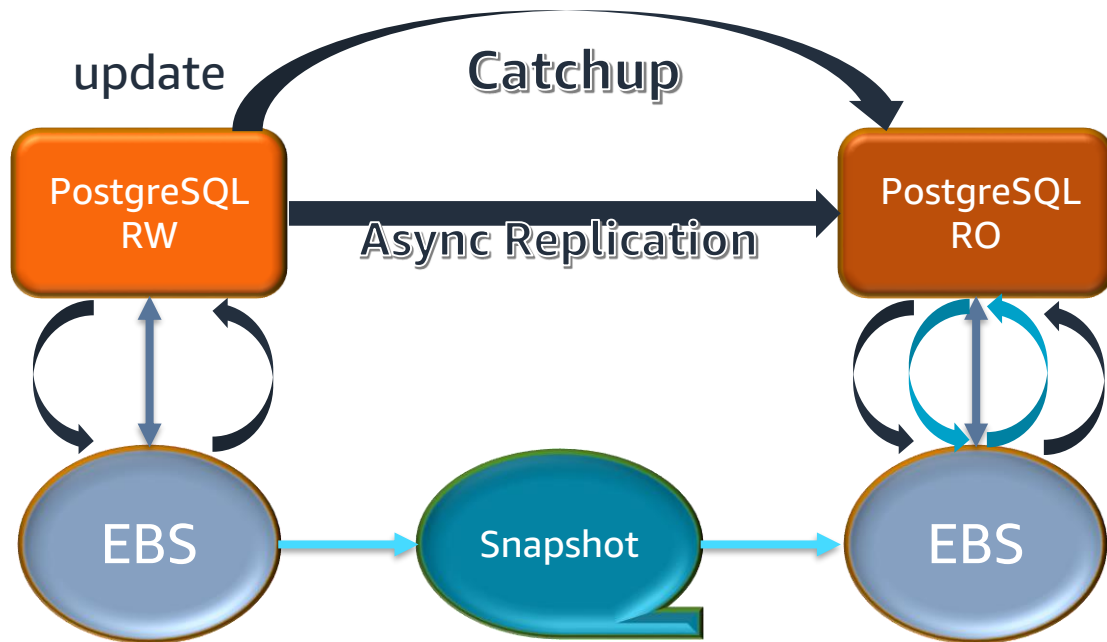
Amazon Aurora Gives >2x Lower Response Times

