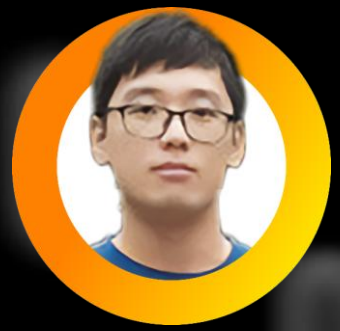
FUJITSU

How to Hack on Logical Replication

- Insights from contributors



Zhijie Hou

Senior Engineer

Hayato Kuroda

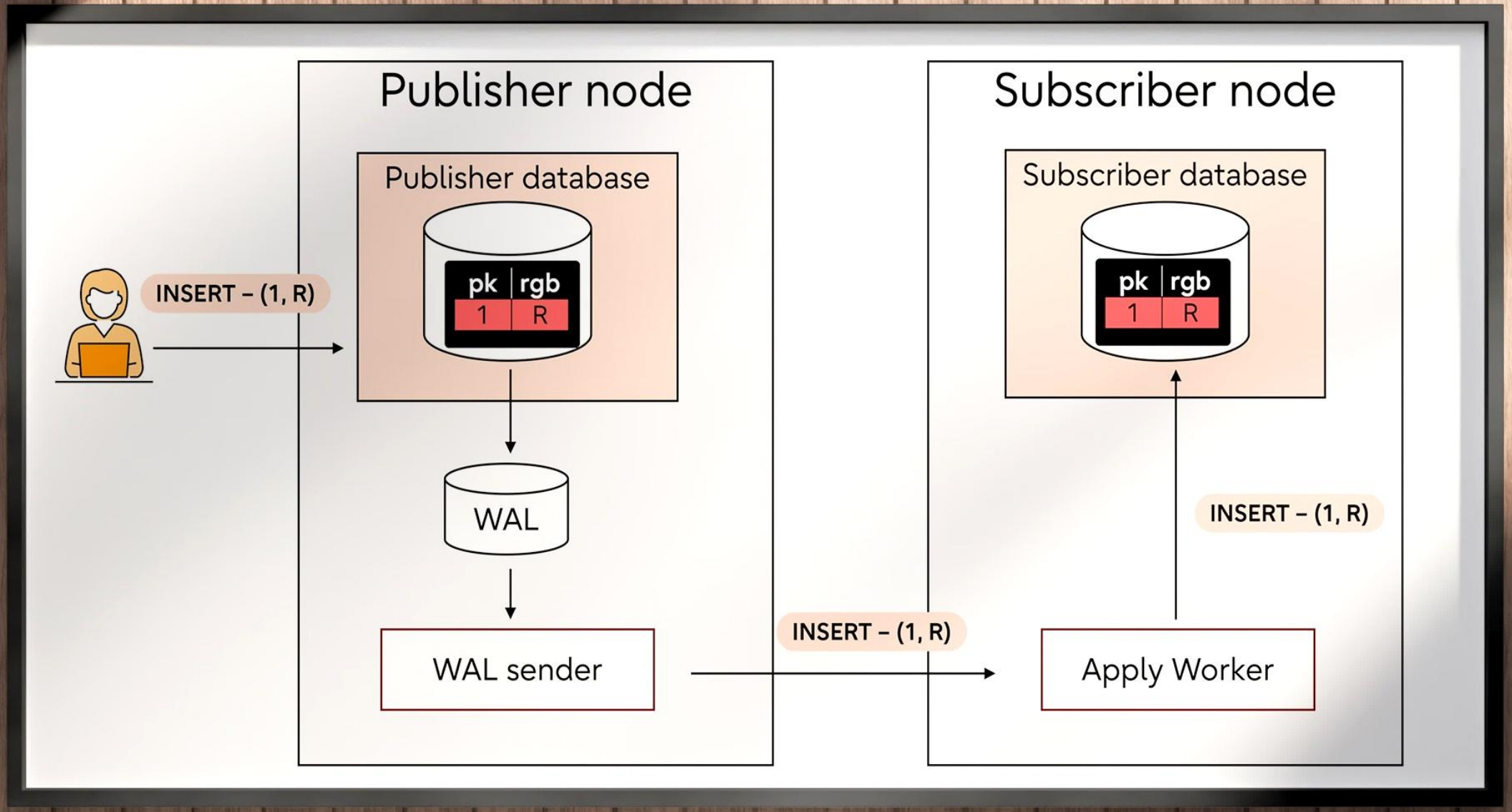
Data Engineer

Agenda

- What’s logical replication
- Why, what to hack ?
- How to hack ?

