

Die 10 besten PostgreSQL-Replikationsstrategien für Ihr Unternehmen

Borys Neselovskyi | Sales Engineer, EDB

23 March 2022



Agenda

- How physical and logical replication works in PostgreSQL
- Differences between synchronous and asynchronous replication
- Advantages, disadvantages and challenges in multi-master replication
- Which replication strategy is more suitable to different use cases.



POLL:

Do you currently have external professional support for your databases?

- 1) Yes
- 2) No

Replication/HA options (10 solutions/variations)

- Shared disk failover
- File System (block device) replication
- WAL shipping: Archiving
- WAL shipping: Streaming replication
- WAL shipping with Synchronous commit
- Native logical replication
- Trigger based Primary-Standby replication
- SQL-based Replication Middleware
- Asynchronous multi-master Replication
- Synchronous multi-master Replication

<https://www.postgresql.org/docs/14/different-replication-solutions.html>

No products mentioned:
None forgotten

Legend:

Native Postgres

External tooling

POLL:

Which replication solution are you currently using?

- 1) None
- 2) Native WAL archiving
- 3) Native WAL streaming
- 4) Native synchronous commit
- 5) Native logical replication
- 6) 3rd party replication solution: Replicated disks replication
- 7) 3rd party replication solution: Multi-master replication

“

“If your data is important enough to replicate, then you should also ensure professional support for the databases.”

