

Neon - cloud-native storage backend for PostgreSQL

Heikki Linnakangas <heikki@neon.tech>

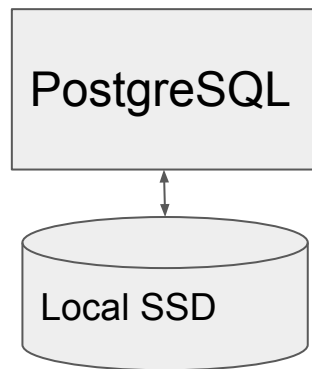
What is Neon?

- Startup
 - founded March 2021
- New storage system for PostgreSQL
 - Open source
- Cloud service
 - `psql -h pg.neon.tech`
 - In beta, invite code “**pgconfeu**”

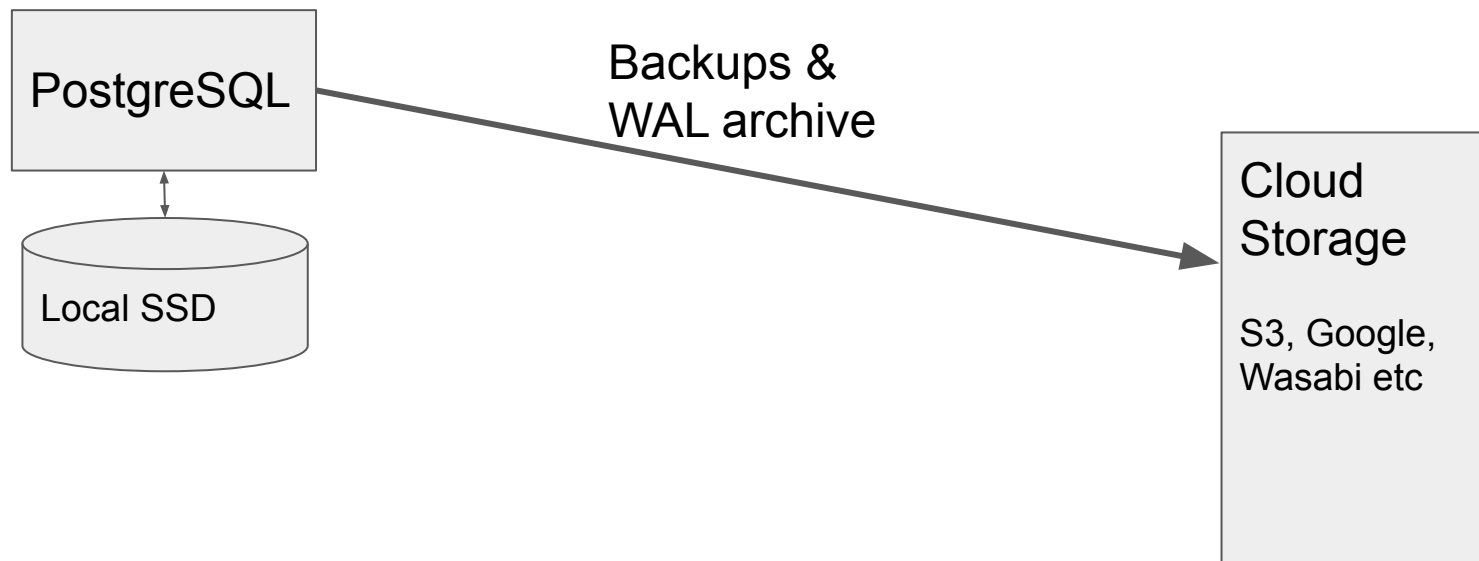
What is Neon?

