

Warm standby done right

Heikki Linnakangas / Pivotal

# This presentation

- About built-in tools
  - Not about repmgr, WAL-e etc.
  - You probably should use those tools though!
- Not about monitoring, heartbeats etc.

# Part 1. Continuous archiving

# Setting up a WAL archive

- `wal_level=archive`
- `archive_mode=on`
- `archive_command='...'`

## 24.3.1. Setting up a WAL archive

*Depending on the application and the available hardware, there could be many different ways of "saving the data somewhere": we could copy the segment files to an NFS-mounted directory on another machine, write them onto a tape drive (ensuring that you have a way of identifying the original name of each file), or batch them together and burn them onto CDs, or something else entirely.*

*PostgreSQL User Manual*

# archive\_command

according to the manual

- *It is important that the archive command **return zero exit status if and only if it succeeds**. Upon getting a zero result, PostgreSQL will assume that the file has been successfully archived, and will remove or recycle it. However, a nonzero status tells PostgreSQL that the file was not archived; it will try again periodically until it succeeds.*
- *The archive command should generally be designed to **refuse to overwrite any pre-existing archive file**. This is an important safety feature to preserve the integrity of your archive in case of administrator error (such as sending the output of two different servers to the same archive directory).*

# archive\_command

according to the manual

```
# Unix
```

```
archive_command =
```

```
'test ! -f /mnt/server/archivedir/%f &&  
  cp %p /mnt/server/archivedir/%f'
```

```
# Windows
```

```
archive_command =
```

```
'copy "%p" "C:\\server\\archivedir\\%f"'
```

*This is an example, not a recommendation, and might not work on all platforms.*

# archive\_command

## Gotcha 1:

No `fsync()` in the example command. If the archive server dies before the file has been flushed to disk, it might be lost.



# archive\_command

## Gotcha 2:

If the server crashes immediately after archiving a segment, the server might try to archive the same file again after restart.

# How to write a robust archive\_command

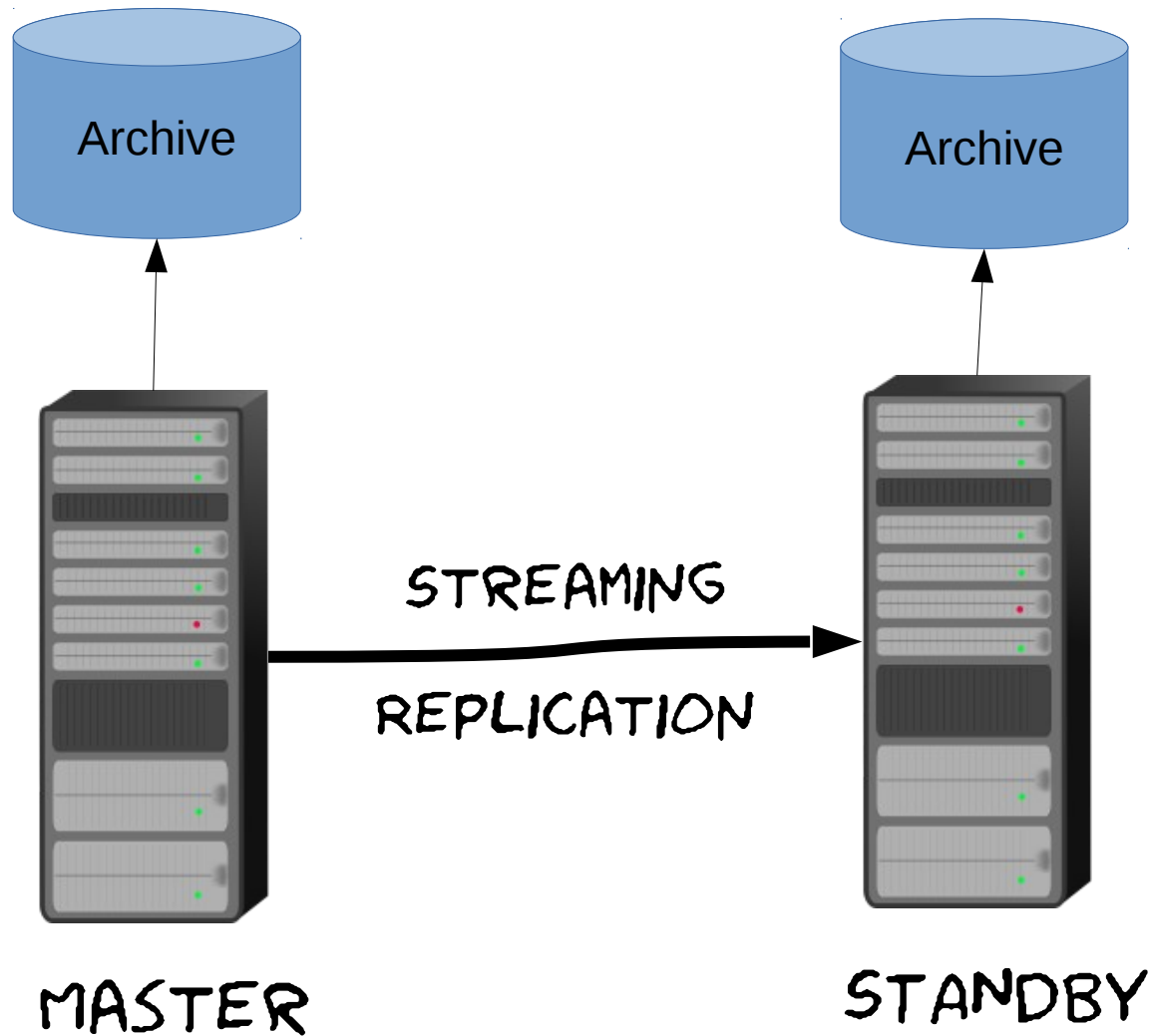
- Only return zero on success
- Issue `fsync()` before returning
- If the file already exists in the archive, check that the contents are identical, and return success.