Michael Willer
EDB

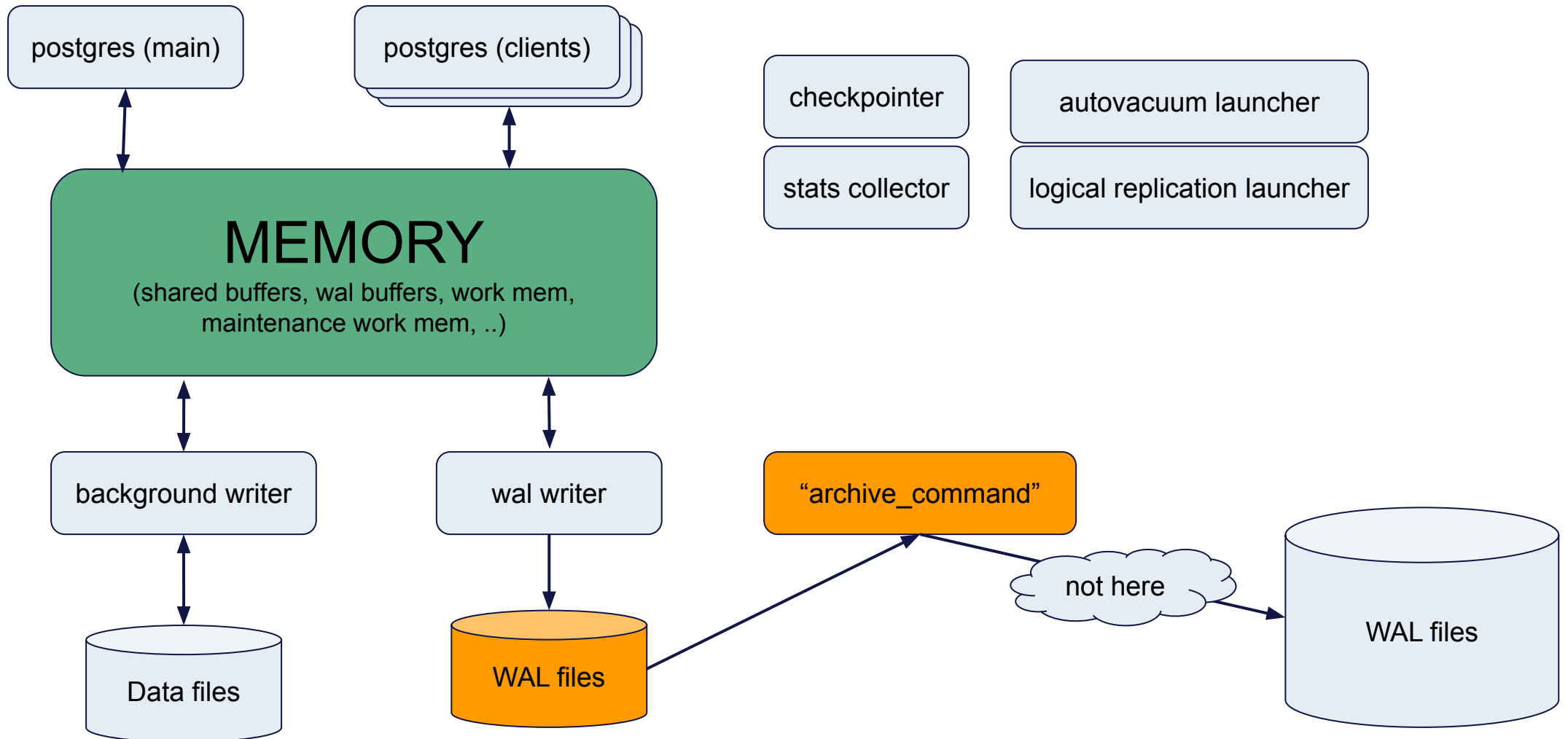


“

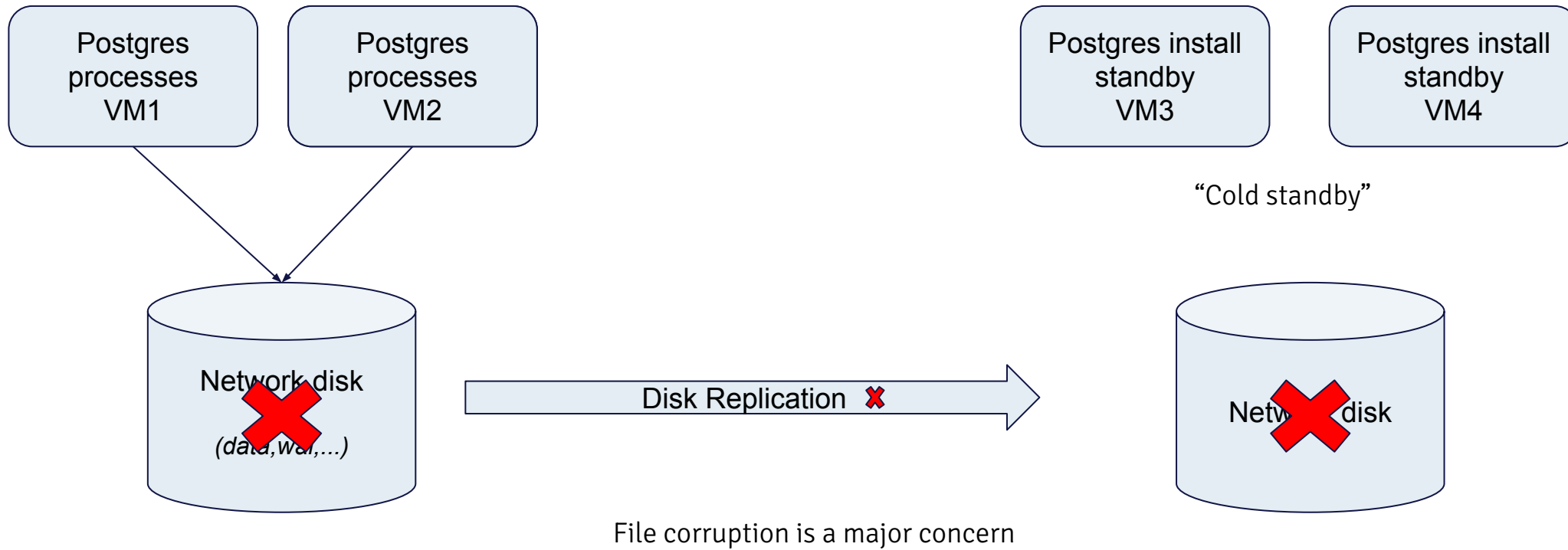
“Your Data is important. That is why it is crucial to keep data redundant and save. Configuration of High Availability Landscapes can be very complex and should be well planned. Therefore, you should make sure that you have competent and well-trained personal and professional support for your databases.”

”

How PostgreSQL works



Shared disk & Disk replication



Shared disk / Disk replication

Use case

Well,? It works

Evaluation

Good:

No overhead on primary

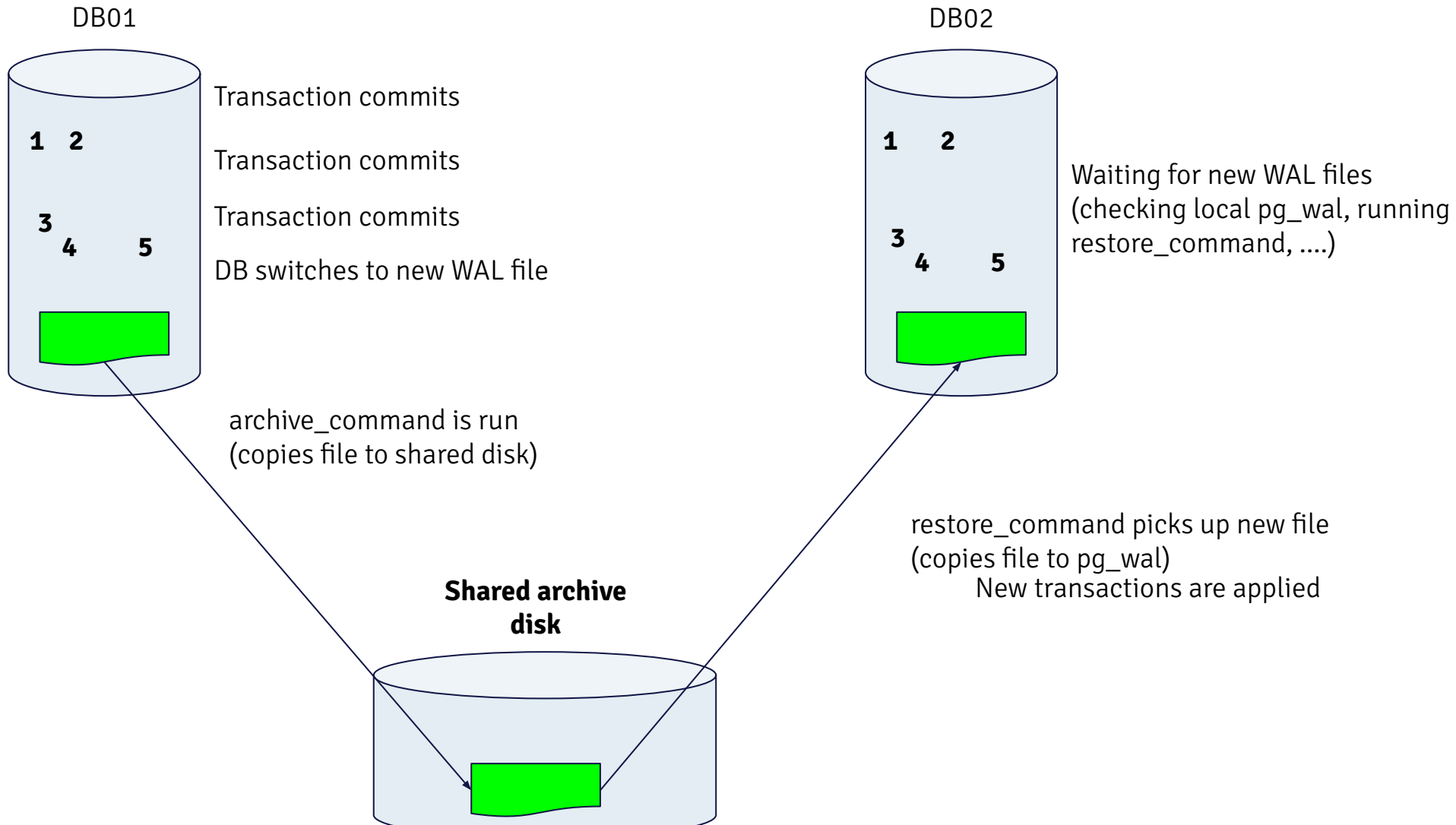
“Easy” to set up for the DBA

Can't lose a transaction (RPO=0?)

Bad:

- Any failover starts with the database going into recovery before opening (RTO might be high)
- Only one server actually accesses the data
- Disk replication has to be **perfect** (all updates done in the correct order)
- *Data corruption in one place*
= *data corruption everywhere*

WAL Shipping: Archiving





WAL Shipping: Archiving

Use case

Easy way to get a copy of the database

Standby doesn't have to be up-to-date to the "minute"

Good solution when communication between the databases isn't possible due to network.

Evaluation

Good:

Very low impact on primary server.