- Storage and compute are separated
- Single writer, multiple readers
- Multi-tenant storage
- Single-tenant compute
 - runs in Kubernetes containers / VMs
- Cheap copy-on-write branching and timetravel query

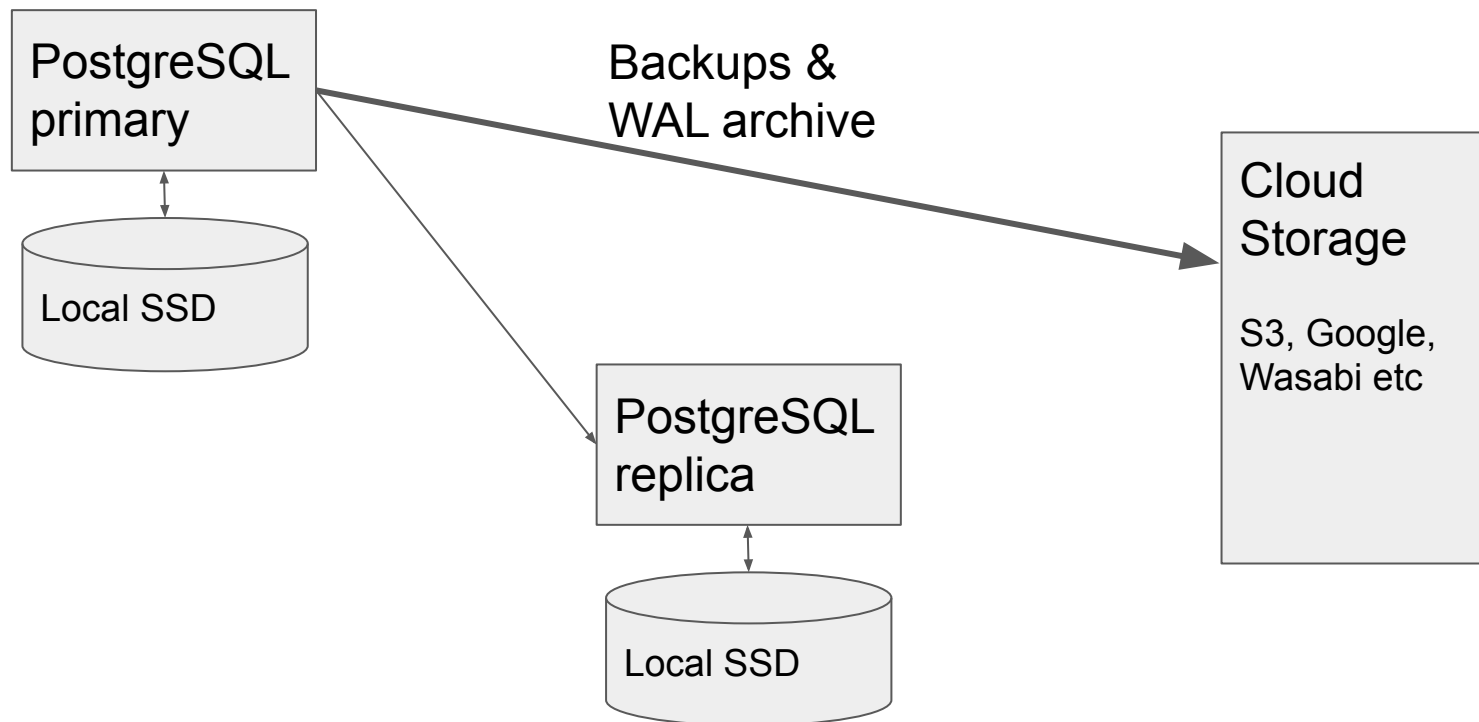
Traditional setup



Traditional setup *in the cloud*



Traditional setup *in the cloud*



Neon architecture

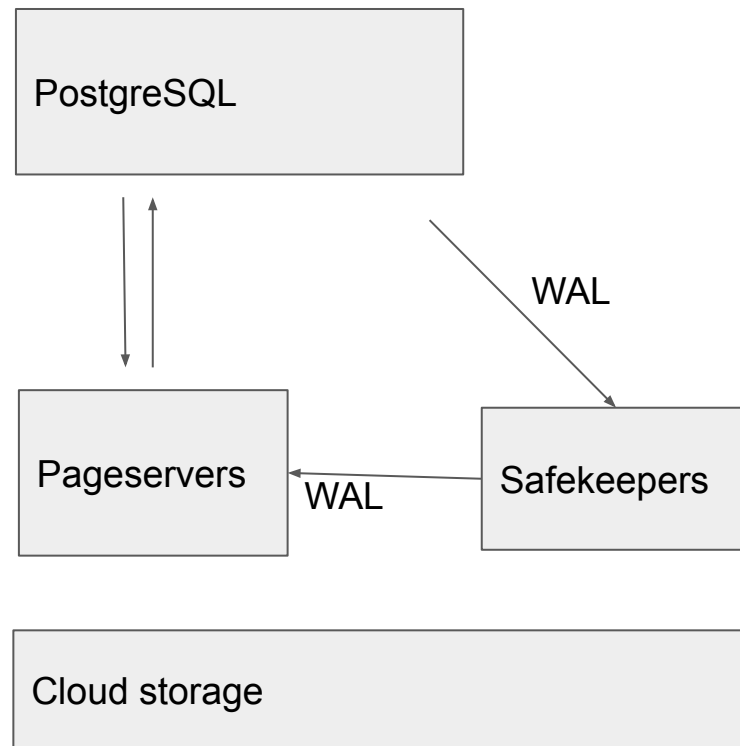
Storage and Compute are separated

Compute = PostgreSQL

- no persistent data
- runs in a container or VM
- starts up in 4 seconds

Storage = Neon storage system

- shared



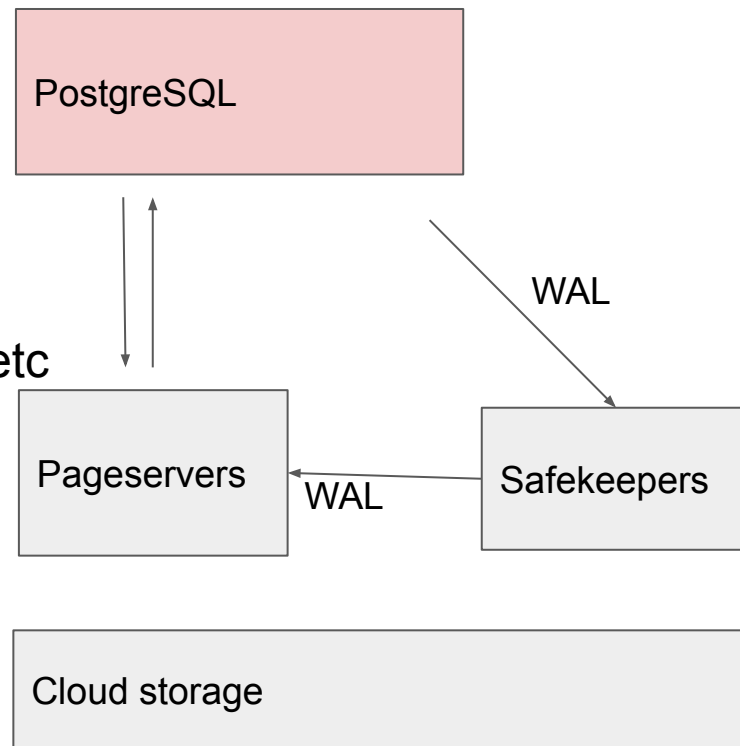
Why?

- Compute can be completely shut down, and started quickly
 - No restore from backup
 - No WAL recovery
 - Serverless!
- Same storage can be shared by multiple read-only nodes
- Scale independently
- Cloud storage is cheap

Storage and Compute are separated

PostgreSQL:

- streams WAL to the safekeepers
- reads pages from pageservers
- write() is a no-op
- local disk only for temporary files, sorting etc



Best platform to run PostgreSQL

We try to minimize changes to PostgreSQL:

- No changes to planner or executor
- Support all extensions, tools
- Support all PostgreSQL index types
- PostgreSQL handles MVCC

