WAL, Standbys and Postgres 9.5

Postgres Open 2015
17th Sept 2015, Dallas
Michael Paquier / VMware

Summary

- About archiving
- … And standbys
- Mixed with magic from Postgres 9.5

Archiving

- Store database crash journal (WAL) independently
- Used by recovery from backups and by standbys
- Configuration

```
wal_level = [archive|hot_standby|logical]
archive_mode = 'on'
archive_command = '...'
```

Quoting the docs

- Return 0 as exit status if it succeeds
  - 0 => Backend assumes archiving is complete
  - Not 0 => Backend will retry archiving
- Be careful of existing files.
  - Sending output of different clusters at the same place?
  - Not overwriting an existing file?
- Reference:
  - http://www.postgresql.org/docs/current/static/continuous-archiving.html

Quoting the docs (2)

- Examples: Any issues here?

```
# Unix
archive_command =
     'test ! -f /mnt/server/archivedir/%f &&
      cp %p /mnt/server/archivedir/%f'

# Windows
archive_command =
    'copy "%p" "C:\\server\\archivedir\\%f"'
```

Quoting the docs (3)

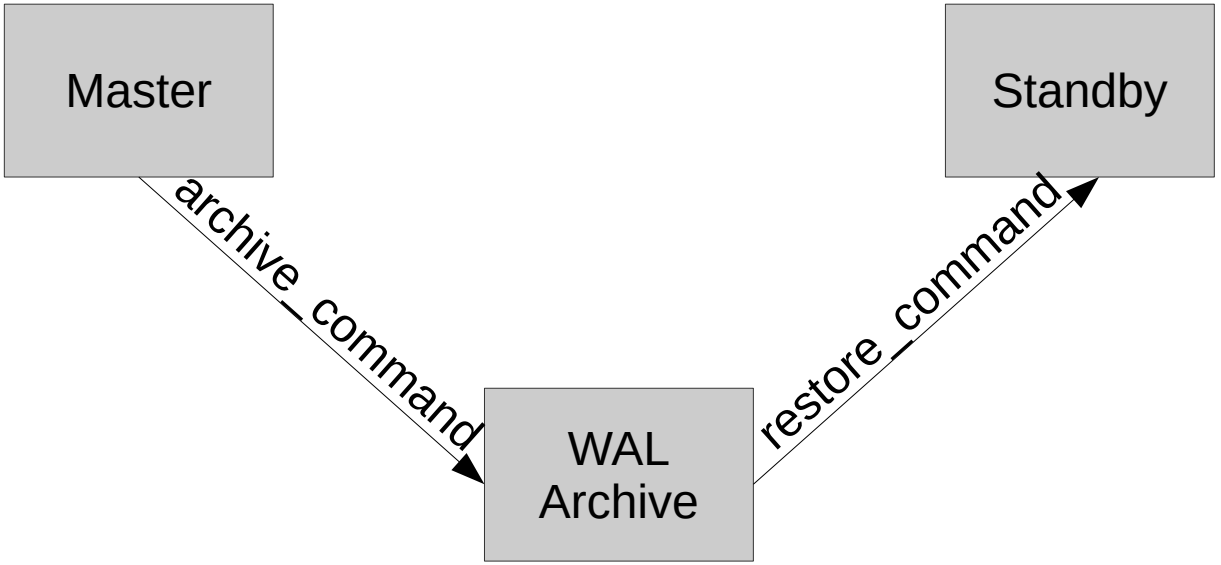
- No fsync() done.
  - host dies before flushing segment to disk
  - Maybe game over
- If backend crashes immediately
  - Segment not switched from .ready to .done
  - May try to archive again same segment
- So…
  - Call fsync before leaving archive_command
  - Check content of existing segments (checksum, etc.)
  - Use a custom script, not only commands

Standbys

- Built from base backup of existing node
  - FS-level backup or snapshot
  - pg_basebackup
  - pg_start_backup() and pg_stop_backup()
- Backup taken from master or other standby
- Can be read-only: hot_standby = on, wal_level >= hot_standby
- Several types of standbys: warm or hot.