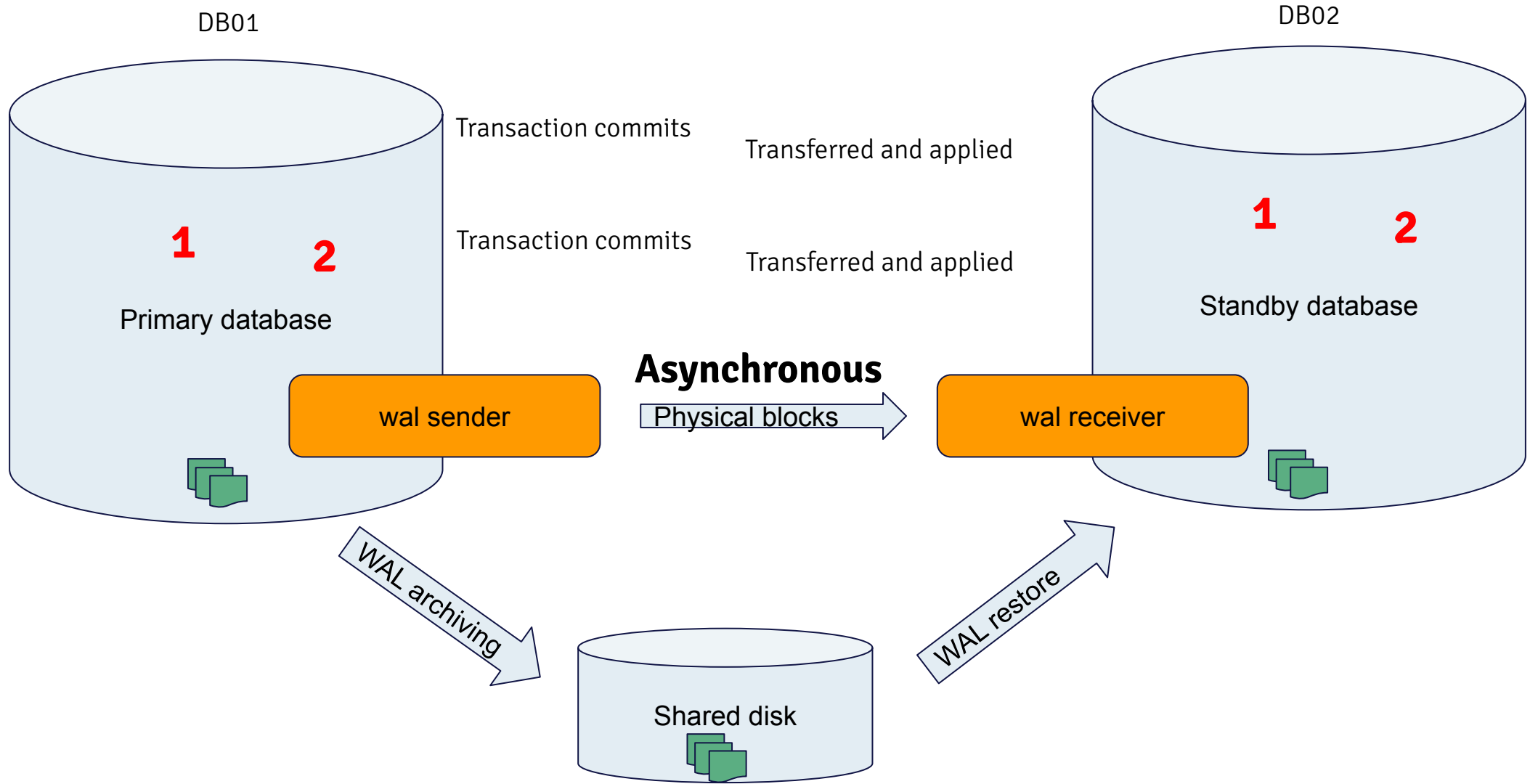
Standby can be used for read/only queries (that don't need up-to-date information).

Bad:

Risk of data loss

(checkpoints frequency can be increased - within reason)

WAL Shipping: Streaming





WAL Shipping: Streaming

Use case

Good for almost all Postgres workloads.

Standby is (almost) identical to the primary database.

Evaluation

Good:

Minor impact on primary server

Standby is almost up-to-date (read/only queries)

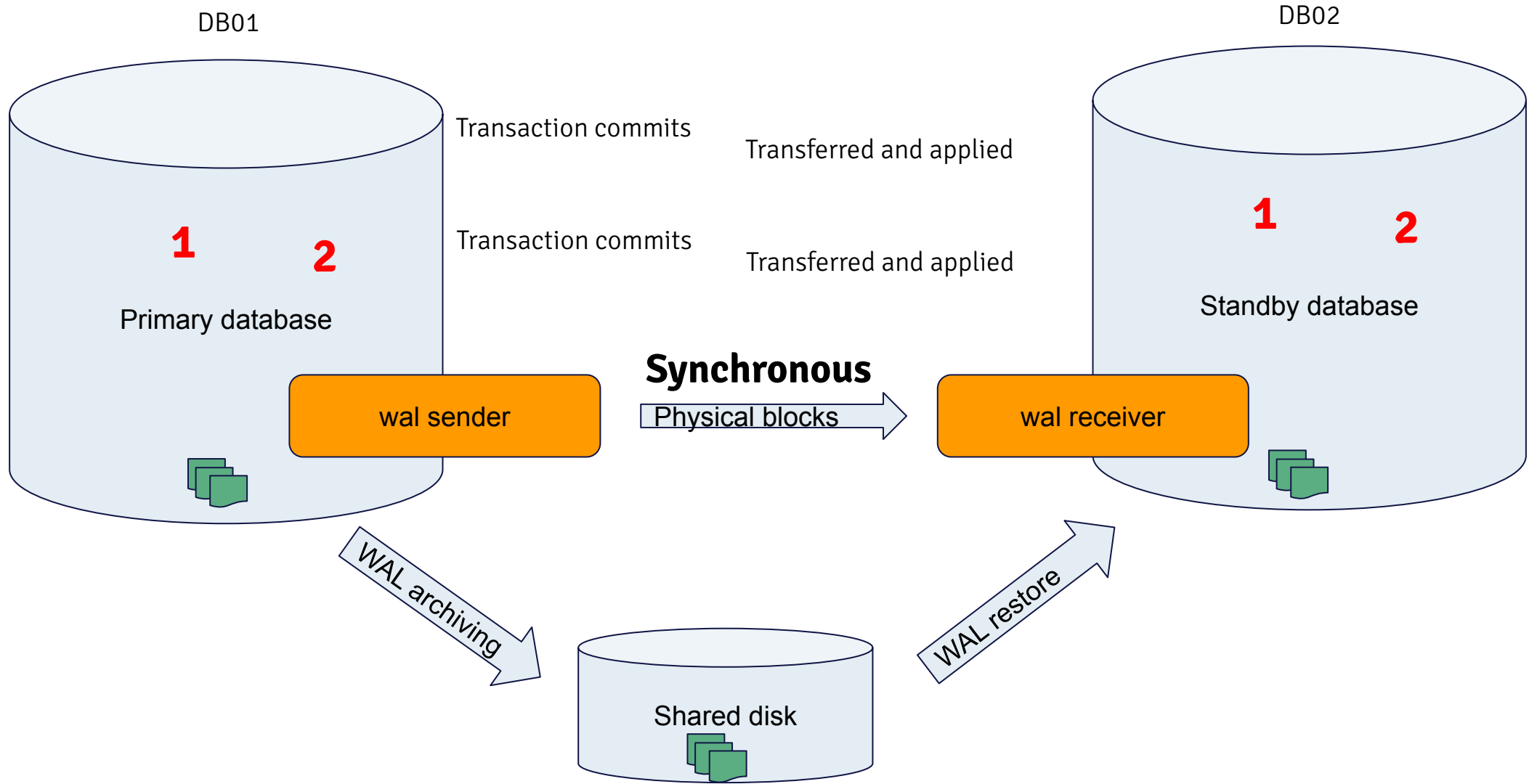
RPO close to 0 (“almost” no data loss)

