PostgreSQL Connections Memory Usage: How Much, Why, and When? (Josef Machytka: Speaker portfolio)

Josef Machytka
2 min read · Feb 9, 2025

(See my bio, other talks in my portfolio and my speaker experience in the covering article.)

Duration: 45 minutes
Target Audience: Database developers, DBAs

This talk explores the memory usage of PostgreSQL connections on Debian/Ubuntu running on x86–64 architecture. It gives an overview of memory management concepts, explaining key metrics like virtual, resident, and proportional memory sizes. It also covers various Linux tools for displaying memory usage.

The second part presents practical measurements of PostgreSQL memory usage, based on data from /proc/PID/smaps and /proc/PID/pagemap. It explains why RSS numbers for PostgreSQL connections in `top` command appear so large after query execution and demonstrates that the actual unique memory usage is only a few dozen megabytes.

Finally, the talk also reveals where work_mem is “hidden” in these statistics, visualizes memory usage during query processing with multiple plots, and discusses queries that exceed the work_mem setting multiple times. It also demonstrates the total memory usage when a query utilizes parallel workers.

Key Takeaways:

- The large RSS values in long-running sessions mostly come from linked shared_buffers
- A newly created connection consumes only up to 10 MB of physical memory, independent of the work_mem setting
- Additional memory is allocated and later released as queries execute
- Work_mem is a “soft maximum limit” — it may not be fully used, but it can also be exceeded

Slides: not available online

Presented at:

- NetApp internal talk 2025.01.27
- (Talk was chosen as a reserve talk for Prague PostgreSQL Developer Day 2025.)

title picture of slides

Postgresql Memory Usage Linux

Edit profile

Written by Josef Machytka

68 Followers · 25 Following

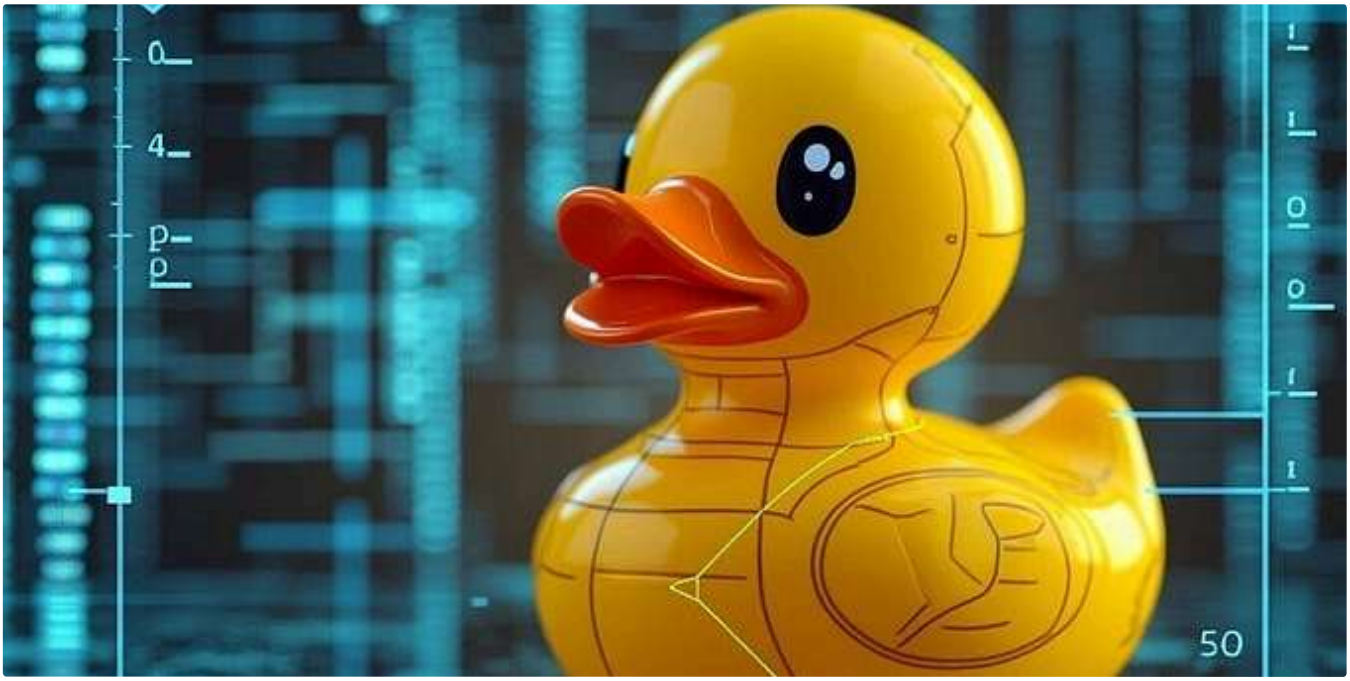
I work as PostgreSQL specialist & database reliability engineer at credativ GmbH.