Replace low-level storage, close to where read() & write() happens

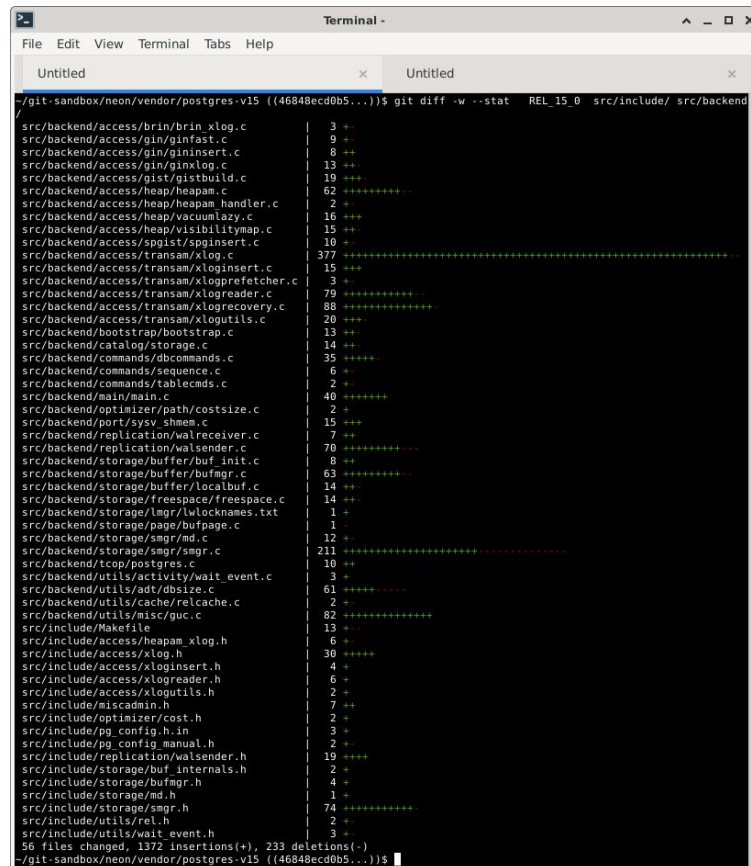
- *Goal is to get all changes into PostgreSQL*

What did you need to change in PostgreSQL?

Most code is in an extension. Core changes include:

- Make smgr API extendable
- Startup without crash recovery
- WAL-log command ID
- Track last-written LSN
- Cache relation sizes
- Prefetching sequential scans