Bad:

Still a risk of data loss (though a lot smaller than with WAL archiving)

Communication between servers is needed

WAL Shipping: Streaming (a.k.a. synchronous commit)



Synchronous Replication

Settings

Parameter **synchronous_standby_names**:

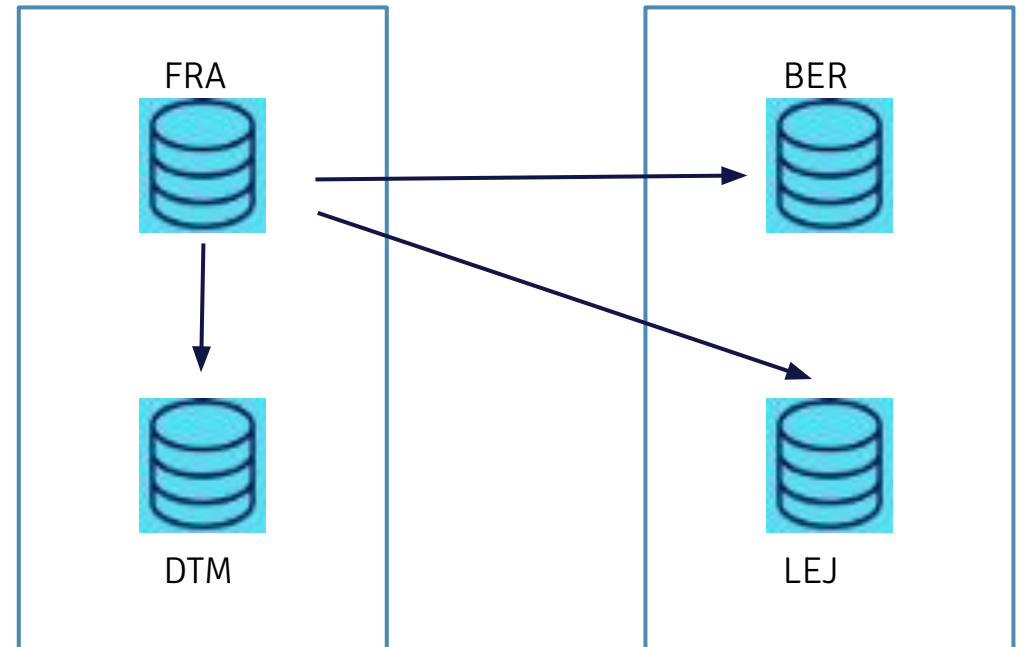
```
FIRST 1 (dtm)
```

```
FIRST 1 (dtm, ber)
```