What is Logical Replication?

FUJITSU



What Is Logical Replication?

- Publisher

```
CREATE TABLE users(id int);
CREATE PUBLICATION mypub FOR TABLE users;
```

- Subscriber

```
CREATE TABLE users(id int);
CREATE SUBSCRIPTION mysub
  CONNECTION 'host=192.168.1.100
              dbname=postgres
              user=repuser
              password=rep123'
  PUBLICATION mypub;
```

Why hack Logical Replication?

FUJITSU

Logical replication is being adopted more widely, yet it still has significant potential for improvement

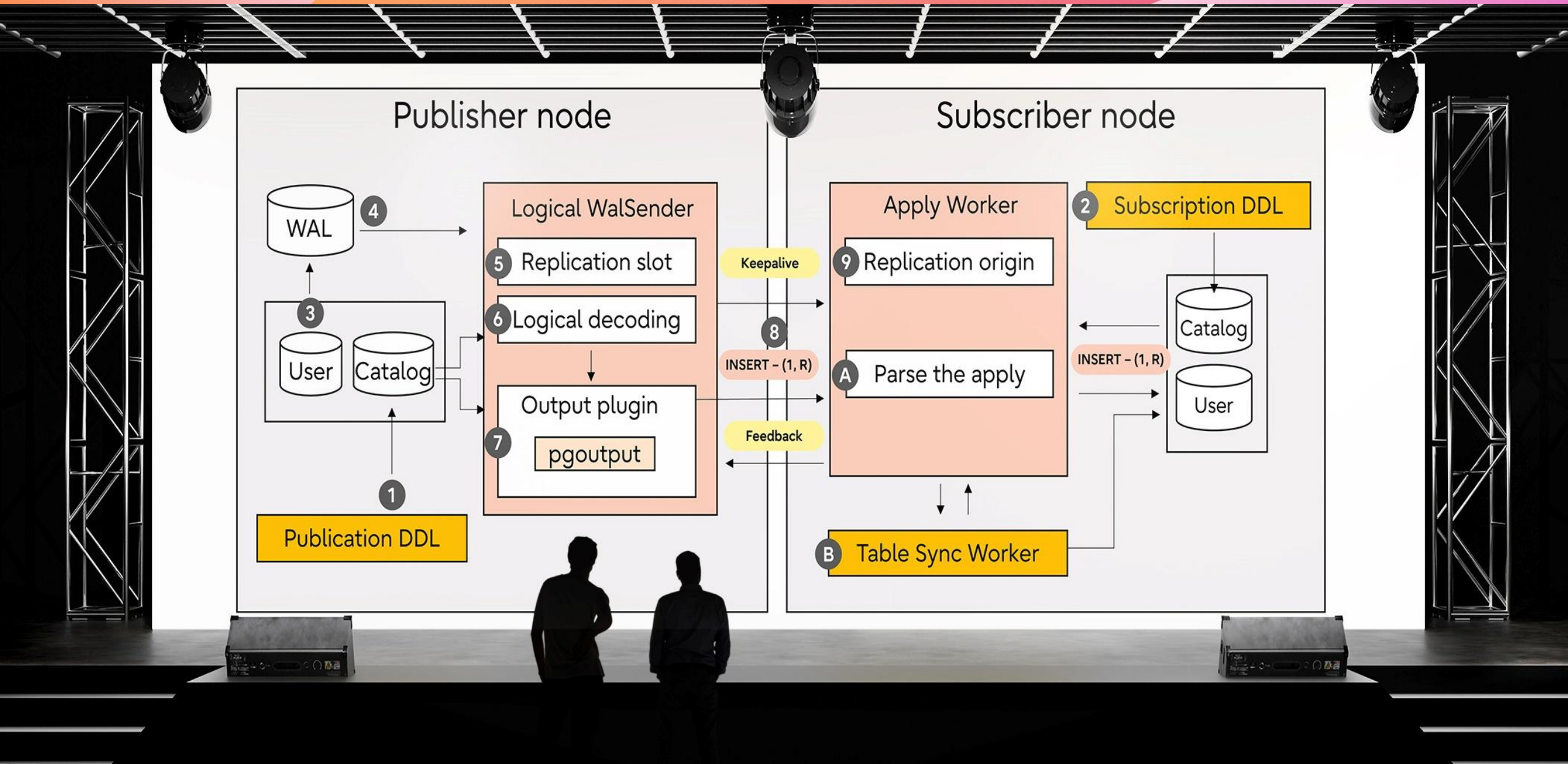
Key issues that still exist:

- Performance bottlenecks (e.g., decoding, apply)
- Lack of many features that other databases already have

Opportunities for contributors to make impactful improvements

What to hack ?