No responses yet

Josef Machytka he/him

What are your thoughts?

More from Josef Machytka

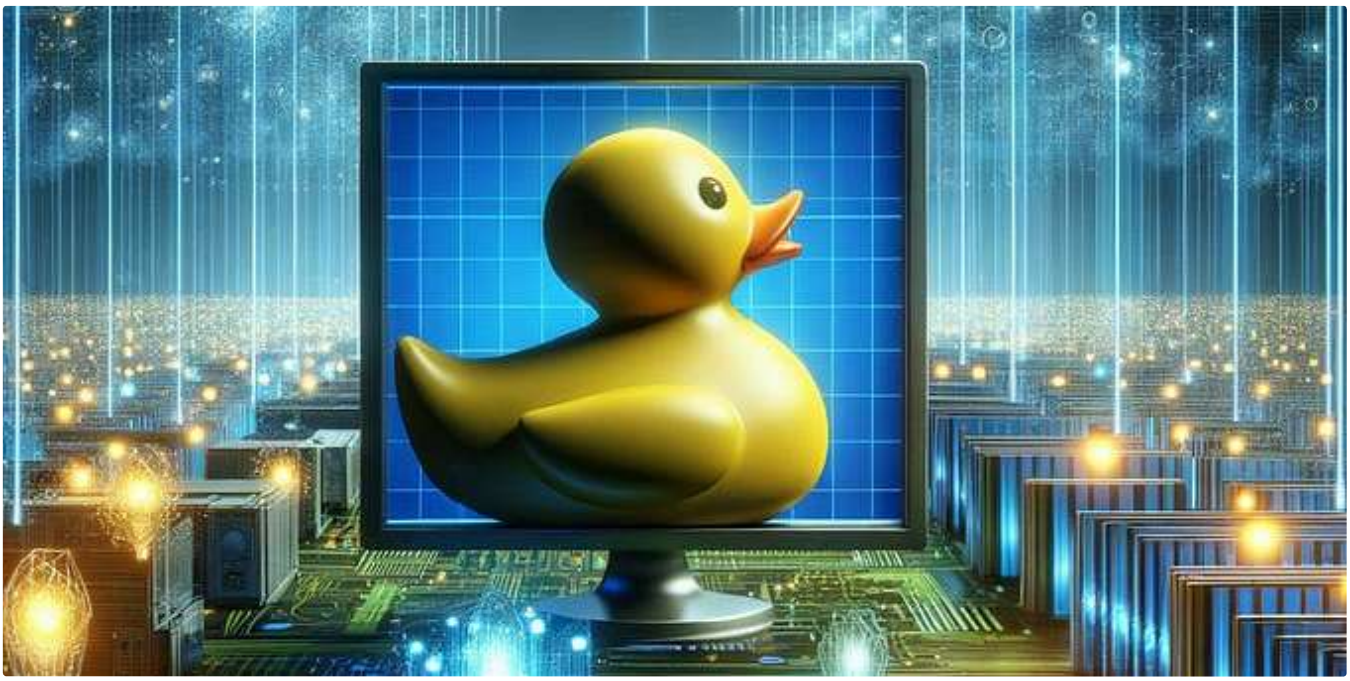


Josef Machytka

DuckDB Database File as a New Standard for Sharing Data?

This is not my original idea; I came across it in an excellent article titled “DuckDB Beyond the Hype” by Alireza Sadeghi. However, it...