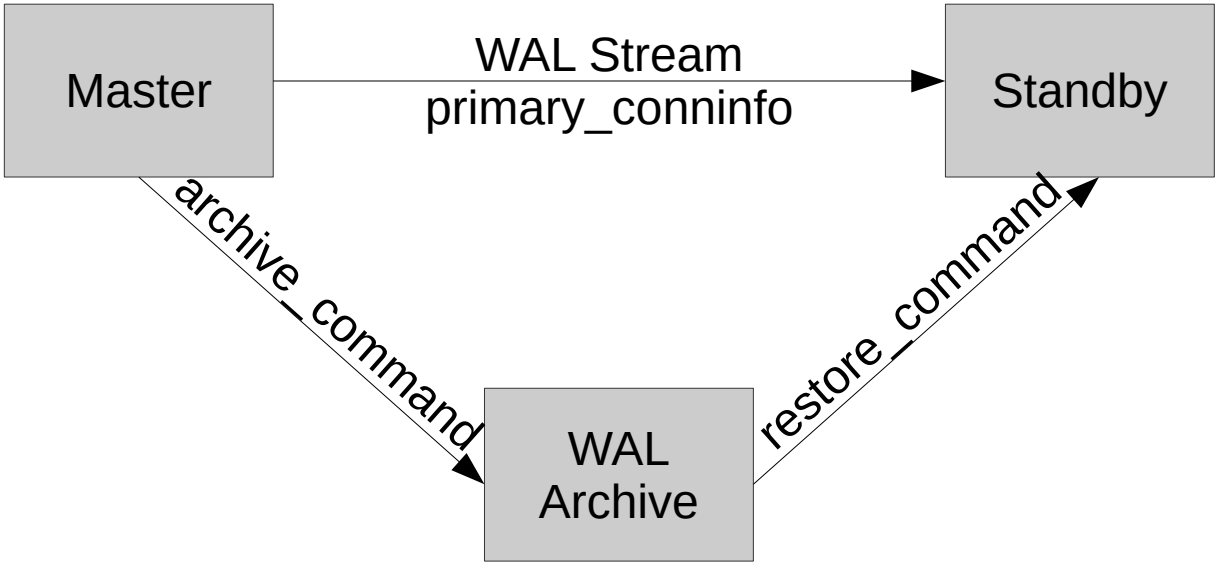
Warm standby

- Consumes WAL from archives via restore_command of recovery.conf.
- pg_switch_xlog() may be useful.
- Only completed segments are fetched and replayed.



Hot Standby

- Consumes WAL via streaming replication
- Can optionally consume WAL archives
- primary_conninfo in recovery.conf
- Can be synchronous



Archive and promotion <= 9.4

Master	Standby	Archive
000000010000000000000022	000000010000000000000022	**000000010000000000000022**
000000010000000000000023	000000010000000000000023	**000000010000000000000023**
000000010000000000000024	000000010000000000000024	**000000010000000000000024**
	Promotion	
000000010000000000000025	**000000010000000000000025**	**000000010000000000000025**
	000000020000000000000025	**000000020000000000000025**
	000000020000000000000026	**000000020000000000000026**

- Standby archives last, partial WAL segment of old timeline at promotion
  - Conflicts if archived from standby **and** master
  - Size of 16MB, with garbage after fork point

Archive and promotion >= 9.5

Master	Standby	Archive
000000010000000000000022 **000000010000000000000023** **000000010000000000000024** **000000010000000000000025**	000000010000000000000022 000000010000000000000023 000000010000000000000024 Promotion **000000010000000000000025** **000000020000000000000025** **000000020000000000000026**	**000000010000000000000022** **000000010000000000000023** **000000010000000000000024** **000000010000000000000025** **000000010000000000000025.partial** **000000020000000000000025** **000000020000000000000026**

- Standby still archives last, partial segment at promotion of old timeline
  - With suffix .partial
  - No name conflict
  - Format not recognized by backend, should be renamed manually if used at recovery

Lost WAL segments

Master	Standby	Archive
000000010000000000000022 **000000010000000000000023** 000000010000000000000024 000000010000000000000025	000000010000000000000022 000000010000000000000023 000000010000000000000024 Promotion **000000010000000000000025** **000000020000000000000025** **000000020000000000000026**	**000000010000000000000022** **000000010000000000000023** **000000010000000000000025.** **partial** **000000020000000000000025** **000000020000000000000026**