FUJITSU



What to hack ?

FUJITSU

Performance improvement

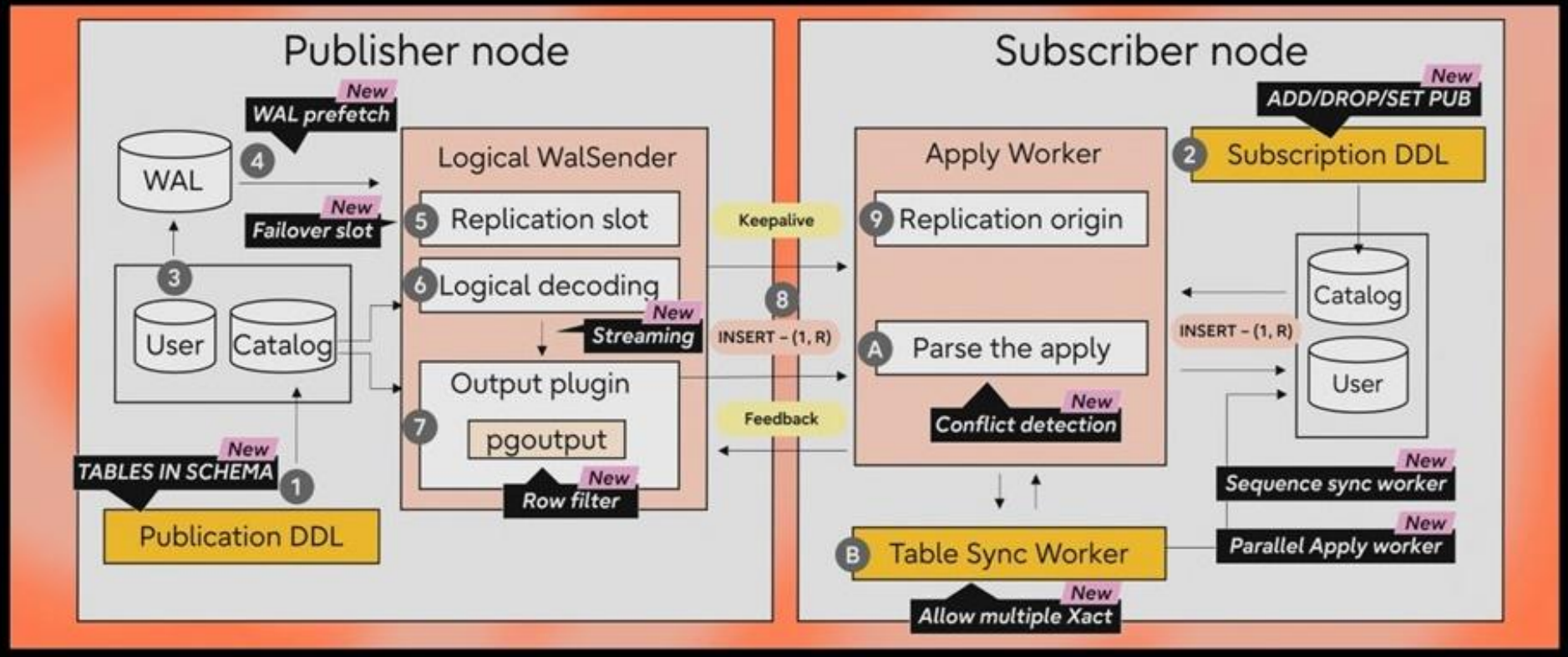
Bug fixes

Code refactoring

New features

New features

FUJITSU



Performance improvement

FUJITSU

Focus on performance-sensitive processes

- Performance friction: locks, latency, and slow SQL
- User impact: degraded responsiveness in critical environments