https://github.com/neondatabase/neon/blob/main/docs/core_changes.md



```

Terminal -
File Edit View Terminal Tabs Help

Untitled x Untitled x

~/git-sandbox/neon/vendor/postgres-v15 ((46848ecd0b5...))$ git diff -w --stat REL_15_0 src/include/ src/backend

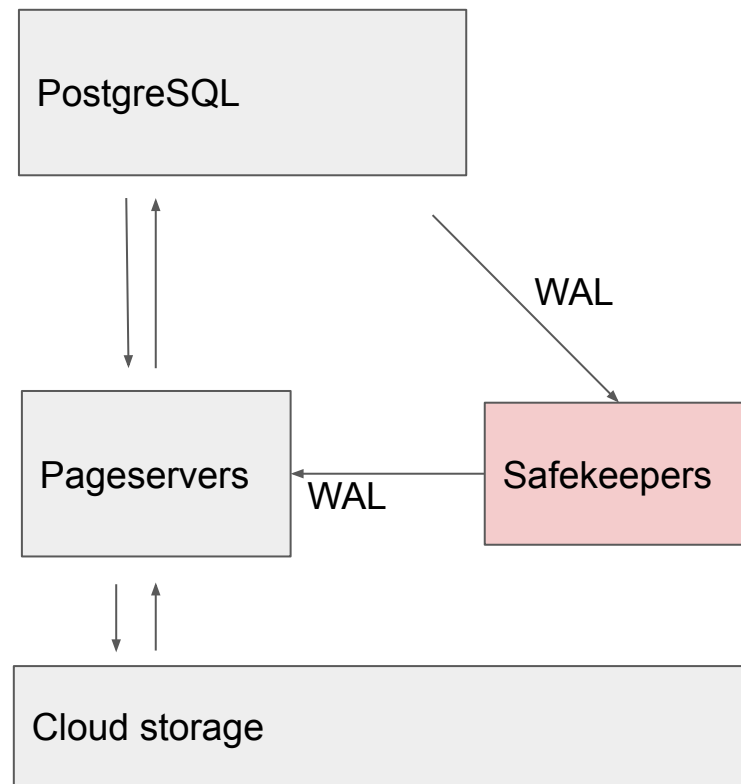
src/backend/access/brin/brin_xlog.c      | 3 +-
src/backend/access/gin/ginfast.c        | 9 +-
src/backend/access/gin/gininsert.c      | 8 +-
src/backend/access/gin/ginxlog.c        | 13 +-
src/backend/access/gist/gistbuild.c     | 19 +
src/backend/access/heap/heapam.c        | 62 ++++++
src/backend/access/heap/heapam_handler.c | 2 +-
src/backend/access/heap/vacuumLazy.c    | 16 +
src/backend/access/heap/visibilitymap.c | 15 +-
src/backend/access/spgist/spginserter.c | 10 +-
src/backend/access/transam/xlog.c       | 377 +
src/backend/access/transam/xloginsert.c | 15 +
src/backend/access/transam/xlogprefetcher.c | 3 +-
src/backend/access/transam/xlogreader.c  | 79 ++++++
src/backend/access/transam/xlogrecovery.c | 88 ++++++
src/backend/access/transam/xlogutils.c   | 20 +
src/backend/bootstrap/bootstrap.c       | 13 +
src/backend/catalog/storage.c           | 14 +-
src/backend/commands/dbcommands.c       | 35 +
src/backend/commands/sequence.c         | 6 +-
src/backend/commands/tablecmds.c        | 2 +-
src/backend/main/main.c                 | 40 +
src/backend/optimizer/path/costsize.c    | 2 +-
src/backend/port/sysv_shmem.c           | 15 +
src/backend/replication/walreceiver.c    | 7 +-
src/backend/replication/walsender.c      | 70 +
src/backend/storage/buffer/buf_init.c    | 8 +-
src/backend/storage/buffer/bufmgr.c     | 63 +
src/backend/storage/buffer/localbuf.c    | 14 +-
src/backend/storage/freespace/freespace.c | 14 +-
src/backend/storage/lmgr/lwlocknames.txt | 1 -
src/backend/storage/page/bufpage.c       | 1 -
src/backend/storage/smgr/md.c            | 12 +-
src/backend/storage/smgr/smgr.c          | 211 +
src/backend/tcop/postgres.c              | 10 +
src/backend/utills/activity/wait_event.c | 3 -
src/backend/utills/adf/objsize.c         | 61 +
src/backend/utills/cache/relcache.c      | 4 -
src/backend/utills/misc/guc.c            | 82 +
src/include/Makefile                      | 13 +-
src/include/access/heapam_xlog.h          | 6 +-
src/include/access/xlog.h                 | 30 +
src/include/access/xloginsert.h           | 3 +-
src/include/access/xlogreader.h           | 6 +-
src/include/access/xlogutils.h            | 2 -
src/include/miscadmin.h                   | 7 +-
src/include/optimizer/cost.h              | 2 -
src/include/pg_config.h.in                | 3 -
src/include/pg_config_manual.h            | 2 -
src/include/replication/walsender.h       | 19 +
src/include/storage/buf_internals.h       | 2 -
src/include/storage/bufmgr.h              | 4 -
src/include/storage/md.h                  | 1 -
src/include/storage/smgr.h                | 74 +
src/include/utills/rel.h                  | 2 -
src/include/utills/wait_event.h           | 3 -
56 files changed, 1372 insertions(+), 233 deletions(-)
~/git-sandbox/neon/vendor/postgres-v15 ((46848ecd0b5...))$