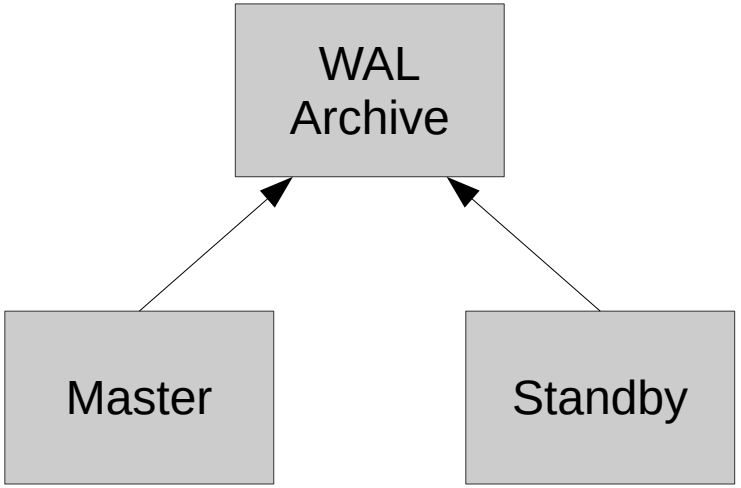
- Master crashed before archiving all segments
  - Game over to recover from older backups on new timeline
  - Series of backups now useless

archive_mode = 'always'

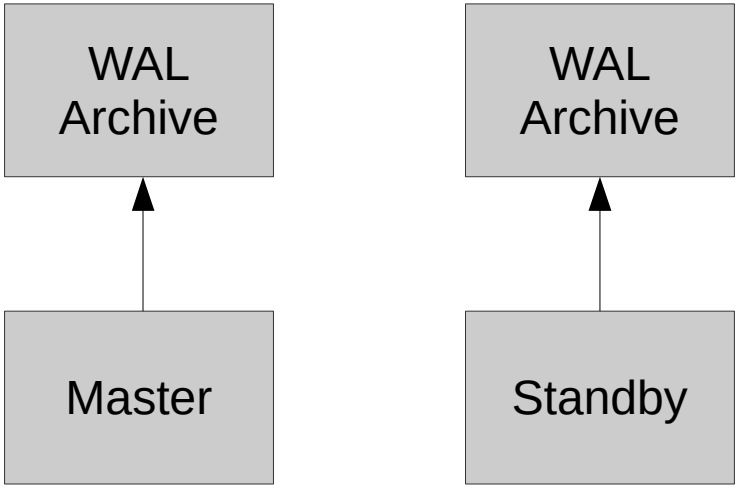
Master	Standby	Archive
000000010000000000000022 **000000010000000000000023** 000000010000000000000024 000000010000000000000025	**000000010000000000000022** **000000010000000000000023** **000000010000000000000024** Promotion **000000010000000000000025** **000000020000000000000025** **000000020000000000000026**	**000000010000**000000000022 **000000010000**000000000023 **000000010000000000000024** **000000010000000000000025.partial** **000000020000000000000025** **000000020000000000000026**

- No changes on master node compared to 'on'
- In recovery mode
  - standby will archive segments whose reception is finished
  - For <= 9.4, forcibly switched to .done
  - May want to use same archive_command. Or not.

archive_mode = 'always' (2)