Dec 30, 2024 24 3



Josef Machytka

Quick and Easy Statistics and Histograms with DuckDB

DuckDB is an exceptional tool that demonstrates how tasks requiring sometimes considerable manual effort in other tools can be accomplished...

Dec 16, 2024 26

Josef Machytka

PostgreSQL JSONB Operator Classes of GIN Indexes and Their Usage

Throughout 2024, I worked on an internal project exploring the use of JSONB data in PostgreSQL and its various indexing options. During...

Jan 8 5

```
D pivot pg.sales on (product,year) using sum(sales_amount) group by salesperson order by salesperson;
```

salesperson varchar	(Laptop, 2022) double	(Laptop, 2023) double	(Phone, 2022) double	(Phone, 2023) double	(Tablet, 2022) double	(Tablet, 2023) double
Alice	1200.0	1400.0	800.0	900.0	300.0	400.0
Bob	1000.0	1100.0	600.0			
Charlie	1100.0	1200.0	700.0	800.0	500.0	600.0

```
D pivot pg.sales on (year,product) using sum(sales_amount) group by salesperson order by salesperson;
```

salesperson varchar	(2022, Laptop) double	(2022, Phone) double	(2022, Tablet) double	(2023, Laptop) double	(2023, Phone) double	(2023, Tablet) double
Alice	1200.0	800.0	300.0	1400.0	900.0	400.0
Bob	1000.0	600.0		1100.0		
Charlie	1100.0	700.0	500.0	1200.0	800.0	600.0

```
D pivot pg.sales on (year) using sum(sales_amount) group by salesperson order by salesperson;
```

Josef Machytka

Easy and Intelligent Pivot Tables with DuckDB

After exploring the various capabilities of DuckDB in my earlier articles, I want to focus more on its powerful data analytical...

Dec 4, 2024 7 1

See all from Josef Machytka

Recommended from Medium

Query Optimization Method	Execution Time	Performance Improvement (vs. No Index)
No Index	42,049 ms (≈42 sec)	Baseline
With B-tree Index	9,684 ms (≈9.7 sec)	77% faster
With Chunk-Skipping Index + Columnstore	304 ms (0.3 sec)	99.28% faster

In Timescale by Team Timescale

Handling Billions of Rows in PostgreSQL

Here's how to scale PostgreSQL to handle billions of rows using Timescale compression and chunk-skipping indexes.

Jan 17 72 1

Kiran Maan

STOP Using Python Dictionaries Like This!!!

Sometimes...I see that some people use Python dictionaries incorrectly.