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

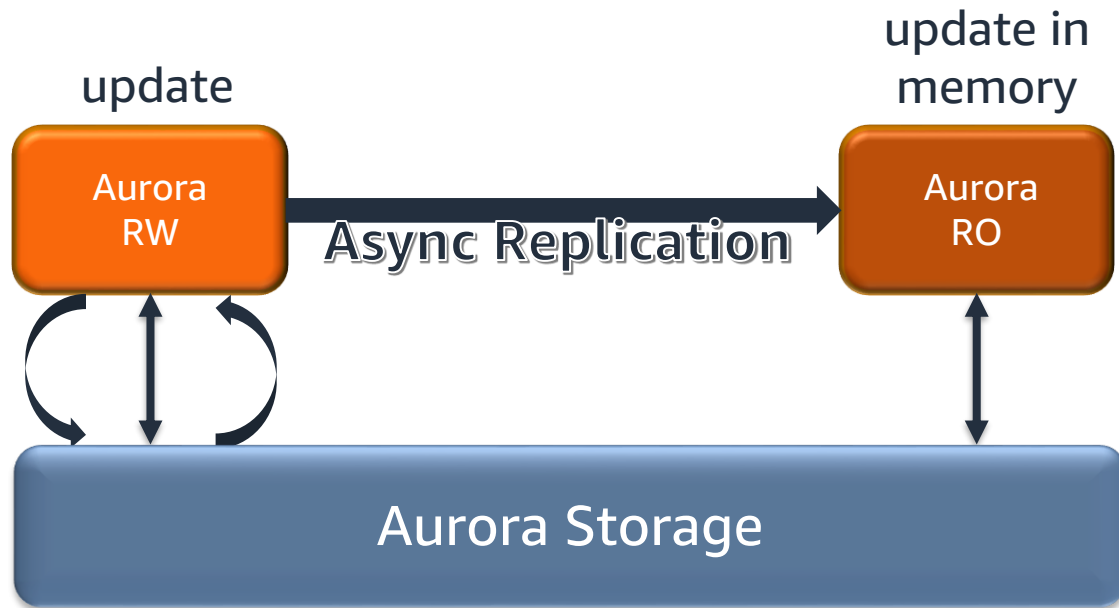


Replicas and Clones

Replicas—PostgreSQL



Replicas—Amazon Aurora

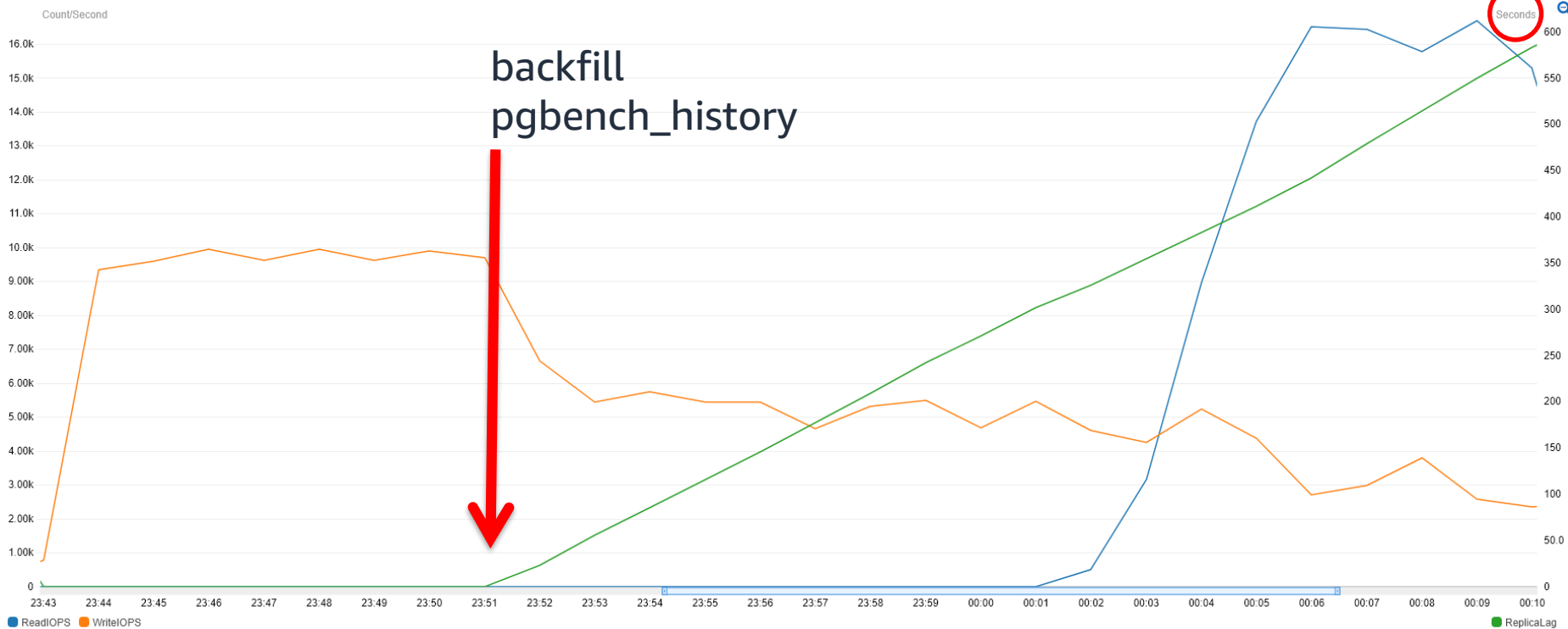


pgbench Benchmark



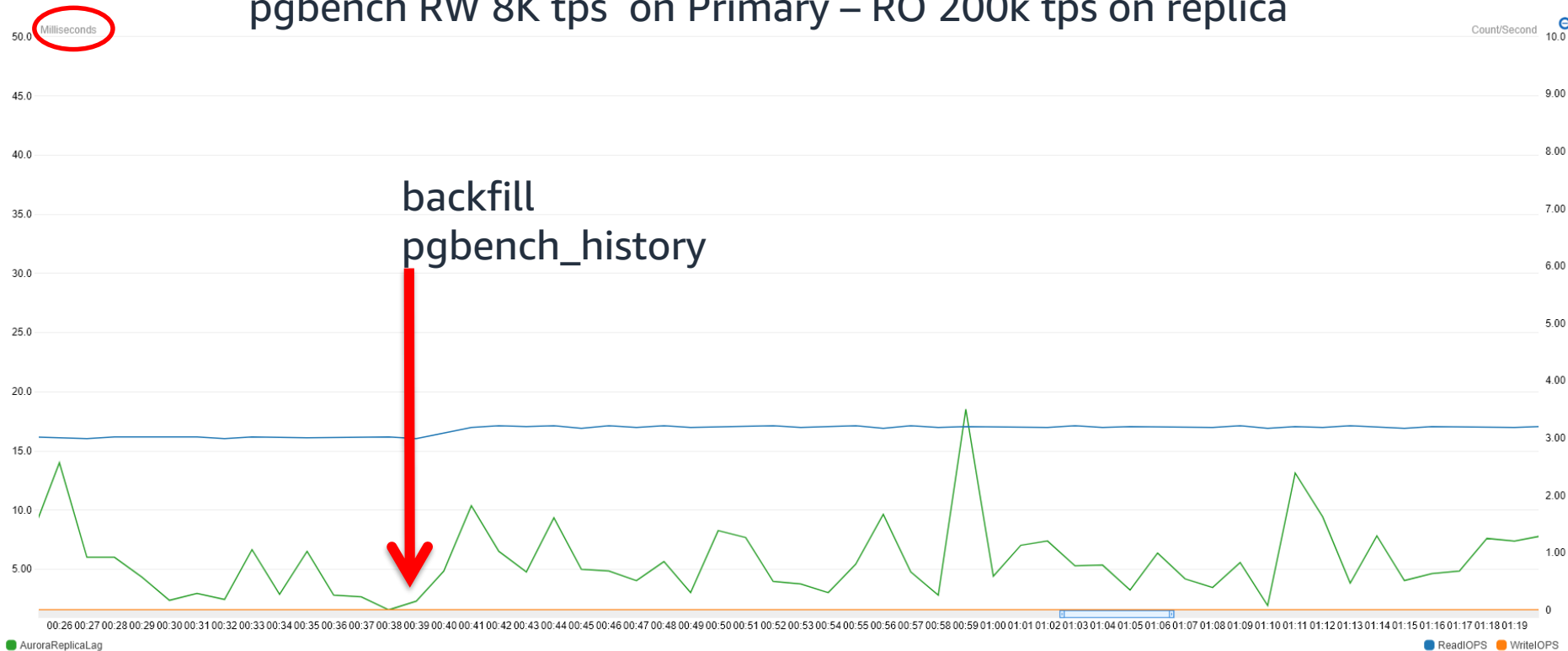
Replicas—Backfill on PostgreSQL

