dbtune

# A benchmark study on the impact of PostgreSQL server parameter tuning

P2D2 conference, Prague

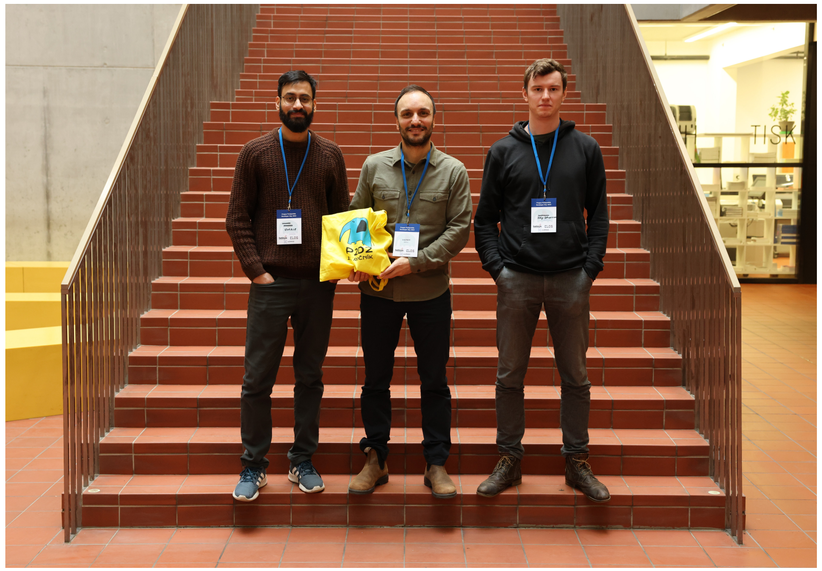
January 28, 2026

**Luigi Nardi**

Founder & CEO, DBtune

# PG Developer Day Prague training

January 31st, 2023




*On the left, a photo of our training session. On the top right three members of the DBtune team and on the bottom the main track.*

# About me

Mixed background in industry and academia

Among other things:
- Ph.D. CS at Sorbonne
- Research Staff at Stanford University
- Associate Professor in AI at Lund University

