PostgreSQL Commitfest Metrics

A Quantitative Analysis

by Andreas Scherbaum (EDB) + Jimmy Angelakos (pgEdge)

PGConf.dev 2026-05-20

What this talk is **not**

- A critique of any contributor
- A critique of any committer
- A claim that anything is broken
- A diagnosis, this is an observation
- A recommendation going forward

What this talk **is**

The question we want to open:

- What changes would?
  - Improve the contributor experience
  - Lower barriers
  - Attract new contributors
  - While not compromising our standards for technical excellence
- And we have data!

Who are we?

Andreas Scherbaum

- WarehousePG Principal Engineer, EDB
- PostgreSQL Major Contributor
- PostgreSQL Europe Board of Directors
- Meetup & Conference Organizer
- Author, “PostgreSQL - Datenbankpraxis für Anwender, Administratoren und Entwickler“ (German)

Jimmy Angelakos

- Staff Software Engineer, pgEdge
- PostgreSQL Significant Contributor
- PostgreSQL Europe Diversity Committee
- Author, PostgreSQL Mistakes and How to Avoid Them
- Co-author, PostgreSQL 16 Administration Cookbook
- `pg_statviz` PostgreSQL extension

Four numbers

- 42.6% of authors submitted **exactly one** patch
- 35.2% of patches **never** committed
- 46 patches stuck for **2+ years**
- One patch is open for **3,750 days**

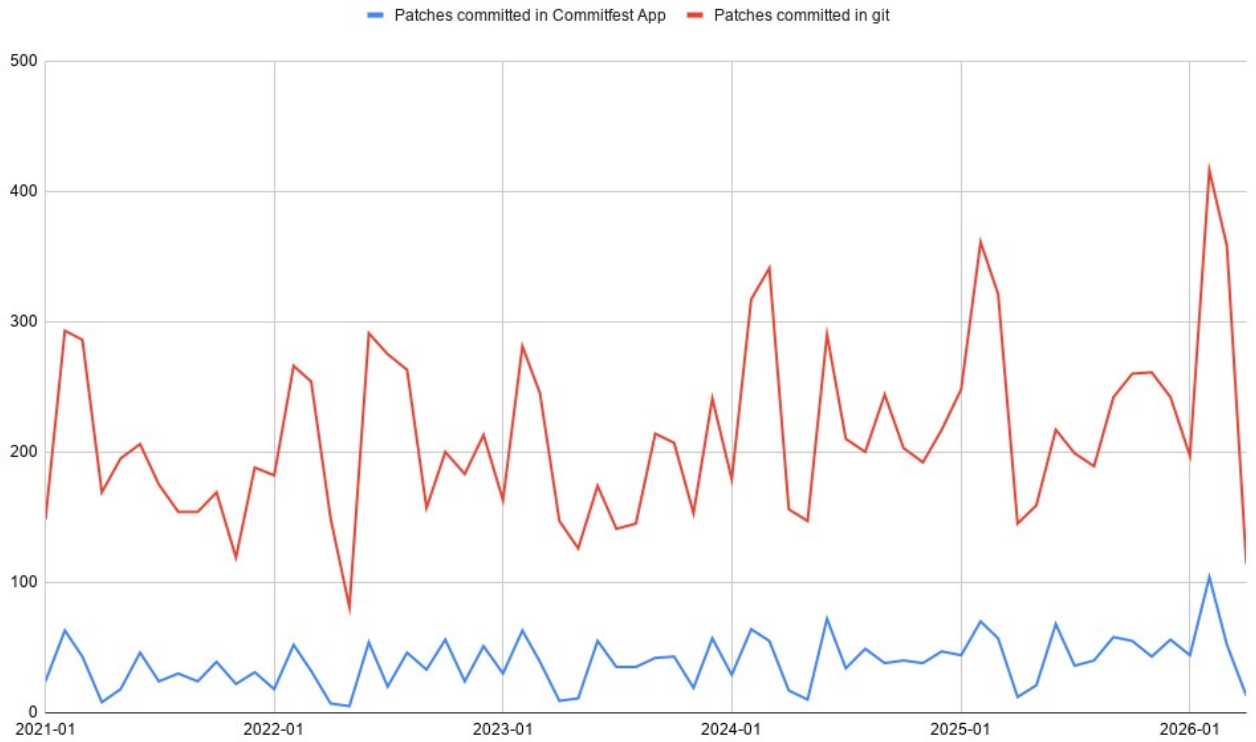
What is Commitfest?