```

Write path

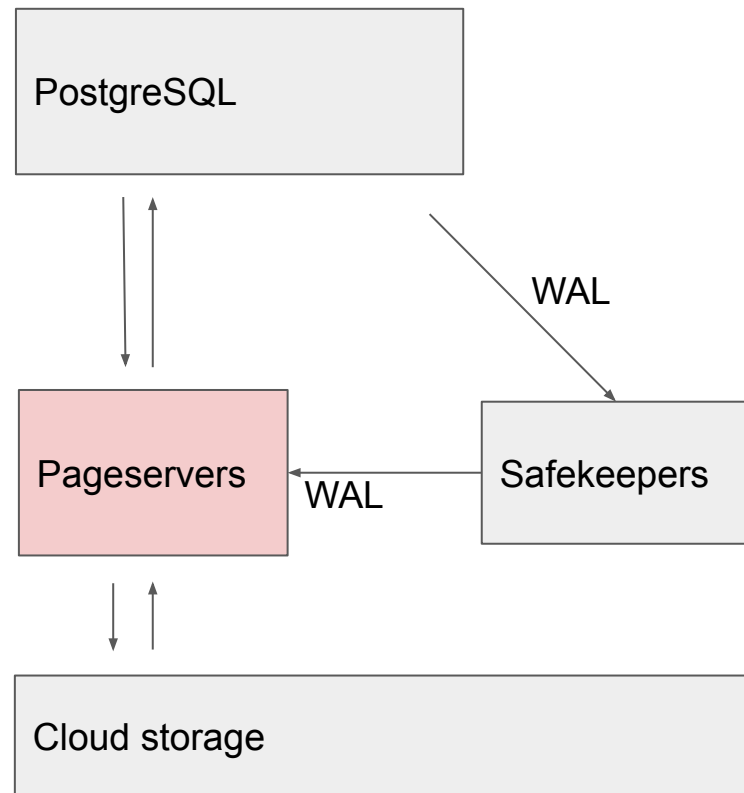
PostgreSQL streams the WAL to the safekeepers

- Three running safekeepers
- Consensus algorithm based on Paxos
- Ensures durability of recent transactions
- WAL is stored on local SSDs



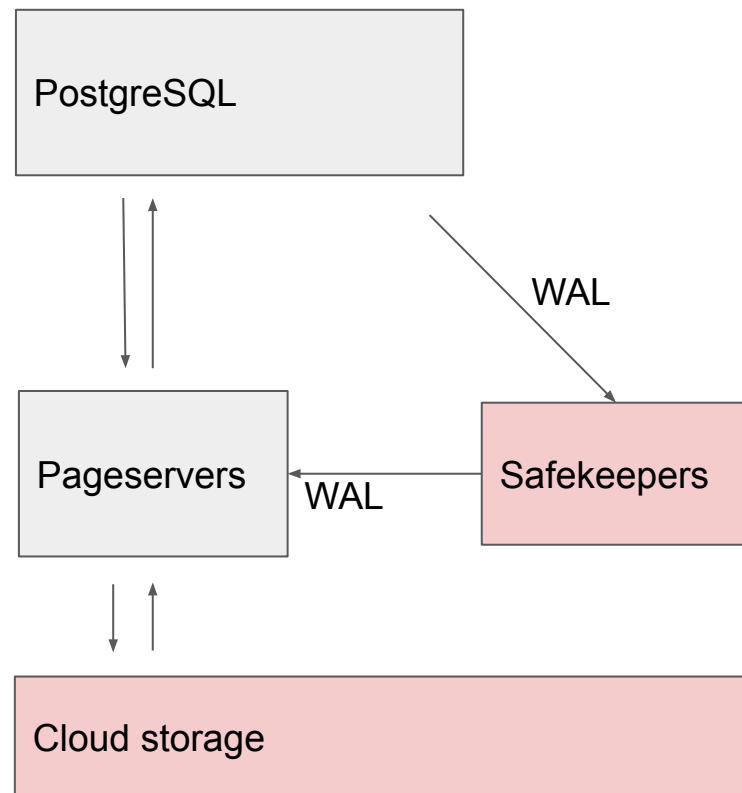
Write path: Pageservers

- Digests the PostgreSQL WAL
- Re-orders and processes it into immutable files
- Uploads files to cloud storage
- Local SSDs for caching



