Streaming Replication, the basics

PGDAY BELGIUM 2019

Stefan Fercot

17 May 2019

DALIBO
L'expertise PostgreSQL

Who Am I?

- Stefan Fercot
- aka. pgstef
- PostgreSQL user since 2010
- involved in the community since 2016
- @dalibo since 2017

DALIBO
L'expertise PostgreSQL

Dalibo

- Services

Support Training Advice

- Based in France
- Contributing to PostgreSQL community

Introduction

DALIBO
L'expertise PostgreSQL

Write-Ahead Log (WAL)

- transactions written sequentially
  - COMMIT when data are flushed to disk
- WAL replay after a crash
  - make the database consistent

DALIBO
L'expertise PostgreSQL

PostgreSQL WAL

- REDO log only
  - no UNDO log (yet)
  - instant rollback

DALIBO
L'expertise PostgreSQL

Structure

- WAL is divided into WAL segments
  - each segment is a file in pg_wal directory

```
$ ls pg_wal/
00000001000000010000002E
00000001000000010000002F
000000010000000000000030
...
```

Filenames

- `000000010000000100000002E`
  - 00000001 : TLI
  - 000000010000002E : LSN
    - 00000001 : log id
    - 0000002E : segment number

Checkpoints

- flush all data pages to disk
- write a checkpoint record
- recycle / remove old WAL

Archiving

- old WAL segments are deleted / recycled after a checkpoint
- can also be archived with `archive_command`

Allows online backups and Point-in-Time Recovery

DALIBO
L'expertise PostgreSQL

Replication

- apply WAL when generated on a standby server
  - using WAL archives (files)
  - or by **streaming** over a TCP connection

Streaming Replication

- architecture/compile flag dependent
- whole cluster only
- read-only standby
- no built-in cluster management
- no (easy) fail-back

DALIBO
L'expertise PostgreSQL

Setup

L'expertise PostgreSQL

wal_level

```
wal_level = 'replica'
```

max_wal_senders

`max_wal_senders=10` (default from v10)

Authentication

- On primary
  - `CREATE ROLE replicator WITH LOGIN REPLICATION;`
  - … and setup a password
  - in `pg_hba.conf`
    - `host replication replicator standby_ip/32 md5`

Data initialization

```
$ pg_basebackup -D /var/lib/pgsql/11/data \
    -h primary -U replicator -R -P
```

- before v10, add `-X stream`

recovery.conf

- standby_mode
- primary_conninfo
- recovery_target_timeline

standby_mode

- `standby_mode=on`
- continuous recovery by fetching new WAL segments
  - using restore_command
  - by connecting to the primary server

primary_conninfo

- `primary_conninfo = 'user=replicator host=primary'`
- connection string to the primary server

recovery_target_timeline

- particular timeline for recovery
  - `latest` is useful in a standby server
- new timeline created after a recovery
  - to identify the series of WAL records generated afterwards

PostgreSQL 12 changes

“Integrate recovery.conf into postgresql.conf” (2018-11-25)

- recovery.signal / standby.signal
- pg_basebackup **-R** append `postgresql.auto.conf`

Start

```
# systemctl start postgresql-11
```

Processes

On primary:

- `walsender replicator ... streaming 0/3BD48728`

On standby:

- `walreceiver   streaming 0/3BD48728`

Monitoring

- lag
  - amount of WAL records generated in the primary
  - not yet received / applied on the standby
- `pg_current_wal_lsn()` on the primary
- `pg_last_wal_receive_lsn()`, `pg_last_wal_replay_lsn()` on the standby

pg_stat_replication

On primary:

```
usename          | replicator
application_name | walreceiver
state            | streaming
sent_lsn         | 0/3BD48728
write_lsn        | 0/3BD48728
flush_lsn        | 0/3BD48728
replay_lsn       | 0/3BD48728
sync_state       | async
...
```

pg_stat_wal_receiver

On standby:

```
status                  | streaming
received_lsn            | 0/3BD48728
received_tli            | 1
...
```

Fail-over

DALIBO
L'expertise PostgreSQL

Split-brain

- if standby server becomes new primary
  - make sure the old primary is no longer the primary
- avoid situations where both systems think they are the primary
  - lead to confusion and ultimately data loss

Check-up before clean promote

On primary:

```
# systemctl stop postgresql-11
$ pg_controldata -D /var/lib/pgsql/11/data/ \
| grep -E '(Database cluster state)|(REDO location)'
Database cluster state:               shut down
Latest checkpoint's REDO location:    0/3BD487D0
```