**Part 2:** `archive_mode=always`

# archive\_mode=always

- New feature in 9.5
- archive\_mode = off | on | **always**
  - off: no archiving
  - on: archiving is enabled in master
  - **always: archiving is enabled in master and standby (and archive recovery)**

# Continuous archiving with a standby

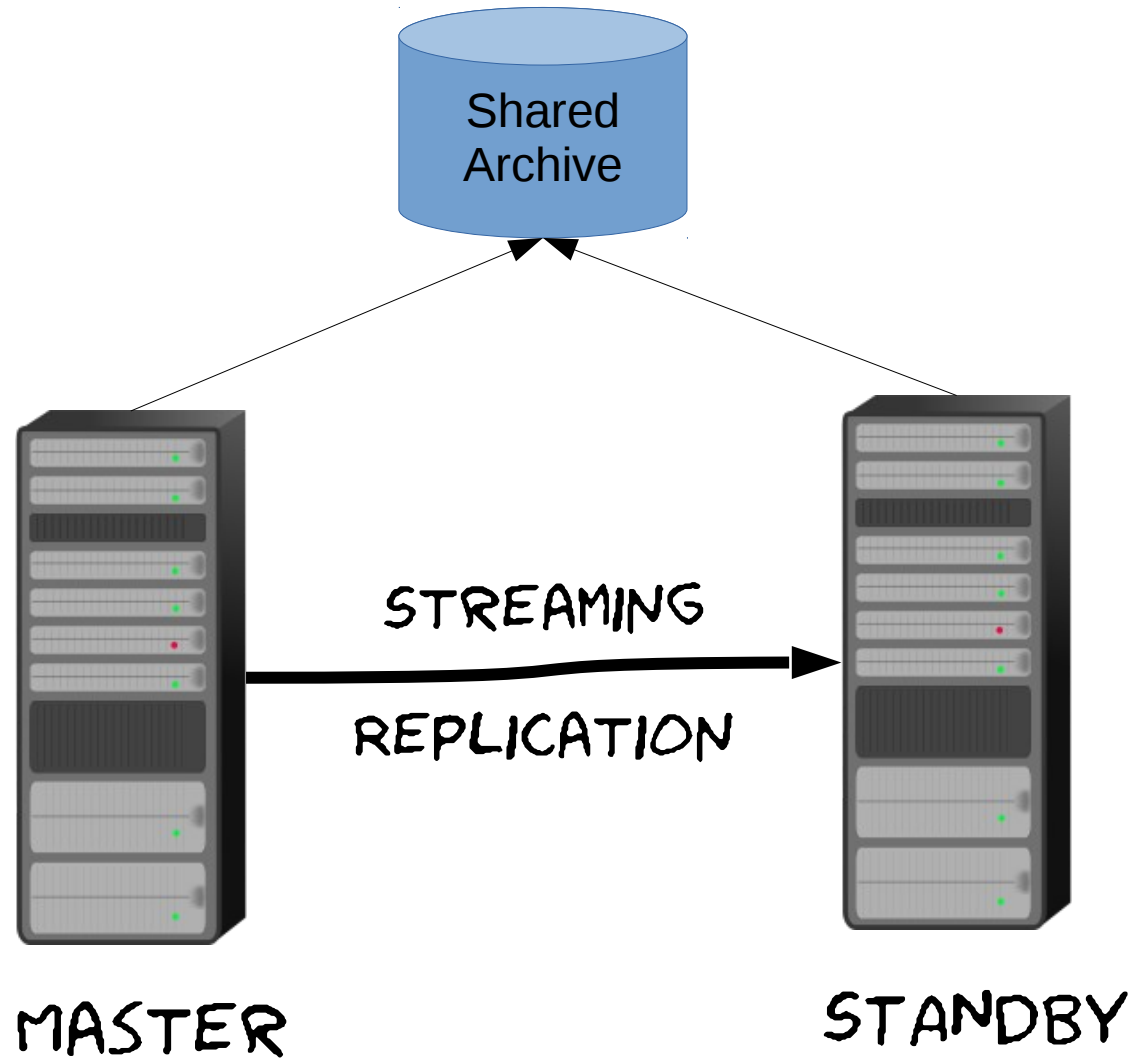


→ WAL data flow

- `Archive_mode=always` allows setting up separate archives in master and standby
- But that's not all

# Part 3: Continuous archiving with master and standby sharing the archive

# Continuous archiving with shared archive



→ WAL Data flow



# Naive approach

- `archive_mode=on`
- **Same** `archive_command` on both servers
- All set?

# Promotion

1. The master dies
2. `pg_ctl promote`
3. ???

# Promotion

Standby will:

1. recover any remaining WAL it had streamed,
2. create a new timeline,
3. copy the last, partial segment to the new timeline, and start writing WAL,
4. start archiving from the new timeline

# Master, standby, WAL archive

Master

00000001000000000000000012  
00000001000000000000000013  
00000001000000000000000014

Standby

00000001000000000000000012  
00000001000000000000000013  
00000001000000000000000014

WAL archive

00000001000000000000000012  
00000001000000000000000013  
00000001000000000000000014

# Master, standby, WAL archive

Master

00000001000000000000000012  
00000001000000000000000013  
00000001000000000000000014  
00000001000000000000000015  
00000001000000000000000016  
*00000001000000000000000017*

Standby

00000001000000000000000012  
00000001000000000000000013  
00000001000000000000000014  
00000001000000000000000015  
00000001000000000000000016  
*00000001000000000000000017*

WAL archive

00000001000000000000000012  
00000001000000000000000013  
00000001000000000000000014

*Partial segment, being written to*

# Master, standby, WAL archive after promotion. 9.4 and below

Master	Standby	WAL archive
00000001000000000000000012	00000001000000000000000012	00000001000000000000000012
00000001000000000000000013	00000001000000000000000013	00000001000000000000000013
00000001000000000000000014	00000001000000000000000014	00000001000000000000000014
00000001000000000000000015	00000001000000000000000015	
00000001000000000000000016	00000001000000000000000016	
00000001000000000000000017	00000001000000000000000017	00000001000000000000000017
	00000002000000000000000017	

1. Standby creates a new timeline
2. Standby archives the partial segment

# Detour: Partial segment

- 16MB in size, but the rest contains garbage.
- Indistinguishable from a completed segment
- Problems:
  - If the master continues running, and archives the completed segment later.
  - If the standby had fallen slightly behind and the master had already archived the completed segment.

