credativ

Postgres with many Data

To MAXINT and beyond

Patrick Lauer <patrick.lauer@credativ.de>

04.09.2025

Patrick Lauer

credativ

- <patrick.lauer@credativ.de>

- ca.2 years at credativ
- Professional Services Consultant
- Mostly Databases, especially PostgreSQL
- Originally Sysadmin/Devops/Platform Engineer
- Gentoo Linux Developer

credativ GmbH

- Founded 1999 in Mönchengladbach
- Close ties to Open-Source Community
- 40 Open-Source experts
- Consulting, development, training, support (3rd-level / 24x7)
- Open-Source infrastructure with Debian, Kubernetes and Proxmox
- Open-Source databases with PostgreSQL
- DevOps with Ansible, Puppet, Terraform and others
- Since 2025 independent owner-managed company again

Overview

credativ

- (my) History
- The root of most problems
- PostgreSQL limits
- Hardware-Limits
- PostgreSQL with many data
- Scaling issues

credativ

(my) History

How I ended up here

credativ

- ca. 2015: New Job as Sysadmin, using lots of PostgreSQL
- Chris Travers: Postgres at 10TB and beyond
- … we hired Chris
- a few weeks later: Chris Travers: Postgres at 20TB and beyond
- Elasticsearch doesn't scale, we replace it with PostgreSQL:
- pgconf.ru: Wiktor Kerr does a talk on Bagger

Fun with PostgreSQL

credativ

- Bagger: Petabyte-sized Logging with PostgreSQL
- internal project: 400TB+ statistics/rollups, distributed over 32+ Servers
- internal project: 20TB+ in 400k+ tables, one table per event-type
- internal project: 200TB+ in a single DB, specific data extracted for pattern analysis (fraud etc.)
- PostgreSQL was always our default

We are not alone

credativ

- Operational hazards of managing PostgreSQL DBs over 100TB (adyen) (earlier today!)
- pgconf.de .eu: Gitlab
- Chris Travers / One More Data
- and many many more

Lesson 0

credativ

- Best to not have many data
- Many data = many problem

Normalization of Deviance

credativ

- Always close to the limits
- Risk perception slowly shifts
- ”Best practise” doesn’t apply
- ”and then we disabled fsync to make it run faster”
- ”it’s an append-only workload, let’s disable autovacuum”

credativ

The root of most problems

MVCC

credativ

- Multi-Version Concurreny Control
- Needed to have stable (repeatable) results with concurrent queries
- PostgreSQL: every row (tuple) has lifetime / visibility
- Consequence: No in-place updates!

ARIES

credativ

- Algorithms for Recovery and Isolation Exploiting Semantics
- Crash safety is handled by serializing all changes into a log that can be replayed
- Checkpoint allows truncating log as data before checkpoint is guaranteed to be persisted
- In PostgreSQL this is implemented as WAL
- LSN (Log Sequence Number) orders each distinct unit of change
- Enables crash recovery and point-in-time recovery

Why this is challenging

credativ

- WAL: Single mutex for all database changes!
- WAL limits theoretical scalability
- MVCC needs cleanup: Vacuum
- (Auto)Vacuum creates write amplification
- failing to run Vacuum creates performance issues

The NoSQL option

credativ

- If we don’t care about persistence we can go faster
- If we don’t care about correctness we can go faster
- Does this satisfy your requirements?

credativ

PostgreSQL limits

Limits

credativ

- Documentation says:
- number of databases: 4,294,950,911
- relations* per database: 1,431,650,303
- rows per table: limited by the number of tuples that can fit onto 4,294,967,295 pages (ca. 32TB**)
- these limits are unlikely to be reached
- *Relation is anything that is stored in the catalog table pg_class: tables, views, sequences, indexes, materialized views, partitioned tables and partitioned indexes.
- ** with 8KB Blocksize

