The journey towards active-active replication in PostgreSQL

Jonathan Katz
Principal PM - Tech
AWS

Amit Kapila
Senior Director
Fujitsu

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

- Overview of PostgreSQL replication
- Active-active replication: Use cases and requirements
- Evolution of logical replication to support "active-active"
- Roadmap for PostgreSQL to support active-active

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Overview of PostgreSQL replication

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What is replication?

- Copying data between systems
- Physical replication
  - Copies data exactly as it appears on disk
  - Only works between same major versions of PostgreSQL
- Logical replication
  - Copies data in a format that can be interpreted by other systems
    - "pgoutput" is the default; can create your own "decoding plugins" (e.g. wal2json)
  - Publisher / subscriber model
  - Can replicate between heterogeneous systems

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Current replication use-cases for PostgreSQL

- High availability
- Load balancing read queries
- Change data capture (CDC)
- Extract-transform-load (ETL)
- Data warehousing
- Online major version upgrades
- System migrations
- Data residency (to a degree)

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Replication deployment models

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Active-standby deployment model

- One primary (active), one or more replicas (standby)
- Choice of synchronous / quorum commit or asynchronous
- Use-cases
  - High availability
  - Read load balancing

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Active-standby advantage and tradeoffs

Advantages

- Simple consistency model: one "source of truth"
- Simple for application development

Tradeoffs

- "Extra work" in promoting a standby
  - Heartbeat
  - Determine "best available" standby
  - Write traffic redirection

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Active-active deployment model

- One or more primaries (active) that replicate between each other
  - Can also include standbys, but not in "high availability set"
- Use-cases
  - High availability
  - "Blue / green" deployments (upgrades, application changes)
  - System migrations

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Active-active advantage and tradeoffs

Advantages	Tradeoffs
• "No failover" – redirect write traffic	• Requires conflict detection / resolution • Applications need to be designed for active-active

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What does PostgreSQL need to support active-active?

- PostgreSQL already supports active-active*
- Logically replicate between partitions across different publishers
- Some extensions / 3rd party tools provide "active-active" support
- (Spoiler: PostgreSQL 16 supports bidirectional replication)

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What does PostgreSQL need to *better* support active-active?

- Features that allow PostgreSQL to support active-active natively:
  - Replication of all/most objects
  - Replication of all/most commands
  - Improvements to conflict detection
  - Conflict resolution / conflict statistics
  - Node synchronization
  - (Two-phase commit (2PC) transaction manager?)

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Evolution of logical replication to support active-active

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Evolution of logical replication in PostgreSQL

9.4 — 2014
- Logical decoding
- Replica identity
- Replication slots

9.5 — 2016 (Jan)
- Track commit timestamp
- Track replication progress

9.6 — 2016 (Sep)
- Support generic WAL messages for logical decoding

10 — 2017
- Logical Replication

11 — 2018
- Replicate TRUNCATE
- Reduce memory usage

12 — 2019
- Allow replication slots to be copied

13 — 2020
- logical_decoding_work_mem
- Replicate partitioned table
- max_slot_wal_keep_size

14 — 2021
- Streaming of in-progress transactions
- Decoding of prepared transactions
- Allow data to be transferred in binary mode
- Performance improvements

15 — 2022
- Replication of prepared transactions
- FOR ALL TABLES IN SCHEMA
- Row filter/Column Lists
- SKIP transaction
- Disable subscription on error

Logical replication enhancements in PostgreSQL 16

Allow filtering data based on origin during replication

Syntax

```
CREATE SUBSCRIPTION sub1 CONNECTION ... PUBLICATION pub1 WITH (origin = none);
```

- options: `none`, `any`
- Setting `origin` to `none` means that the subscription will request the publisher to only send changes that don't have an origin.
- Setting `origin` to `any` means that the publisher sends changes regardless of their origin.
- The default is `any`.
- This allows to setup n-way logical replication as it can be used to prevent loops when doing bi-directional replication

