dbtune

A benchmark study on the impact of PostgreSQL server parameter tuning

Swiss PGDay

June 27, 2025

Luigi Nardi, Ph.D.

Founder & CEO, DBtune

About me

B.Sc and M.Sc. Computer Engineering at La Sapienza — Rome (Italy)

- 2006 M.Sc. thesis at LAAS-CNRS — Toulouse (France)
- 2007 Ph.D. Computer Science at Université Pierre et Marie Curie — Paris (France)
- 2011 Software Engineer at Murex SAS — Paris (France)
- 2014 Postdoc Imperial College London (UK)
- 2017 Research Staff at Stanford University (USA)
- 2019 Assistant Professor in AI/ML at Lund University (Sweden)
- 2021 Founder & CEO at DBtune — Malmö (Sweden)
- 2024 Associate Professor in AI/ML at Lund University (Sweden)

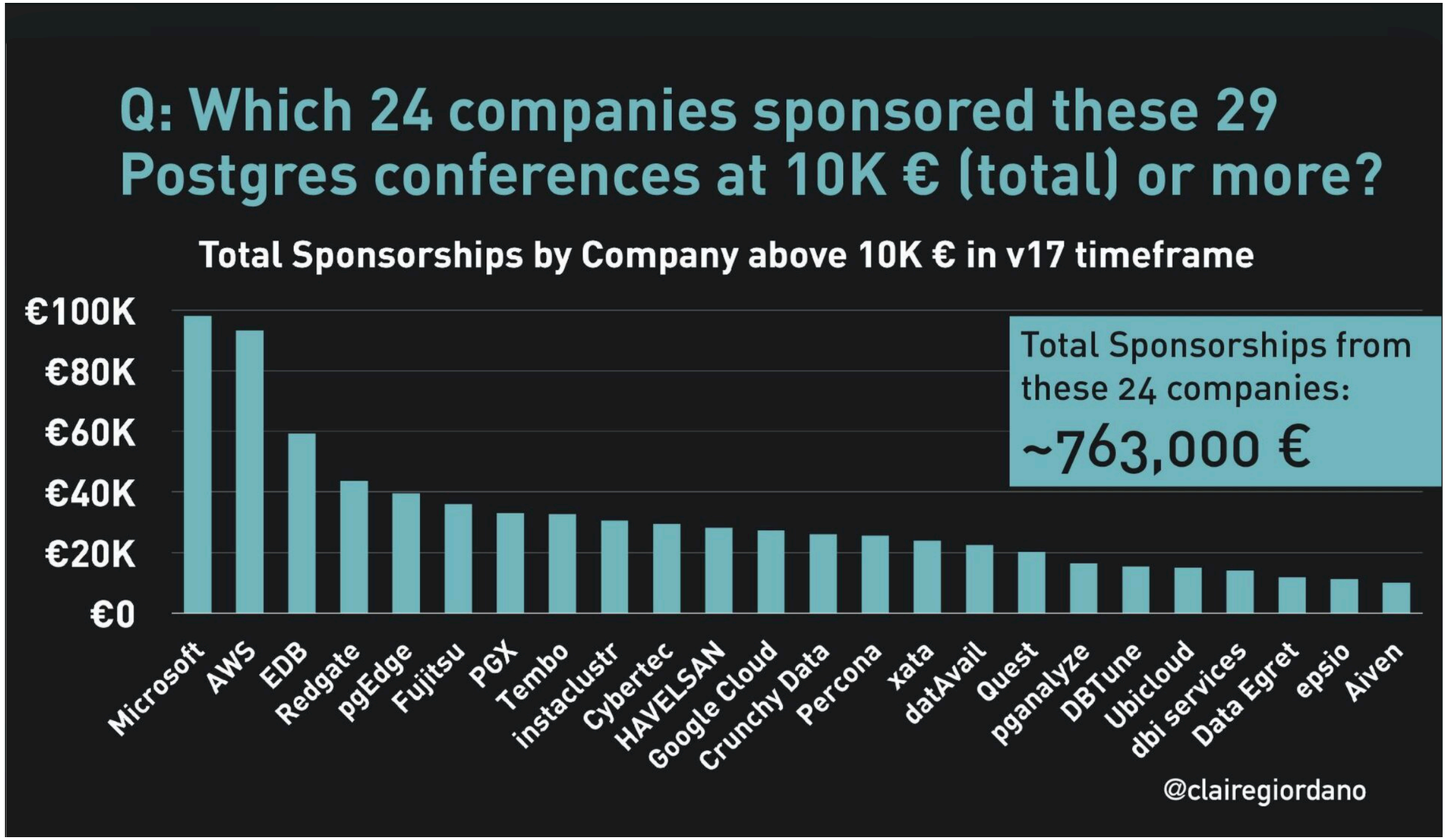
About DBtune

*DBtune is an **AI-powered** PostgreSQL server parameter **tuning** service.*

*Spun out of research at Stanford University, DBtune autonomously optimizes the configuration of databases through **machine learning**.*

*It observes, iterates and adapts until converging and delivering the **optimal** server configuration for any individual workload, use case and machine.*

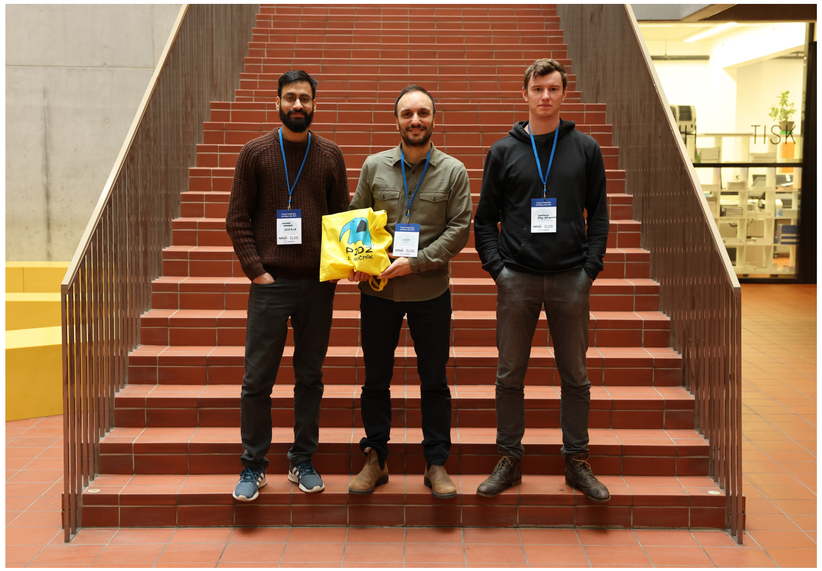
DBtune in the top 20 PostgreSQL sponsors



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

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, Umair, Luigi and Filip, who delivered the training, and bottom the full event.