Generate profiles

- perf record/report

Perform performance tests regularly

Performance improvement

1/3

- Set up the publication

```
CREATE TABLE users(id int);
CREATE PUBLICATION mypub FOR TABLE users;
SELECT pg_create_logical_replication_slot('origin', 'plugin');
```

- Workload – pgbench
- Save the end position

```
SELECT pg_current_wal_lsn();
```

- Evaluate the performance

```
time pg_recvlogical -S consume --endpos 0/xxxxxx --start
                    -o proto_version=3
                    -o publication_names=mypub -f /dev/null
```

Performance improvement

2/3

- Set up the publication

```
CREATE TABLE users(id int);
CREATE PUBLICATION mypub FOR TABLE users;
```

- Set up the subscription

```
CREATE TABLE users(id int);
CREATE SUBSCRIPTION mysub .. PUBLICATION mypub
        WITH (enable=false);
```

- Workload – pgbench
- Evaluate the performance

```
ALTER SUBSCRIPTION mysub .. ENABLE;
-- Count the time till when the apply worker catches up.
```

Performance improvement

3/3

- Commit 6ce1608
  - Reduce relcache access in WAL sender streaming logical changes
- During a test for what the walsender behaves when no changes are replicated, found something weird

```
--12.83%--pgoutput_change
  |--11.84%--get_rel_sync_entry
    |--4.76%--get_rel_relispartition
    |--4.70%--get_rel_relkind
```

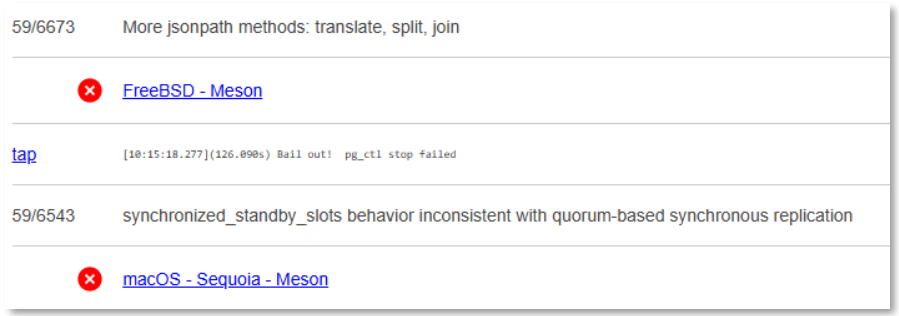
- Found the issue is that we were unconditionally calling these function even if the change is not replicated

Bug fix

Track the PostgreSQL BuildFarm Recent Failures

Branch: master			
Alias	System	Status	Flags
dodo	Raspbian GNU/Linux 10 gcc 15.x (nightly build)	10:42 ago InstallCheck-C [a63bbc8] Details	
drongo	Windows Server 2019 Visual Studio 2019	1 day 05:43 ago subscriptionCheck [80156ce] Details	
alimoche	Debian GNU/Linux 10 gcc 8.3.0	1 day 11:59 ago pgsql-Git [] Details	

Track the PostgreSQL Patch Tester



It would be easier to catch fresh bugs after a new commit

Bug fix

- When a new feature is committed, good to analyze whether it has any intersection with logical replication
- Commit fc6600f
  - Fix replica identity check for MERGE

> Fix replica identity check for MERGE.
>
> When executing a MERGE, check that the target relation supports all actions mentioned in the MERGE command. Specifically, check that it has a REPLICA IDENTITY if it publishes updates or deletes and the MERGE command contains update or delete actions. Failing to do this can silently break replication.
>
> Author: Zhijie Hou <houzj.fnst@fujitsu.com>

Code refactoring

FUJITSU

Reduces a considerable amount of code duplication

Easier to
read
and maintain

Extracts duplicate code into a centralized function

Reduces maintenance burden by eliminating the need to update multiple places when adding common features