Limits

credativ

- Documentation says:
- columns per table: 1,600**
- field size: 1 GB (TOAST size limit)
- field size (jsonb): 256MB
- large objects: oh dear - please avoid this
- shared buffers is limited to 256GB **
- ** with 8kB blocksize

Limits

credativ

- There are relatively strict limits on the size and shape of objects we can persist
- There are fewer limits on the numbers of such objects

credativ

Hardware-Limits

Hardware

credativ

- Servers can get very big
- 256+ CPU-Cores, 4TB+ RAM is easy (but not cheap)
- large systems often show interesting issues with NUMA (pgconf.eu talk by Andres Freund)
- storage can get very very large - >1PB in 1U is easy
- SAN/NAS mostly limited by wallet size

Hardware

credativ

- Largest single system I'm aware of:
- IBM z17
- up to 208x 8-core CPUs @ 5.5Ghz
- up to 64TB RAM
- nice hardware, but is it an effective solution to our problems?

Clustering

credativ

- Data is maybe too big for a single server -
- ”Somehow” distribute PostgreSQL over multiple servers
- Application sharding?
- or maybe citus, pgdog, Greenplum/CloudberryDB, PostgresXL/XC
- Distributed systems are either ”slow” or lose ACID features

Sharding

credativ

- Citus: No global transaction synchronization
- Partial visibility of transactions possible, ”eventually consistent”
- Application sharding: emulate transactions outside the DB?
- 2-Phase-Commits are slow and complex
- Performance: every query has some network latency added, overload of single components possible
- Relational database without ACID? Why not NoSQL or some other modern solution?

Chris' rule of thumb

credativ

- Chris Travers: "Every time you grow a system by one order of magnitude some components will become a bottleneck"
- … but we don't know which component
- Constantly observe, adapt, adjust

Lesson 1

credativ

- A lot of problems are mostly financial
- Big Data, Big Budget
- Some problems remain that can’t be squashed with money directly

credativ

PostgreSQL with many data

practical limits

credativ

- ”Billion Tables” talk (PGCon 2013): filesystem can become a bottleneck
- ”Velocity”: WAL is limited to 1-2GB/s*, this is a global throughput limit
- even small databases can be difficult (high rate of change, long-running transactions)

Data at rest

credativ

- Data that is not queried has very little cost
- Once VACUUM FREEZE is done there is no further maintenance
- Only mutation (changing data) has maintenance cost

Small databases

credativ

- 1GB: ”fits in CPU-cache”, everything is reasonably fast
- backup/restore in an instant
- ALTER TABLE needs a second or three
- Tables and indexes are few enough to be manually managed

1TB

credativ

- Data (usually) doesn't fit into RAM
- Tablescans are generally slow, indexes needed
- Too many indexes cause problems too!
- Autovacuum is often difficult to handle and needs careful adjustment
- Autovacuum: Defaults are very conservative
- Performance tuning: fill factor, tuple_cost, checkpoints
- Query tuning:
- Slow query log, EXPLAIN (ANALYZE, BUFFERS [,...])

10TB

credativ

- Backup and Restore times can exceed SLA / RTO
- Replication can be challenging: How long does it take to clone a new replica?
- Large tables may benefit from partitioning (helps autovacuum and can speed up queries)
- Sequences - int vs. bigint: easy to overflow int, best to use bigint all the time
- Wraparound vacuums - when autovacuum can’t keep up properly
- Number of tables might require automation

100TB

credativ

- ”Velocity” (rate of change) limited by total throughput of WAL writer
- Autovacuum can write changes to WAL faster than replicas can apply the changes
- Self-DoS: Query results can become huge
- Many tables vs. large tables: different pain points
- Partitioning can make query plans huge, and can make query planning take a very long time
- Schema changes can take a long time, and lock out other queries

1PB

credativ

- Expensive to have this much hardware attached to one server
- RAM-Bandwidth as bottleneck?
- ”blast radius”: What happens if this server is unavailable?
- Can a single server manage the required amount of connections?
- HA / DR is needlessly exciting
- I am not aware of any single PostgreSQL install this large (yet)