Bi-directional setup

Publisher

```
CREATE TABLE mytbl(c1 int primary key);
CREATE PUBLICATION mypub FOR TABLE mytbl;
```

Subscriber

```
CREATE TABLE mytbl(c1 int primary key);
CREATE SUBSCRIPTION mysub CONNECTION 'dbname=postgres' PUBLICATION mypub;
CREATE PUBLICATION mypub FOR TABLE mytbl;
```

Publisher

```
CREATE SUBSCRIPTION mysub CONNECTION 'dbname=postgres port=5444' PUBLICATION mypub;
INSERT INTO t1 values(1);
```

Publisher's server log

```
ERROR:  duplicate key value violates unique constraint "t1_pkey"
DETAIL: Key (c1)=(1) already exists.
```

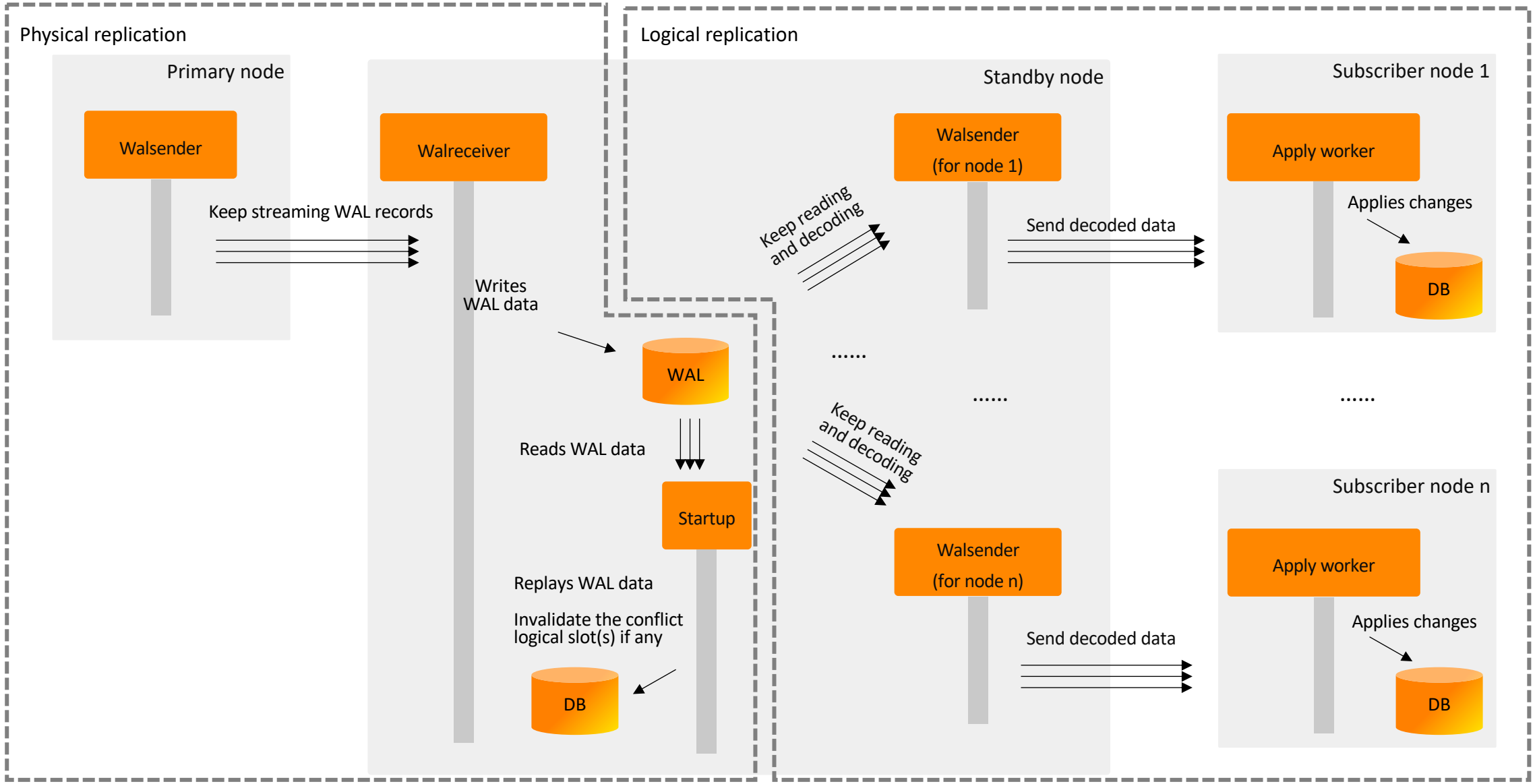
Publisher

```
ALTER SUBSCRIPTION mysub SET(origin=none);
```