Refactors for better understandability

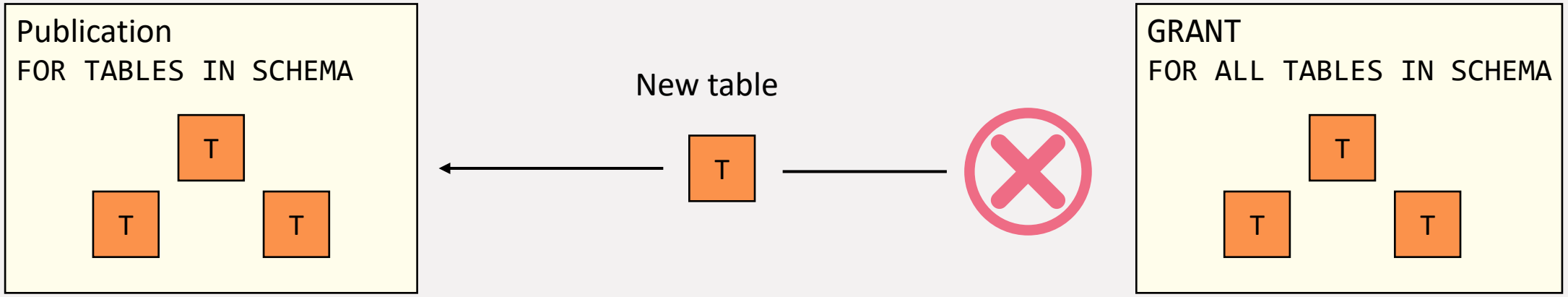
Makes it easier for future developers to build additional functionality on top

How to hack the Publisher

Extending publication DDL

FUJITSU

- Coding is not difficult
- Consider the semantics and avoid misleading
- e.g., `FOR TABLES IN SCHEMA`
  - Originally proposed as `FOR ALL TABLES IN SCHEMA`
  - But it works differently with `GRANT … FOR ALL TABLES IN SCHEMA`
  - When new table is created in the schema…



Restriction bypass

FUJITSU

DDL-time validation may look straightforward, but lazy evaluation may be better in some cases

e.g., row filter, **`CREATE PUBLICATION … WHERE …`**

- Only replica identity (RI) columns are usable in the `WHERE` clause
- Initially we tried to validate all cases at DDLs, but found it's a burden
  - Must detect not only on **`CREATE/ALTER PUBLICATION`**, but also **`DROP INDEX`**/ **`ALTER TABLE ... REPLICA IDENTITY`**/ **`ATTACH PARTITION`** and more!

Upgrade / downgrade compatibility

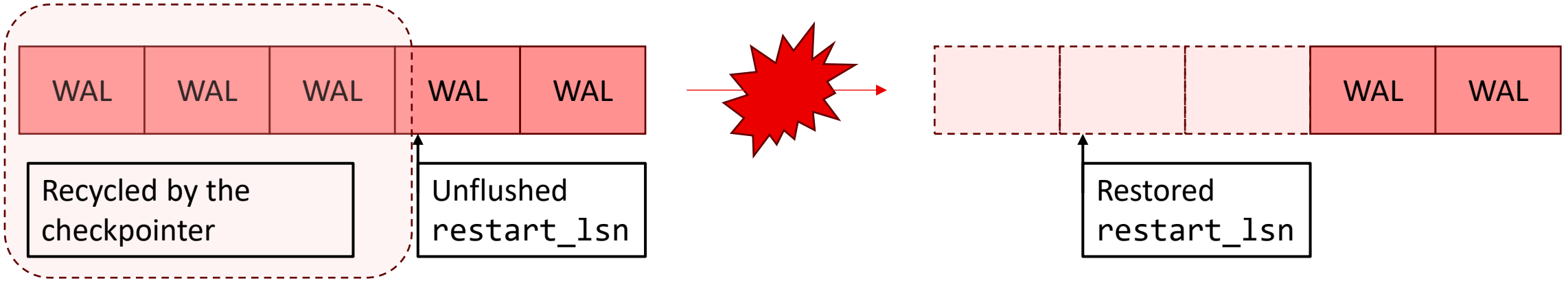
FUJITSU

- Publisher or subscriber may be the older version
- Be careful with SQL API signatures
- Keep a path for older versions
- e.g. Table synchronization
  - Initially tried to just replace the SQL function signatures, but the publisher side might be the older version
  - Therefore, the code still has the path to communicate with older nodes

