
AWS
re:Invent
NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

DAT303

Top 5 things to know when migrating from Oracle to PostgreSQL

Nathan Fuzi

Principal Product Manager
Amazon RDS for Oracle, AWS

Krishna Sarabu

Sr Database Specialist Solutions Architect
AWS

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

High availability & disaster recovery

Migration

Data concurrency and consistency

Design best practices

Optimizing PostgreSQL database for high concurrent workloads

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Drivers for migration



Infrastructure

- Hardware end of life
- Lack of redundancy or automation
- Failure at different layers

Scalability

- Change instance sizes as required
- Scale across regions as required





Cost optimization

- Pay for what you use
- Open source options, license savings
- Fully managed databases

Innovation

- Modernize your database tier
- Choose the right database for your workload



aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

RDS PostgreSQL engines and features

Currently supports engines from 9.7 to 14 (1)

Within those versions you can use the following PostgreSQL features:

- Fully managed
- Extensions: PostGIS, hstore, aws_s3 and more
- Highly secure
- Full text search dictionaries
- Advanced data types such as Range and JSON
- Foreign data wrappers
- Server-side programming PL/PgSQL, plv8, PL/Perl
- Other core functionalities listed here

1. Get latest version here: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_PostgreSQL.html#PostgreSQL.Concepts.General.DBVersions

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

High availability & disaster recovery

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

High availability & disaster recovery options

	Oracle	PostgreSQL		
		Self-managed	RDS	Amazon Aurora
High availability	Oracle RAC, Oracle DataGuard	Streaming replication, tools such as Patroni	Multi-AZ DB instance Multi-AZ DB cluster	Multi-AZ
Disaster recovery	Oracle DataGuard	Streaming replication, tools such as Patroni	Cross region read replica	Aurora Global Database
	Backups & cross region backup copy	Backups & cross region backup copy	Automated backups & backup replication	Automated backups & backup replication
Other	Oracle GG, Logical standby	Logical replication	Logical replication	Logical replication

aws

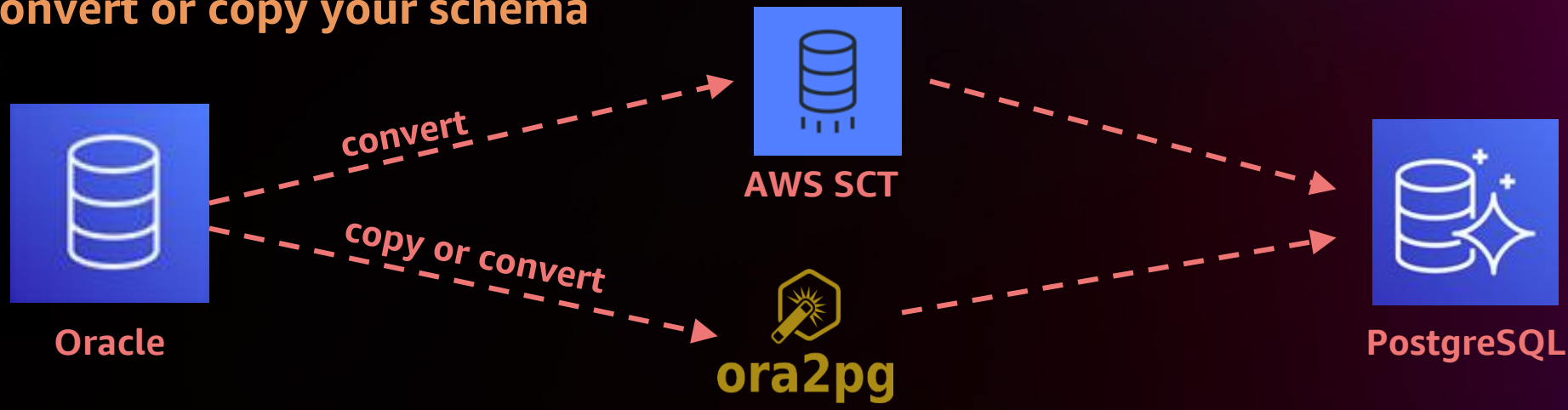
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Migration

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Database migration process

Step 1: Convert or copy your schema



Step 2: Move your data



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Data concurrency & consistency