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

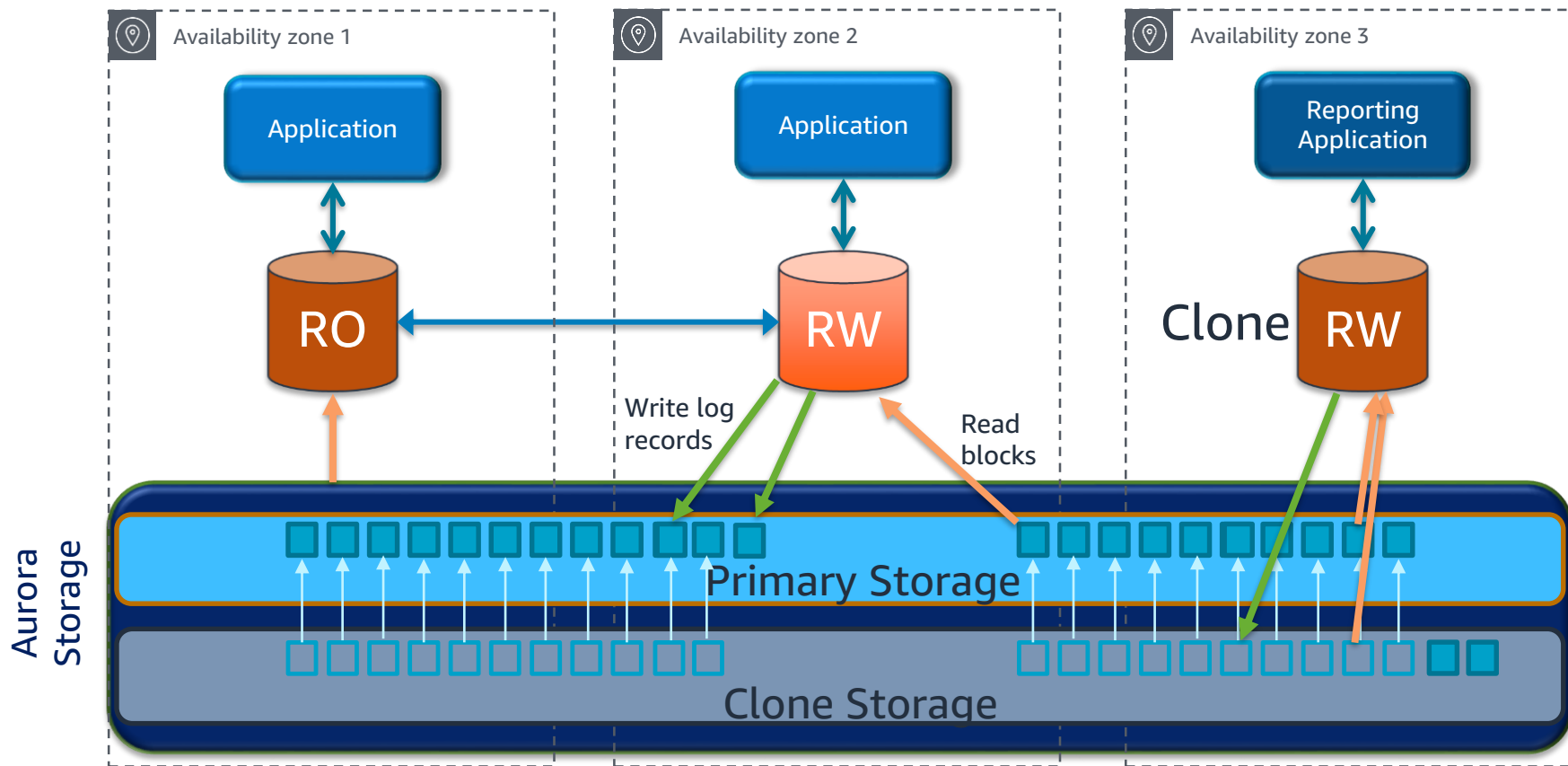


Replicas—Backfill on Amazon Aurora

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

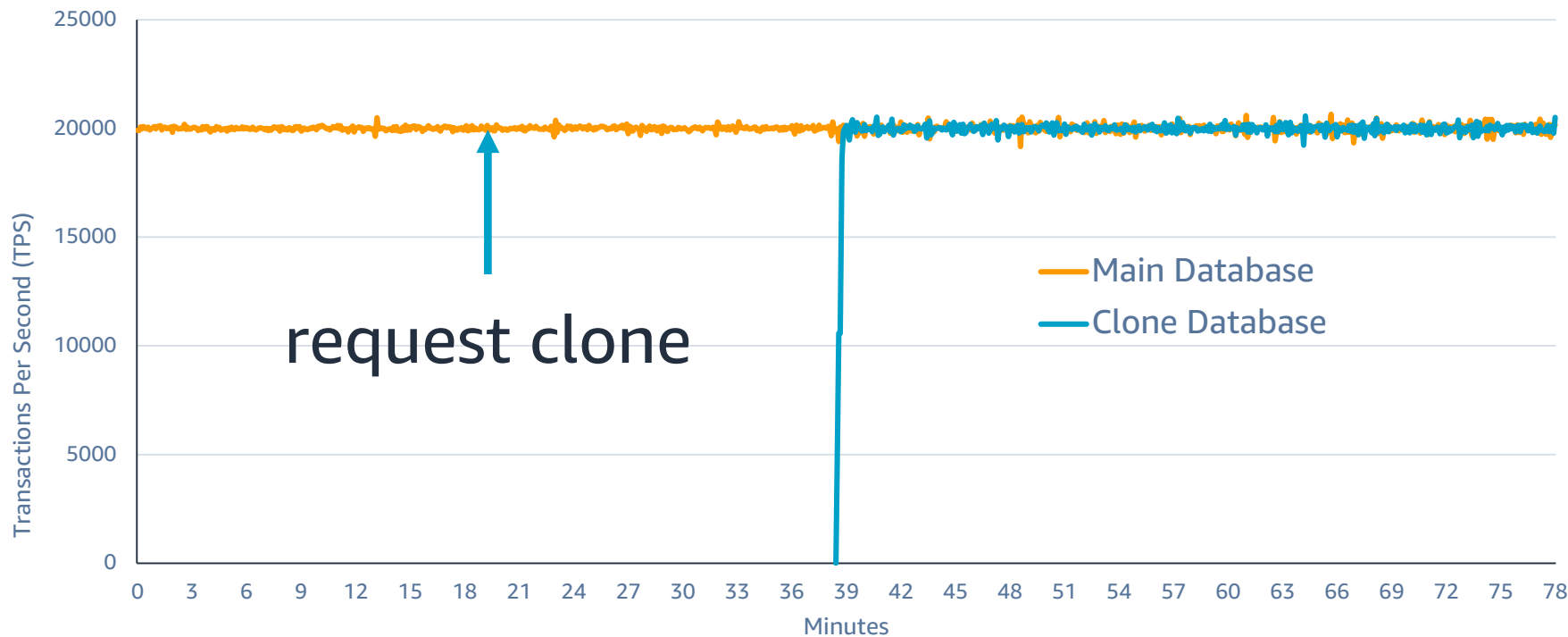


Fast Clones



Fast clone example

PGBench RW Scale 10K - Target Rate 20K TPS