Special handling for replication slot

FUJITSU

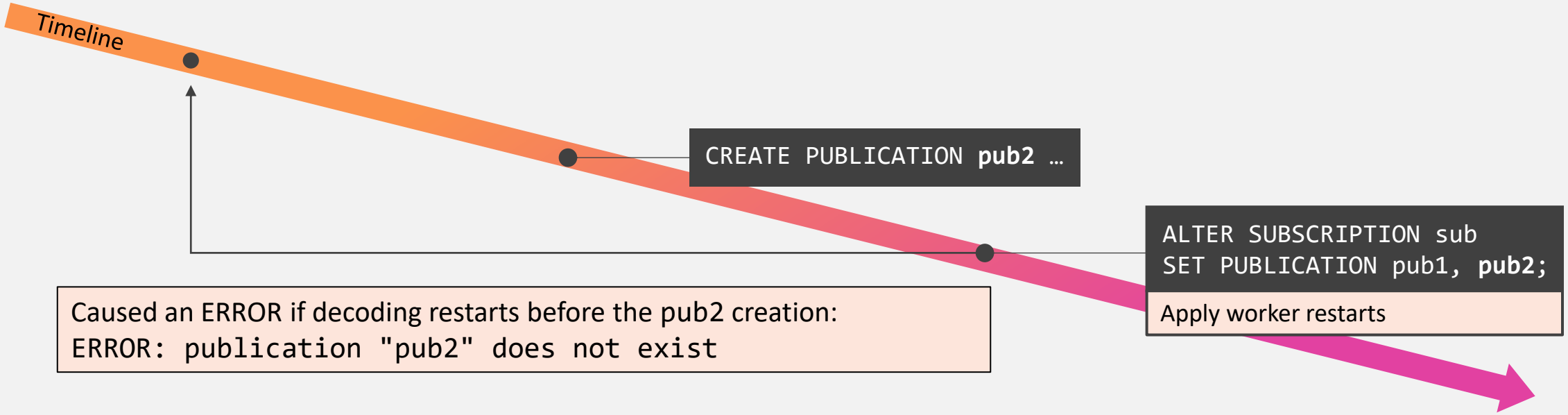
- Replication slot is **not** a transactional object
- Creating slots cannot be reverted
  - A slot may be retained if `CREATE SUBSCRIPTION` failed at the later point
- Updating slots does not guarantee flushing to disk - *ca307d5*
  - Checkpointer could recycle WALs based on the unflushed `restart_lsn`
  - If the backend crashes, `restart_lsn` might point to recycled WAL



Catalog lookups in output plugins

FUJITSU

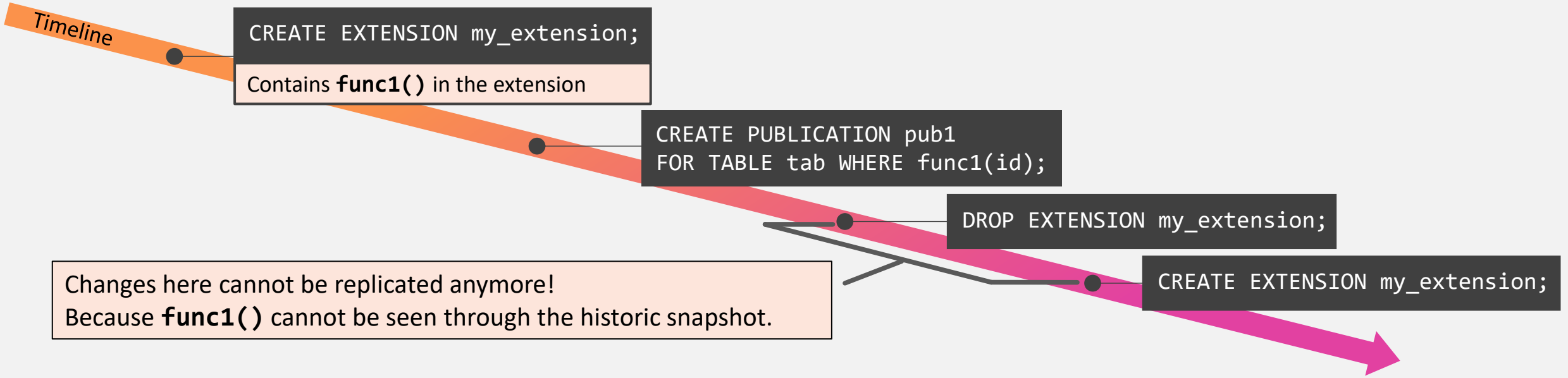
- Logical decoding uses the **historic** snapshot
- It respects the snapshot of decoding transaction
- e.g., `ALTER SUBSCRIPTION ... SET PUBLICATION`
  - There was a bug: replication cannot be resumed after the command