UNDO, MVCC, bloating, transaction wraparound & vacuum

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Multi version concurrency control

- Snapshot isolation
  - Readers don’t block writers & writers don’t block readers
- Statement-level read consistency
  - Read committed transaction isolation
- Transaction-level read consistency
  - Repeatable read transaction isolation

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

MVCC: performance considerations

- Multiple row versions
- Visibility using Transaction ID (XMIN & XMAX)
  - XID is a 4-byte integer
  - Wrap-around after 2.1 billion transactions
- Auto vacuum
  - Removes the dead tuples, index entries pointing to dead tuples
  - Freeze xids, update visibility maps
  - Gather statistics
- Long running transactions affect performance

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Design best practices

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Design best practices

- Tablespaces
  - Provides no real benefit as PostgreSQL creates file per each segment
- Case folding
  - PostgreSQL folds all metadata to lower-case Vs upper-case in Oracle
- Synonyms
  - Use search path
- Sequences
  - Sequences are cached per session in PostgreSQL
- Data types
  - PostgreSQL has 64 base types and can be extended for more

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Data types

- Most migration tools translate Oracle *Number* to PostgreSQL *Numeric*
  - *Number* has 38 digits while *Numeric* has 131,072 digits (16,383 decimals)
  - Performance differences between *Numeric* and *bigint*
- Working with large objects
  - Using *text* for CLOB and *bytea* for BLOB
    - TOASTed
    - Allows values up to 1 GiB
  - Using *largeobject (lo* extension)
    - Allows values up to 4 TiB
    - Up to 2^32 large objects

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Index types

PostgreSQL has more and different types of indexes than Oracle

- B-tree
- Hash
- GIN
- GiST
- SP-GiST
- BRIN

PostgreSQL can use indexes on LIKE queries

```
CREATE INDEX idx_users_lname
  ON users USING gin (lname gin_trgm_ops);
```

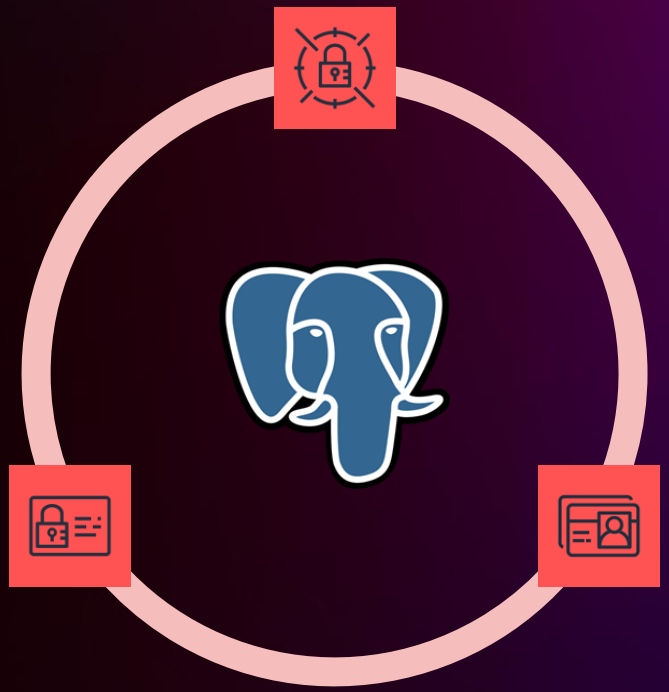
```
 EXPLAIN SELECT * FROM users WHERE lname LIKE '%ing%';

                        QUERY PLAN
-----------------------------------------------------
  Bitmap Heap Scan on users (cost=8.00..12.02 rows=1 width=654)
    Recheck Cond: ((lname)::text ~~ '%ing%'::text)
    -> Bitmap Index Scan on idx_users_lname
         (cost=0.00..8.00 rows=1 width=0)
         Index Cond: ((lname)::text ~~ '%ing%'::text)
```

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Security

- Authentication
  - PostgreSQL local accounts (SCRAM, MD5)
  - AWS Identity and Access Management (IAM)
  - Kerberos
- Authorization
  - Roles & privileges
- Encryption



aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Optimizing PostgreSQL for high concurrent workload