Malmö PostgreSQL User Group (M-PUG)

M-PUG organizers

Ellyne Phneah
DBtune

Dr. 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 on community PG and Aiven DBaaS
- Conclusions

What is database 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

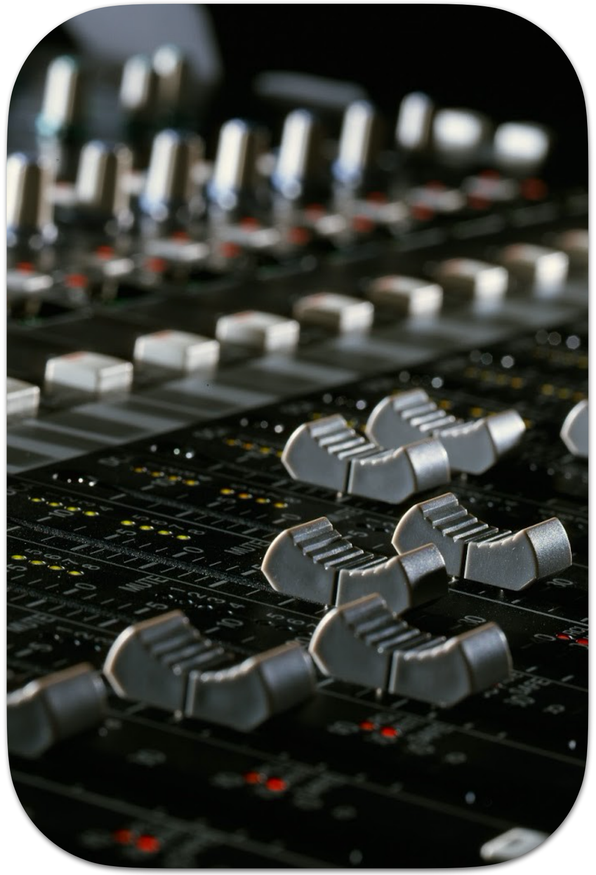
- Impacts system performance
  - Throughput and latency
- Improves scalability / stability / SLA

Business perspective

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

PostgreSQL server parameter tuning

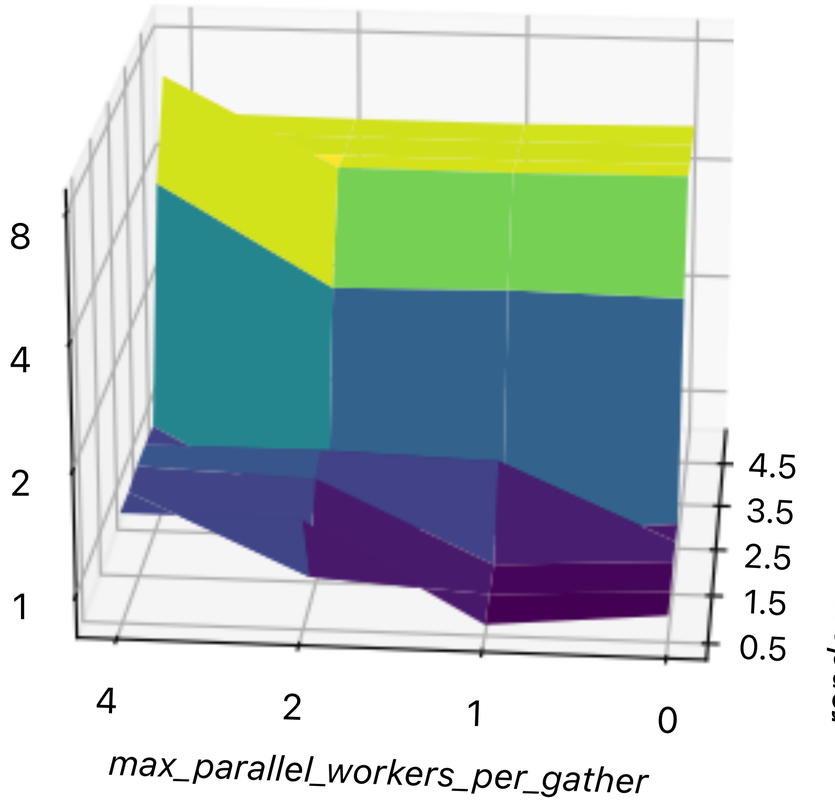
- Adjusting knobs to best fit the workload
- PostgreSQL parameters that are typically important:
  work_mem, *shared_buffers*, *max_wal_size*, etc.
- Example *max_parallel_workers_per_gather*:
  Max # of workers started by a Gather or Gather Merge node
- Example *random_page_cost*:
  Planner's cost of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