Catalog lookups in output plugins | Another example

FUJITSU

- Row filters can use only immutable built-in functions
- Manually created functions are prohibited
  - It uses the historic snapshot, may behave unexpectedly
- If we allowed functions from extensions…



Replica identity can be different on publisher and subscriber

FUJITSU

- Publisher and subscriber can have different RI settings
  - Subscriber must be able to identify rows using the key columns sent by the publisher
  - e.g., Publisher has two-column PK (left) and
    Subscriber has a different PK (right)

Publisher

```
order_details
    (order_id INT,
     product_id INT, …
     PRIMARY KEY (order_id, product_id))
```

Subscriber

```
order_details
    (order_id INT,
     product_id INT, …
     PRIMARY KEY (order_id))
```

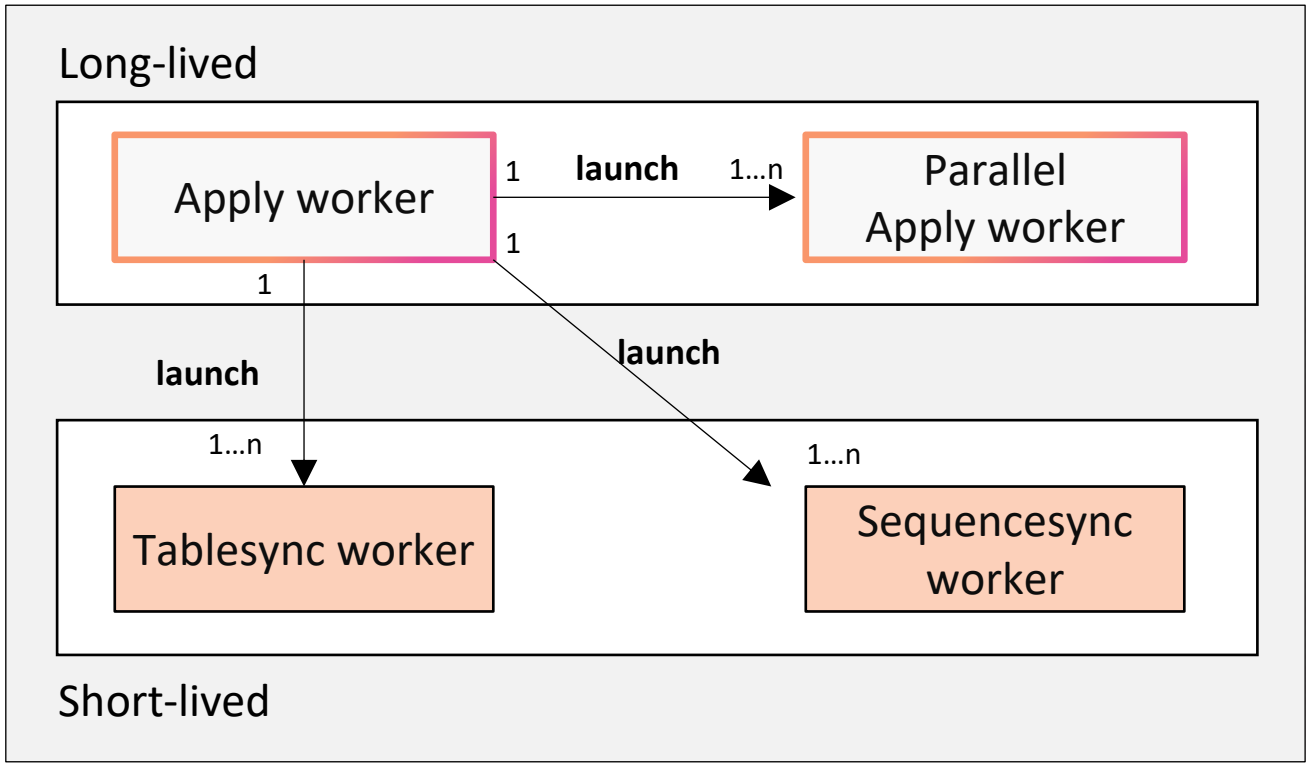
How to hack the Subscriber

Apply Worker mechanics

FUJITSU

Please remember the architecture before hacking

- (Normal, leader) apply worker
- Parallel apply worker
- Tablesync worker
- Sequencesync worker