# Partial segment in 9.5

- The partial segment is archived with the `.partial` suffix
- Not restored automatically. You can copy it into `pg_xlog` manually and remove `.partial` suffix
  - This shouldn't be necessary under normal circumstances



# Master, standby, WAL archive after promotion

Master	Standby	WAL archive
00000001000000000000000012	00000001000000000000000012	00000001000000000000000012
00000001000000000000000013	00000001000000000000000013	00000001000000000000000013
00000001000000000000000014	00000001000000000000000014	00000001000000000000000014
00000001000000000000000015	00000001000000000000000015	
00000001000000000000000016	00000001000000000000000016	
00000001000000000000000017	00000001000000000000000017.p artial	00000001000000000000000017.p artial
	00000002000000000000000017	00000002000000000000000017
	00000002000000000000000018	00000002000000000000000018
	00000002000000000000000019	

1. Standby creates a new timeline
2. **Standby renames the old segment as .partial**
3. Standby archives the partial segment
4. Standby startups up as master and starts archiving

# Master, standby, WAL archive after promotion

Master

00000001000000000000000012  
00000001000000000000000013  
00000001000000000000000014  
00000001000000000000000015  
00000001000000000000000016  
00000001000000000000000017

Standby

00000001000000000000000012  
00000001000000000000000013  
00000001000000000000000014  
00000001000000000000000015  
00000001000000000000000016  
00000001000000000000000017.p  
artial  
00000002000000000000000017  
00000002000000000000000018  
00000002000000000000000019

WAL archive

00000001000000000000000012  
00000001000000000000000013  
00000001000000000000000014  
00000001000000000000000017.p  
artial  
00000002000000000000000017  
00000002000000000000000018  
00000002000000000000000019

***Segments 15 and 16 are never archived!***

# Missing segments

- If the master had not archived all the segments before it crashed
- Bye bye backups

# `archive_mode=always` to the **rescue**

- Can also be used with a single archive, to ensure there are no gaps.
- (my original patch added a separate `archive_mode=shared` mode for this)

```
archive_mode=always
```

- Have same `archive_command` in master and standby
- Both servers will attempt to archive all segments to same location.
- Careful, race conditions!
  - Master and standby will try to archive the same file at the same time

# Shared archive summary

- Use `archive_mode=always`
- Make sure your `archive_command` is concurrency-safe and handles duplicates
- Don't be alarmed if you see `.partial` files
- Still make sure your `archive_command` calls `fsync()`