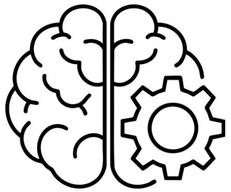
Since 2020, Founder & CEO at DBtune

@luinardi

dbtune

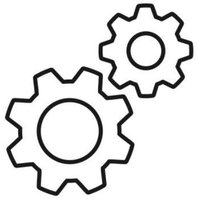
**What**

*DBtune is an AI-powered database tuning service*



**Where**

Spun out of research at Stanford University

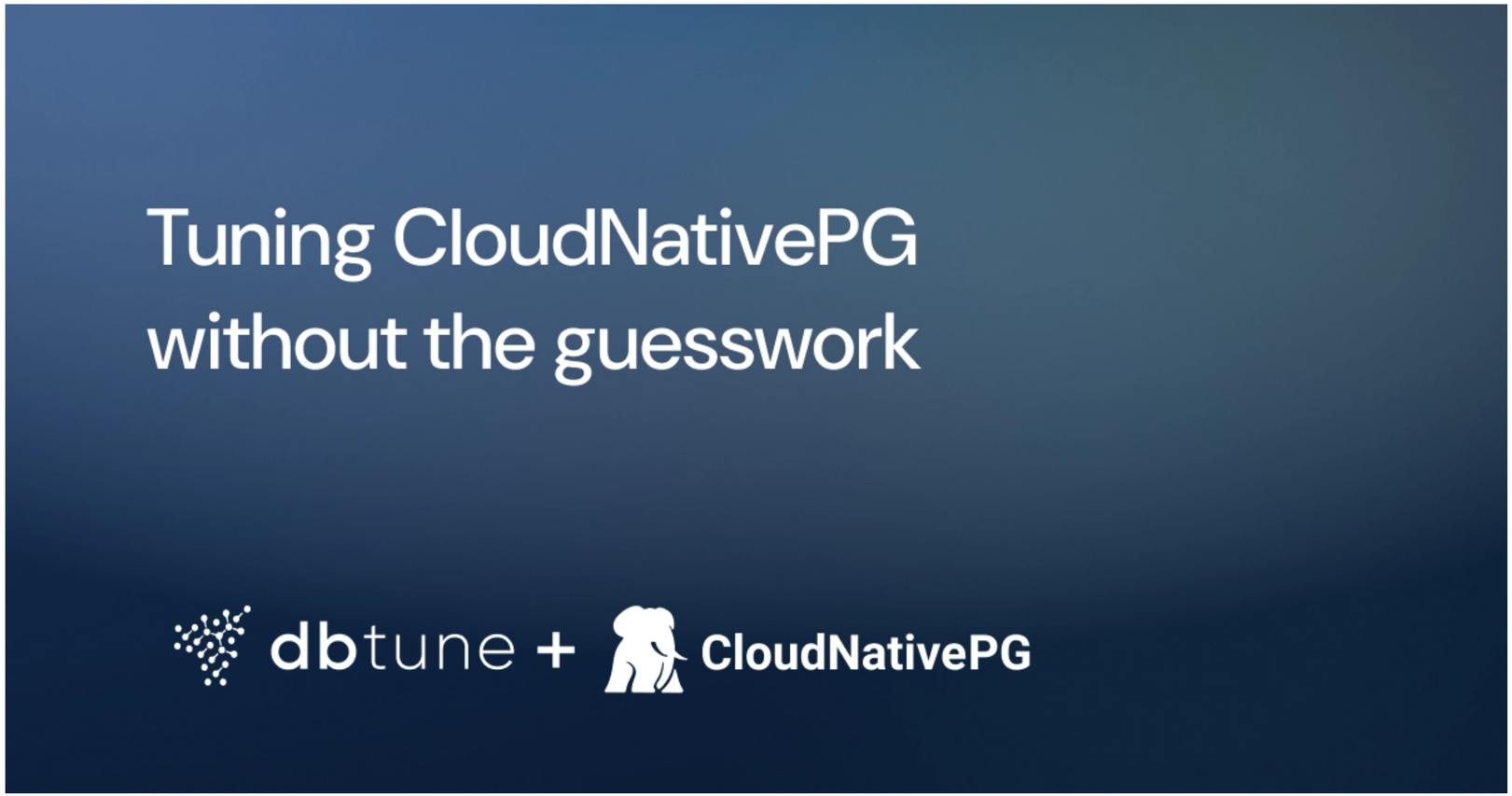


**How**

Tunes for a specific workload, use case and machine

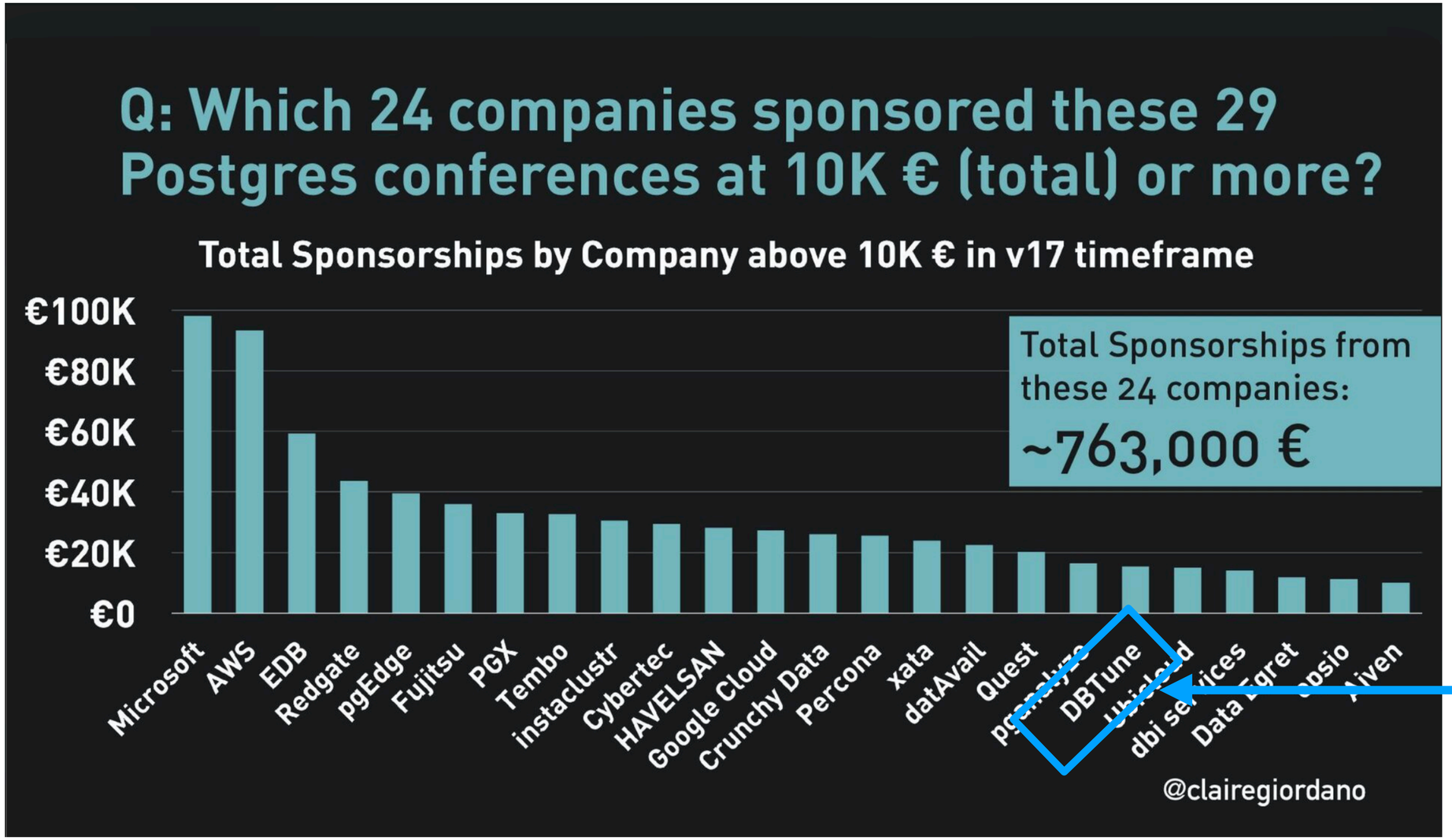
# DBtune now supports CloudNativePG

## Public release date: January 27, 2026 at P2D2




Read the blog post: https://www.dbtune.com/blog/tuning-cloudnativepg-without-the-guesswork

# DBtune in the top 20 PostgreSQL sponsors




https://speakerdeck.com/clairegiordano/whats-in-a-postgres-major-release-an-analysis-of-contributions-in-the-v17-timeframe-claire-giordano-pgconf-eu-2024

# Malmö PostgreSQL User Group (M-PUG)

## M-PUG organizers

**Ellyne Phneah**
DBtune

**Luigi Nardi**
DBtune

**Daniel Gustafsson**
Microsoft

**Dennis Rilorin**
Redpill Linpro

- The group is officially recognized by PostgreSQL Europe
- Regular meetups every 4-8 weeks in Malmö — Top speakers
- We are building a vibrant PostgreSQL community in the region

# What is this talk about?

- Is it worth tuning your server parameters? A benchmark study
- In this talk you won’t learn how to manually tune your PostgreSQL server

# Outline

- Introduction to PostgreSQL server parameter performance tuning
- Quantitative illustrative examples
- How do we solve this today?
- Results synthetic and prod workloads
- Conclusions

# What is PostgreSQL tuning?

# What is database tuning?

## Keeping the database fit and responsive

- Databases change, grow and slow down
- Not all workloads and machines are the same
- **Tuning adapts a database to its current use-case, load and machine**
- It is a ‘dark-art’ yet an integral part of any DBA and developer’s job
- Tuning includes query, **server parameters***, index, OS parameters, etc.

*We will focus solely on PostgreSQL server parameter tuning

# Why does it matter?

## Technical perspective

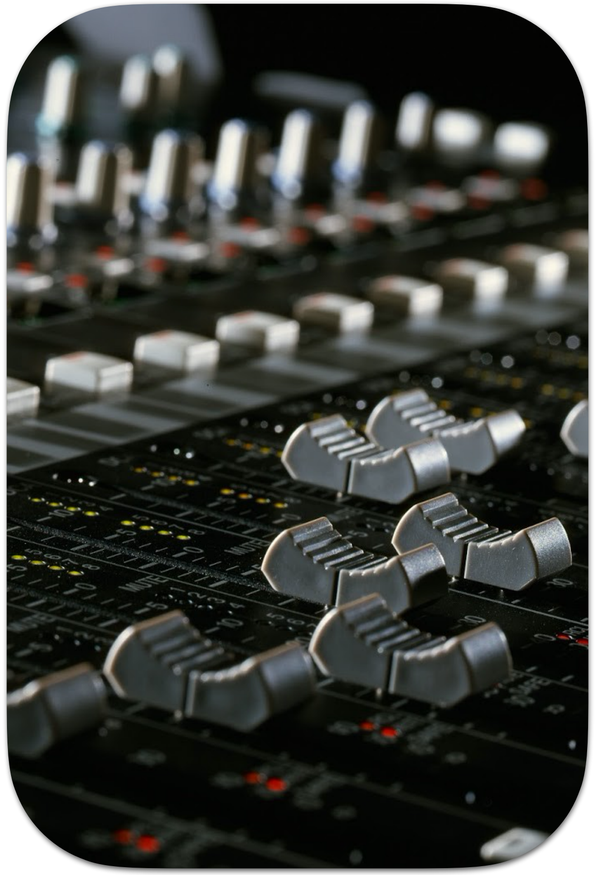
- Impacts system performance
  - Throughput and latency
- Improves scalability / stability / SLA

## Business perspective

- Decreases infrastructure spend
- Higher end-user satisfaction
- Reduces downtime
- Increases productivity
- Saves energy (ESG)