Allow logical decoding from standby

- This requires `wal_level = logical` on both primary and standby.
- Invalidate logical slots on standby
  - when the required rows are removed on primary.
  - when the wal_level on the primary server is reduced to below logical.
- Check the conflicting field in pg_replication_slots to know if the slot is invalidated due to conflict.
- This feature allows workload distribution by allowing subscribers to subscribe from standby when primary is busy.

Logical replication from standby



Allow non-superusers to create subscriptions

- Non-superusers must have been granted pg_create_subscription role.
- Non-superusers must additionally have CREATE permissions on the database in which the subscription is to be created.
- Non-superusers are required to specify a password for authentication.
- Superusers can set password_required=false for non-superusers that own the subscription.

Allow apply process to perform operations with the table owner's privileges

- `CREATE SUBSCRIPTION sub1 … WITH (run_as_owner = false);`
- The subscription owner needs to be able to SET ROLE to each role that owns a replicated table.
- If the table owner doesn't have permission to SET ROLE to the subscription, SECURITY_RESTRICTED_OPERATION is imposed.
- If the subscription has been configured with `run_as_owner = true`, then no user switching will occur.
- This also means that any user who owns a table into which replication is happening can execute arbitrary code with the privileges of the subscription owner.

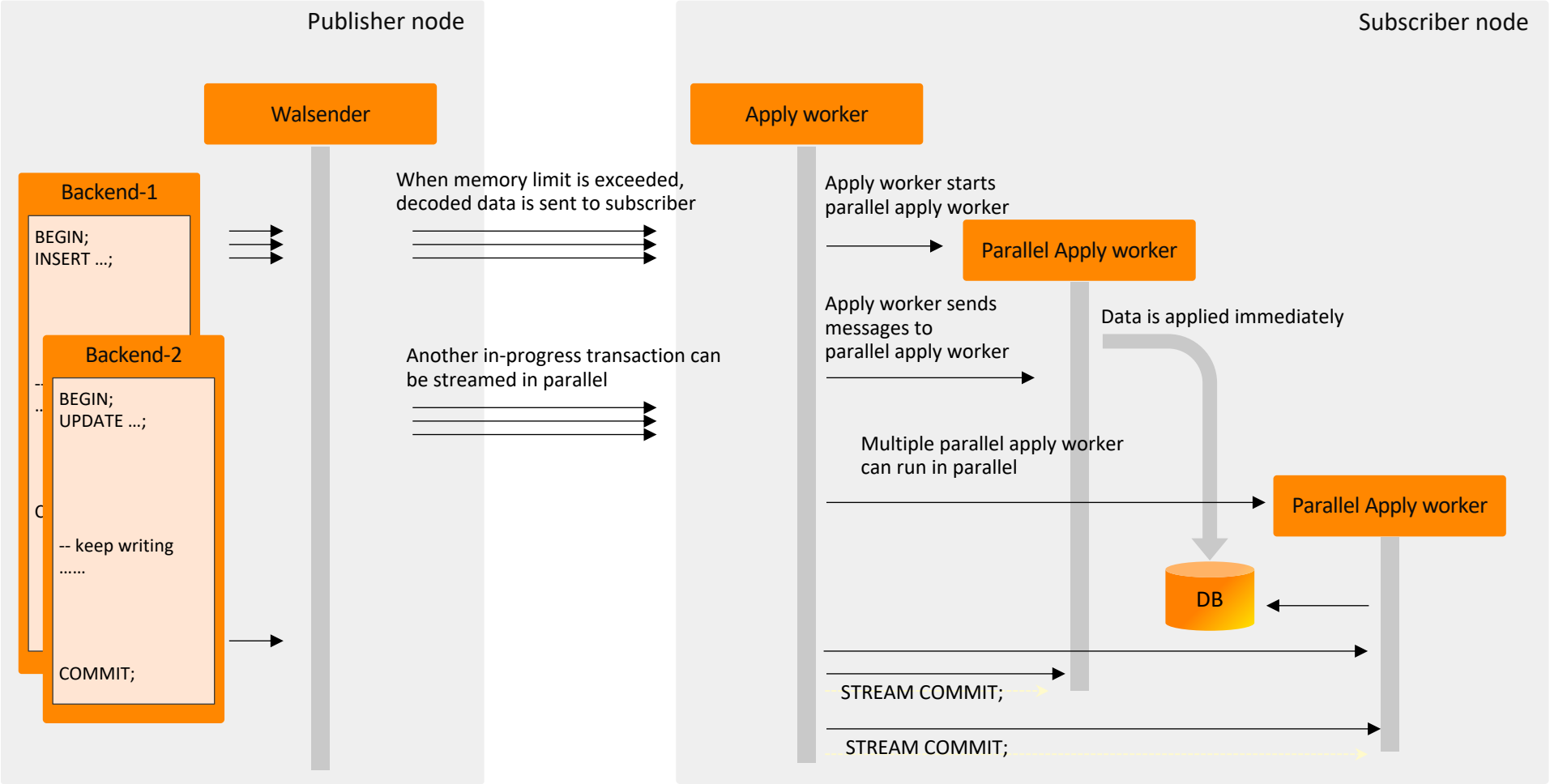
Allow large transactions to be applied in parallel

- Performance improvement in the range of 25-40% has been observed.
- Each large transaction is assigned to one of the available workers. The worker remained assigned till the transaction completes.
- max_parallel_apply_workers_per_subscription indicates the maximum number of parallel apply workers per subscription.