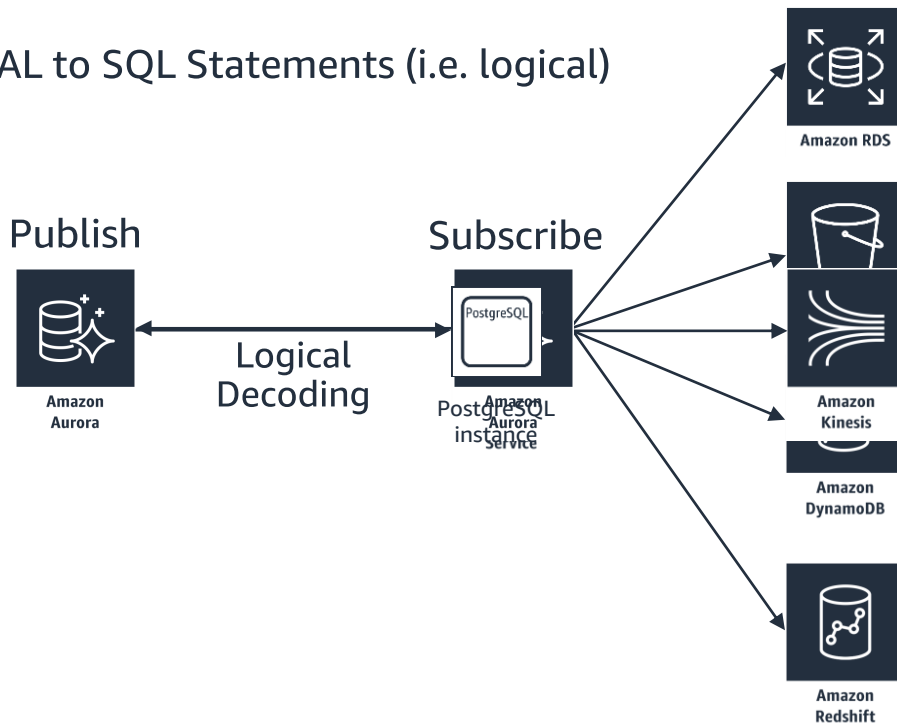


Replication

Logical Replication Support

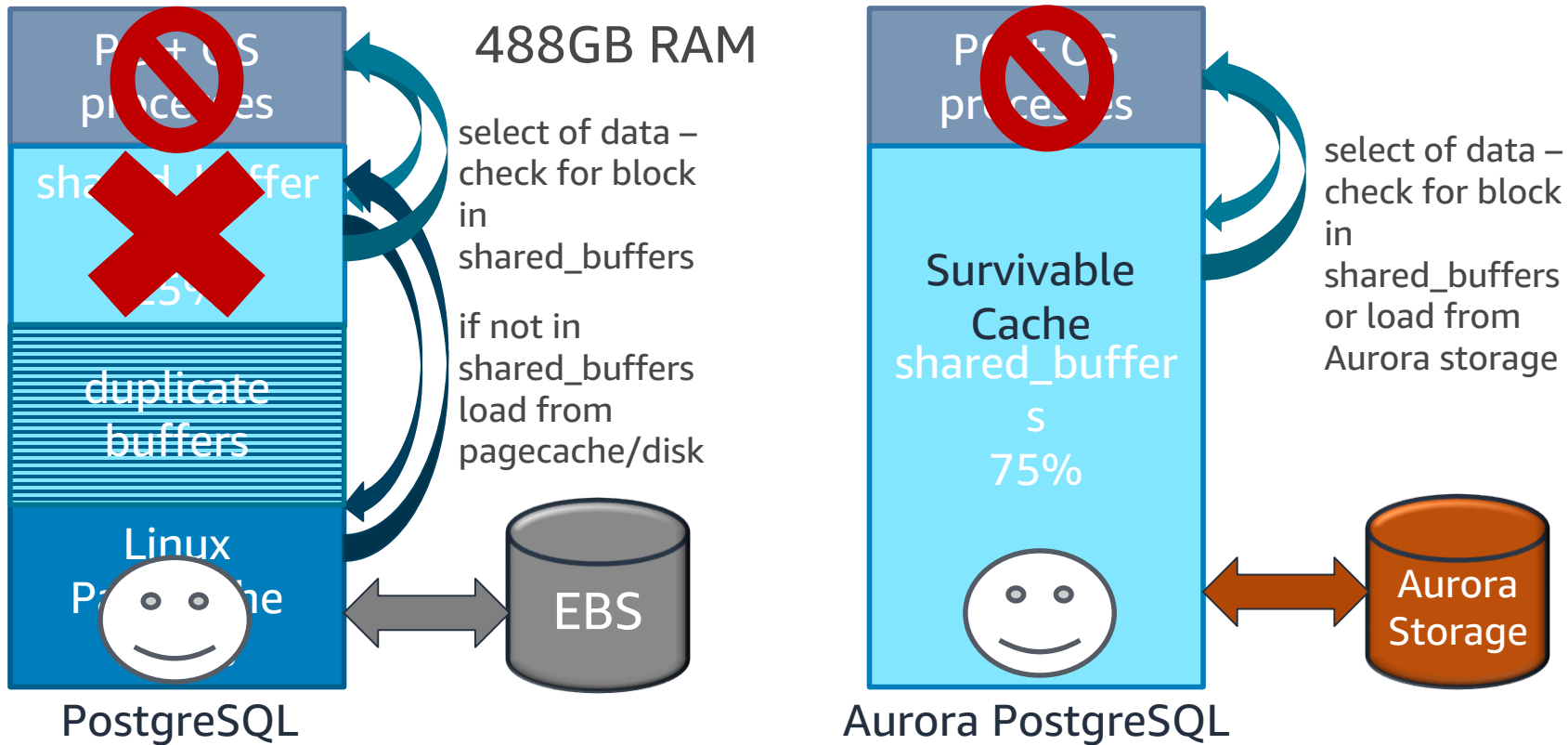
Converts Physical Changes (WAL to SQL Statements (i.e. logical)

- Logical Decoding Plugin
(test_decoder, decoder_raw, wal2json)
- V10 – Publish / Subscribe to another PostgreSQL Instance