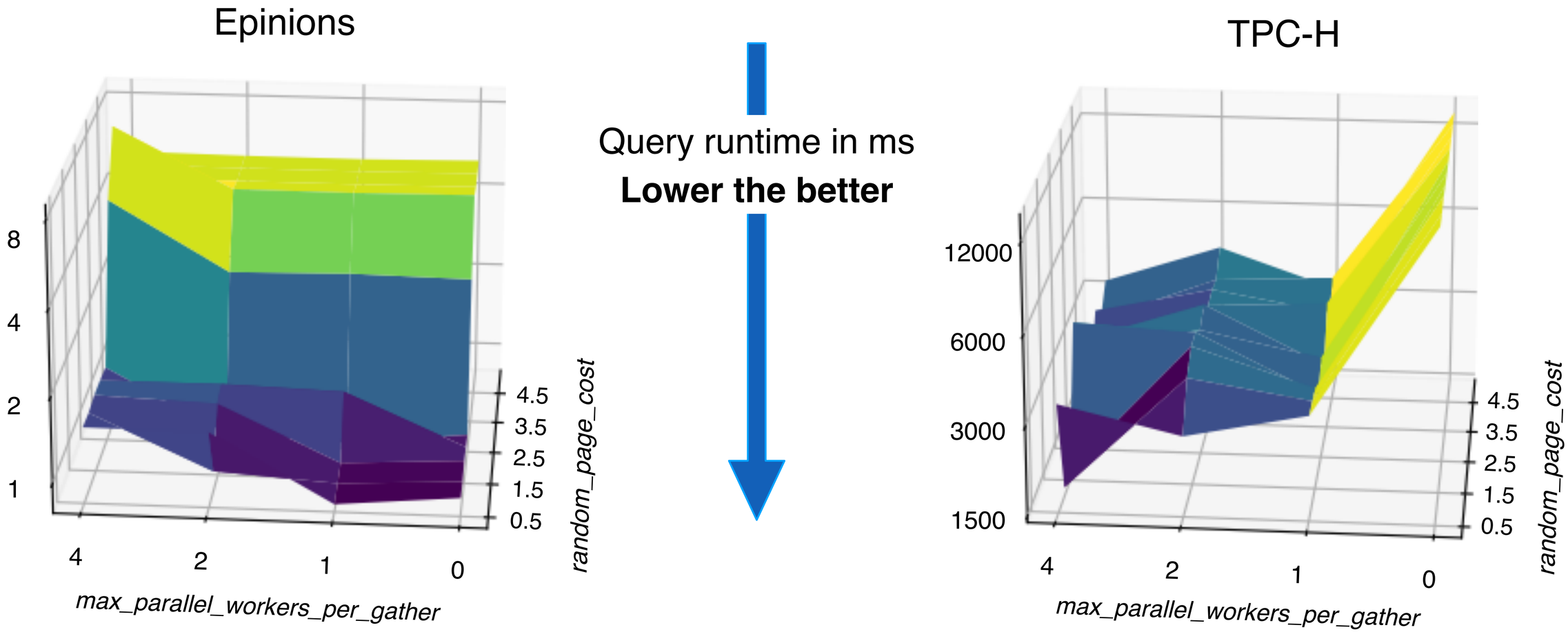
# PostgreSQL server parameter tuning

- Adjusting knobs to best fit the workload
- PostgreSQL parameters that are typically important:
  *work_mem*, *shared_buffers*, *max_wal_size*, etc.
- Example *max_parallel_workers_per_gather*:
  Max # of workers started by a Gather or Gather Merge node
- Example *random_page_cost*:
  Planner's cost of a non-sequentially fetched disk page
- These parameters highly depend on the application



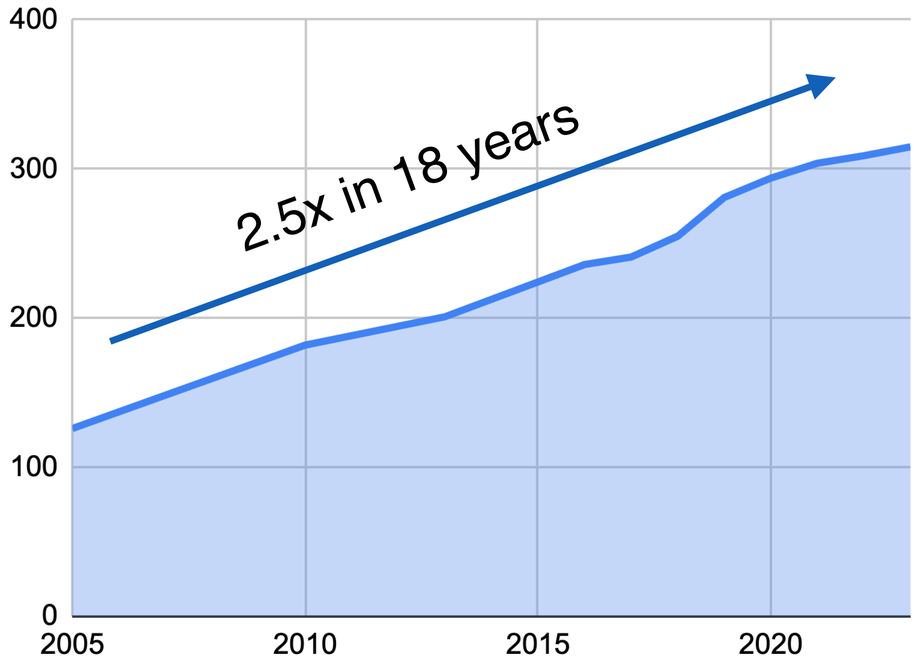
# Average query runtime tuning

## for *max_parallel_workers_per_gather* and *random_page_cost*



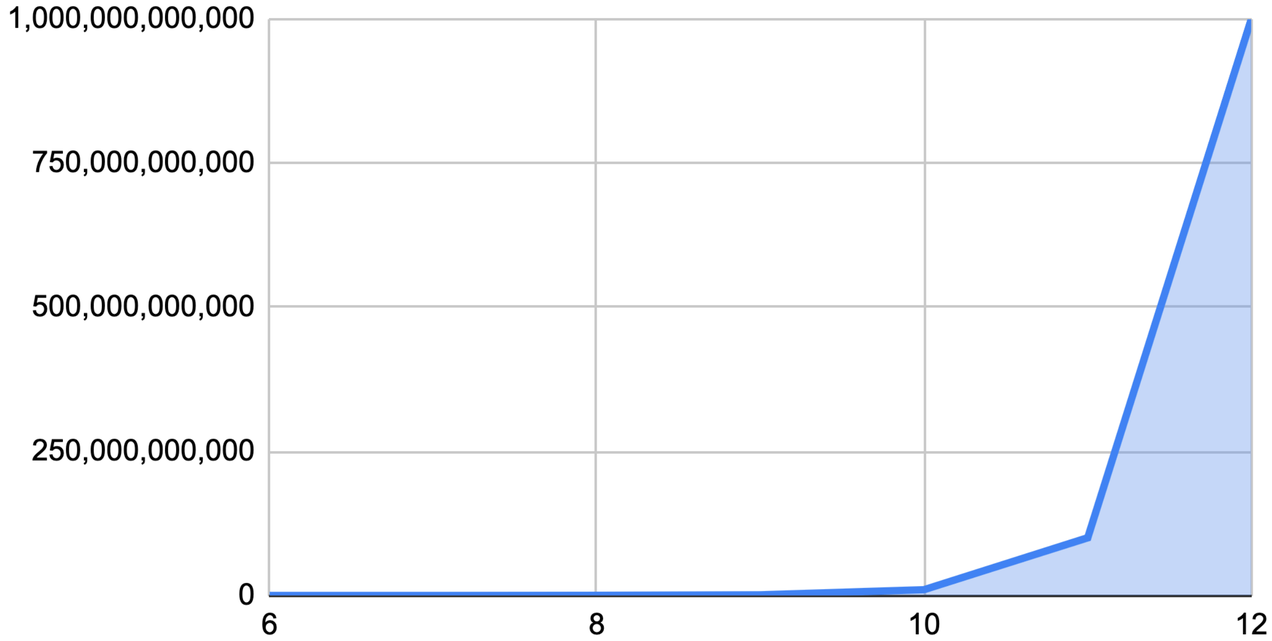
# Complexity is growing over time

The number of parameters
is growing **linearly**




PostgreSQL number of parameters

The number of configurations
is growing **exponentially**




Example of complexity with 12 parameters

# How is parameter tuning tackled today by DBAs and developers?

Tuning guru

## Manual

Slow
Takes days

Painstaking
Needs high expertise

Ineffective
Tune again in a week

Inadequate
Seasonal workload

## Heuristics

PGTune

POSTGRESQL CONFIGURATOR

One-size-fits-all
Uses generic rules

Workload agnostic
Not bespoke

Ineffective
Tune again in a week

Inadequate
Seasonal workload

## AI approach

dbtune

AI agent that learns by observation, adapts to changing workloads, and autotunes under minimal supervision