- Be aware of name conflicts



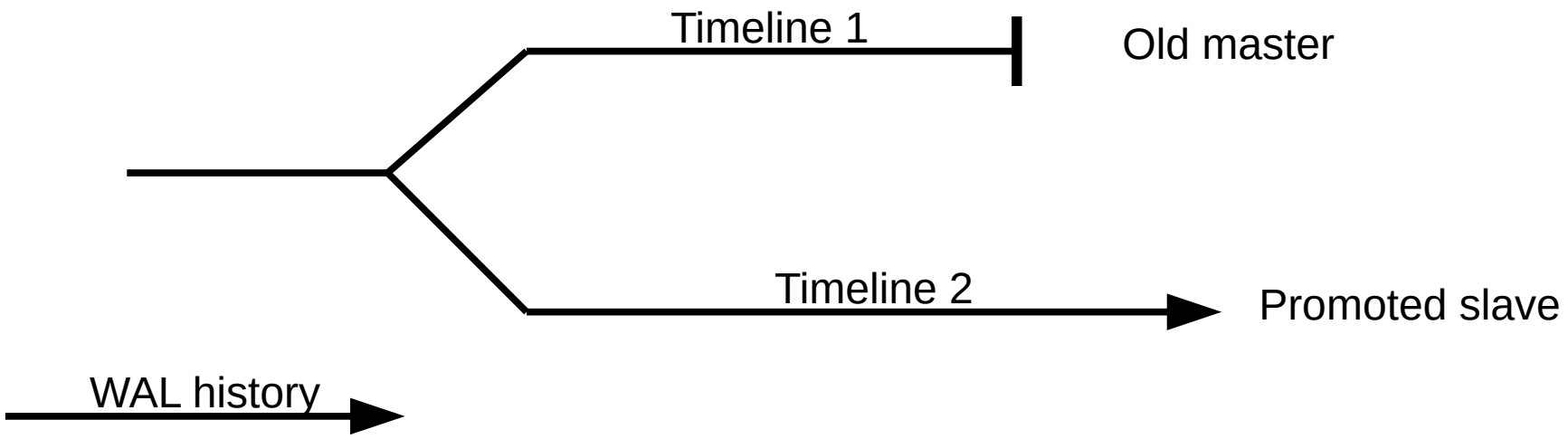
- Be careful with restore_command

pg_receivexlog

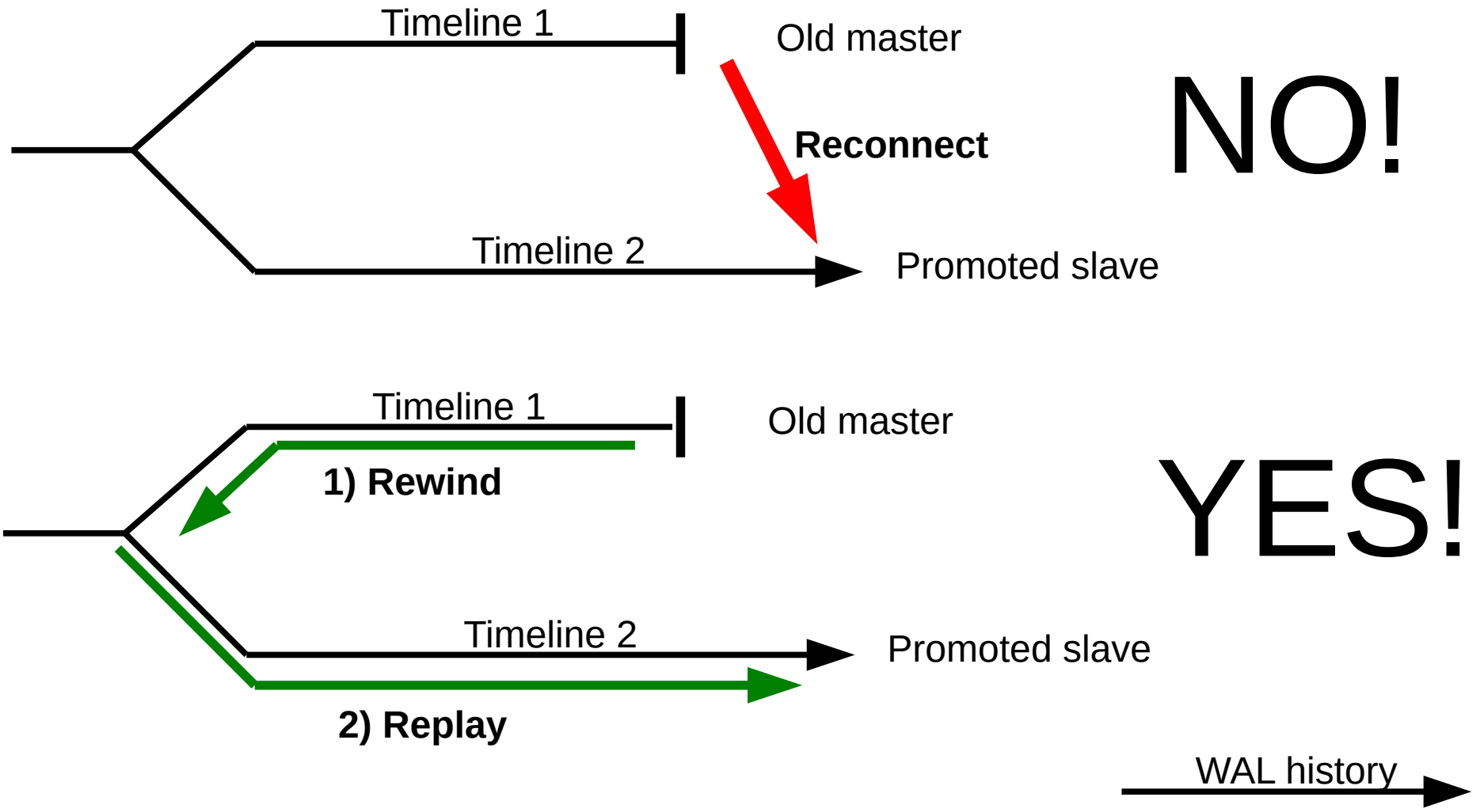
- Useful to leverage archiving
- Segments marked as .partial if not completed
- Should be used on archive host
- New features
  - Support for timeline switches in 9.3
  - Replication slots in 9.4
  - Synchronous mode in 9.5
- Take advantage with cascading replication and standbys (works <= 9.2)