## Part 4: pg\_receivexlog

# pg\_receivexlog

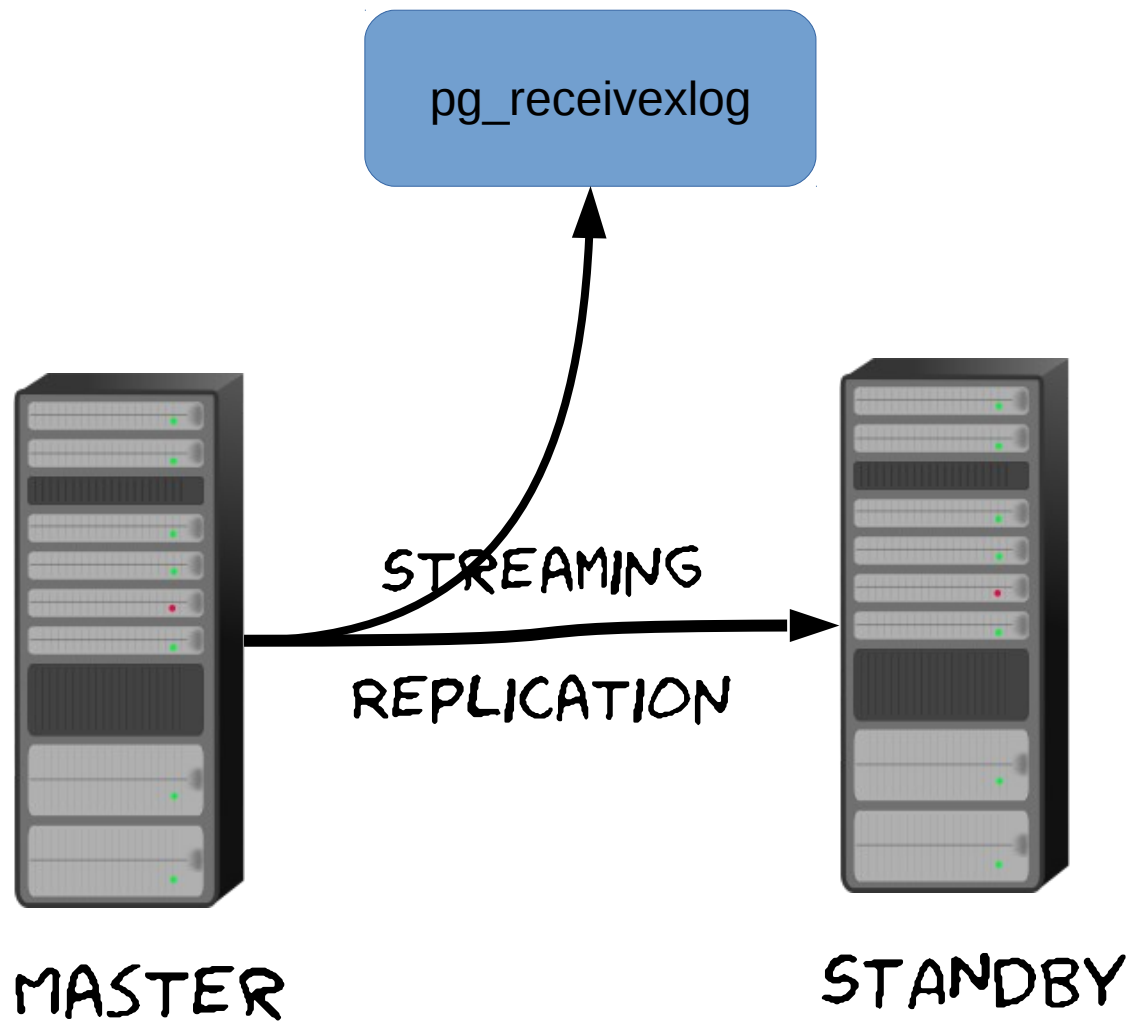
- Runs in the archive server
- Connects to master with streaming replication
- Keeps up-to-date, not just segment granularity



# pg\_receivexlog

- Added 9.2
- Follows timeline changes since 9.3
- Replication slot support in 9.4
- --synchronous option added in 9.5

# pg\_receivexlog



→ WAL Data flow

# pg\_receivexlog

Setup:

```
pg_receivexlog -D wal-archive  
--slot="receivexlog_archive" --create-slot
```