Feb 20 303 6

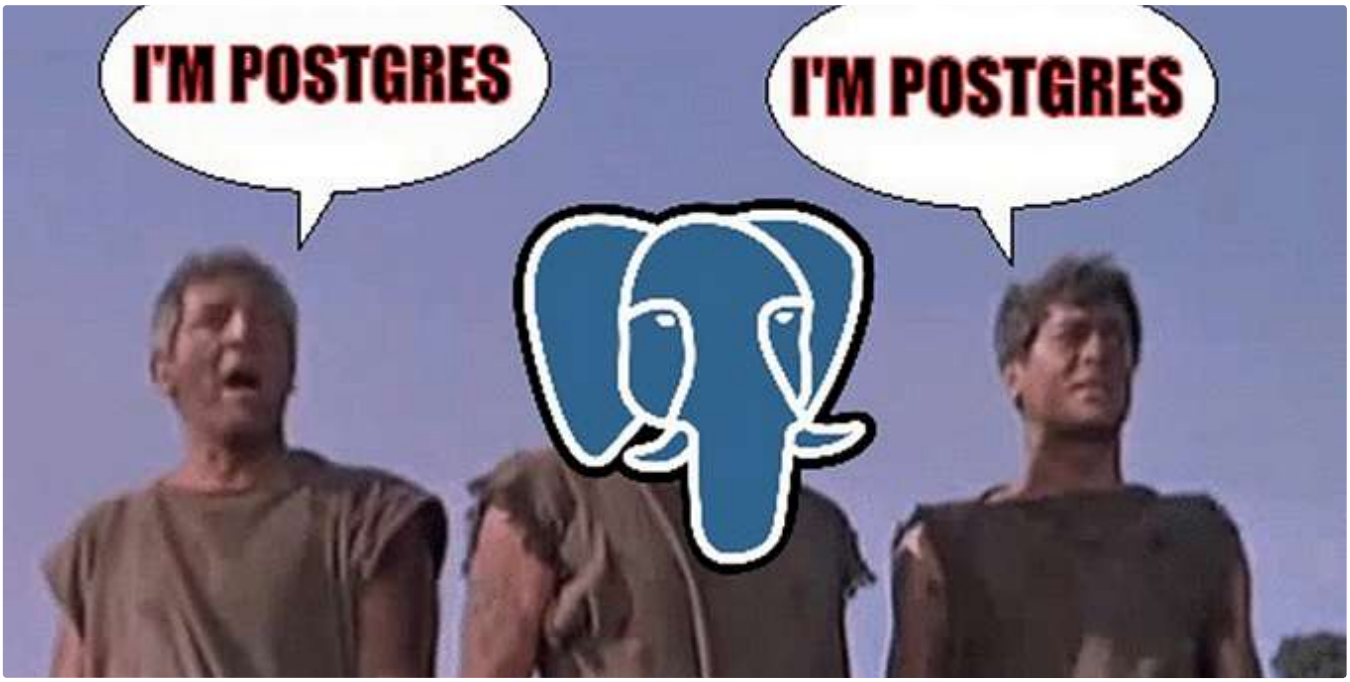
Lists

General Coding Knowledge

20 stories · 1930 saves

Staff picks

819 stories · 1637 saves

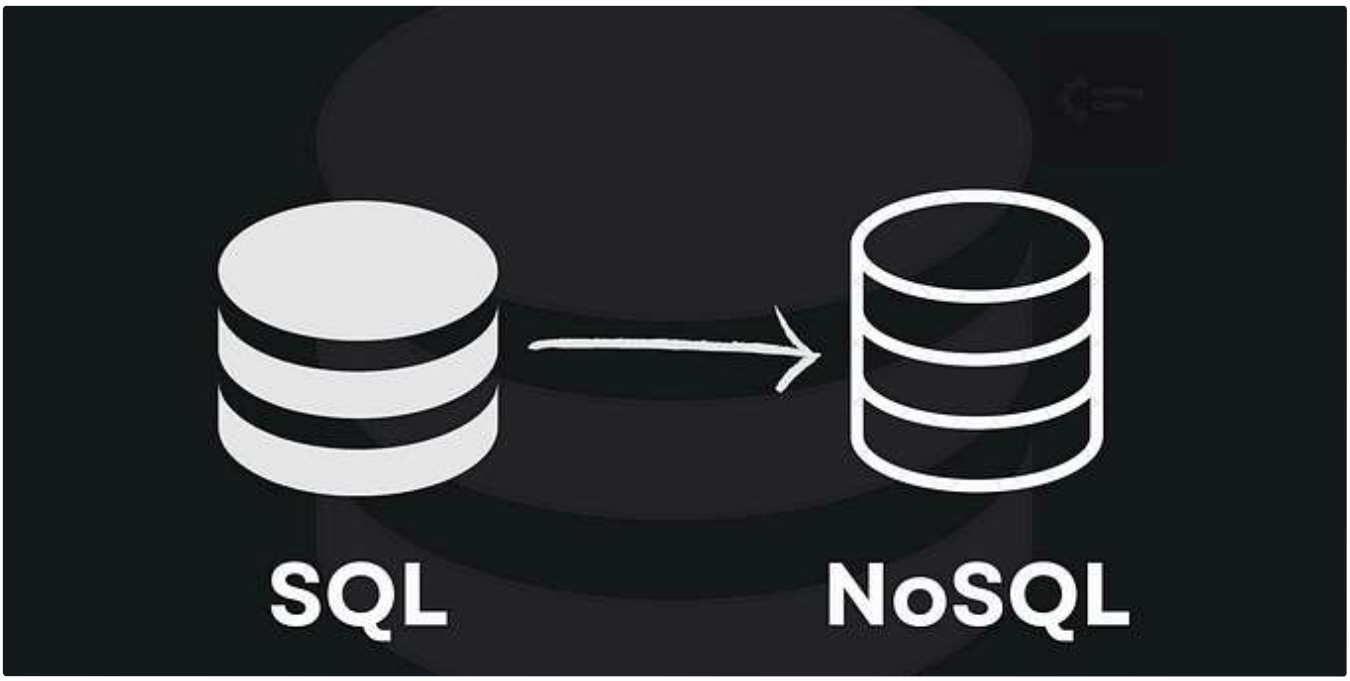


Mayur (Do not drink & database)