# Heuristic-based server parameter tuning

## Heuristics

PGTune

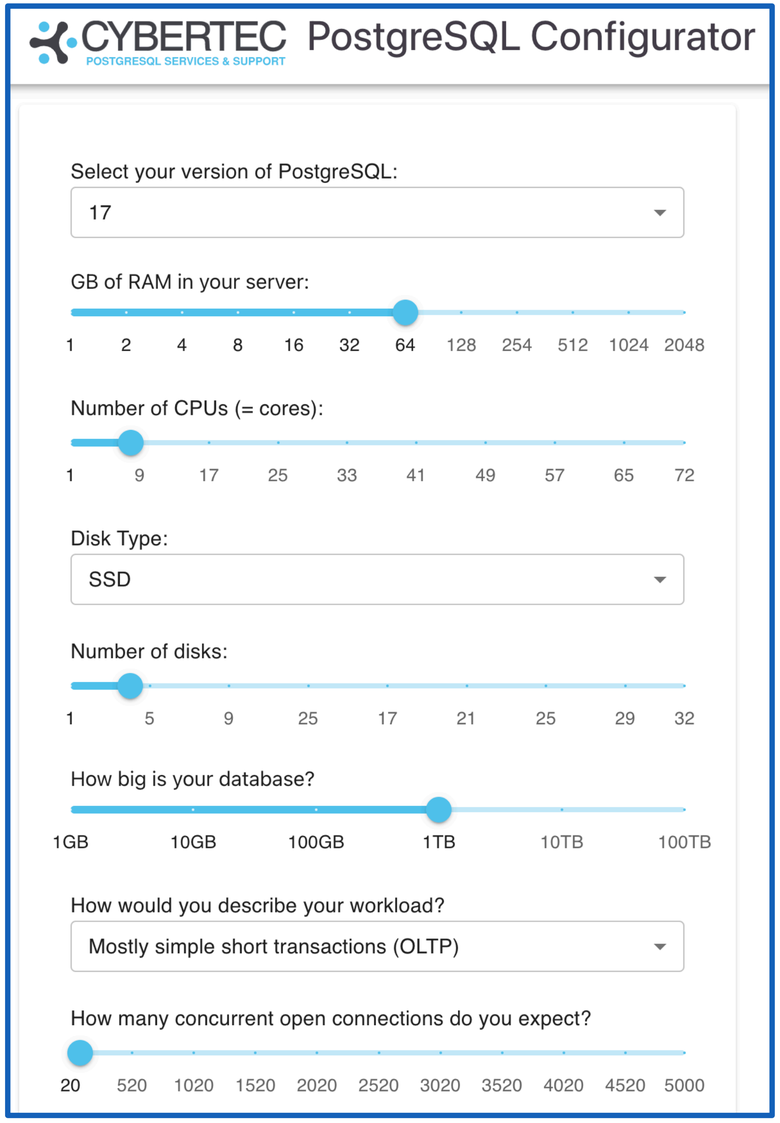
POSTGRESQL CONFIGURATOR

### One-size-fits-all
Uses generic rules

### Workload agnostic
Not bespoke

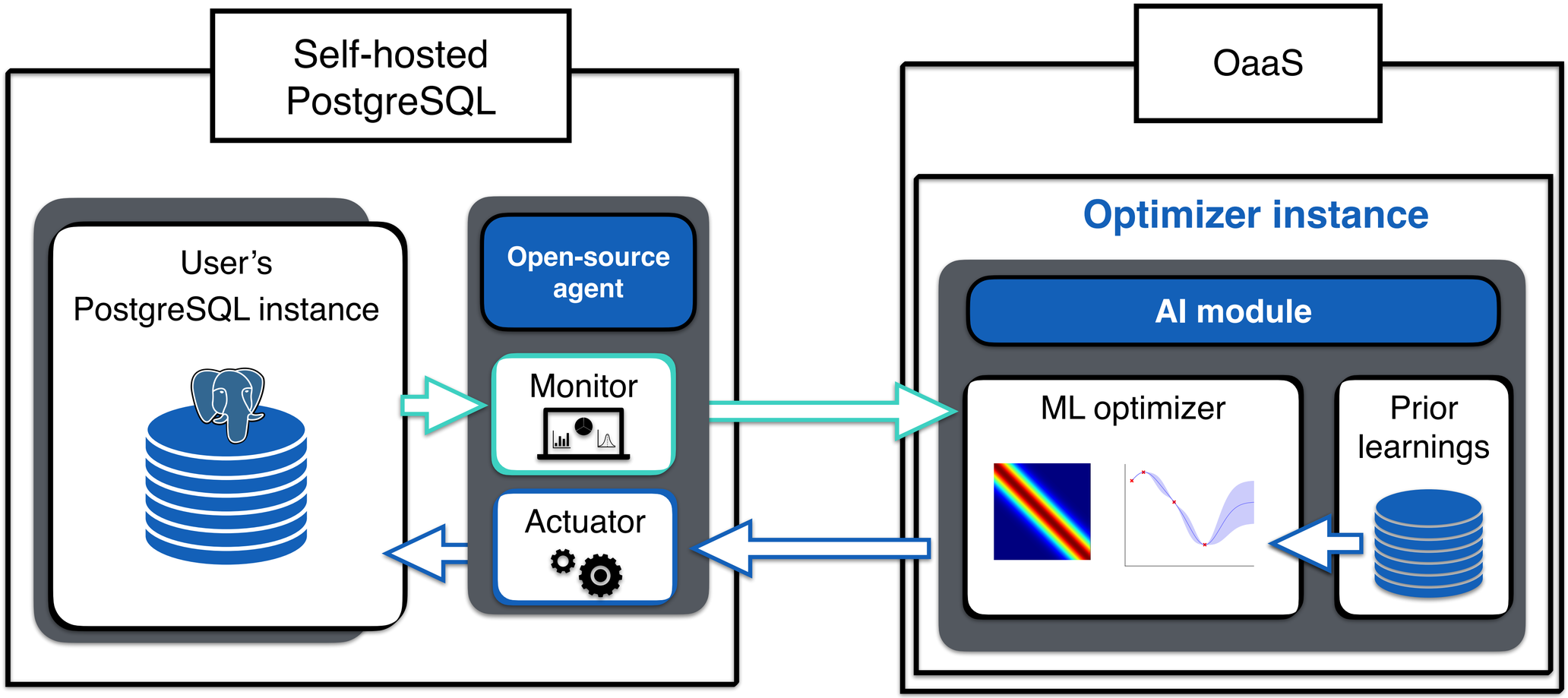
### Ineffective
Tune again in a week

### Inadequate
Seasonal workload

# In this presentation we will use the DBtune free-tier to automate tuning

## High-level architecture view for the PostgreSQL Optimizer-as-a-Service (OaaS)



# How often do you tune?

On-going requirements

- Scale your cloud instance up or down
- Change in application workload
- Change in data distribution and query planner

Specific moments

- Migrate from Oracle to PostgreSQL
- Migrate from on-prem to the cloud — Or vice-versa
- Migrate PostgreSQL version
- Experience downtimes

# The reality of how most enterprises treat manual parameter tuning today

- Tuning is typically **reactive** to something going wrong — Not **proactive**
- Often engage expensive external resources / experts
- Different workloads are not treated differently
- Modus operandi: Throw more hardware / compute at any issue ($$$)

# Example of PostgreSQL parameters tuned in this presentation

## Database reload (15 params)

- work_mem
- *max_parallel_workers*
- *max_parallel_workers_per_gather*
- *effective_io_concurrency*
- *bgwriter_lru_maxpages*
- *random_page_cost*
- *sequential_page_cost*
- *bgwriter_delay*
- *max_wal_size*
- *min_wal_size*
- *checkpoint_completion_taget*