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

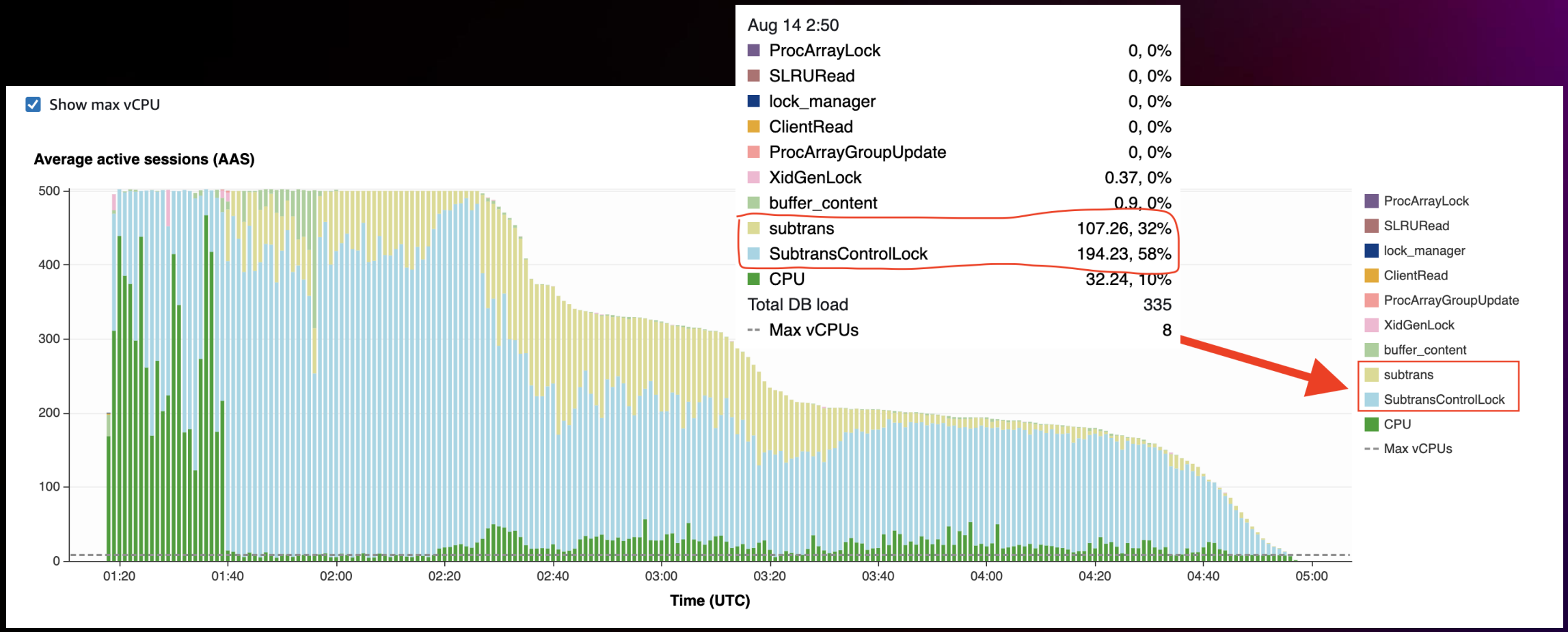
Lightweight locks (LWLock)

- Controls access to shared memory structures
  - Equivalent of latches in Oracle
- Potential heavy contention places
  - SubtransSLRU
    - Process waiting to access SLRU cache for a subtransaction
  - BufferContent
    - Process waiting to acquire LWLock on shared memory buffer
  - LockManager
    - Process waiting to add or examine locks for backend process

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Subtransactions SLRU cache contention