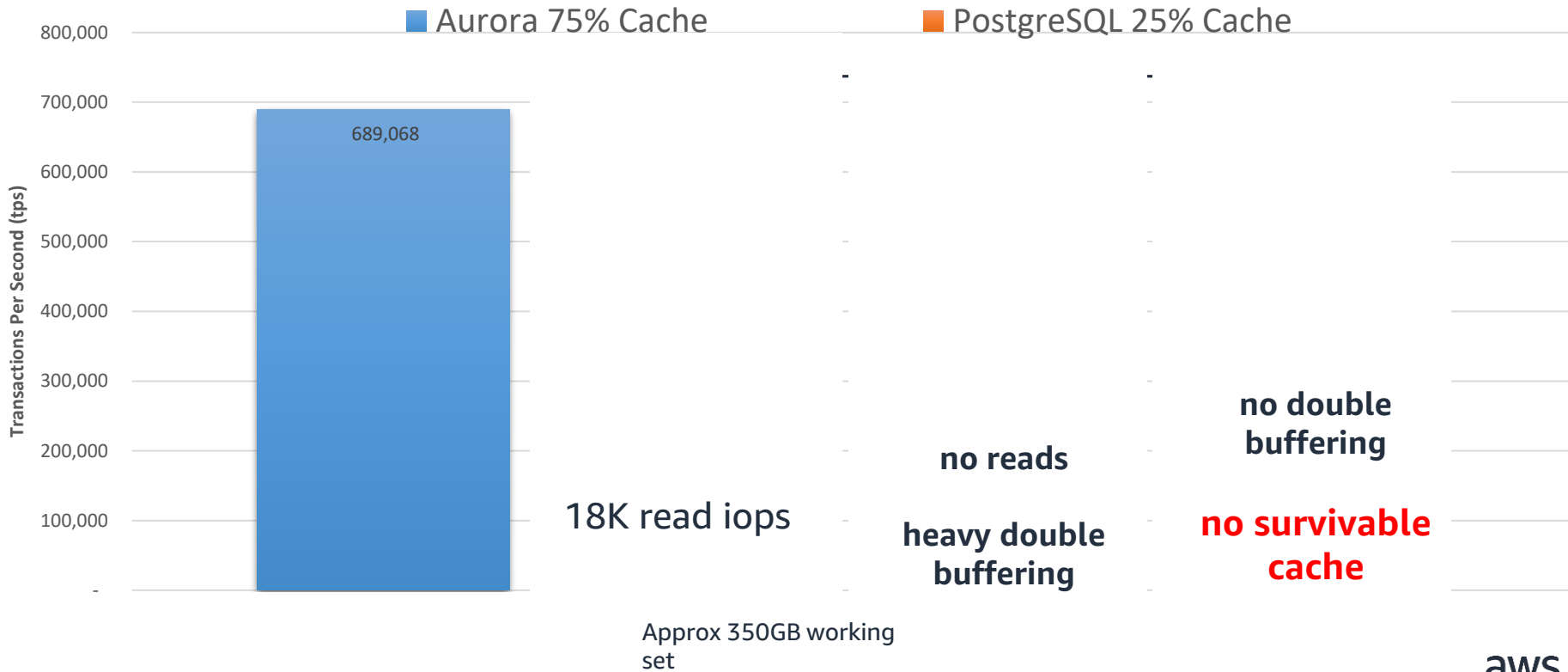
Caching

Caching Changes—No Double Buffering



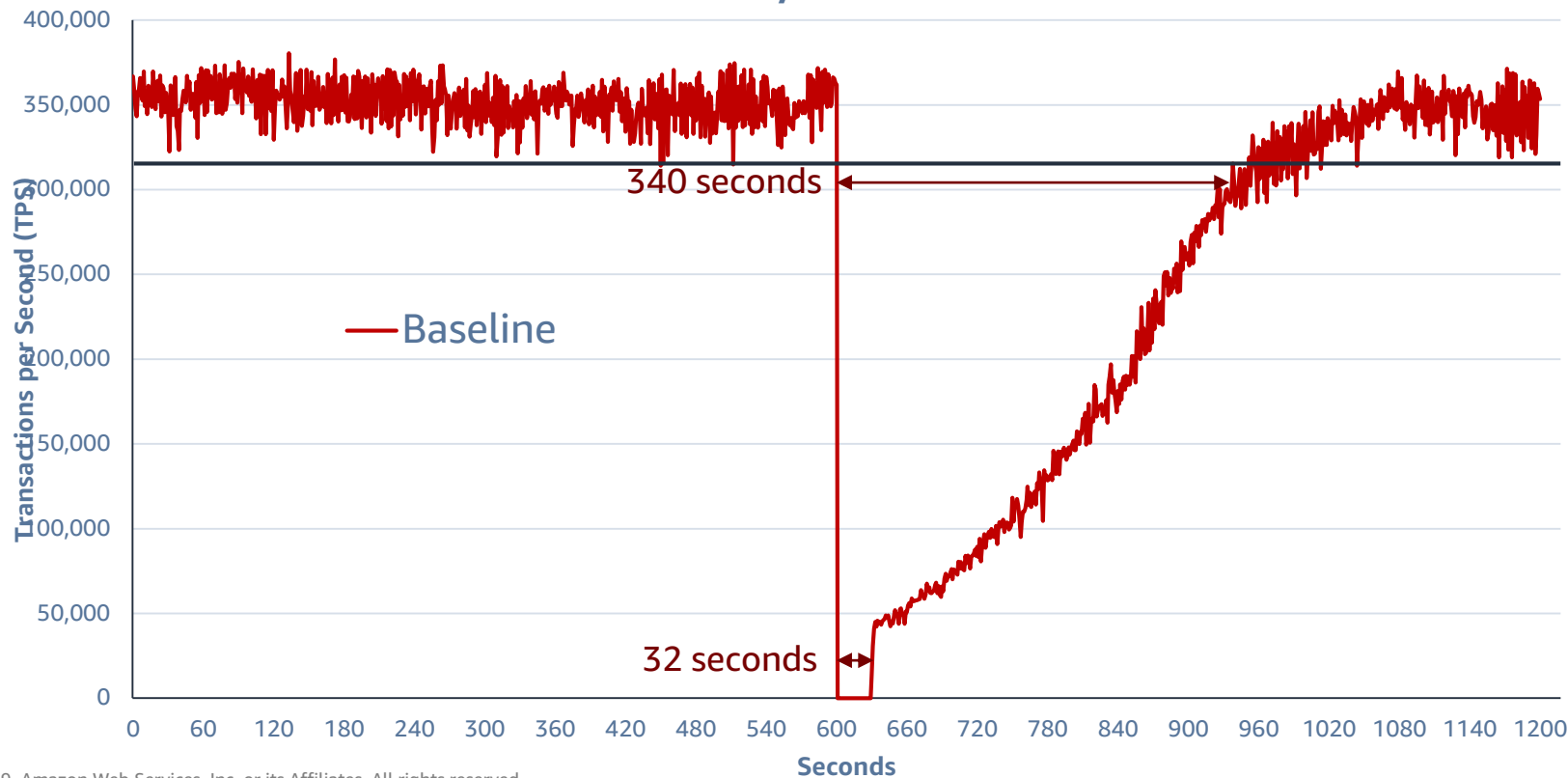
Caching Changes—No Double Buffering

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

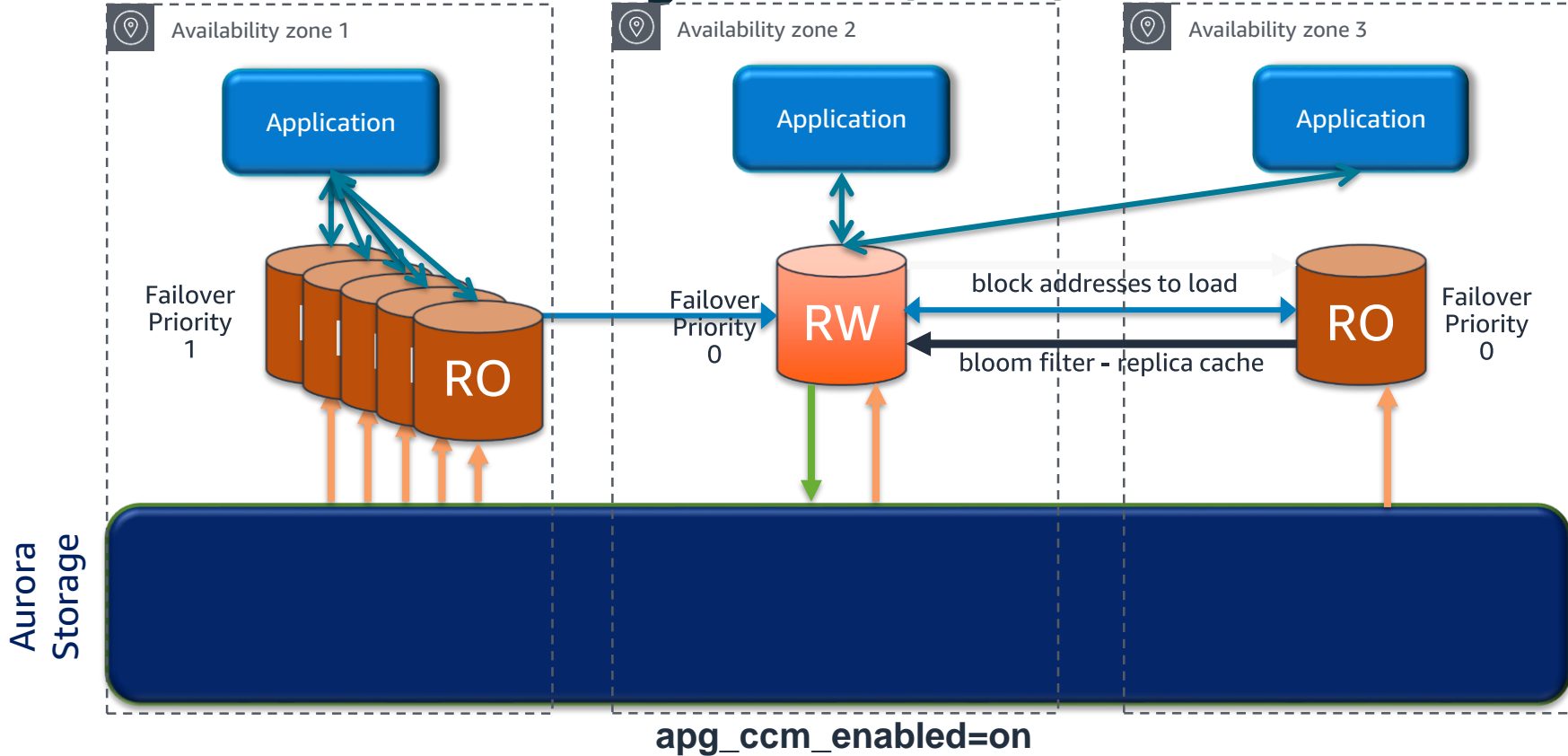


Cluster Cache Management - Failover

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

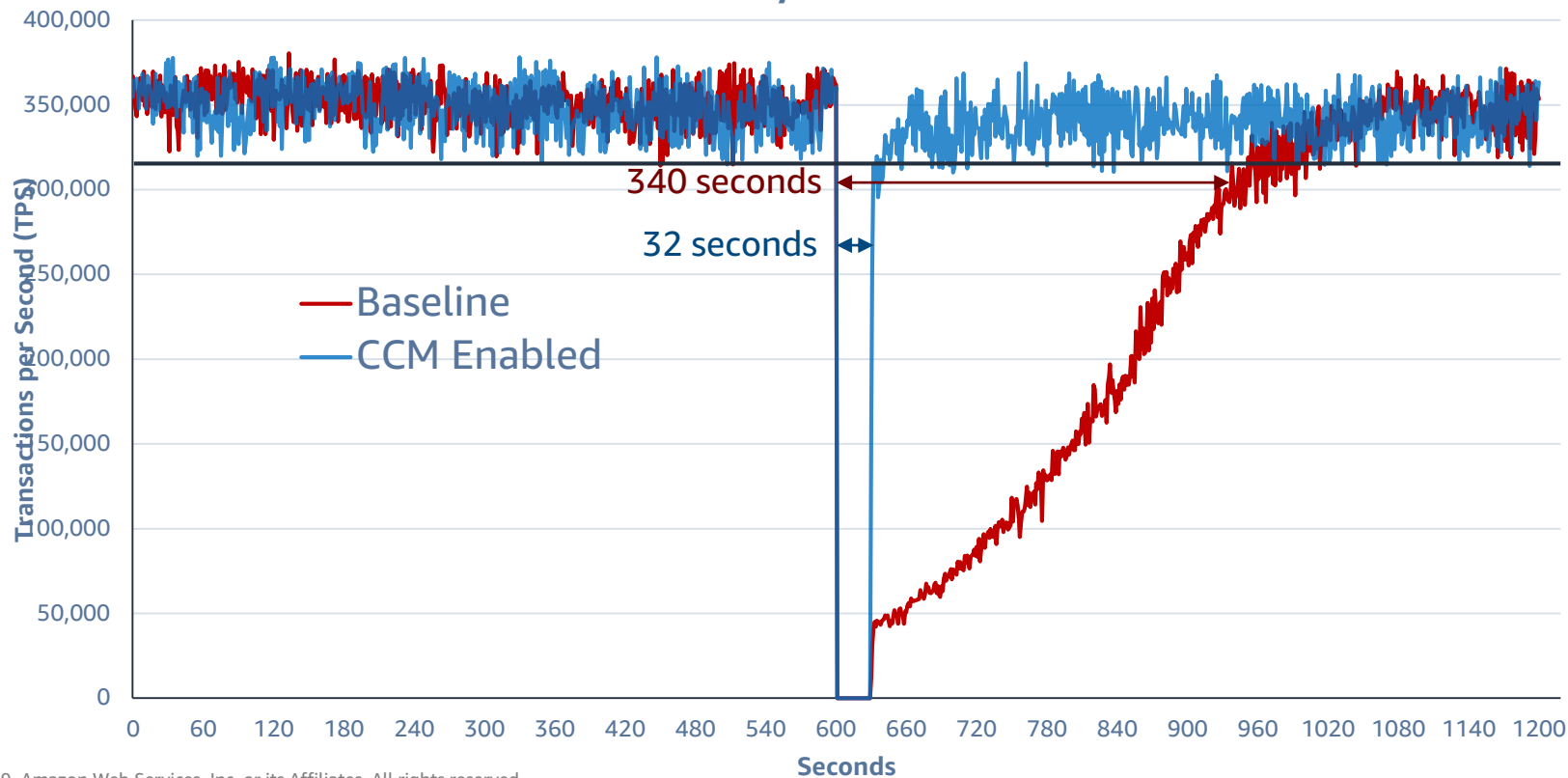


Cluster Cache Management (CCM) Feature



Cluster Cache Management

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



Performance

Performance Insights

Database load

Average active sessions (AAS)

5m

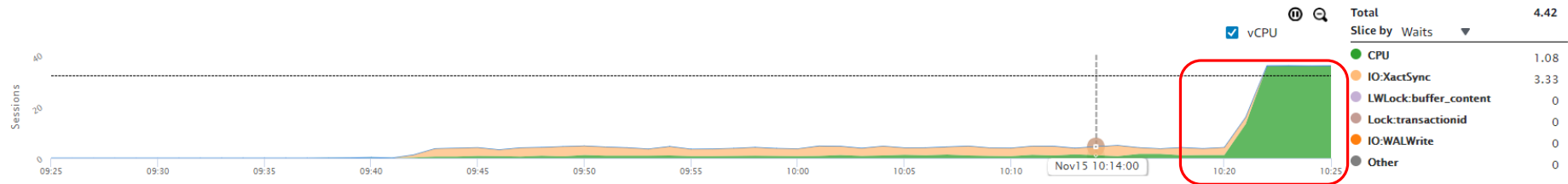
1h

5h

24h

1w

All



Waits SQL Hosts Users

Search SQL queries

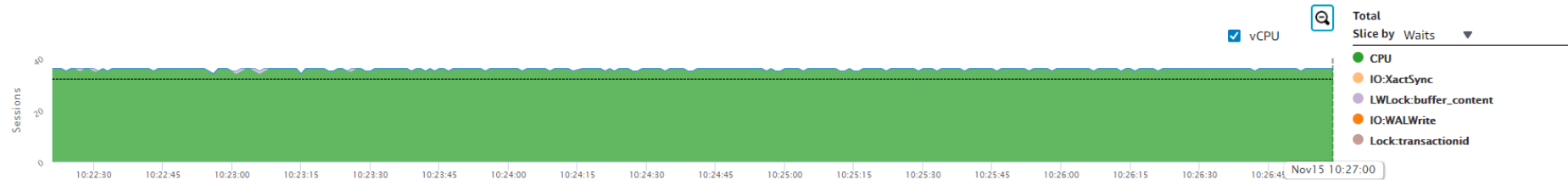
	Load By Waits (AAS)	SQL
▶	2.52	select sum(delta), sum(bbalance) from pgbench_history h , pgbench_branches b where b.bid = h.bid and b.bid in (?, ?, ?, ?) and mtime between now() - interval ? and now() - interval ?;
▶	2.03	END;
▶	0.41	UPDATE pgbench_accounts SET abalance = abalance + ? WHERE aid = ?;
▶	0.17	SELECT abalance FROM pgbench_accounts WHERE aid = ?;
▶	5.7e-2	UPDATE pgbench_branches SET bbalance = bbalance + ? WHERE bid = ?;
▶	5.3e-2	UPDATE pgbench_tellers SET tbalance = tbalance + ? WHERE tid = ?;
▶	3.1e-2	select sum(delta), sum(bbalance) from pgbench_history h , pgbench_branches b where b.bid = h.bid and b.bid in (?, ?, ?, ?) and mtime between now() - interval ? and now() - interval ?;
▶	2.3e-2	UPDATE pgbench_accounts SET abalance = abalance + ? WHERE aid = ?;
▶	2e-2	UPDATE pgbench_tellers SET tbalance = tbalance + ? WHERE tid = ?;
▶	2e-2	UPDATE pgbench_branches SET bbalance = bbalance + ? WHERE bid = ?;