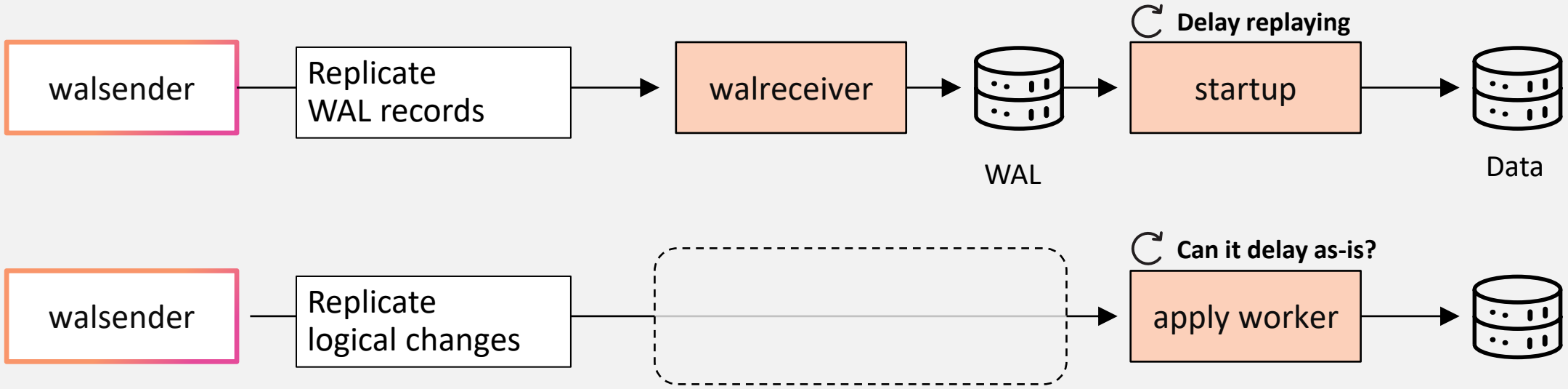
Adapting physical replication ideas

FUJITSU

- Remember that the two replication types are different
- e.g., delayed replication

https://about.gitlab.com/blog/delayed-replication-for-disaster-recovery-with-postgresql/

  - Useful for protecting against user mistakes
  - Can we adopt this for logical replication? – **No**



Multiple worker types mean multiple review targets

FUJITSU

- Adding a new type of worker is sometimes needed
- Check existing code carefully and follow procedures
- Recently fixed bugs related to this point:
  - `e41d954` – Fix lock timeout handling for p.a. worker.
    Due to the wrong setting of signal handler.
  - `1528b0d` – Fix inconsistent origin advance for p.a. worker
    Due to the missing process exit callback.

Replication origin

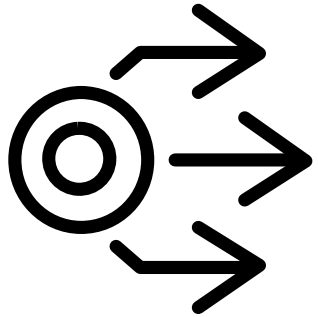
FUJITSU

- Replication origin is used to identify the data source
- If an origin is dropped once and new origin is created, the same origin ID can be assigned.
  - **No way** to distinguish whether the old or the new origin modified old tuples
- Ongoing thread - migration of origins during pg_upgrade
  - Dump `origin_id, name, and remote_lsn` from the old node, and restore them
  - Final goal: allow reporting data conflict correctly

Streaming mode changed what is worth optimizing

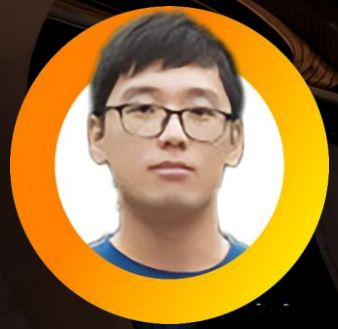
FUJITSU

- Streaming mode was added in PG14
- It is a game changer for large transactions
  Our efforts should focus on extending this mode
  - Parallel apply, improving the mode, etc.
  - Old mode is kept for backward compatibility
- e.g., old patch idea – compression of spilled transaction
  - Potentially useful if streaming is not usable
  - But now streaming mode is enabled by default, the benefit is small



FUJITSU

Thank you



Zhijie Hou

Senior Engineer

Hayato Kuroda

Data Engineer