## Require database restarts (3 params)

- shared_buffers
- *max_worker_processes*

There is an on-going shared_buffers patch to make it dynamically adjustable (see hackers' list)

- Full list can be found here: https://docs.dbtune.com/postgresql/
- Taxonomy of the PostgreSQL server parameters, blog by Marc Linster: https://dbtune.com/blog/dbtunes-multi-dimensional-performance-tuning-space

# HammerDB TPC-C benchmark suite

Open-source database load-testing and benchmarking tool

**TPC-C benchmark**

Wholesale supplier's order and delivery operations across multiple warehouses:

- New order: Creating new customer orders
- Payment: Recording customer payments
- Order status: Querying existing order status
- Delivery: Processing batch deliveries
- Stock level: Checking stock levels

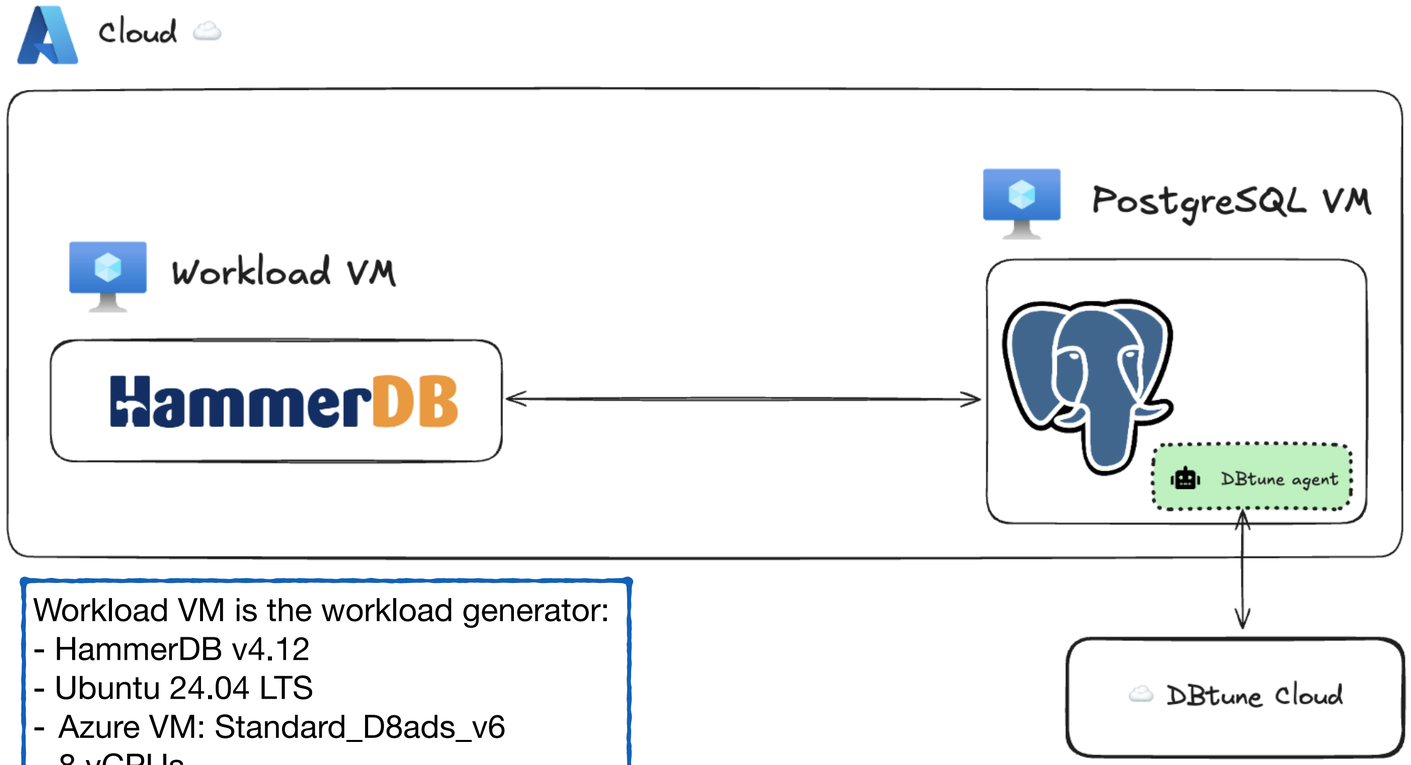
Many users execute transactions concurrently

TPC-C is an OLTP write-heavy workload

Blog: https://www.dbtune.com/blog/dbtune-and-hammerdb by Mohsin Ejaz (DBtune) and Steve Shaw (HammerDB)

# The distributed environment for running PostgreSQL benchmarks

Best practice — Two-node setup one for benchmark runner and one for PG:
Eliminates contention between the workload generator from the db server




PostgreSQL VM, hosting database and process transactions:
- PostgreSQL 17.4
- Ubuntu 24.04 LTS
- Azure VM: Standard_D8ads_v6
  8 vCPUs
  32 GB RAM
  440 GB ephemeral disk

Workload VM is the workload generator:
- HammerDB v4.12
- Ubuntu 24.04 LTS
- Azure VM: Standard_D8ads_v6
  8 vCPUs
  32 GB RAM
  440 GB ephemeral disk

Blog: "DBtune and HammerDB: Your guide to fair PostgreSQL benchmarking" by Mohsin Ejaz (DBtune) and Steve Shaw (HammerDB)

# HammerDB tips and tricks

Extending run duration: From 5 min to 1 day `diset('tpcc','pg_duration','1440')`

- Establishing baseline performance and warmup caches (next slide)
- Observe the system's behavior under sustained load
- Allow sufficient time to complete a tuning session

Setting number of warehouses:

- HammerDB recommends 250-500 per CPU socket: 52 GB of data volume

Setting virtual users:

- Set virtual users to 285 `vuset('vu',285)`
- On 8 vCPUs higher vu creates bottlenecks and a concurrent workload

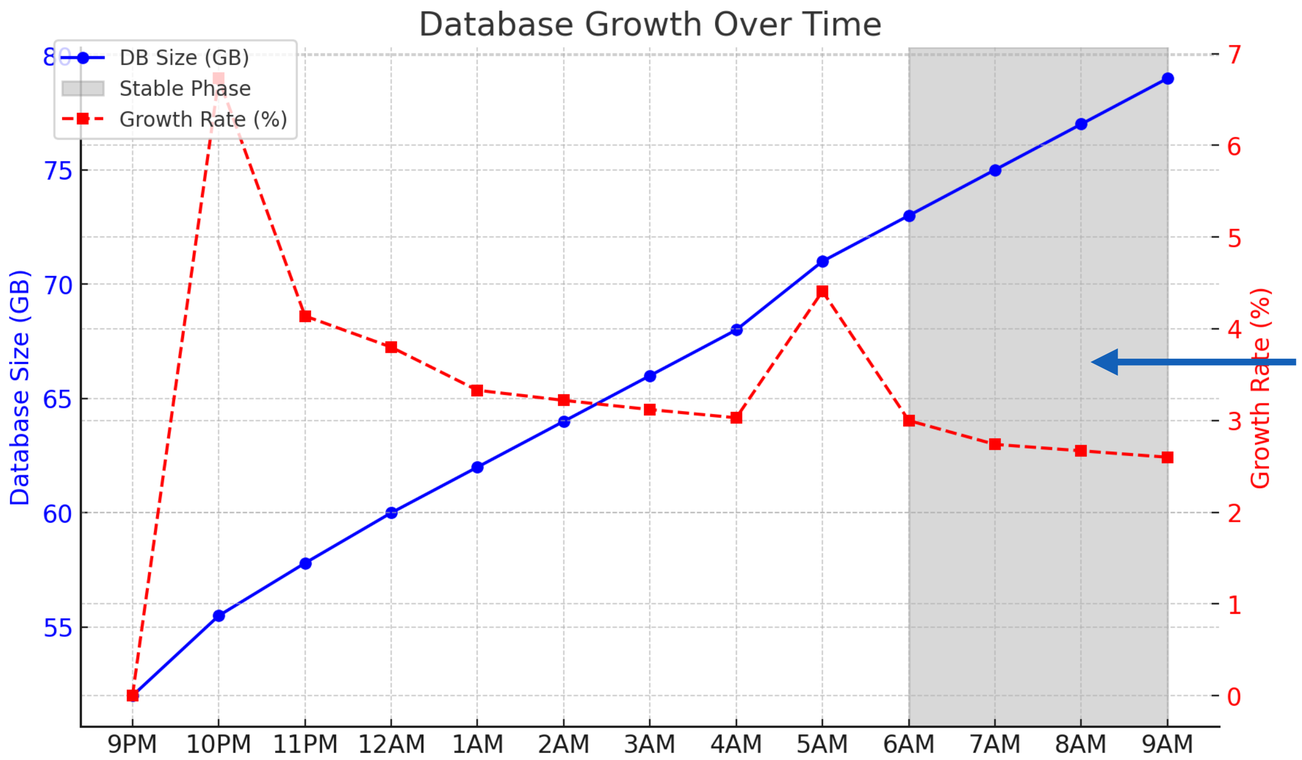
Setting number of connections:

- max_connections from 100 to 300

Blog: https://www.dbtune.com/blog/dbtune-and-hammerdb by Mohsin Ejaz (DBtune) and Steve Shaw (HammerDB)

# Establishing a realistic HammerDB TPCC baseline performance

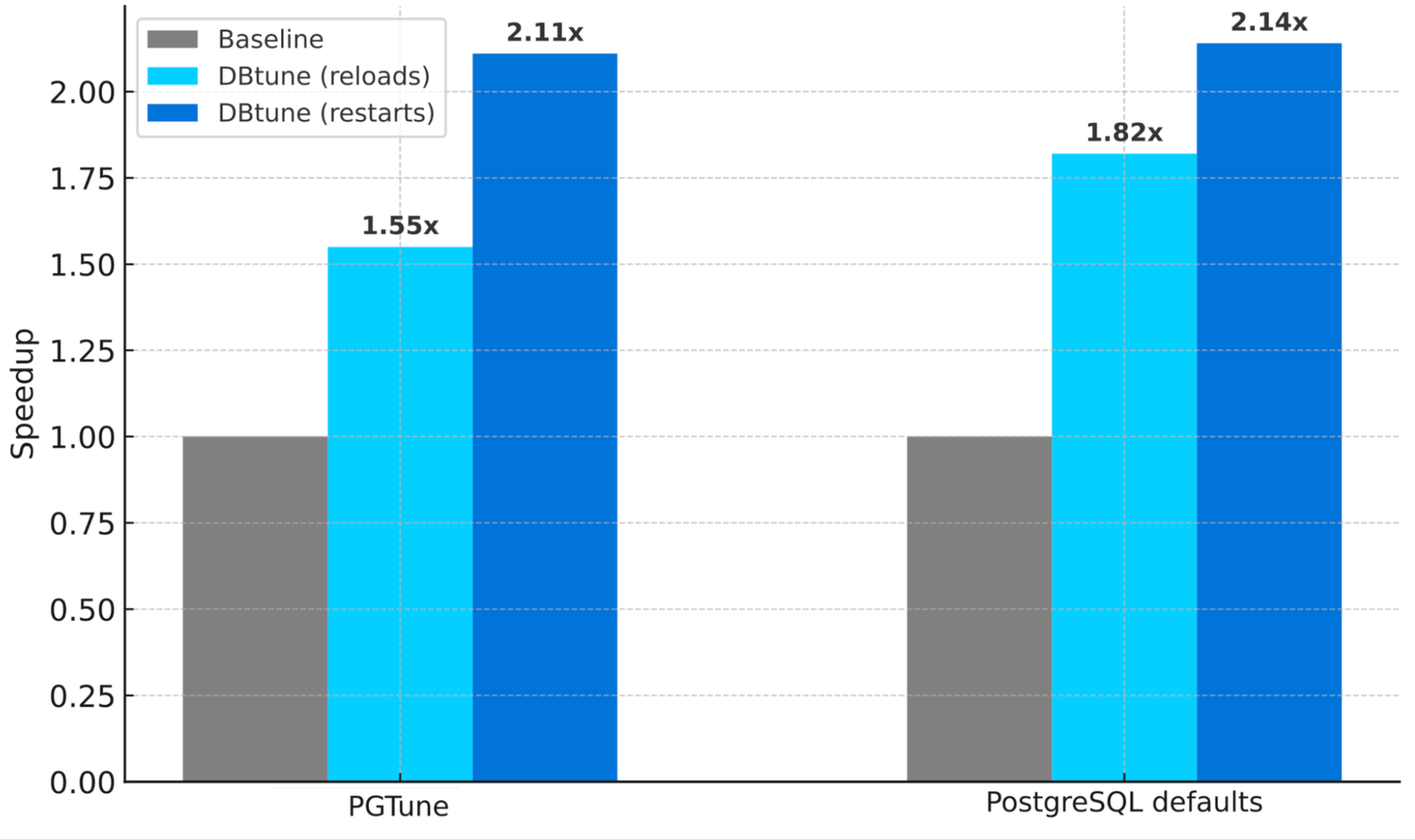
HammerDB performance metrics fluctuate significantly during the initial hours of operation as the system builds up buffer caches, statistics, and query plans




**The grey area is closer to a steady state and mimics the real-world more closely**

Blog: https://www.dbtune.com/blog/dbtune-and-hammerdb by Mohsin Ejaz (DBtune) and Steve Shaw (HammerDB)

# HammerDB TPCC tuning speedup results on TPS*




*The AQR of PG default tuning led to a 4x speedup in the restart scenario

Blog: https://www.dbtune.com/blog/dbtune-and-hammerdb by Mohsin Ejaz (DBtune) and Steve Shaw (HammerDB)

# BenchBase benchmarks

Open-source database benchmark suite https://github.com/cmu-db/benchbase

**ResourceStresser benchmark**

Synthetic benchmark that creates isolated contention on system resources:

- The transactions impose some load on 3 specific resources: CPU, I/O, locks

**TPC-H benchmark**

Decision support system benchmark:

- Business oriented ad-hoc queries and concurrent data modifications
- Examine large volumes of data, execute high-complexity queries, and give answers to critical business questions

**Epinions.com benchmark**

Consumer review website, content management and social media apps:

- Models cross-user interactions like writing product reviews
- 9 transactions: 4 user records, 4 item records, and 1 affecting all tables
- The workload is mixed but primarily read-heavy

**SEATS benchmark**

Simulates an airline ticketing system:

- Complex flight searches, reservations, and customer management
- Many concurrent users with a mixed workload and a 60/40 read/write ratio

# BenchBase tips and tricks

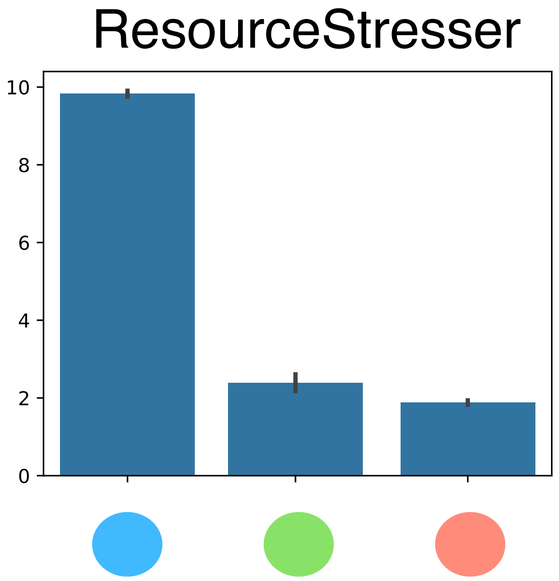
BenchBase benchmarks are configurable:

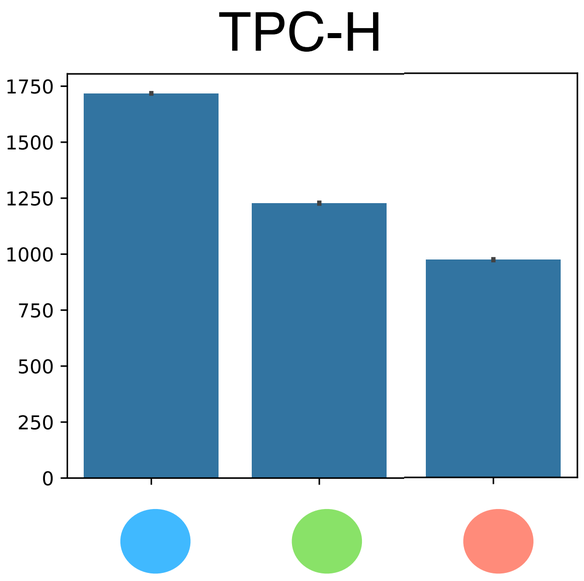
- **scalefactor**: Sets the size of the database
- **terminals**: Sets the number of connections querying the database
- **time**: Duration time of the benchmark in seconds
- **rate**: Transaction injection rate, determines how many TPS attempts to execute — Higher value makes for more intense workloads, commonly set to *unlimited*
- **weights**: Relative frequency of the different query types
- Etc.

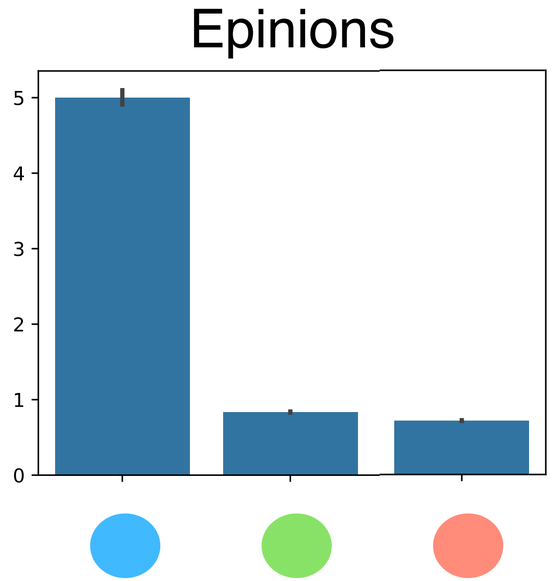
Example on Epinions: scalefactor=10000 (creates an 80 GB database), terminals to 10, time to 86400 (one day), rate=unlimited, and keep the weights as is

# Performance downside of non-restart (reload-only) strategy

## Average query runtime (in ms) on a community PG instance running on EC2




Query runtime in ms

**Lower the better**

Default PostgreSQL configuration & no tuning

*shared_buffers*=25% & reload-only & tuning

Restarts allowed & tuning

# Insurance application use case study — Customer anonymized data

Environment: 16 vCPU, 32 GB RAM, on-prem, primary instance, PG 15

Manually tuned baseline by expert DBA

Automated tuning with DBtune

**Tuning details**

| | |
|---|---|
| Tuning started | 19/06/2025 09:41 |
| Tuning ended | 19/06/2025 15:06 |
| Tuning duration | 5 hours 25 minutes |
| Tuning target | Average query runtime |
| Config application | Restart |