- Not a Patch Tracking tool
- Not a CI interface
  - At least not initially
- A tool only the PostgreSQL Project is using
  - Steep learning curve for new developers

Not a Patch Tracker? (i)

- On average (2021-2026) only 17% of git commits appear in any Commitfest
  - About 6x more git commits than Patch Status = “Committed” in Commitfest
- No audit/track for patches
  - No cross-reference between commit id, mailing list discussion and Commitfest

Not a Patch Tracker? (ii)



Not a CI Interface

- Commitfest exists since 2014
- CFBot exists since ~2018
- CFBot integrated into Commitfest ~2025

ID	7
Title	Foreign table inheritance
CI (CFBot)	Not processed
Stats (from CFBot)	Unknown

What we measured

- 58 commitfests (56 closed, 2 open)
- 6,678 patches
- 899 authors with patches under their name tracked
- December 2014 - April 2026

A statistics refresher

Median

The middle value

- Not the midpoint of min and max
- 5 in 10 are at or below

P90

9 in 10 are at or below

P99

99 in 100 are at or below

Percentiles!
We prefer them to the **average**
The average gets dragged by the tail

Outcomes: what happens to patches?

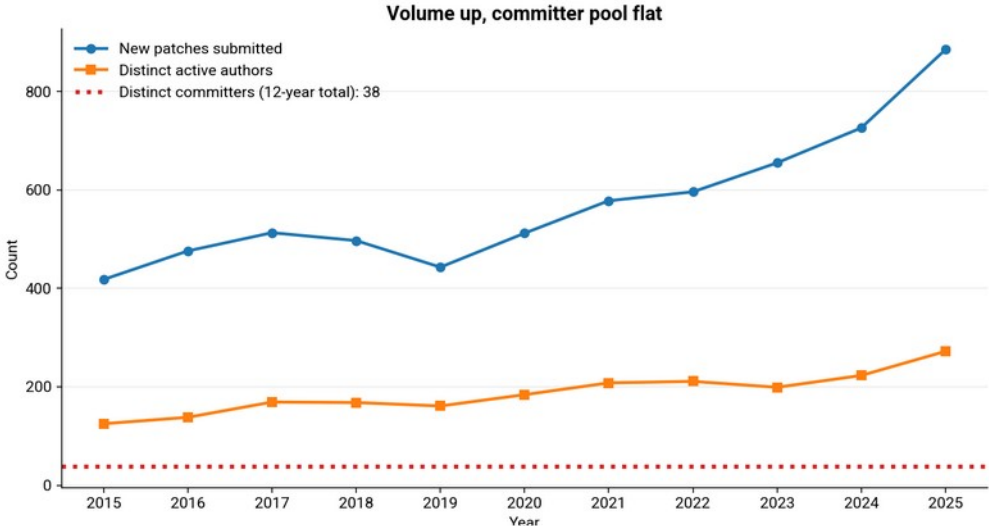
- Committed: 64.8%
- Returned with feedback: 16.5%
- Withdrawn: 6.9%
- Rejected: 3.6%
- Open / rolling: 8.2%

} the 35.2% never committed

- Time to commit since submission
  - Median: 80 days
  - P90: 256 days
  - P99: 711 days
  - Max: 1,980 days

How have we scaled?

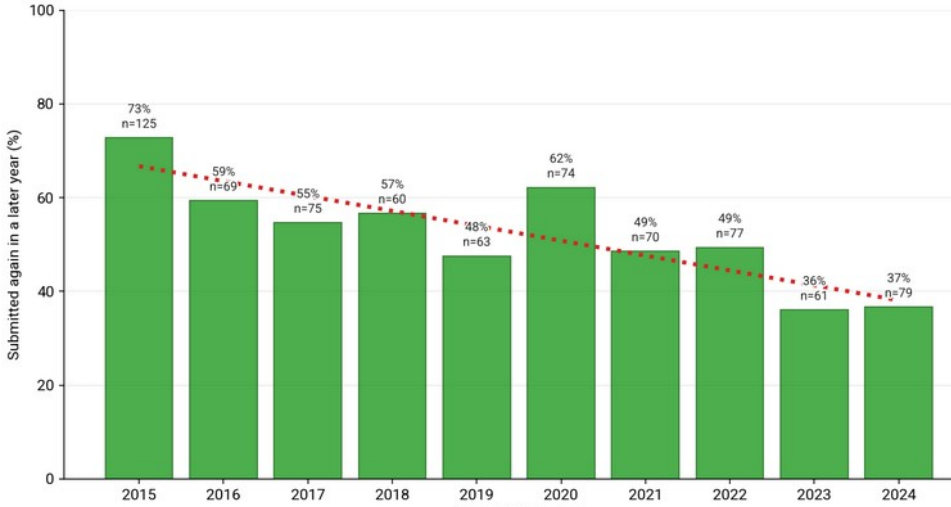
- 2015: 418 patches, 125 authors
- 2020: 512 patches, 184 authors
- 2025: 885 patches, 272 authors
- Roughly 2x growth on both sides
  - Except merges didn’t scale
- Total committers over 12 years: 38 (stable)
- Actually committing: 26.5 (median)



Do first-time authors come back?