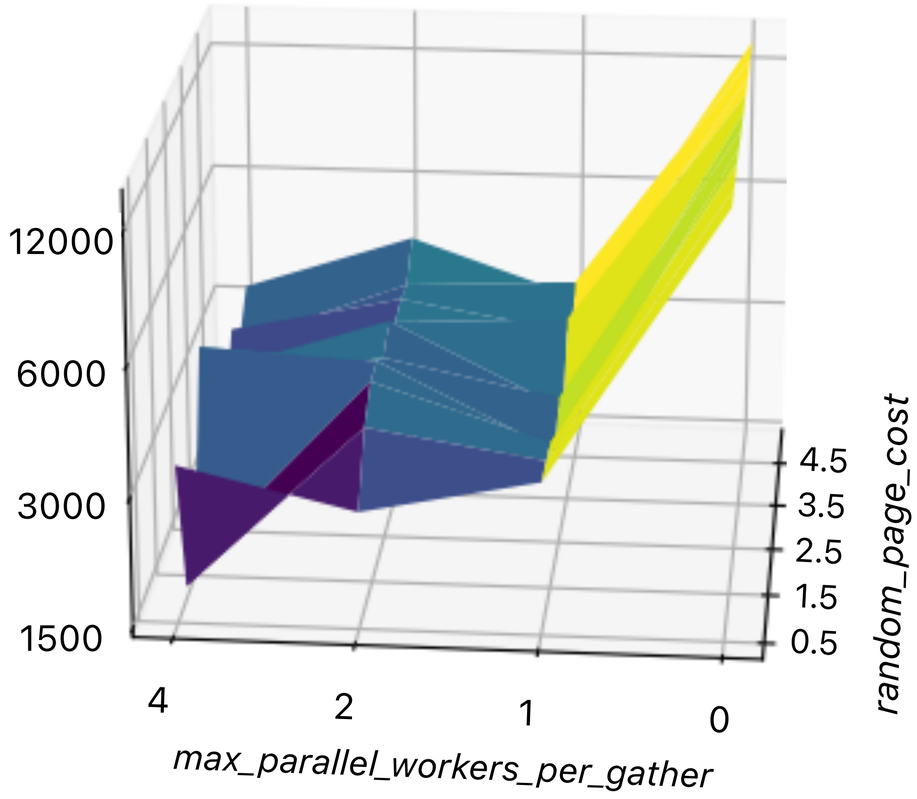
Epinions



Query runtime in ms

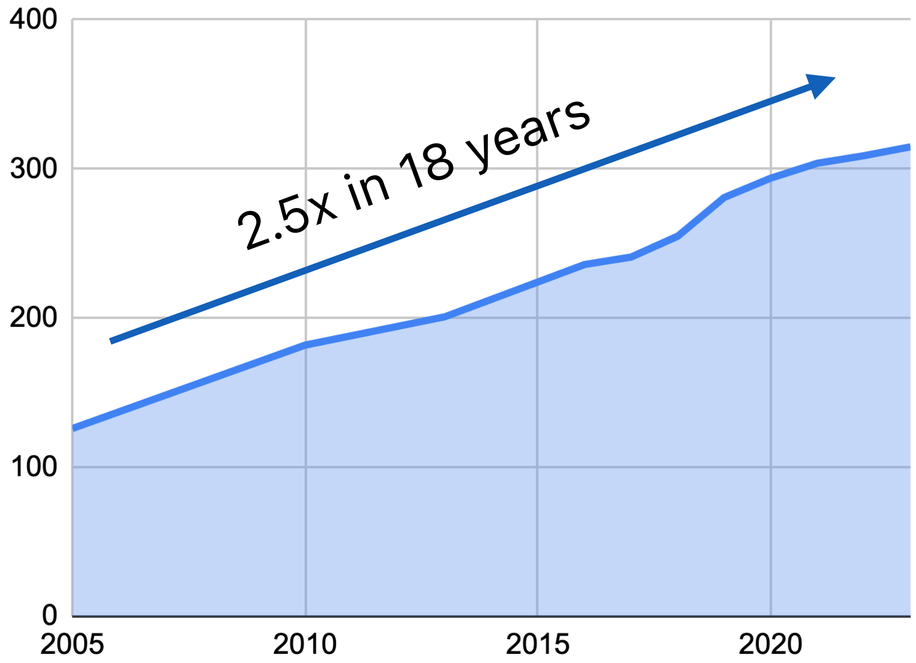
Lower the better

TPC-H



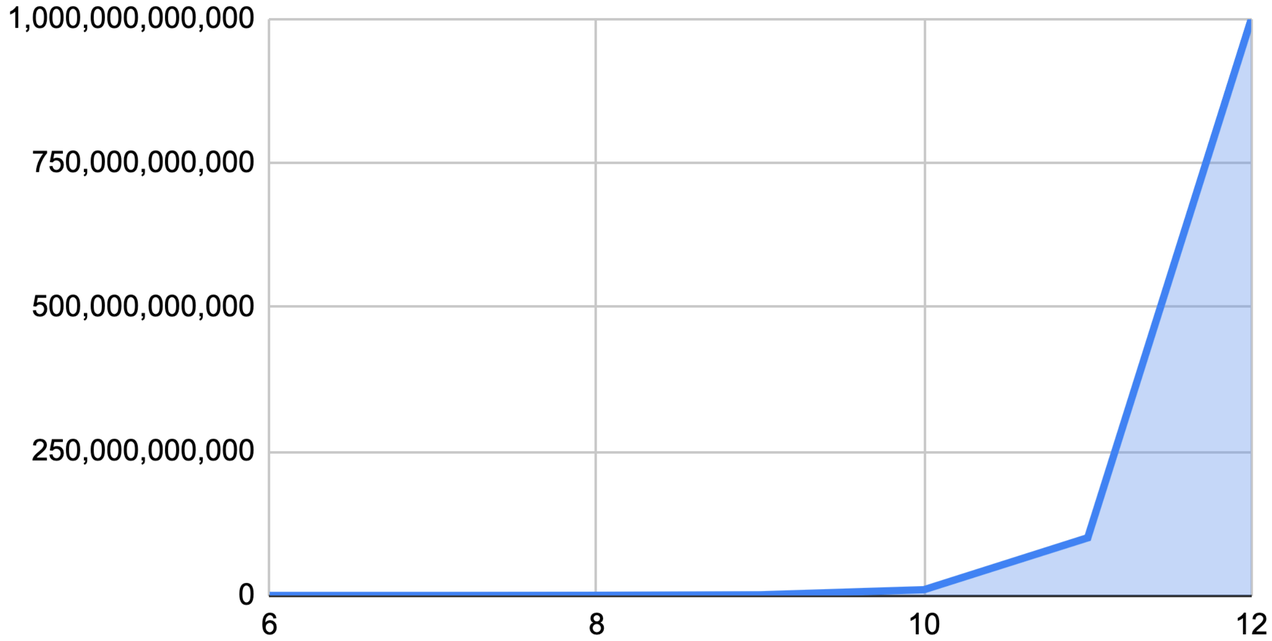
Complexity is growing over time

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

Learn by observation and autotunes

A solution that adapts to changing workloads