Performance Insights

Database load

Average active sessions (AAS)

5m 1h 5h 24h 1w All



Waits SQL Hosts Users

Search SQL queries

Load By Waits (AAS)

SQL

▶	29.81	select sum(delta), sum(bbalance) from pgbench_history h , pgbench_branches b where b.bid = h.bid and b.bid in (?, ?, ?, ?) and mtime between now() - interval ? and now() - interval ?;
▶	3.7	UPDATE pgbench_accounts SET abalance = abalance + ? WHERE aid = ?;
▶	2.3	SELECT abalance FROM pgbench_accounts WHERE aid = ?;

Performance Insights

Database load

Average active sessions (AAS)

5m

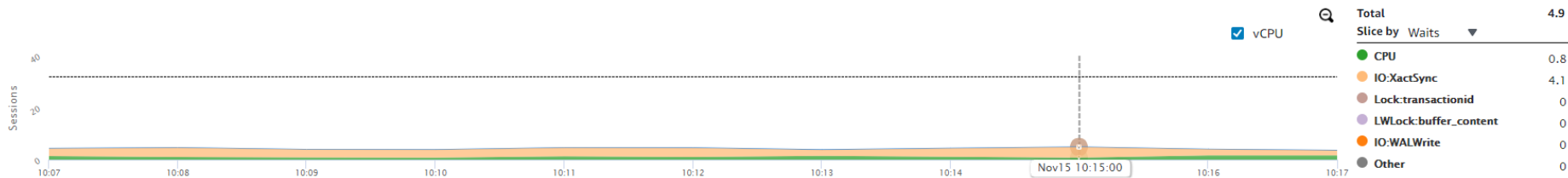
1h

5h

24h

1w

All



Waits SQL Hosts Users

Search SQL queries

Load By Waits (AAS)	SQL
3.12	END;
0.47	select sum(delta), sum(bbalance) from pgbench_history h , pgbench_branches b where b.bid = h.bid and b.bid in (?, ?, ?, ?, ?) and mtime between now() - interval ? and now() - interval ?;
0.14	UPDATE pgbench_accounts SET abalance = abalance + ? WHERE aid = ?;
9.7e-2	UPDATE pgbench_tellers SET tbalance = tbalance + ? WHERE tid = ?;
9.6e-2	UPDATE pgbench_branches SET bbalance = bbalance + ? WHERE bid = ?;
8e-2	SELECT abalance FROM pgbench_accounts WHERE aid = ?;
4.9e-2	select sum(delta), sum(bbalance) from pgbench_history h , pgbench_branches b where b.bid = h.bid and b.bid in (?, ?, ?, ?, ?) and mtime between now() - interval ? and now() - interval ?;
3.6e-2	SELECT abalance FROM pgbench_accounts WHERE aid = ?;
3.6e-2	UPDATE pgbench_accounts SET abalance = abalance + ? WHERE aid = ?;
3.5e-2	UPDATE pgbench_tellers SET tbalance = tbalance + ? WHERE tid = ?;

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=0)
            Index Cond: (bid = b.bid)
```

- stats change?
- config change?
- index change?

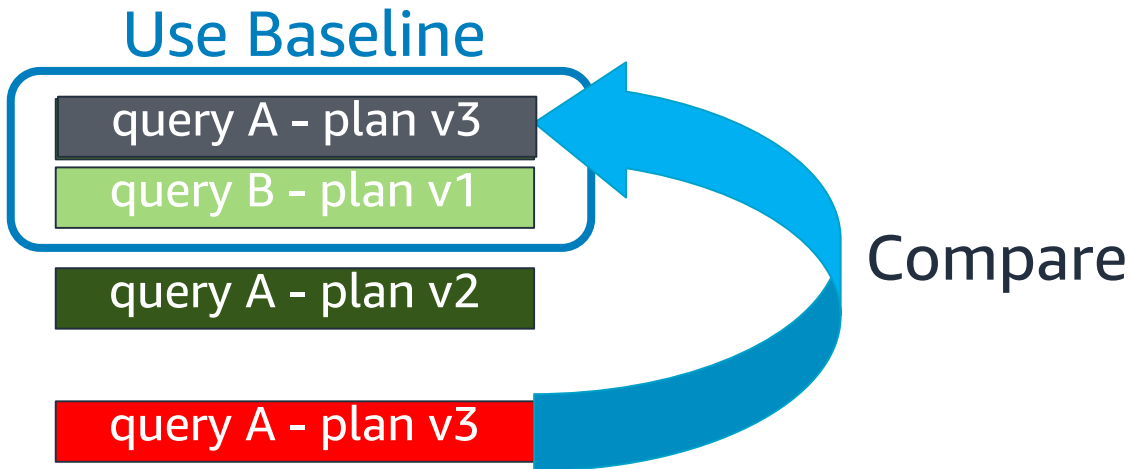
After