Postgres Is

Update: In response to a trademark notice from the PostgreSQL Community Association of Canada, domain has been changed from “Postgres.Is”...

Feb 17 11 1



In Stackademic by Crafting-Code

I Dropped SQL for NoSQL. Our App Now Handles 5x the Traffic

The ‘crazy’ database switch that proved our critics wrong

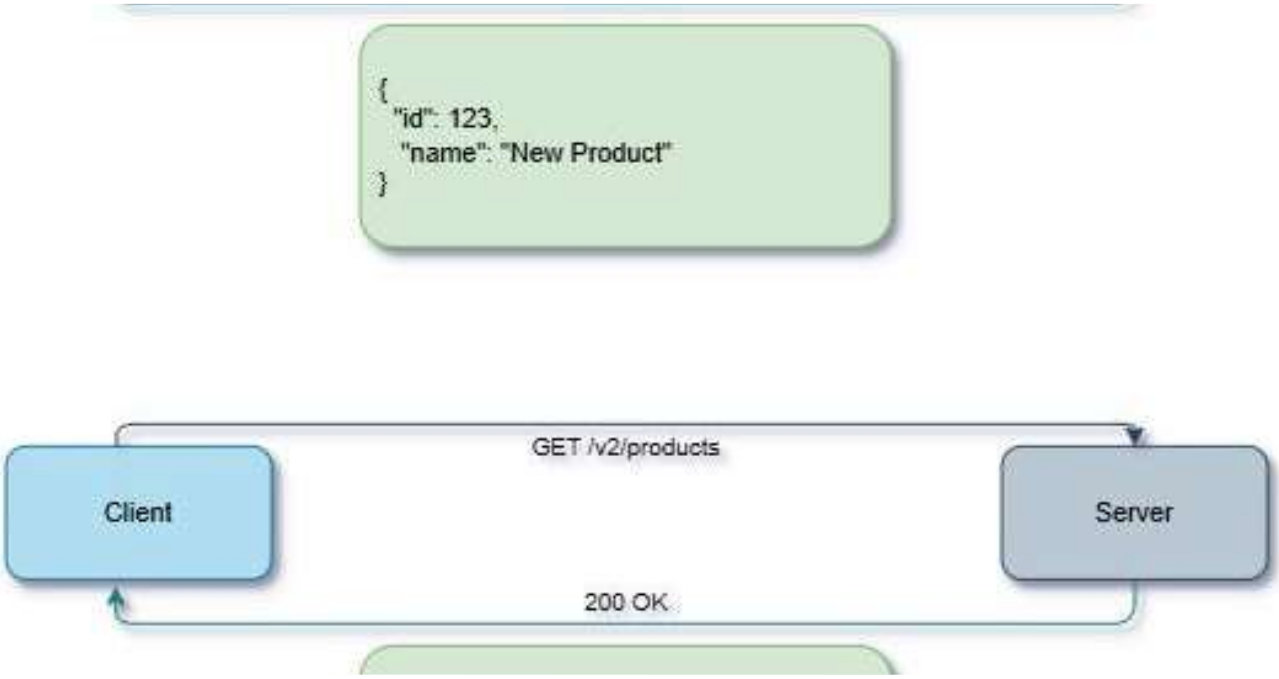
Feb 17 652 22

Shailesh Kumar Mishra

How a Simple Query Brought Down Performance: A PostgreSQL Partition Pruning Mystery

The Curious Case of the Missing Pruning

4d ago 4



In Javarevisited by Sivaram Rasathurai

Stop Messing Up Your API Versions!

If you’re using /v1/products and /v2/products, this article is for you

Feb 8 1.3K 25

See more recommendations