Heuristic-based server parameter tuning

Heuristics

PGTune

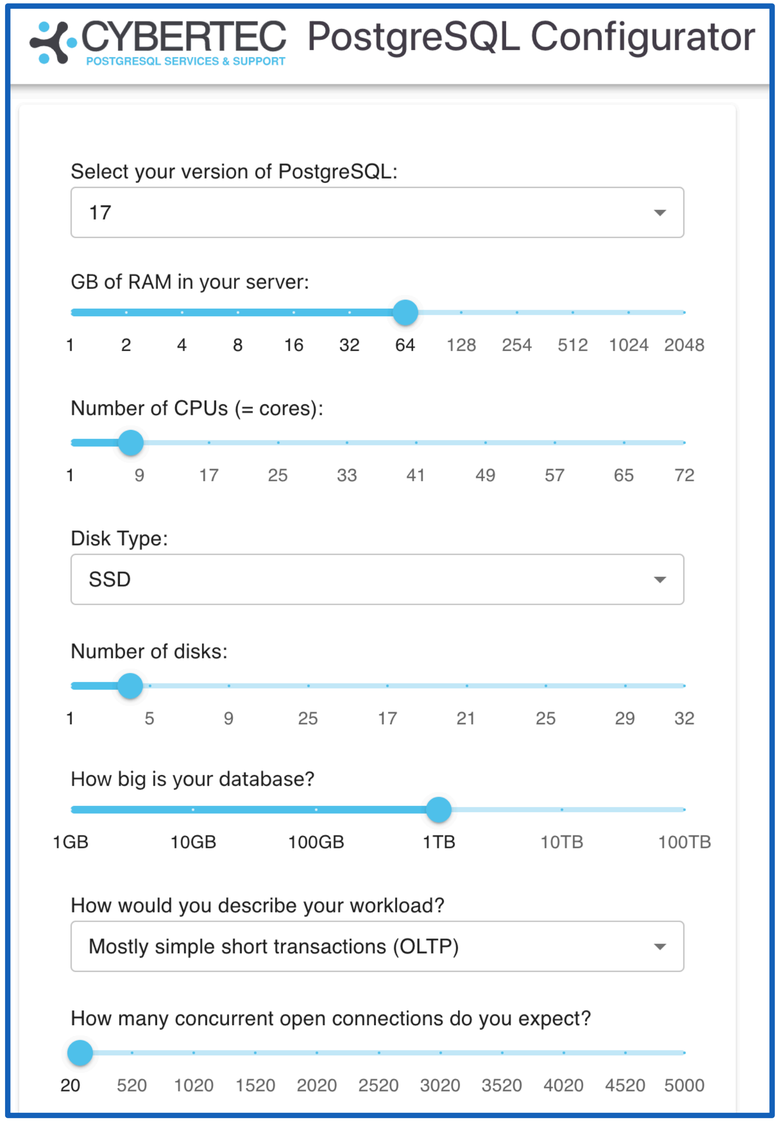
POSTGRESQL CONFIGURATOR

One-size-fits-all
Uses generic rules

Workload agnostic
Not bespoke

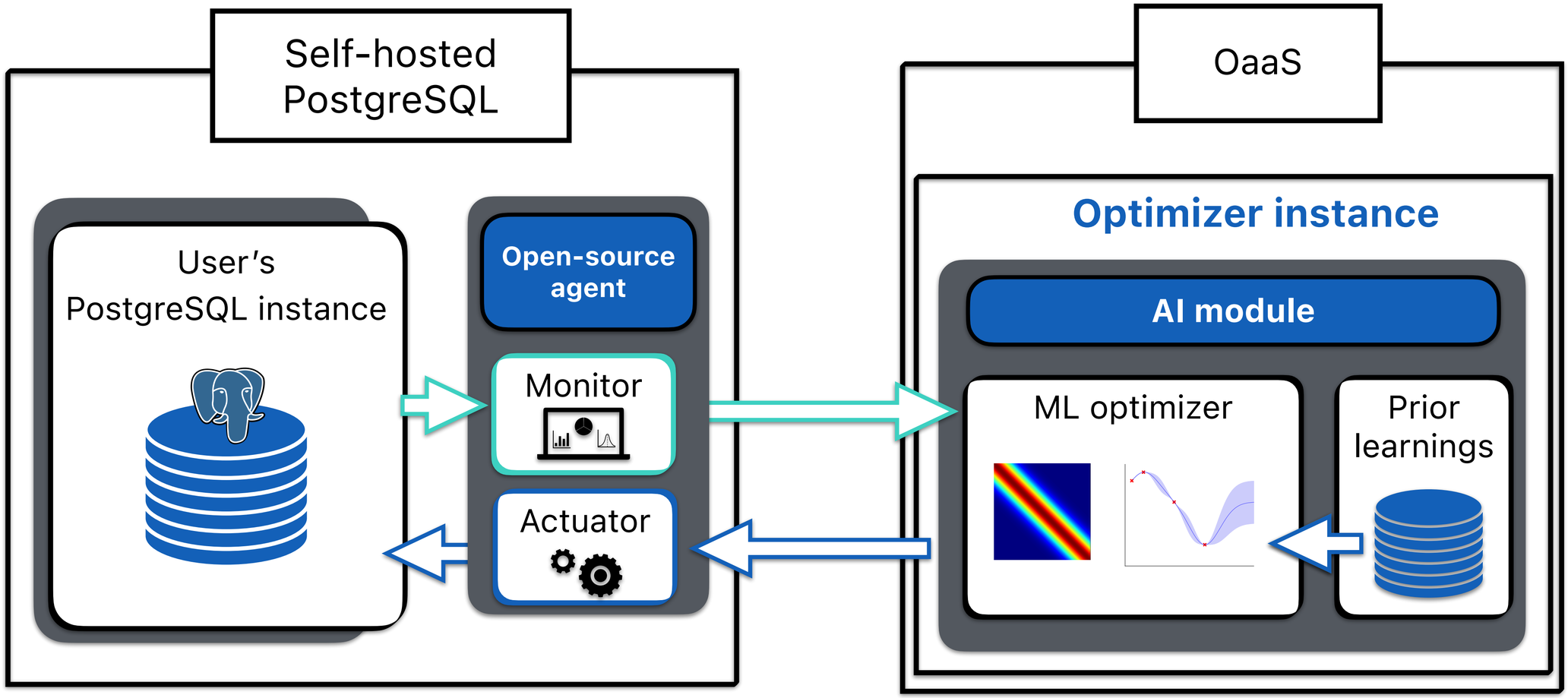
Ineffective
Tune again in a week

Inadequate
Seasonal workload



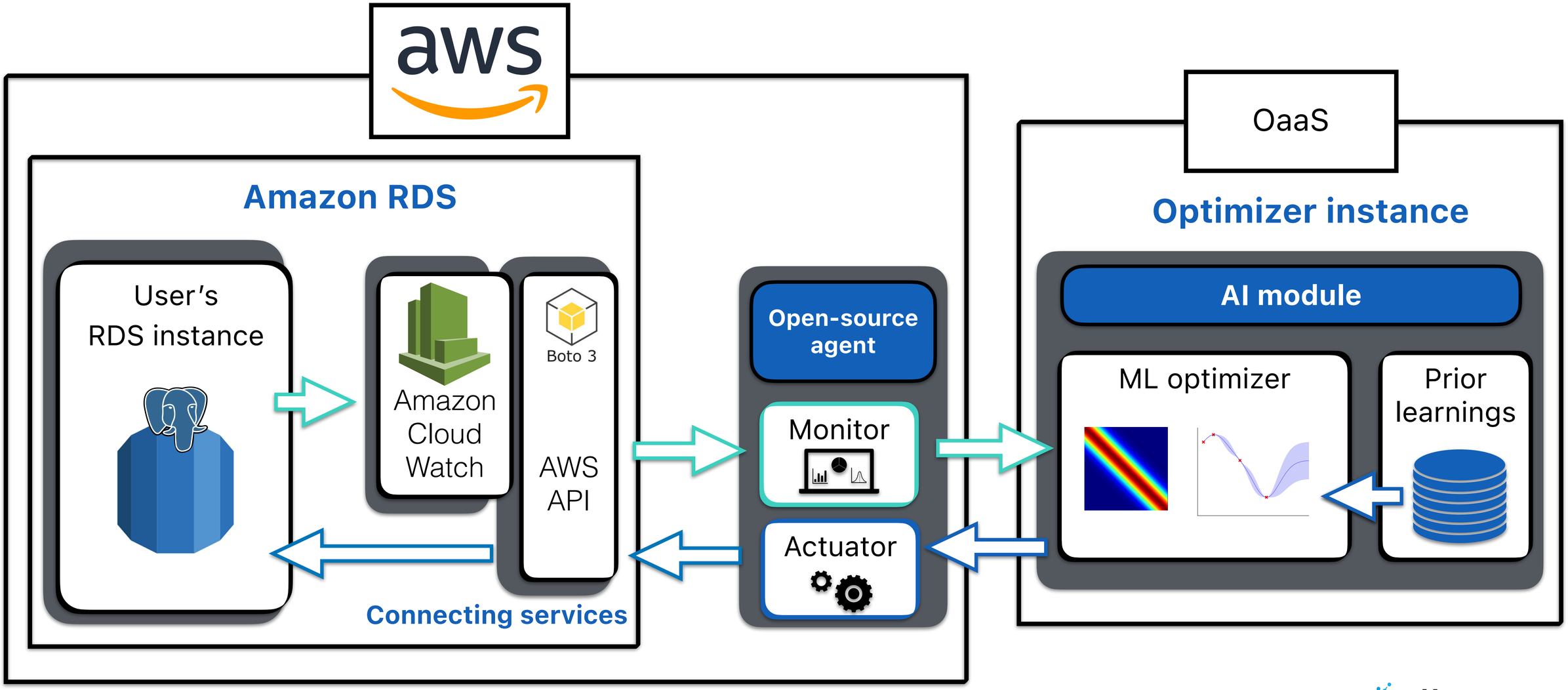
DBtune: PostgreSQL Optimizer-as-a-Service (OaaS)