```
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[]))
```

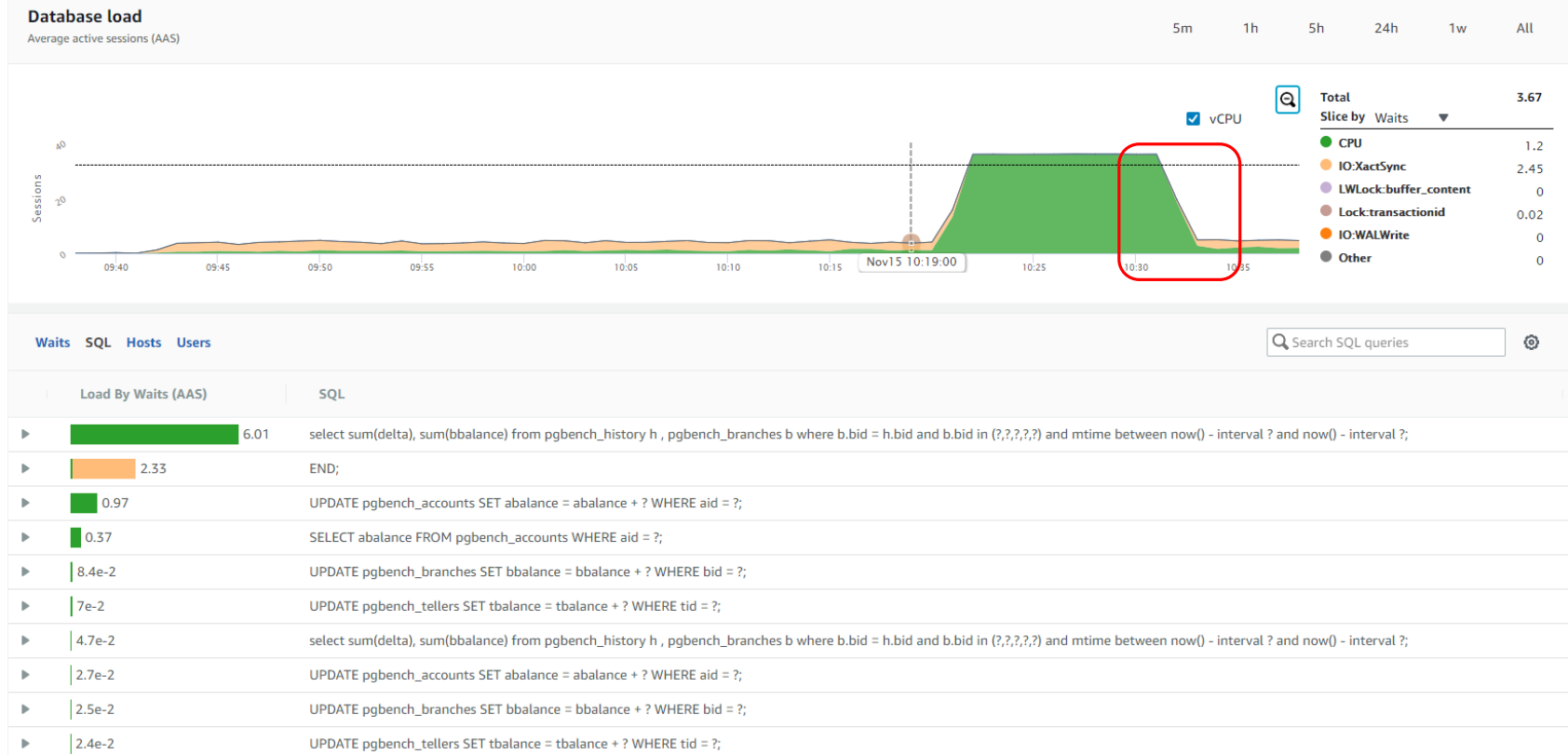
- enable_bitmapscan=off
- enable_indexscan=off

Query Plan Management - QPM

- Capture statements
- Approve statements
- Evolve better plans

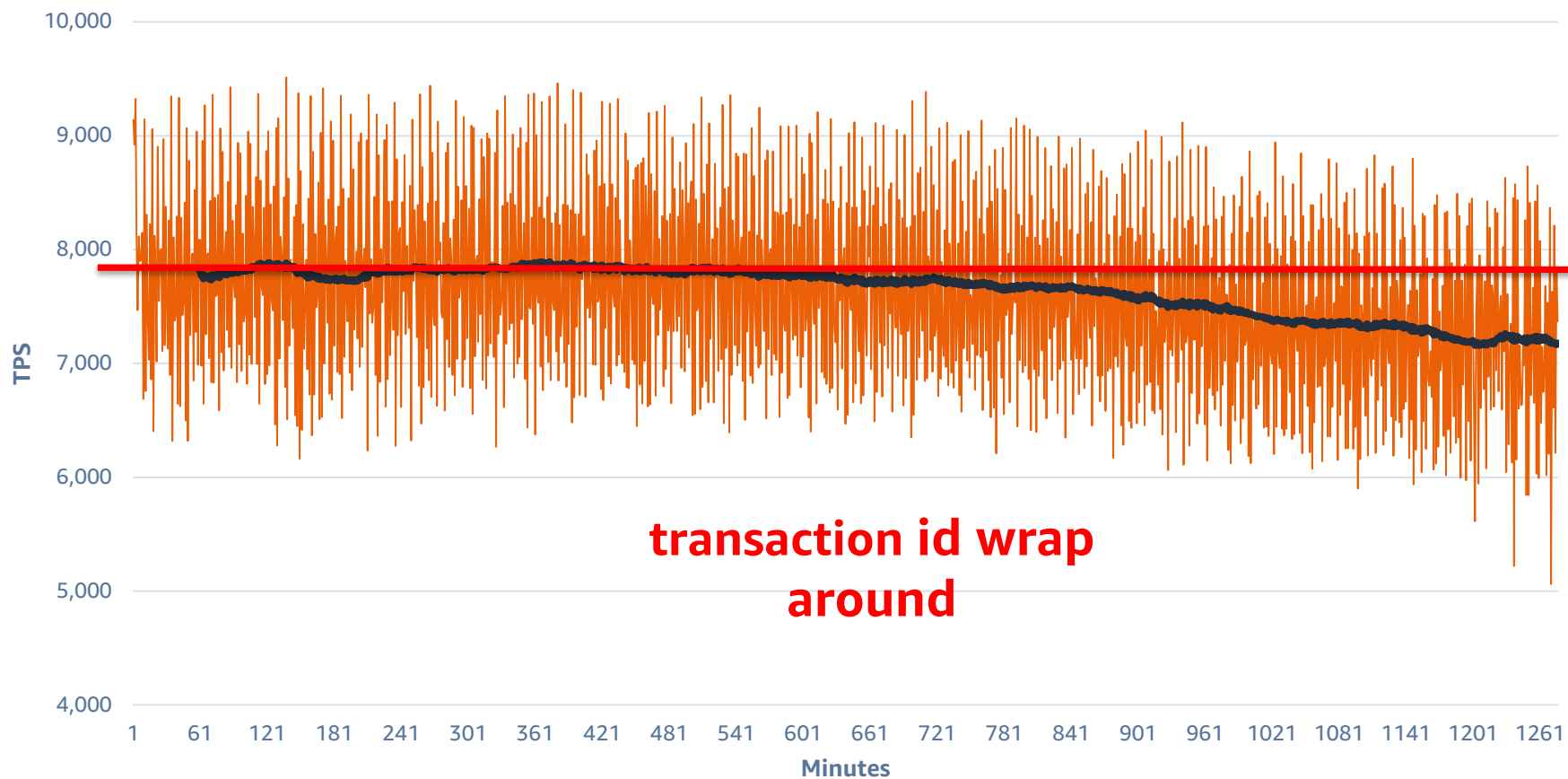


QPM – Use Plan Baselines



Vacuuming

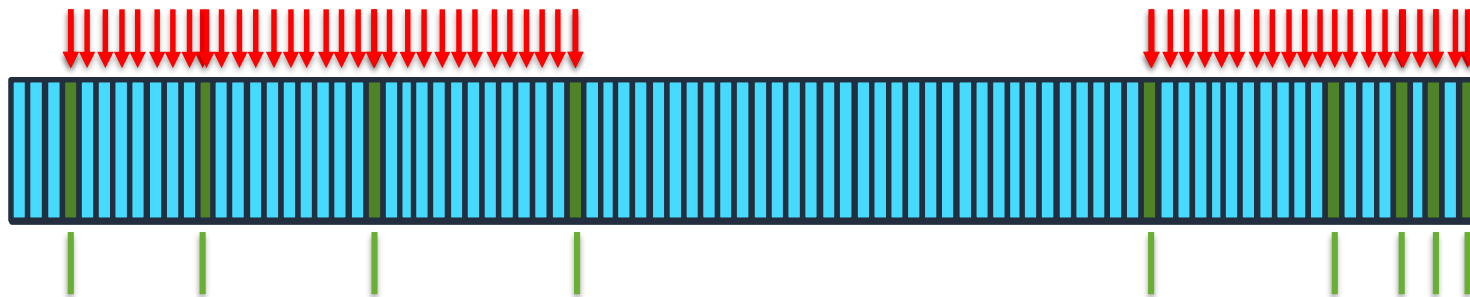
Updates—No Vacuum Running



Intelligent Vacuum Prefetch

PostgreSQL 402 seconds

visibility & frozen map



Aurora PostgreSQL 163 seconds

Submit
Batch I/O
up to
256 blocks



Thank you!