Syntax

```
CREATE SUBSCRIPTION sub1
CONNECTION ...
PUBLICATION pub1 WITH (streaming = parallel);
```

Parallel Apply workers



Allow the use of indexes other than PK and REPLICA IDENTITY on subscriber

- Prior to this feature, using REPLICA IDENTITY FULL on the publisher can lead to a full table scan per tuple change on the subscriber when REPLICA IDENTITY or PK index is not available.
- The index that can be used must be a btree index, not a partial index, and it must have at least one column reference.
- The performance improvement is proportional to the amount of data in the table.

Allow logical replication to copy tables in binary format

Syntax

```
CREATE SUBSCRIPTION sub1
CONNECTION ...
PUBLICATION pub1 WITH (binary = true);
```

- Prior to V16, this option only allows replication to replicate tables in binary format.
- Copying tables in binary format may reduce the time spent depending on column types.
- A binary copy is supported only when both publisher and subscriber are v16 or later.

Roadmap to support active-active deployments

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Logical replication in PostgreSQL 17 and beyond

- DDL replication
  - Deparse the command to pass it in a standard format like JSON
  - Replication of DDL commands
  - Initial sync
- Replication of sequences
- Synchronization of replication slots to allow failover
- Upgrade of logical replication nodes
- Reuse of tablesync workers
- Time-delayed replication

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Features to support active-active deployments

- Logical replication of commands
- Logical replication of sequences
  - Global sequences
- Conflicts
  - Detection
  - Last commits wins resolution
  - Monitoring
- Node initialization, synchronization, resynchronization, pause / resume
- Performance
  - Decoding
  - Apply process
  - Lag catch up

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.