High-level architecture view for self-managed PostgreSQL



DBtune: PostgreSQL Optimizer-as-a-Service (OaaS)

High-level view on Database as a Service (DBaaS) PostgreSQL — Amazon RDS/Aurora



How often do you tune?

Frequent

- Your workload changes — Change queries and application
- Your database grows and table statistics changes
- You scale your cloud instance — Up or down

Infrequent

- You migrate from on-prem to the cloud — Or vice-versa
- You migrate DBMS — E.g., from Oracle to PostgreSQL
- You upgrade your version of PostgreSQL

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 (11 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 (2 params)

- shared_buffers
- *max_worker_processes*

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

Alternatively: 1 restart during maintenance with heuristic defaults

- shared_buffers = 25%
- *max_worker_processes ~ vCPU*

http://docs.dbtune.com/Supported%20databases/PostgreSQL

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

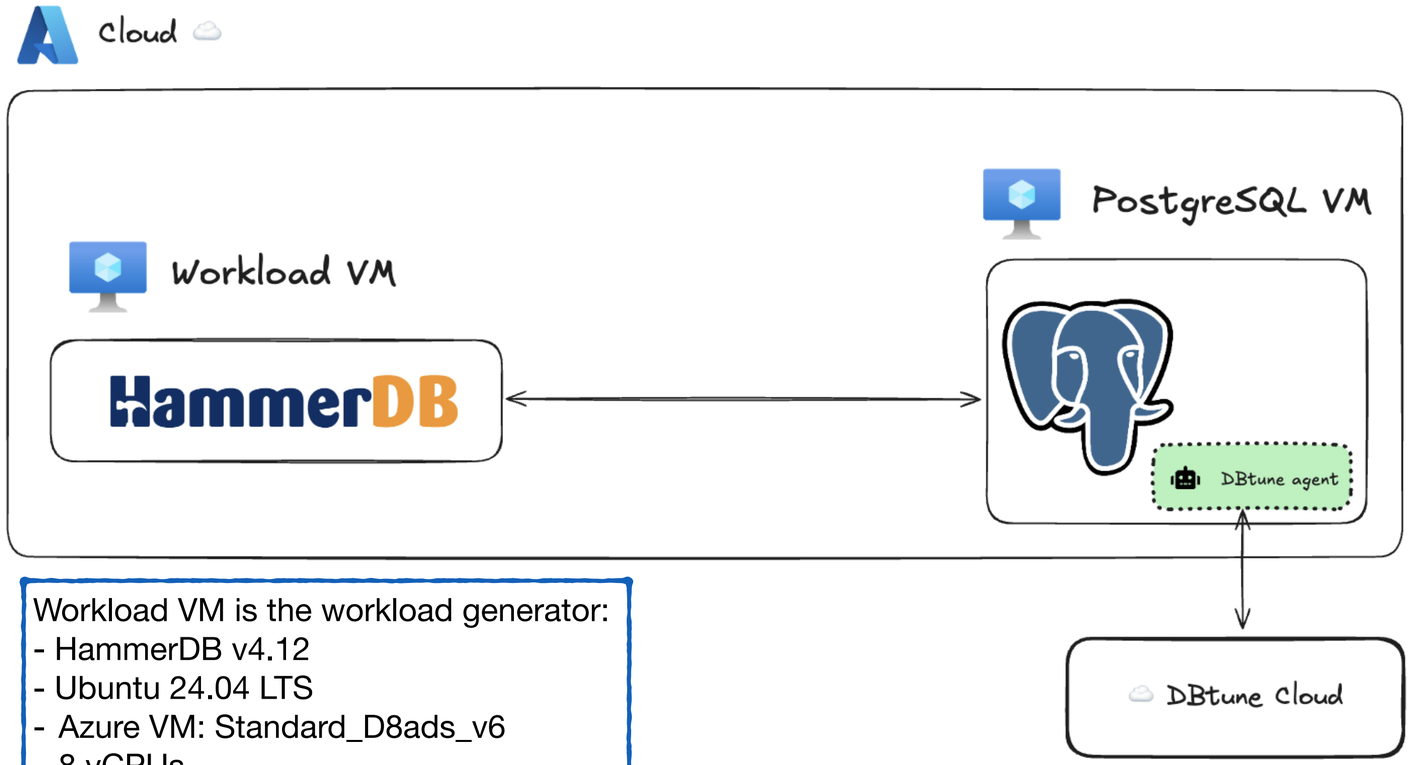
Many users execute transactions concurrently

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 for DBtune's complete 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