pg_rewind (1)

- Resync an existing data directory without new base backup
  - Replug old master to existing cluster => “rewind it”



pg_rewind (2)


Timeline 1
Old master
Reconnect
Timeline 2
Promoted slave
NO!
Timeline 1
Old master
1) Rewind
YES!
Timeline 2
Promoted slave
2) Replay
WAL history

pg_rewind (3)

- Scan old master data folder
  - Start from WAL fork point
  - Record data blocks touched
- Copy all changed blocks from promoted slave to old master
- Copy clog, conf files...
- Replay WAL from master, starting from last checkpoint before WA forked
- WAL format refactored for block tracking

pg_rewind (4)

- Largely faster than a new base backup
- Limitations
  - Need wal_log_hints = on or data checksum
  - Might not have all WAL for replay, can copy them though
  - No handling of timeline switches (WIP for 9.6)

WAL reader facility (1)

- Series of routines to decode WAL records in xlogreader.c
- Used by pg_xlogdump, WAL decoding, pg_rewind.
- Available since Postgres 9.3
- 9.5 facilitates block tracking: no need to know record type

WAL reader facility (2)

- Example of code: pg_wal_blocks
- https://github.com/michaelpq/pg_plugins/

```
$ pg_wal_blocks $PGDATA/pg_xlog/000000010000000000000001 2>&1
Block touched: dboid = 16384, relid = 16385, block = 0
[...]
Block touched: dboid = 16384, relid = 16385, block = 2
[...]
```

min_wal_size and max_wal_size

- checkpoint_segments removed
- max_wal_size
  - Maximum WAL size between checkpoints
  - Soft limit
- min_wal_size
  - Control of WAL segment recycling
  - Useful to handle spikes in WAL usage

wal_compression?

- Not really WAL compression...
- Compression of full-page writes in WAL records
  - Image of block saved in WAL during modification after checkpoint.
- Compression with pglz
  - CPU consumer (may want to switch to lz4 in future)
  - Pushed in src/common, available for extensions
- Reduction of sequential I/O induced by WAL at the cost of some CPU.

wal_compression - performance

- Reduction of WAL size, for example:
  - 15% for FPW with UUID datatype
  - 27% for FPW with integers
- Reduction of recovery time, less segments to fetch.
  - Local machine, 500MB of FPWs.
  - UUID: 14.7s/15.3s
  - Integers: 18.5s/21.8s
- Synchronous replication
  - Bottleneck may be WAL record length when master and standby hosts are close
  - Contention with SyncRepLock
  - Can increase overall output

wal_compression - security

- Compression rate of a page gives hints on content
- Now only PGC_SUSET
- Hide WAL position to lambda users
  - pg_current_xlog_position()
  - pg_current_xlog_insert_location()
  - pg_last_xlog_receive_location()
  - pg_last_xlog_replay_location()

```
REVOKE ALL ON FUNCTION pg_current_xlog_location() FROM PUBLIC;
-- etc.
```

wal_retrieve_retry_interval

- In 9.5, to control interval of time to fetch WAL from source after failure, either WAL archive or WAL receiver.
- Useful for warm standbys
  - Limit requests to archive server
  - Accelerate detection of archived segment

Thanks!
Questions?