% of authors who returned with first patch in:

- 2015: 73% ↓
- 2017: 55% ↓
- 2019: 48% ↓
- 2021: 49% →
- 2023: 36% ↓
- 2024: 37% →
- 2025 / 2026: too soon to tell



Who does the work?

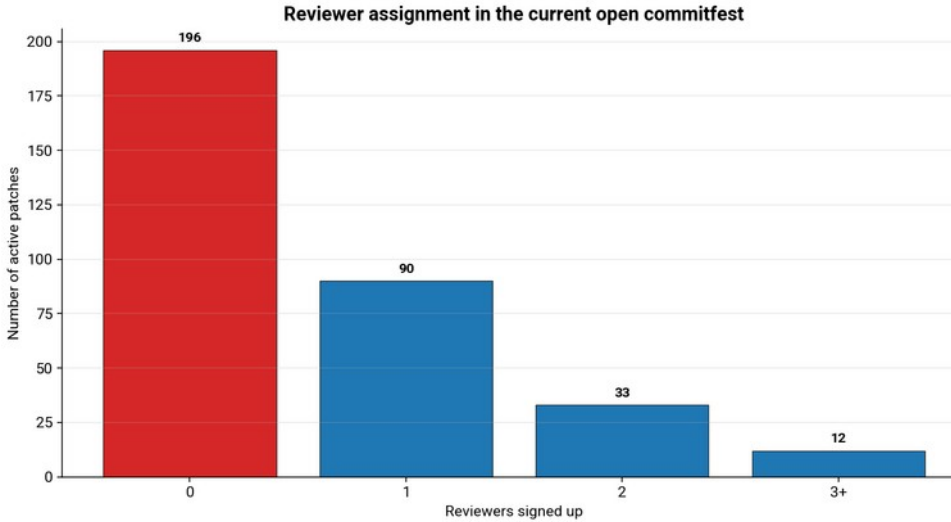
- Top 5 committers: 55% of all commits
- Top 10 committers: 74%
- 4,435 total commit credits
  - Just from Commitfests
  - Mostly external contributions (?)

Review bandwidth

- 60.5% of patches have a reviewer signed up
- Among those, average: 1.41 reviewers
- 72% of reviewed patches have **exactly one** reviewer
- Across all history: 72.1% of "Needs review" patches had **no** reviewer
- NOTE: Commitfest review only
  - `pgsql-hackers` review happens too, but isn't in the data

Current commitfest(s)

- 331 active patches
- 196 of them (59.2%) have **zero** reviewers
- Median wait for those: 54 days
- P90: 244 days
- Oldest: **892** days



Ready for Committer

- Is it really a bottleneck?
- 54 patches are RfC right now
- Median time in RfC: 3 **days**
- P90: 30 days
- P99: 95 days
- Oldest: 354 days
- The backlog is **upstream** of RfC, not at it

The rollover funnel

(i)