On standby:

```
$ psql -c 'CHECKPOINT;'
$ pg_controldata -D /var/lib/pgsql/11/data/ \
| grep -E '(Database cluster state)|(REDO location)'
Database cluster state:               in archive recovery
Latest checkpoint's REDO location:    0/3BD487D0
```

Promote

- `pg_ctl promote [-D datadir] [-W] [-t seconds] [-s]`
- `trigger_file` in recovery.conf

Logs after promote

```
LOG:  received promote request
LOG:  redo done at 0/3BD487D0
LOG:  last completed transaction was at log time ...
LOG:  selected new timeline ID: 2
LOG:  archive recovery complete
LOG:  database system is ready to accept connections
```

DALIBO
L'expertise PostgreSQL

Fail-back

- old primary as a standby
  - full copy of the new primary
  - pg_rewind
    - `--source-pgdata`
    - `--source-server`

pg_rewind

- rewinding a cluster until its divergence with another
- needs `wal_log_hints` or data checksums
- `--dry-run`

pg_rewind (2)

```
$ pg_rewind -D /var/lib/pgsql/11/data/ \
  --source-server="user=postgres host=primary" -P
connected to server
servers diverged at WAL location 0/3BD48840 on timeline 1
rewinding from last common checkpoint at 0/3BD487D0 on timeline 1
reading source file list
reading target file list
reading WAL in target
need to copy 196 MB (total source directory size is 561 MB)
200806/200806 kB (100%) copied
creating backup label and updating control file
syncing target data directory
Done!
```

Troubles

What if the connection between primary and standby fails?

DALIBO
L'expertise PostgreSQL

Replication slots

- primary does not remove WAL segments
  - until received by all standbys
- `pg_create_physical_replication_slot('slot_name');`
- `primary_slot_name`
- `max_replication_slots = 10` (default from v10)

Log-shipping

Don’t prevent the removal of old WAL segments, use the archives!

- `restore_command`
- `archive_cleanup_command = 'pg_archivecleanup /path/to/archive %r'`

PITR

Combine with PITR backups for easier fail-backs!

- online backups
- the standby use archives from the PITR repository
  - to catchup the primary
- faster standby creation through backup restore
  - or refresh an old one

DALIBO
L'expertise PostgreSQL

Synchronous replication

- `synchronous_commit`
  - off
  - local
  - remote_write
  - on
  - remote_apply
- can be applied **by transaction**

synchronous_standby_names

- Single (9.1)
  - `synchronous_standby_names = s1,s2,s3`
- First (9.6)
  - `synchronous_standby_names = 2(s1,s2,s3)`
- Quorum (10)
  - `synchronous_standby_names = ANY 2(s1,s2,s3)`

Hot standby and conflicts

- `DROP TABLE` on primary…
  - cannot wait for the end of queries on standby
- on standby (`max_standby_archive_delay` and `max_standby_streaming_delay`)
  - delay application of WAL record
  - or cancel the conflicting query

Early cleanup

- cleanup on the primary
  - according to MVCC rules
  - remove row versions still visible to a transaction on the standby
- `hot_standby_feedback`
  - or replication slots…

Updates

- different minor release on primary and standby usually works
  - not advised!
- update the standby servers first

Tools

DALIBO
L'expertise PostgreSQL

Automated Fail-over

- Patroni
- repmgr
- PAF

Patroni

- Python
- “template” for high-availability
  - with ZooKeeper, etcd, Consul or Kubernetes
- integrates with HAProxy

DALIBO
L'expertise PostgreSQL

repmgr

- fewer prerequisites
- easier for manual processing
  - repmgrd for automatic fail-over
  - witness to avoid split-brain
- no connection management

DALIBO
L'expertise PostgreSQL

PAF

- agent for Pacemaker/Corosync
  - linux HA
  - possible management of other services
- connection routing with virtual IP
- STONITH

PITR

- pgBackRest, …
  - … but that’s for another talk!

pgBackRest Main Features

- custom protocol
  - local or remote operation (via SSH)
- multi-process
- full/differential/incremental backup
- backup rotation and archive expiration
- parallel, asynchronous WAL push and get
- Amazon S3 support
- encryption
- …

DALIBO
L'expertise PostgreSQL

Logical Replication

- reconstructs changes by row
- replicates row content, not SQL statements
- table-level partial / bi-directional replication
- data replication only
  - no schema
  - no sequences
- suitable for data distribution
  - but not for **HA** !

DALIBO
L'expertise PostgreSQL

Conclusion

- consolidated during 9.x versions
- out of the box in 10
  - wal_level
  - max_wal_senders
  - …

DALIBO
L'expertise PostgreSQL

Thank you for your attention!

DALIBO
L'expertise PostgreSQL