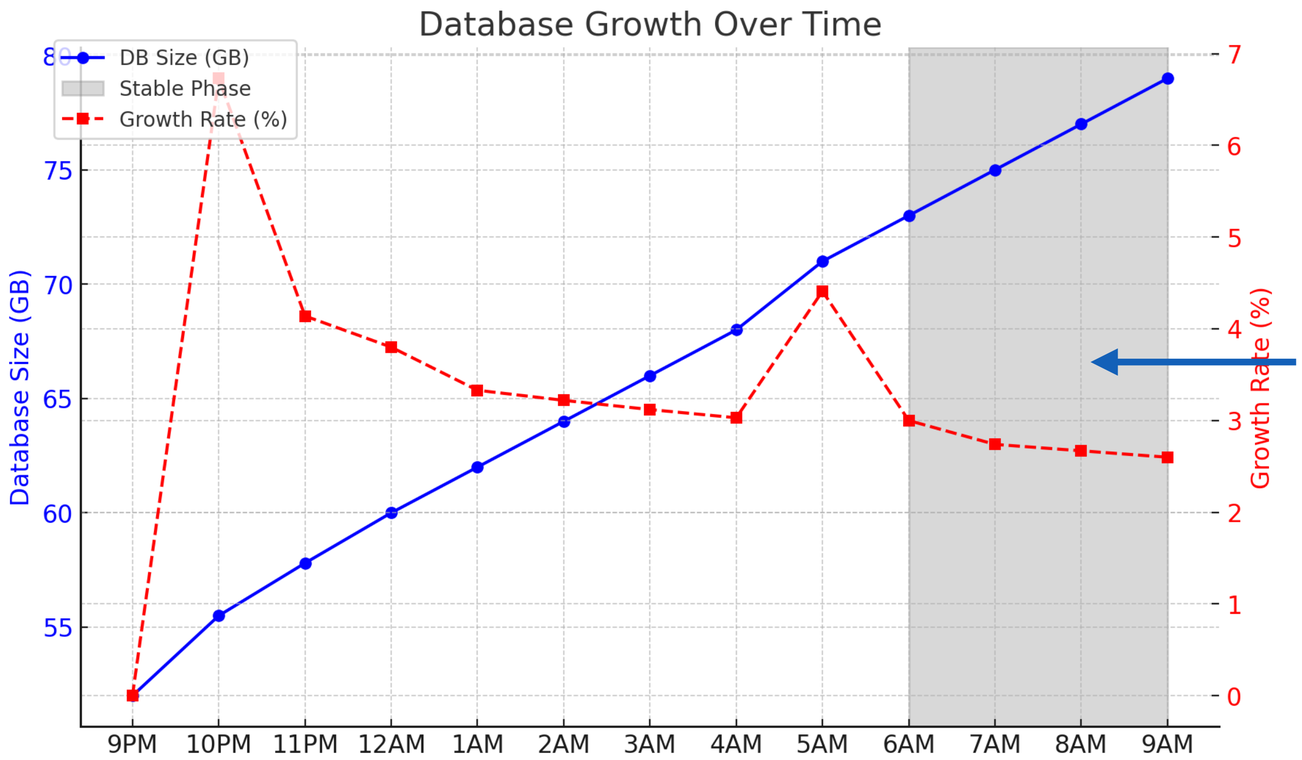
Setting number of connections:

- 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

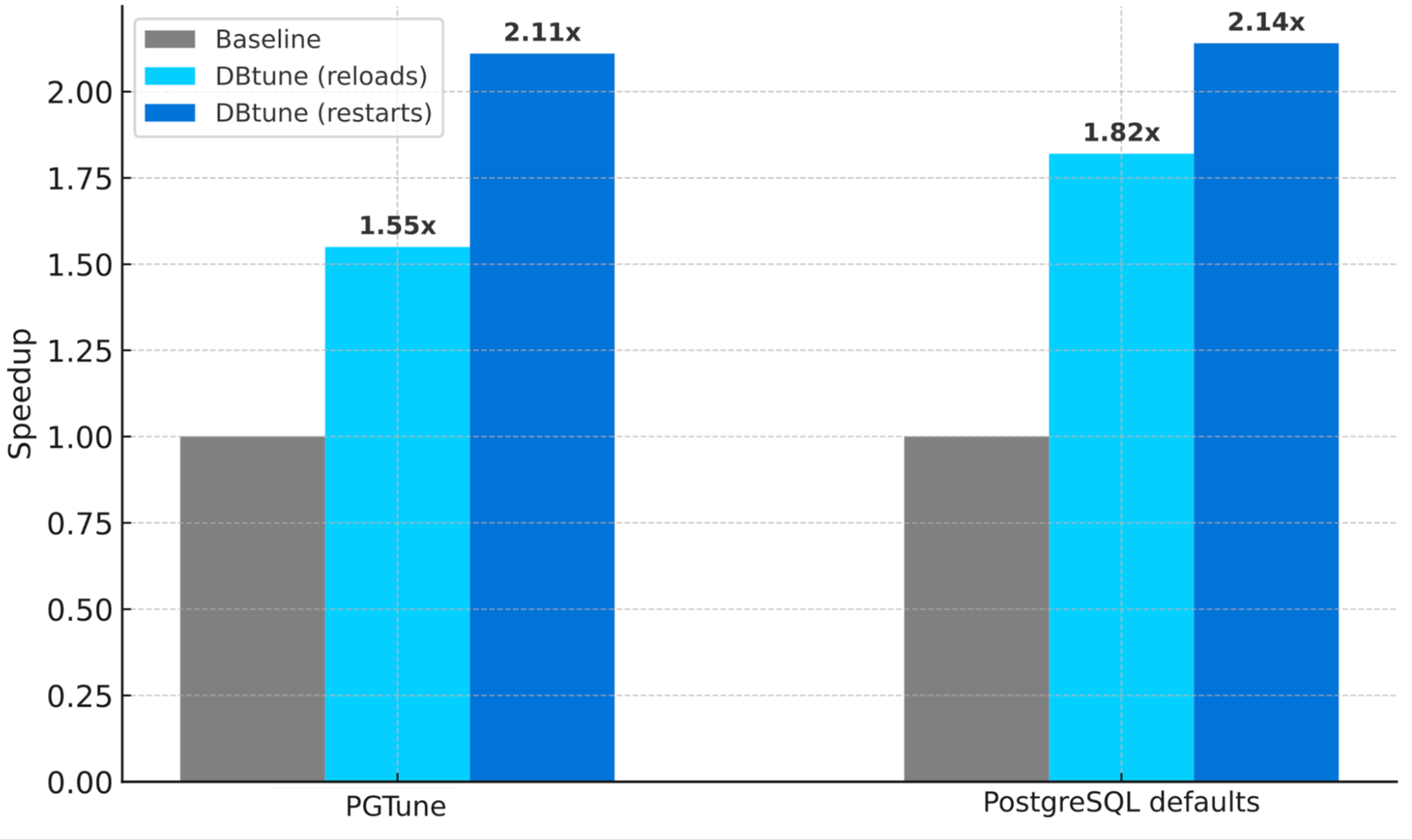
HammerDB performance metrics fluctuate significantly during the initial hours of operation as the system builds up buffer caches, statistics, and query plans



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

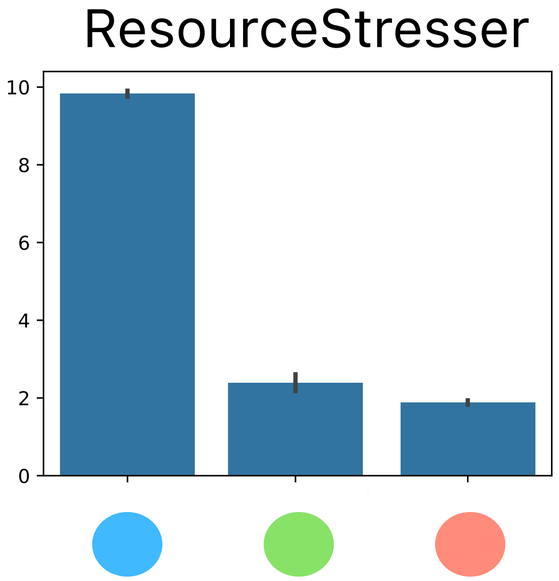
BenchBase benchmarks are configurable:

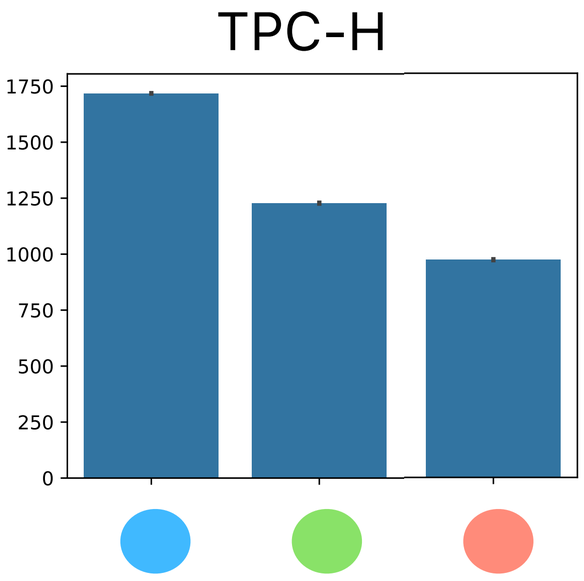
- **scalefactor**: Sets the size of the database
- **terminals**: Sets the number of connections querying the database
- **time**: Duration time of the benchmark in seconds
- **rate**: Transaction injection rate, determines how many TPS attempts to execute — Higher value makes for more intense workloads,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

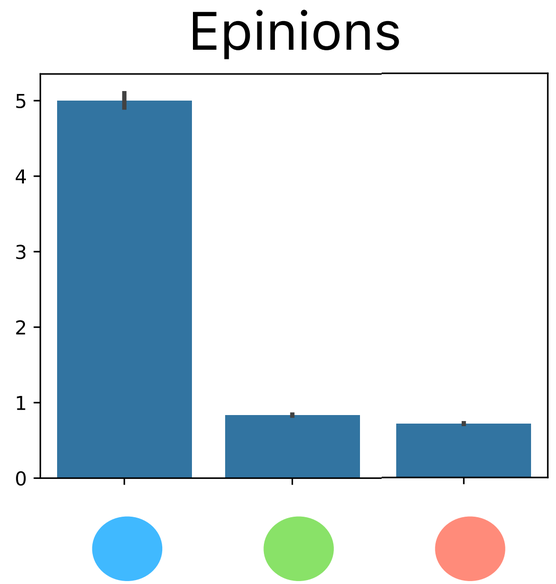
Example on Epinions: scalefactor=10000 (creates an 80 GB database), terminals to 10, time to 86400 (one day), rate=unlimited, and keep the weights as is

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

Average query runtime 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

Aiven for PostgreSQL (DBaaS)

Comparison against the Aiven defaults (baseline)

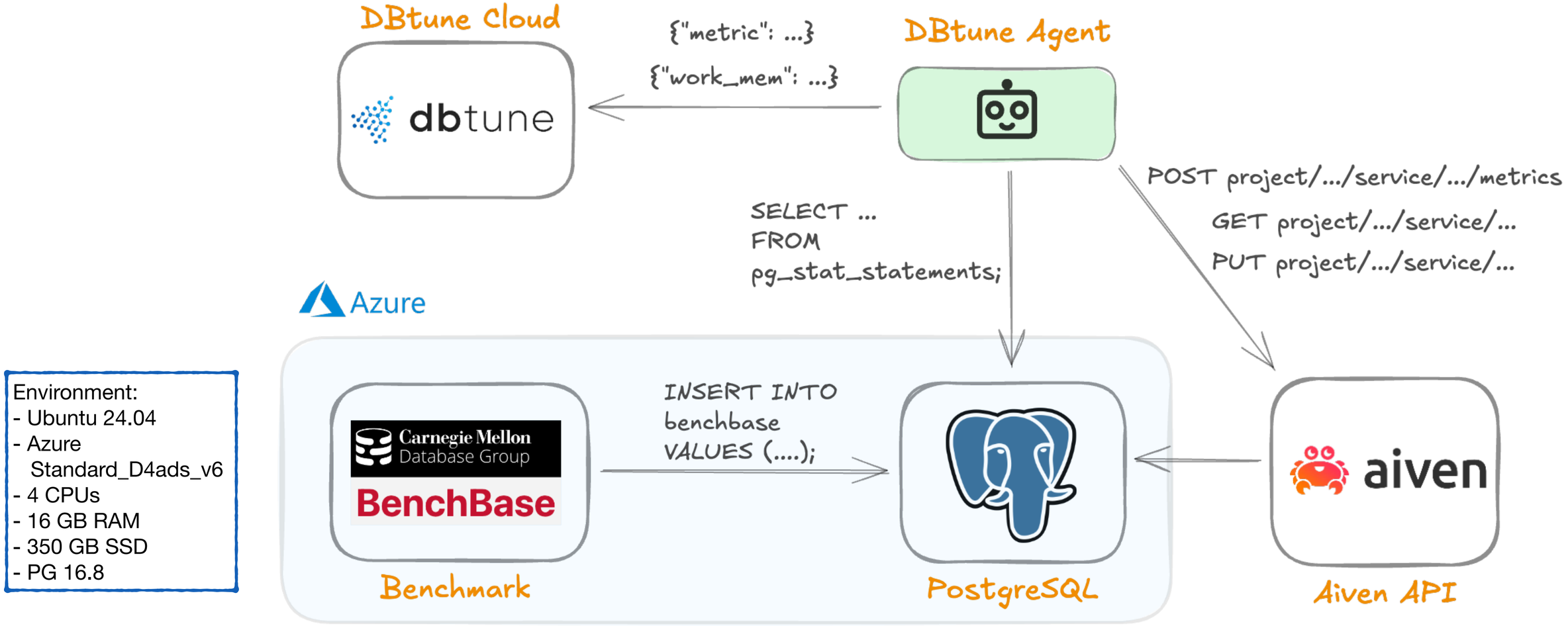
The Aiven DBaaS API gives access this subset of parameters

- work_mem
- *max_parallel_workers*
- *max_parallel_workers_per_gather*
- *effective_io_concurrency*
- *bgwriter_lru_maxpages*
- *random_page_cost*
- *sequential_page_cost*
- *bgwriter_delay*
- shared_buffers
- *max_worker_processes*

Blog: https://dbtune.com/blog/ai-powered-aiven-for-postgresql-server-tuning by Mohsin Ejaz (DBtune) and Eddie Bergman (DBtune)

Aiven for PostgreSQL (DBaaS)