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

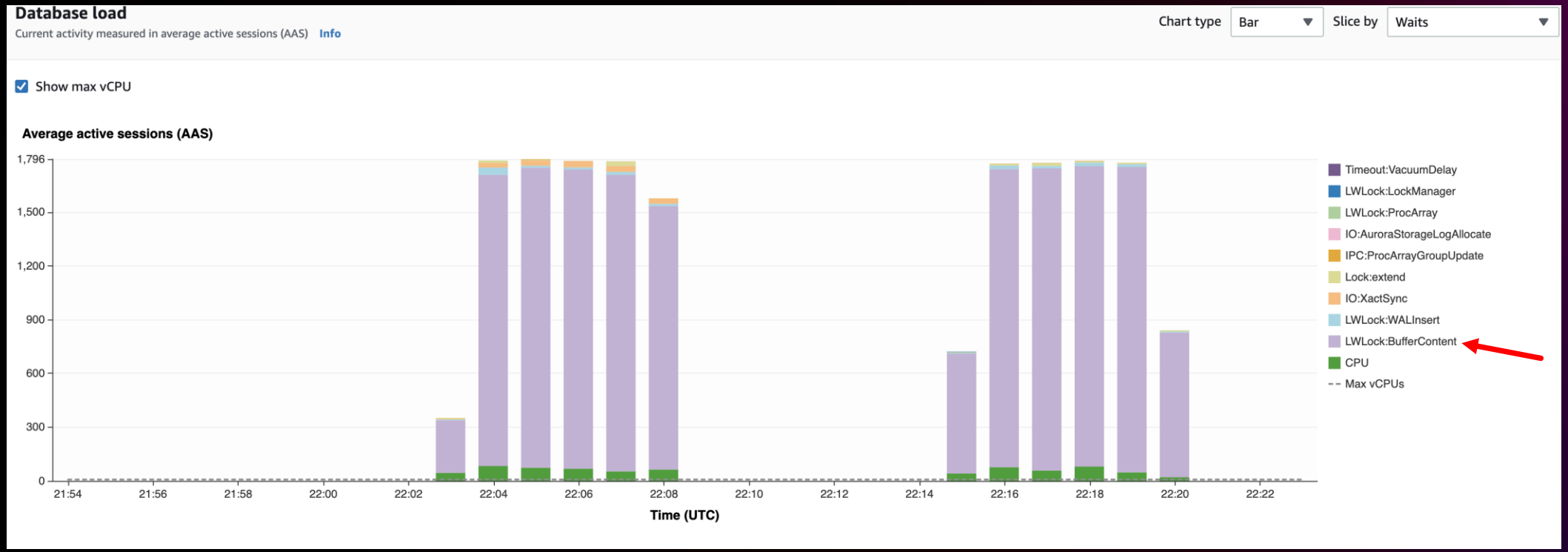
Subtransactions SLRU cache contention

Likely cause	Optimization
Excessive use of SAVEPOINTs, EXCEPTIONs by large concurrent sessions ORM tools with subtransactions setting enabled Long running transactions	Reduce SAVEPOINTs & EXCEPTIONs Turn off subtransactions parameter in ORM tools # of subtransactions < 64