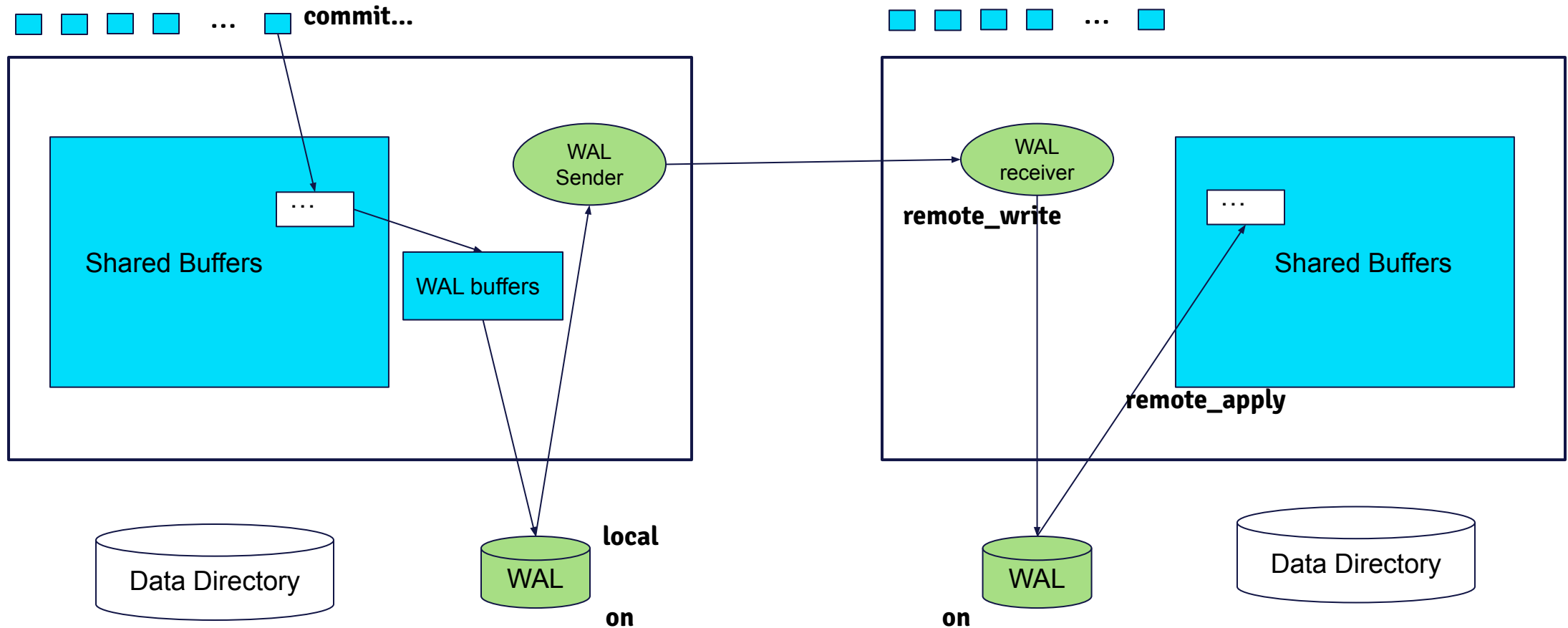
```
ANY 2 (dtm, ber, lej)
```

Parameter **synchronous_commit**:

```
off/local/remote_write/on/remote_apply
```



Reduce data loss → synchronous replication





WAL Shipping: Synchronous commit

Use case

Same as asynchronous streaming

- + need the standby to be fully up-to-date

Evaluation

Good:

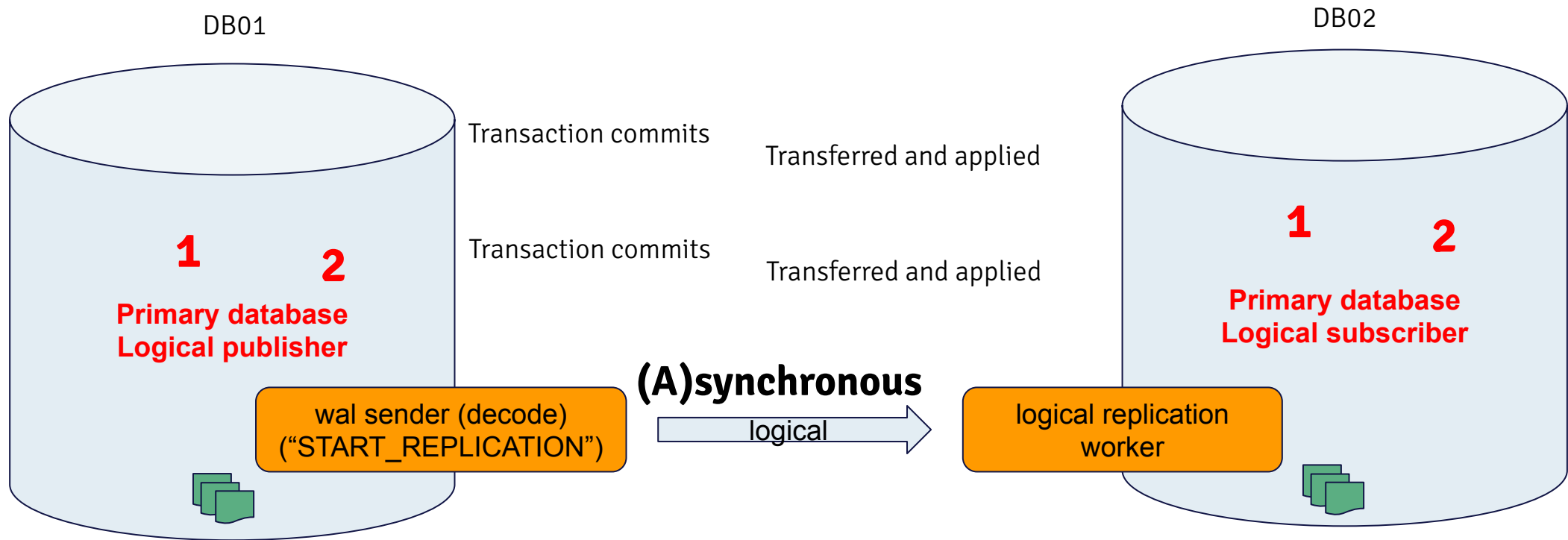
As asynchronous streaming

RPO=0 (no data loss)

Bad:

If standby is unavailable - all commits hang (adding more databases is a workaround, see *synchronous_server_names* for information)

Native logical replication





Native logical replication

Use case

One or more or tables replicated.

Can span postgres versions and OS
(Major version upgrades)

Consolidate several data sources to one
(BI/Reporting server)

INSERT/UPDATE/DELETE/TRUNCATE

Evaluation

Good:

Replicating a select set of tables

Low impact on publishing server

Bad:

No DDL replication

No conflict resolution

REPLICATION IDENTITY is required for updates.
(insert is fine without).

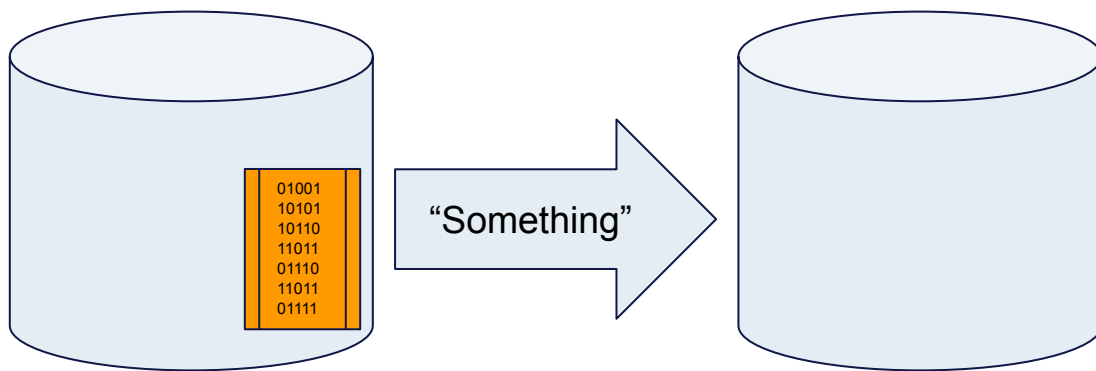
The final four...

- Trigger based Primary-Standby replication
- SQL-based Replication Middleware
- Asynchronous Multimaster Replication
- Synchronous Multimaster Replication

All are non-native

Trigger-based replication

Triggers capture data changes
(Something) picks up the changes and
pushes them to the other database(s).



Good:

Easy to understand and (sometimes) easy to set up.

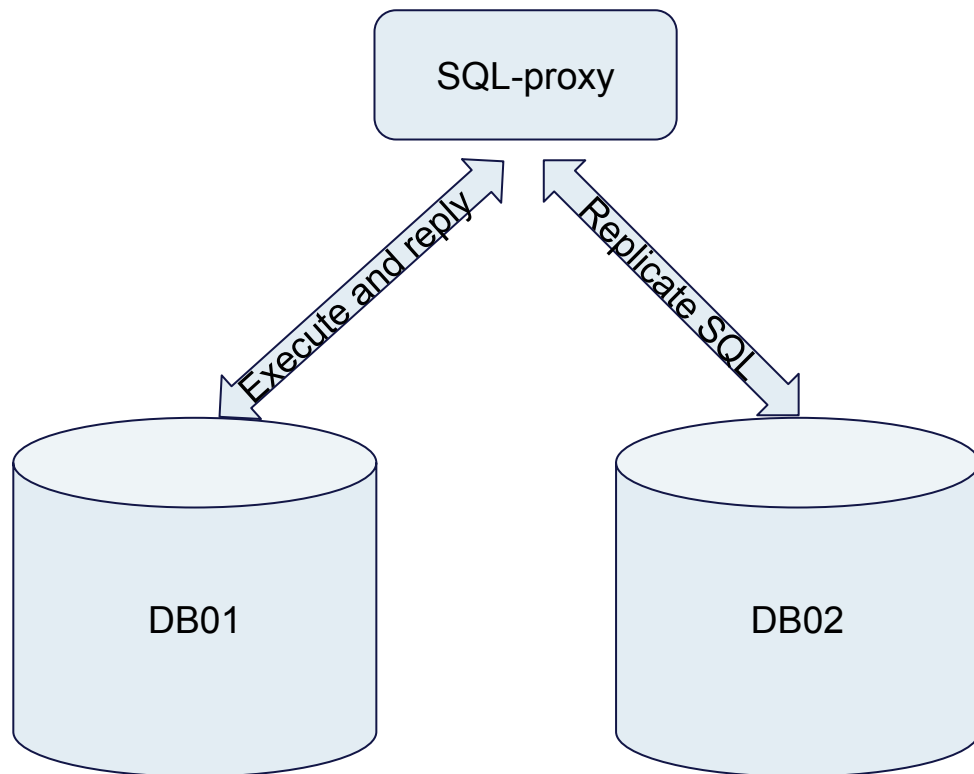
Bad:

Tends to be fragile

Tends to be slower than WAL-decoding solutions

SQL-based replication

Proxy captures data changes
Replicates all updates to other servers



Good:

Easy to understand and (sometimes) easy to set up.

Can provide read-only routing as well

Bad:

Which SQL is an update?
Which isn't?

INSERT INTO
UPDATE

??????

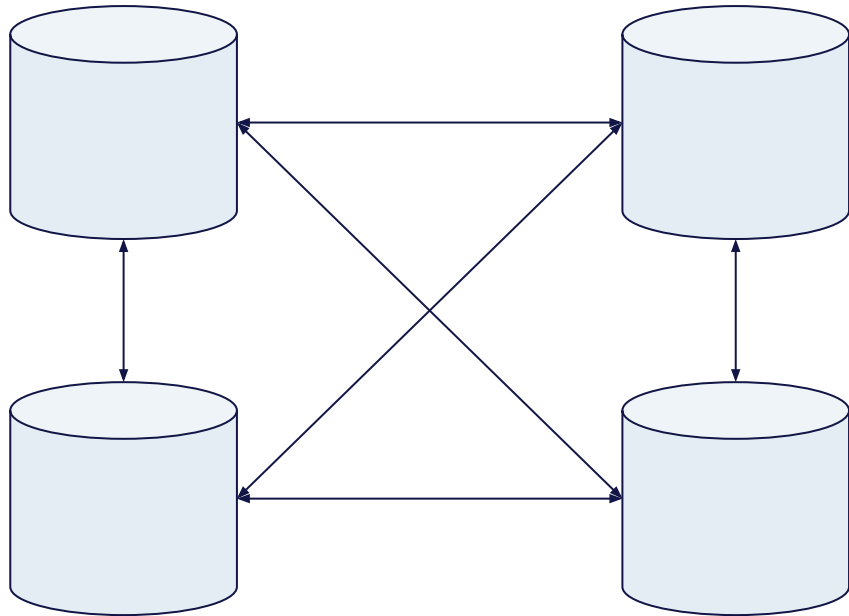
TRUNCATE ...
CREATE TABLE
DROP TABLE

SELECT myfunction();

??????

....

Multi-master, active/active, ...



Fully synchronous multi-master currently not available, though some products have synchronous features

Good:

Highly available.

Applications can access and update *any* DB

Scalable (to a degree).

Bad:

Can be difficult to configure (get assistance)

Distributed systems are generally complex.

No distributed locks (SELECT FOR UPDATE, ...)

Beware of conflicts!

(The chosen solution must have solid conflict handling and avoidance)

POLL:

Which replication solution would you want to use?

- 1) None
- 2) Native WAL archiving
- 3) Native WAL streaming
- 4) Native synchronous commit
- 5) Native logical replication
- 6) 3rd party replication solution: Replicated disks replication
- 7) 3rd party replication solution: Multi-master replication

References

[Demo script for replication](#)

Script to demo the different types of replication.

[Replication Engine Potpourri](#)

A comparison of several replication solutions, strengths and weaknesses.

[Defining High Availability](#)

A walk-through of what High Availability looks like for Postgres clusters.

[Nominally Bidirectional](#)

Blog post about a setup that you should never do, and why it doesn't work



Q&A



Thank you for joining!



As a thank you for attending, we'll be running a prize draw for a €25 voucher!

Keep an eye on your inbox to see if you're the lucky winner.



Borys Neselovskyi

borys.neselovskyi@enterprisedb.com
[linkedin.com/in/neselovskyi/](https://www.linkedin.com/in/neselovskyi/)

You can find more information at:

<http://enterprisedb.com>
info@enterprisedb.com