**Performance summary**

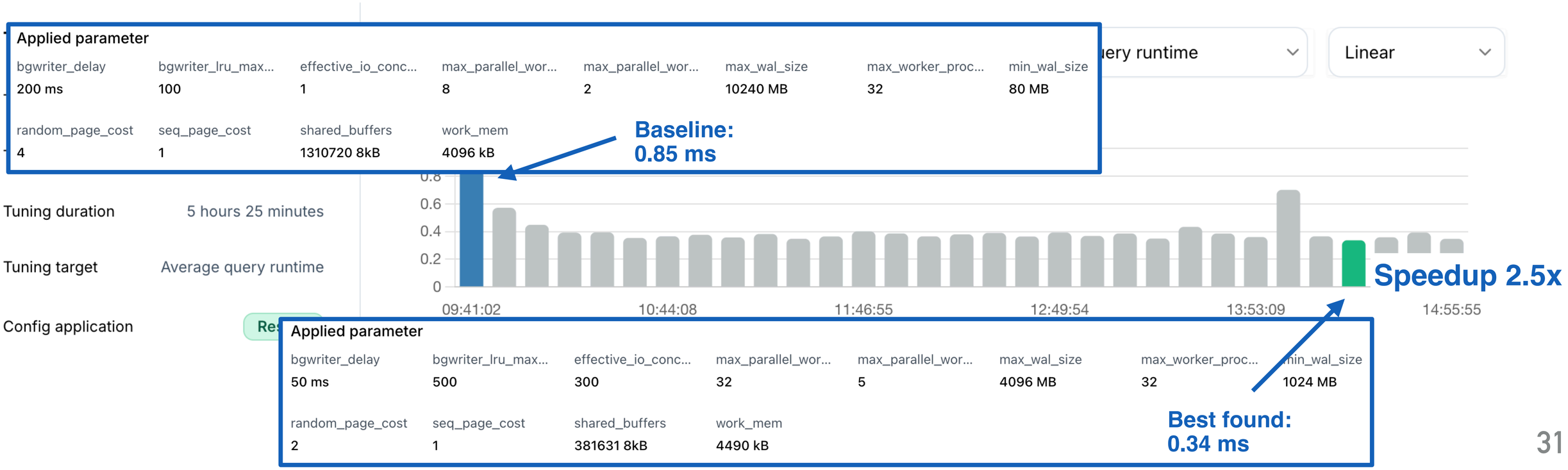
Lower number is better

# Insurance application use case study — Customer anonymized data

Environment: 16 vCPU, 32 GB RAM, on-prem, primary instance, PG 15

Manually tuned baseline by expert DBA

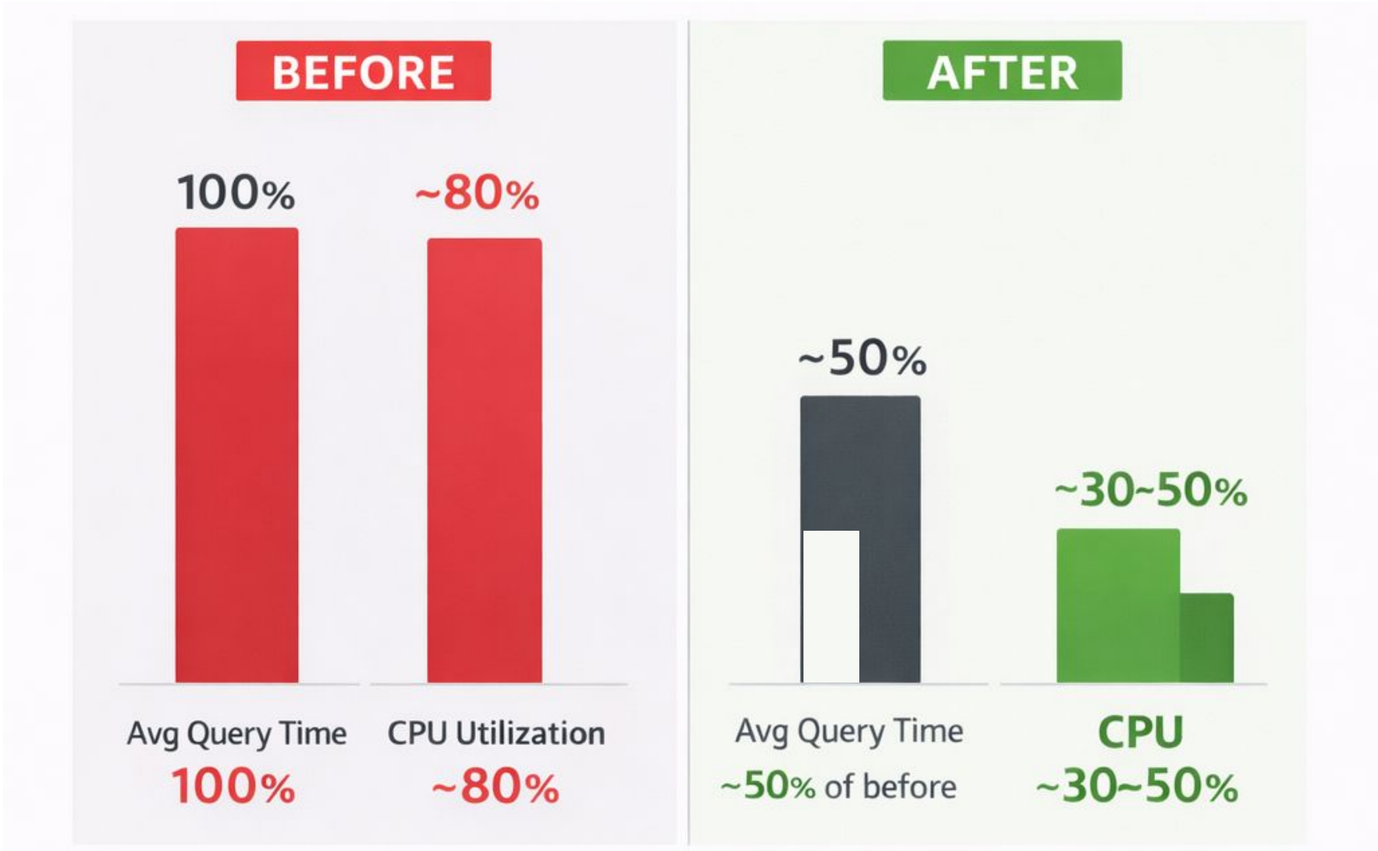
Automated tuning with DBtune



# In production: Workforce management platform by Papershift

Environment: Amazon RDS m5.8xlarge, 32 vCPU, 128 GB RAM, PG 17.6

Baseline by RDS, automated tuning with DBtune

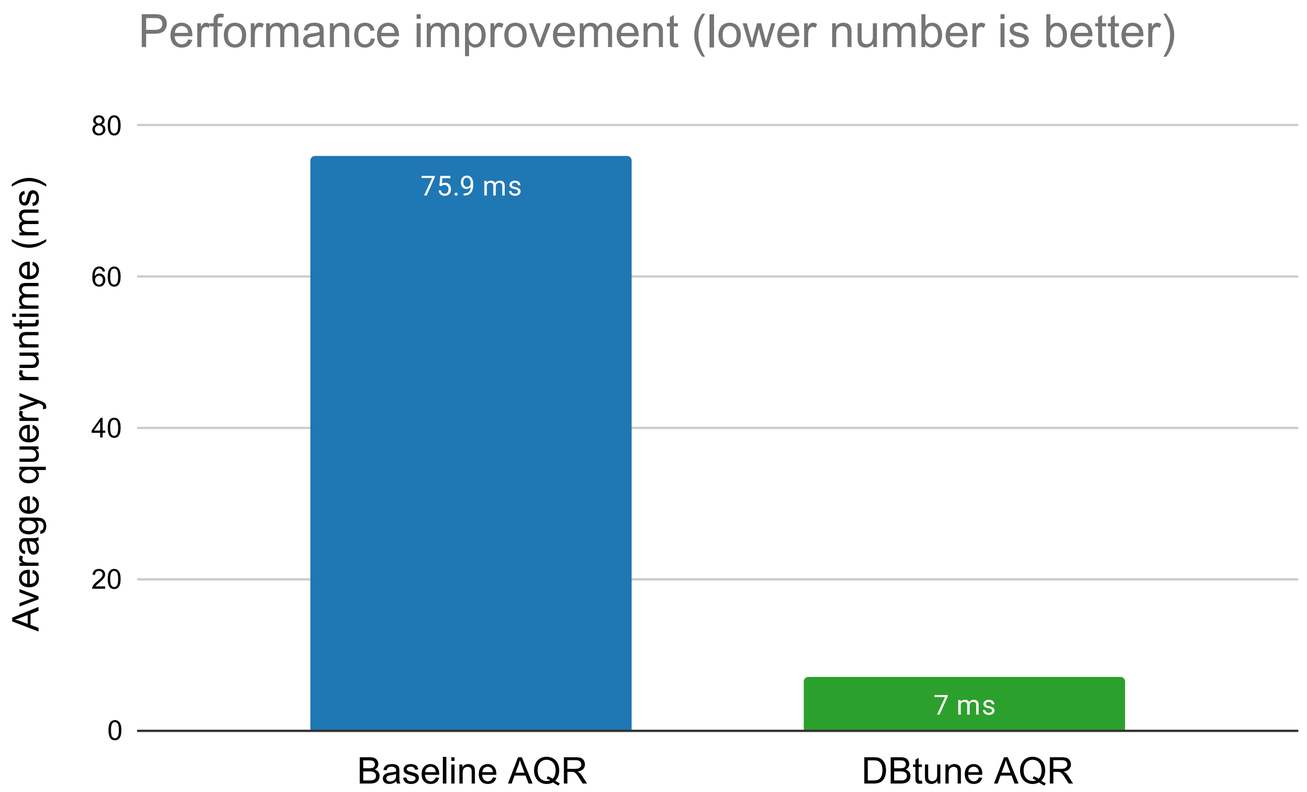



Blog: https://dev.to/floriansuchan/how-we-used-dbtune-to-cut-our-postgres-query-time-by-50-on-aws-rds-2a5e
Figure: https://www.linkedin.com/posts/vibhork_postgresql-autonomouspostgres-autonomousdatabase-activity-7408886284879835137-2i_m

# In production: Digital content service by Midwest Tape

Environment: Amazon RDS r6g.12xlarge, 48 vCPU, 412 GB RAM, PG 14.17

Baseline by RDS, automated tuning with DBtune




Blog: https://www.dbtune.com/blog/how-midwest-tape-achieved-a-10x-performance-boost-with-postgresql-tuning-on-aws-rds

# Conclusions and food for thought

- Realistic benchmarking:
  - Running a benchmark exposes many variables
  - These depend on the SKU and on the benchmark application itself
  - Benchmarks without configuring them doesn’t lead to realistic scenarios
- Tuning PostgreSQL is essential for fair database benchmarking
- Ultimately the speedup from server parameter tuning depends on:
  - How good is your baseline
  - How up to date is your baseline — things change and become untuned
- Point of view shifts from tuning as a specific task to a maintenance activity

# Questions and additional resources

- Blog: DBtune and HammerDB: Your guide to fair PostgreSQL benchmarking
- Useful links: DBtune synthetic workload tutorial GitHub and video

@luinardi

luigi@dbtune.com

dbtune

DBtune newsletter