High-level view of the distributed benchmarking and tuning environment

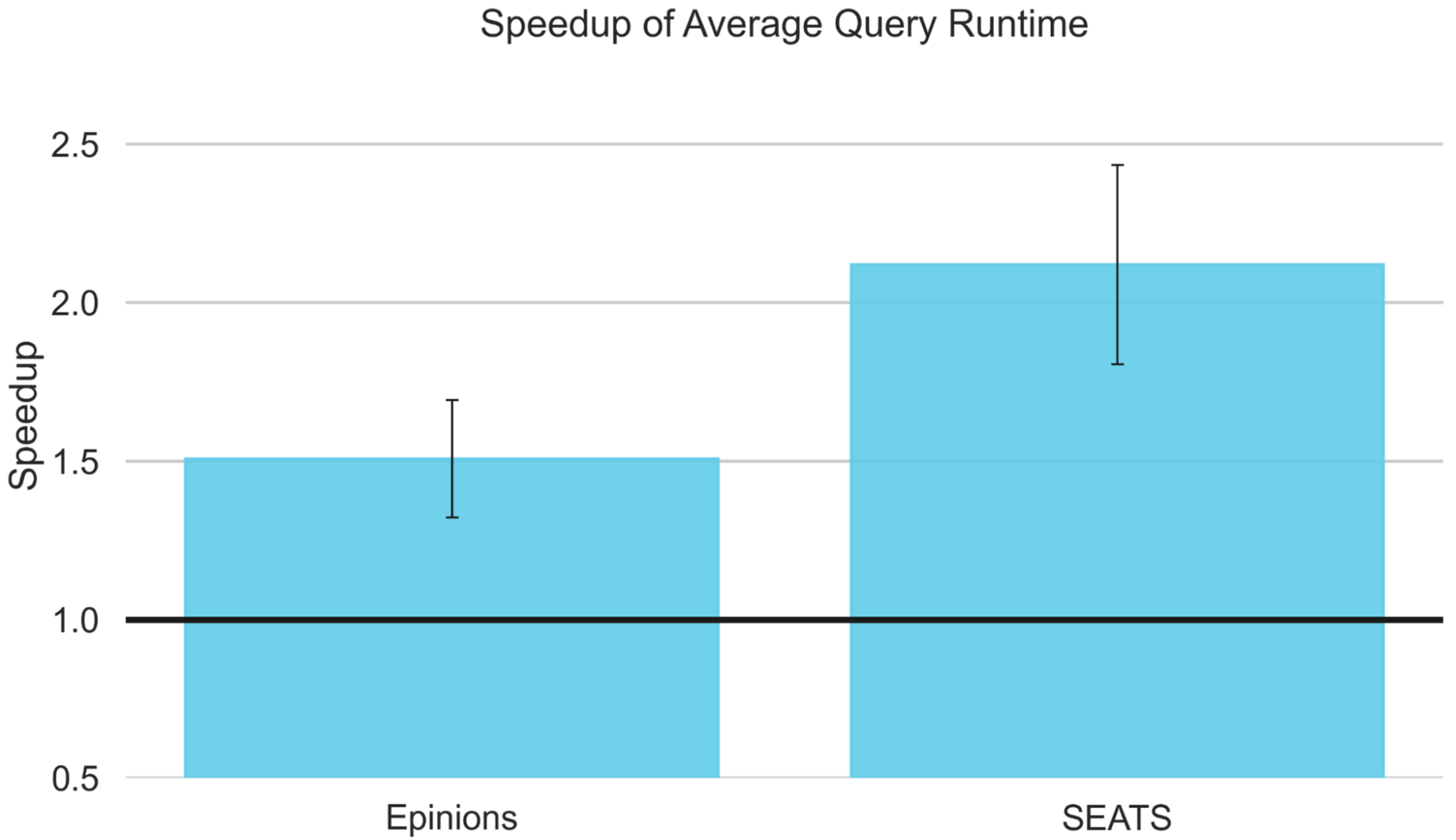


Environment:
- Ubuntu 24.04
- Azure Standard_D4ads_v6
- 4 CPUs
- 16 GB RAM
- 350 GB SSD
- PG 16.8

Blog: https://dbtune.com/blog/ai-powered-aiven-for-postgresql-server-tuning by Mohsin Ejaz (DBtune) and Eddie Bergman (DBtune)

Aiven for PostgreSQL

BenchBase results compared against Aiven defaults on a subset of parameters



Blog: https://dbtune.com/blog/ai-powered-aiven-for-postgresql-server-tuning by Mohsin Ejaz (DBtune) and Eddie Bergman (DBtune)

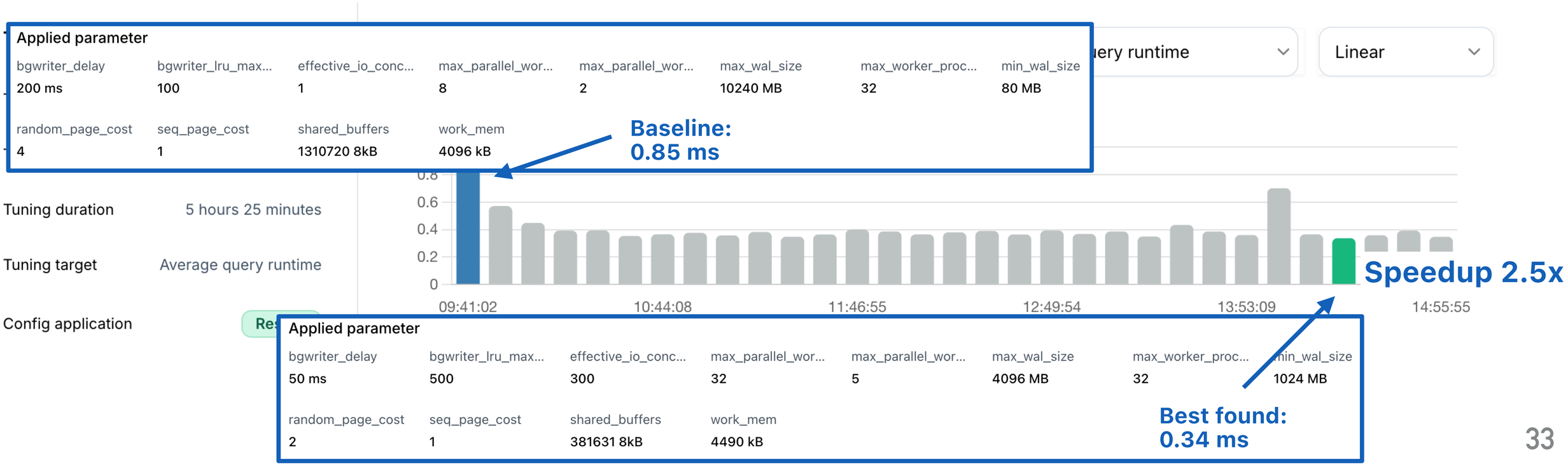
Real-world application use case study

PostgreSQL server parameter tuning in the wild

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

Manually tuned baseline by expert DBA

Automated tuning with DBtune



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
- Blog: From good to great: AI-powered Aiven for PostgreSQL server tuning (demo)
- Useful links: DBtune synthetic workload tutorial GitHub and video
- Panel at PGConf India on the AI revolution in PostgreSQL
- Demo on how DBtune works

@luinardi

luigi@dbtune.com

dbtune