Commit rate

How many commitfests a patch appears in

- 1 CF: 74%
- 3 CFs: 49%
- 5 CFs: 54%
- 7 CFs: 41%
- 11 CFs: 17%

The long tail

One patch

- Has appeared in 41 commitfests
- Was first submitted in 2018
- Is still active

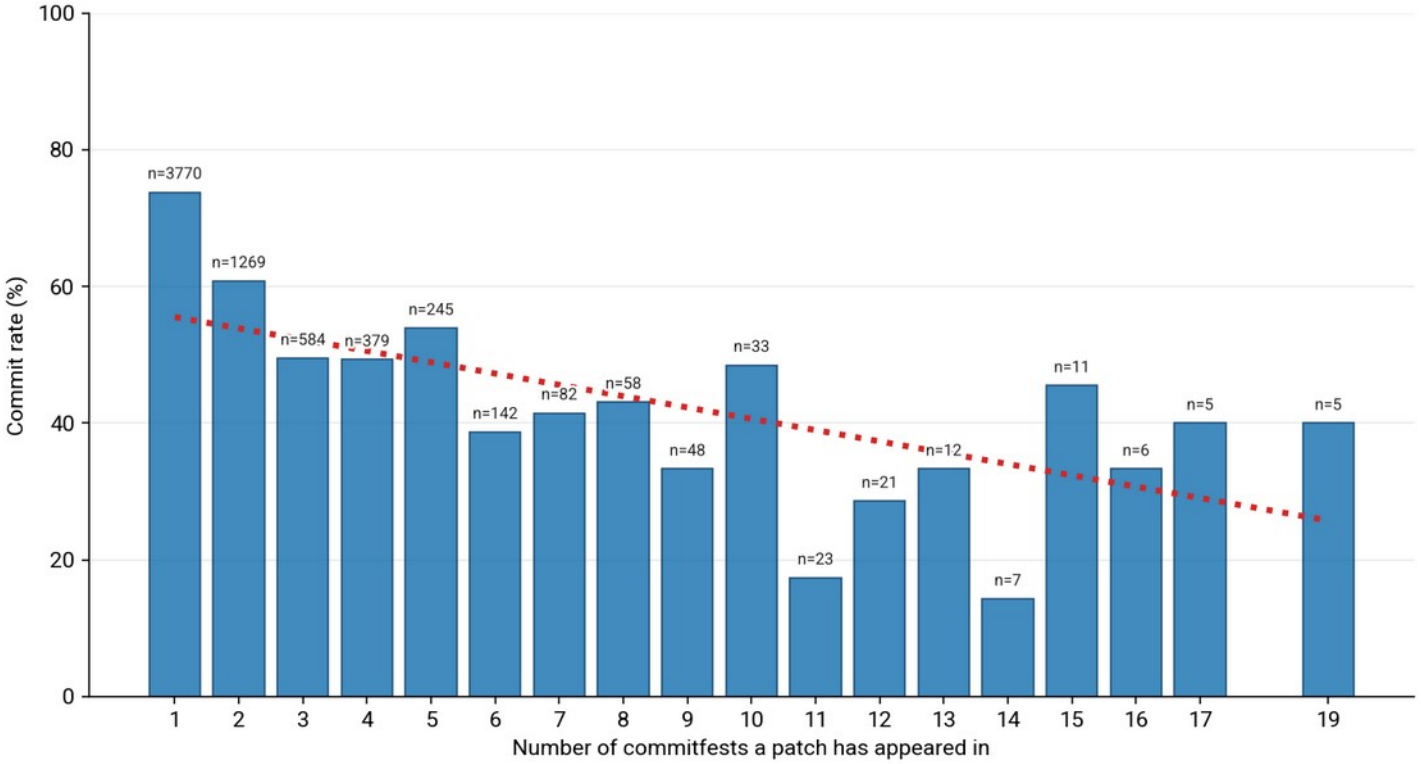
So…

First pass: high commit rate

- Each rollover drops that chance sharply

The rollover funnel

(ii)



The long tail

- One patch first submitted in early 2016 is **still open**
- Over 3,750 days
- That's **10 years**
- Not committed, not rejected, not withdrawn
- CF tracker has no closure path short of formal rejection

“There are known unknowns” *— Some guy, once*

- **What we know:** what we just showed you
- **What we don’t:**
  - Retention and commit rates for 2025-2026, too soon
  - Mailing list review, not on the CF tracker
  - Also: ML discussions make Commitfest redundant for review
  - ~80% of patches are only on -hackers
- **“Unknown unknowns”:**
  - Suggest through the Q&A 😁

How have other projects navigated this?

- Perl: cautionary tale
  - Cultural drift, lost the next generation to Python
- Linux: culture pivot
  - Code of Conflict → Code of Conduct (professionalized their standards), `MAINTAINERS` file
  - Separated work into many submodules -> different maintainers/reviewers/committers
- Rust: onboarding empathy
  - Helpful compiler diagnostics, governance teams, RFC process
- Django: onboarding empathy
  - DSF-funded Fellows triage and review as paid job

Us & other projects: methodologies differ

- PostgreSQL
  - 36–37% recent first-time authors come back
  - 38 committers in 12 years
- Linux
  - ~2,200 `MAINTAINERS` entries
  - First-timers ~10–15% per release
- Rust
  - ~1,000 contributors in 2015 → ~6,700 in 2025
- Django
  - 2 paid DSF Fellows
  - Small core with merge rights

Our reading

- The data is consistent with a process that is **under strain**
- The patch volume has doubled
- The committer pool is flat
- Patches roll forward instead of resolving
- Patches wait without ever getting a reviewer
- There are no dashboards for showing overall patch status

And the Commitfest app?

- Only a few patches (~17%) are tracked there
- No audit trail whatsoever
- Heavily used to kick off CFBot tests

Where may changes be impactful?

- Making formal closure easier to issue & receive
- Making reviewer routing explicit so patches don’t linger
- Making triage an official task: route, defer, close
- Our suggestions. Yours may be better.

The question

restated

What changes would improve:

- the contributor experience
- lower barriers
- attract new contributors

without compromising our standards for technical excellence?

Thank you

Questions?