Run:

```
pg_receivexlog -D wal-archive  
--slot="receivexlog_archive"
```

# pg\_receivexlog

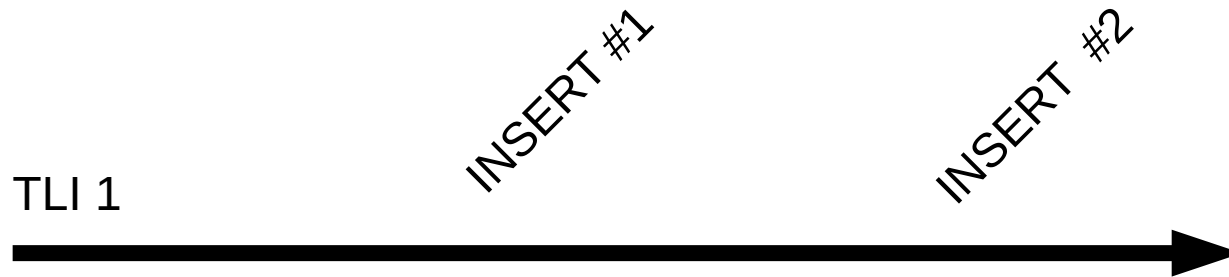
```
$ pg_receivexlog --slot="receivexlog_archive" -D wal-archive -d
"port=5433" -v
pg_receivexlog: starting log streaming at 0/15000000 (timeline 1)
pg_receivexlog: finished segment at 0/16000000 (timeline 1)
pg_receivexlog: finished segment at 0/17000000 (timeline 1)
...
pg_receivexlog: finished segment at 0/28000000 (timeline 1)
pg_receivexlog: finished segment at 0/29000000 (timeline 1)
pg_receivexlog: not renaming "00000001000000000000000029.partial", segment
is not complete
pg_receivexlog: switched to timeline 2 at 0/29DED2C8
pg_receivexlog: finished segment at 0/2A000000 (timeline 2)
pg_receivexlog: finished segment at 0/2B000000 (timeline 2)
...
```

# Shared archive summary

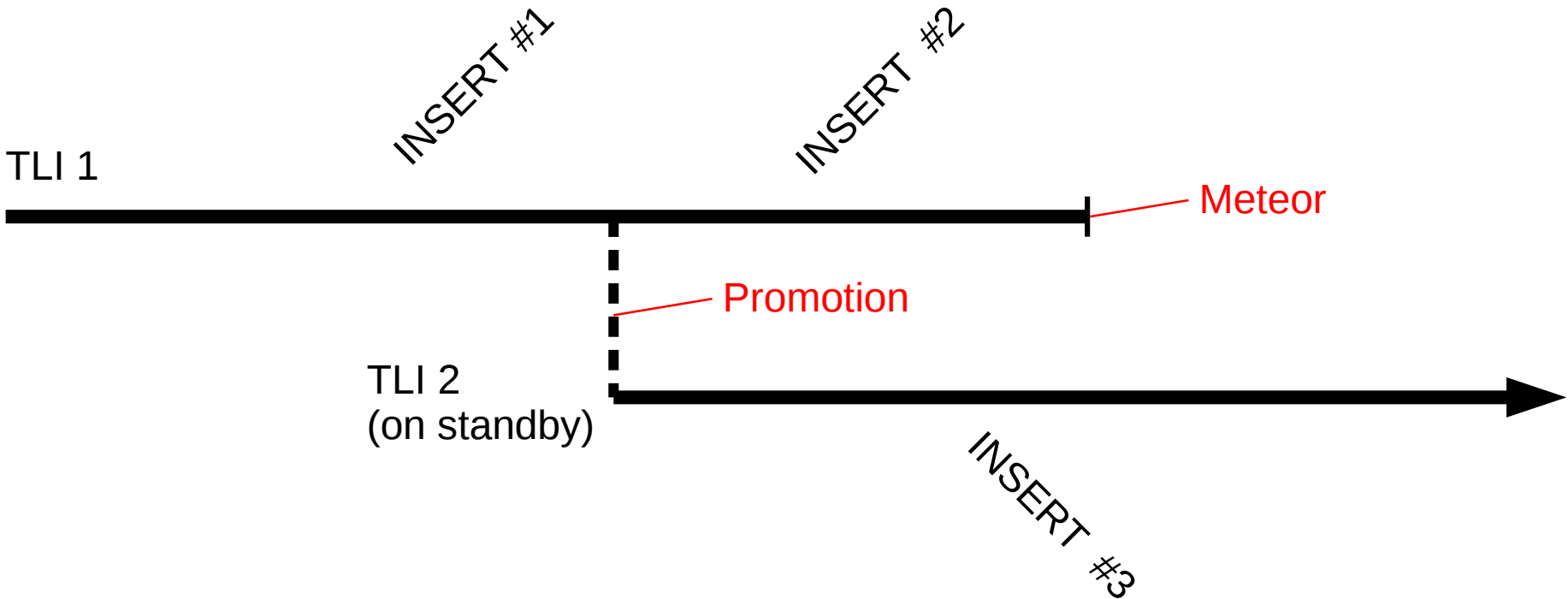
- ~~Use `archive_mode=always`~~
- ~~Make sure your `archive_command` is concurrency-safe and handles duplicates~~
- Use `pg_receivexlog`