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

LWLock: BufferContent



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

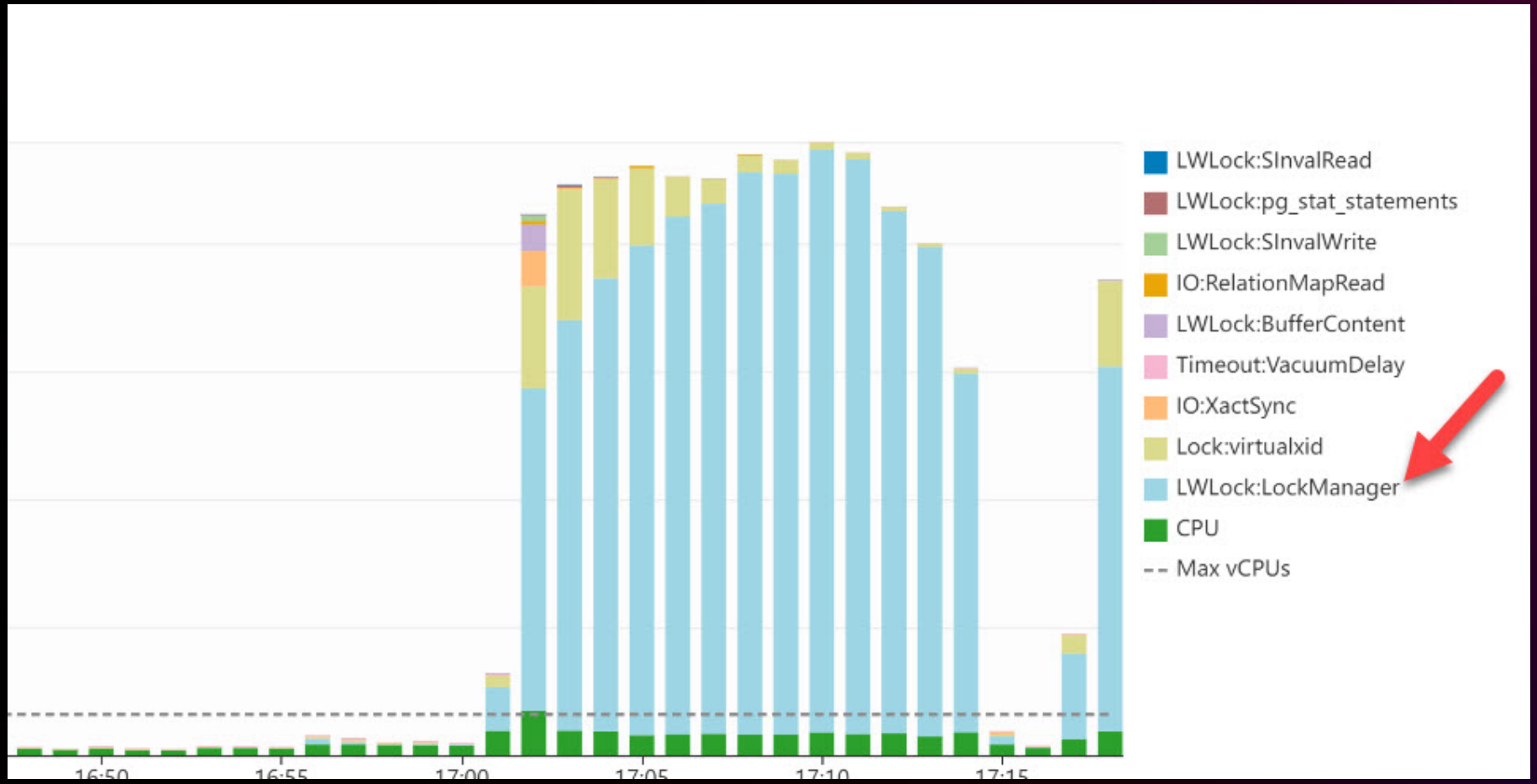
LWLock: BufferContent

Likely cause	Optimization
Large concurrent updates to same block Contention can be more pronounced on tables with a lot of indexes Excessive use of foreign key constraints	Remove unused indexes Remove unnecessary foreign key constraints Use declarative hash partitioning Use database connection pooler (Amazon RDS Proxy, PgBouncer)

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

LWLock: LockManager



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

LWLock: LockManager

Likely cause	Optimization
Session’s use of non-fast path lock method	Use partition pruning
Large concurrent sessions accessing partitioned table with multiple indexes or without pruning partitions	Remove unnecessary indexes Keep number of partitions < 100 per table
Connection storm	Refactor queries for fast path locking Use a database connection pooler

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Prepared statements

Generic plans with prepared statements

- Creates custom plan(s) for first 5 executions
- Creates generic plan if its cost is not so much higher than first 5 executions' average plan cost
- plan_cache_mode = force_custom_plan to force generate custom plan

```
prepare test_prepstmt(date) as select * from test_skewed where val1 = $1;
-- 1 to 5 runs
explain (analyze,settings, costs) execute test_prepstmt('2021-09-26'::date);
Index Scan using test_skewed_val1_idx on test_skewed
   (cost=0.29..10.28 rows=17 width=22) --> 0.00016666666 * 103000 = 17.17
   (actual time=0.026..0.037 rows=32 loops=1)
   Index Cond: (val1 = '2021-09-26'::date)
 --6th run
 explain (analyze, settings, costs) execute test_prepstmt('2021-09-21'::date);
 Index Scan using test_skewed_val1_idx on test_skewed
   (cost=0.29..176.32 rows=3679 width=22) --> 0.1941 * 103000 = 19992.3
   (actual time=0.018..75.644 rows=20031 loops=1)
    Index Cond: (val1 = $1)
```

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Q&A

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

Nathan Fuzi
nfuzi@amazon.com

Krishna Sarabu
ksarabu@amazon.com



Please complete the session survey in the **mobile app**

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.