Durability

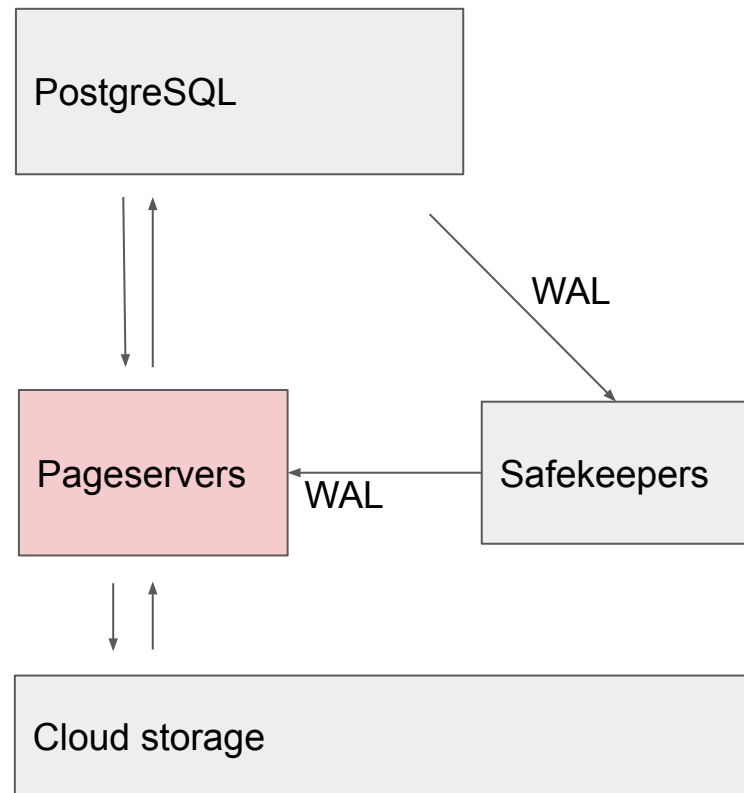
- Recent transactions (= recent WAL) are made durable in safekeepers
- Older WAL is uploaded to cloud storage, in processed format
- Pageservers are disposable



Read path: Pageservers

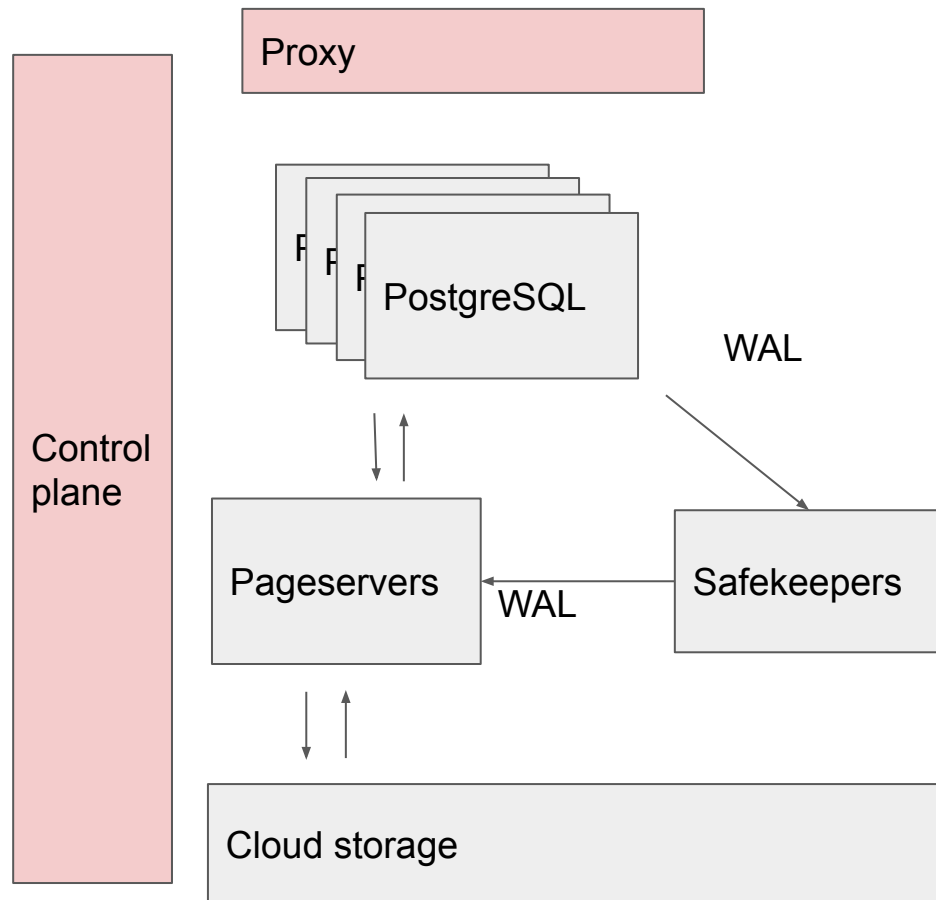
GetPage(Rel id, Block #, **LSN**)