## Part 5: pg\_rewind

# WAL Timelines

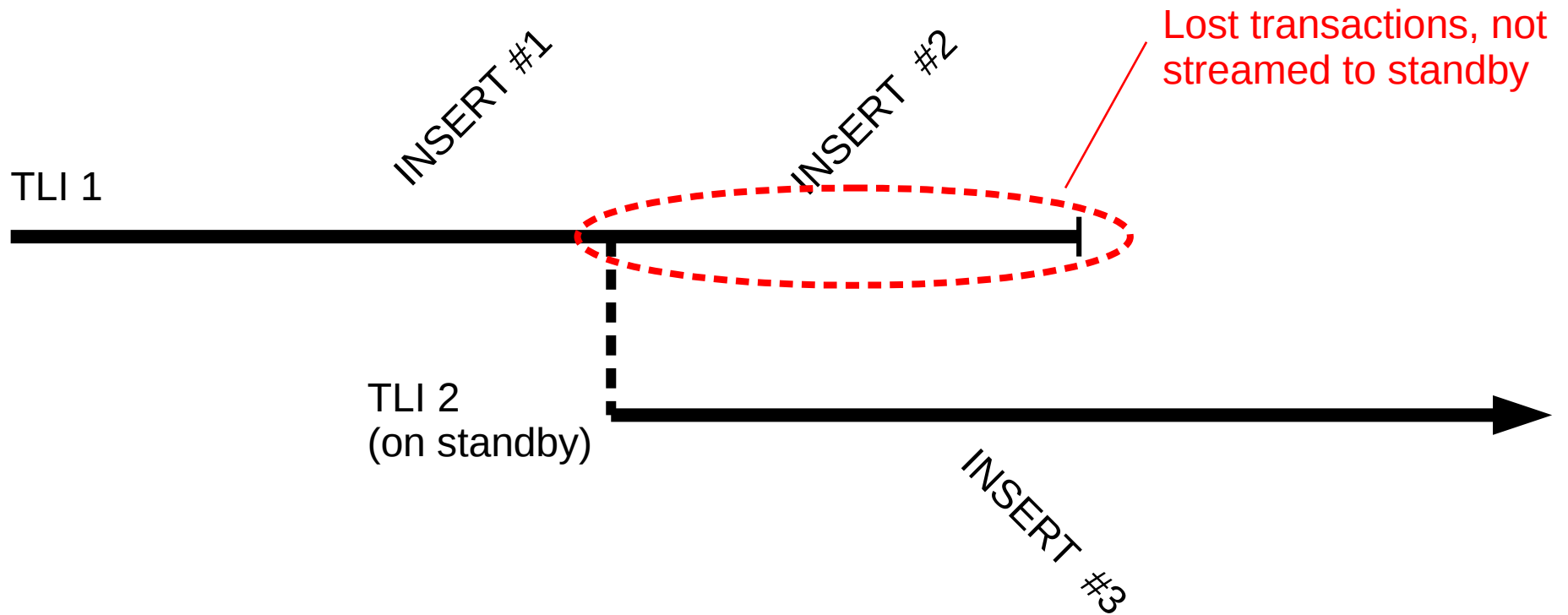


# Promotion





# Lost transactions at failback



# Failback

- Throw away old master's data directory and restore from base backup
- You can use `rsync` to speed it up
- Or you can use `pg_rewind`

# pg\_rewind

- Like `rsync`, but uses the WAL to determine what's changed
- Scans the WAL to figure out what blocks the lost transactions modified
- Copies anything except data files in full

# That's all folks!

- Questions?