Lesson 2

credativ

- Operational problems scale with data size
- Requires having enough skilled employees (or good partners)
- Near the limit you'll find issues that are not well documented

credativ

Scaling issues

Table size

credativ

- MVCC: Data doesn't get deleted immediately as older transactions may still see it
- (Auto)Vacuum: asynchronous cleanup of old data
- (Auto)Vacuum runs a very long time on large tables
- Wraparound-vacuum: transaction-ID is 32bit
- Indexes too large, too bloated, index access is too expensive (B-trees!)
- effective limit of a few TB for a single table

PostgreSQL

credativ

- Write Ahead Log: Write summarized changes into a sequential log, ensures durability, enables crash recovery
- WAL is global bottleneck
- Replication limits rate of change: Replicas usually can’t sustain same write volume as Primary

Backup

credativ

- Backup is at least basebackup plus WAL
- Save WAL for weeks/months? That's huge.
- Backup: basebackup can block WAL-cleanup, can accidentally fill storage
- Backup may acquire locks that prevent DDL, even on replicas
- Restore: How long does it take to restore from backup?

Clients

credativ

- Many connections: just increase max_connections?
- max_connections affects internal datastructures, can cause performance issues
- Many parallel connections can slow down transaction handling
- Every connection is its own process - how does the OS handle thousands of processes?
- Connection Pooler?

Queries

credativ

- Long-running queries can block Autovacuum
- this can cause Table Bloat
- or weird issues with row visibility, index bloat, ...
- If Autovacuum is stalled for a longer time it has more to do to catch up
- If it gets blocked too long: wraparound vacuum
- Long-running queries (can) block DDL - even on replicas!

Concurrency

credativ

- Concurrent queries can have locking issues (deadlock, lock waiting)
- Lock contention: queries might effectively be serialized
- Maximum number of global locks can get exhausted
- Shared_buffers has maximum size, and bigger isn't always faster
- Query diversity can lead to eviction of pages from shared_buffers
- Indexes of large tables might not fit into shared_buffers, index access "slow"

Blast Radius

credativ

- Does failover to a replica work as intended? (e.g. WAL replay can take a long time)
- How long to restore redundancy after a failure?
- How long to restore data (e.g. accidental DROP TABLE)
- Will the projected growth fit into available hardware?

Specialisation

credativ

- If requirements allow for it -
- Disable fsync, use unlogged tables
- Disable Autovacuum (per tablespace?)
- Batch insert + vacuum, less work for autovacuum
- Partitioning, drop partition instead of large deletes
- Replicate data from the application instead of replicating in PostgreSQL
- Tradeoff: convenient vs. fast, cost vs. complexity

The future?

credativ

- Different storage-engines? Columnar Storage, OrioleDB etc.
- Improvements with (auto)Sharding and QueryPlanner - Timescale etc.
- 64bit TransactionIDs ?

The Future?

credativ

- Distributed PostgreSQL - Greenplum, Citus, PostgresXC/XL
- PG-compatible frontend, but new backend - CockroachDB, YugabyteDB, ...
- Autoscaling / Cloud Native? NeonDB

Inspiration

credativ

- Foreign Data Wrappers - just connect to other DBs
- Extensions - add missing functionality
- Just put another DB inside PostgreSQL - pg_duckdb

A different future

credativ

- Maybe we don’t need to save all data
- Maybe we don’t need to keep all data forever
- Maybe we can avoid most of the problems by not having so many data

To think about

credativ

- Datensparsamkeit: best to not have the data
- Budget: Data has cost
- Compliance: Data has (legal) risks
- Staffing: To solve complex issues you need good motivated people
- And sometimes you will need to build something yourself

Takeaway

credativ

- More data, more problems
- Big data, big budget
- Architectural problems vs. operational problems
- (non-financial) cost of data: Compliance etc.

credativ

Questions?