- Replays WAL to reconstruct pages, *on demand*
- Can reconstruct any page **at any point in time**



Control plane and proxy

- Control plane starts and stops compute nodes
- Provide web user interface and user-facing API for creating databases, branches etc.
- Proxy to accept and authenticate user connections



Branching

- Want to try how an index would affect query plans?
- Want to run a long-running query?
- Want to have a fresh copy of your production database for testing?

Create a branch! They are copy-on-write and cheap

Branching

Branching replaces backups, WAL archive and Point-in-time Restore

- Instead of a backup schedule, configure *retention period*

Within the retention period, you can:

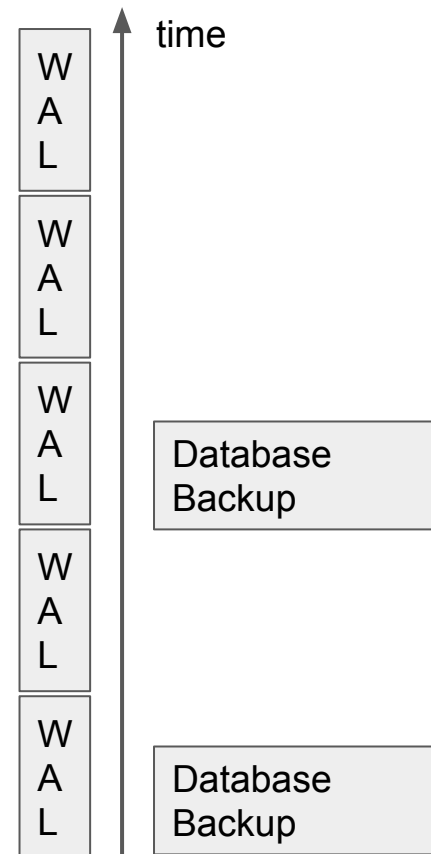
- Create a branch
- Timetravel
- Bisect, when was this row updated?

Traditional Point-In-Time Recovery

- Take daily backup
- and archive the WAL

To restore:

- Restore last backup before the point-in-time
- Replay all the log



Neon does this at *page granularity*

WAL contains a mix of:

- full page images of pages, and
- incremental WAL records.

To reconstruct a page version:

- Find the last image of the page, and
- Replay all the WAL records on top of it

To make this *perform*:

- Reorder and index the WAL
- Materialize and store additional page images

Current status

- Mostly works
- We are learning how to operate a cloud service
- Focused on performance, autoscaling
- Just rebased over PostgreSQL v15

Thank you!

Q & A

<https://github.com/neondatabase/neon/>

Try it: `psql -h pg.neon.tech`

Invite code: `pgconfeu`

Feedback: heikki@neon.tech

