dbtune

Réglage automatisé de PostgreSQL : Explorer l'optimisation des paramètres serveur

A 5-year long journey

PG Day France

June 4, 2024

Luigi Nardi, Ph.D.

Founder & CEO, DBtune

About me

@luinardi

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)

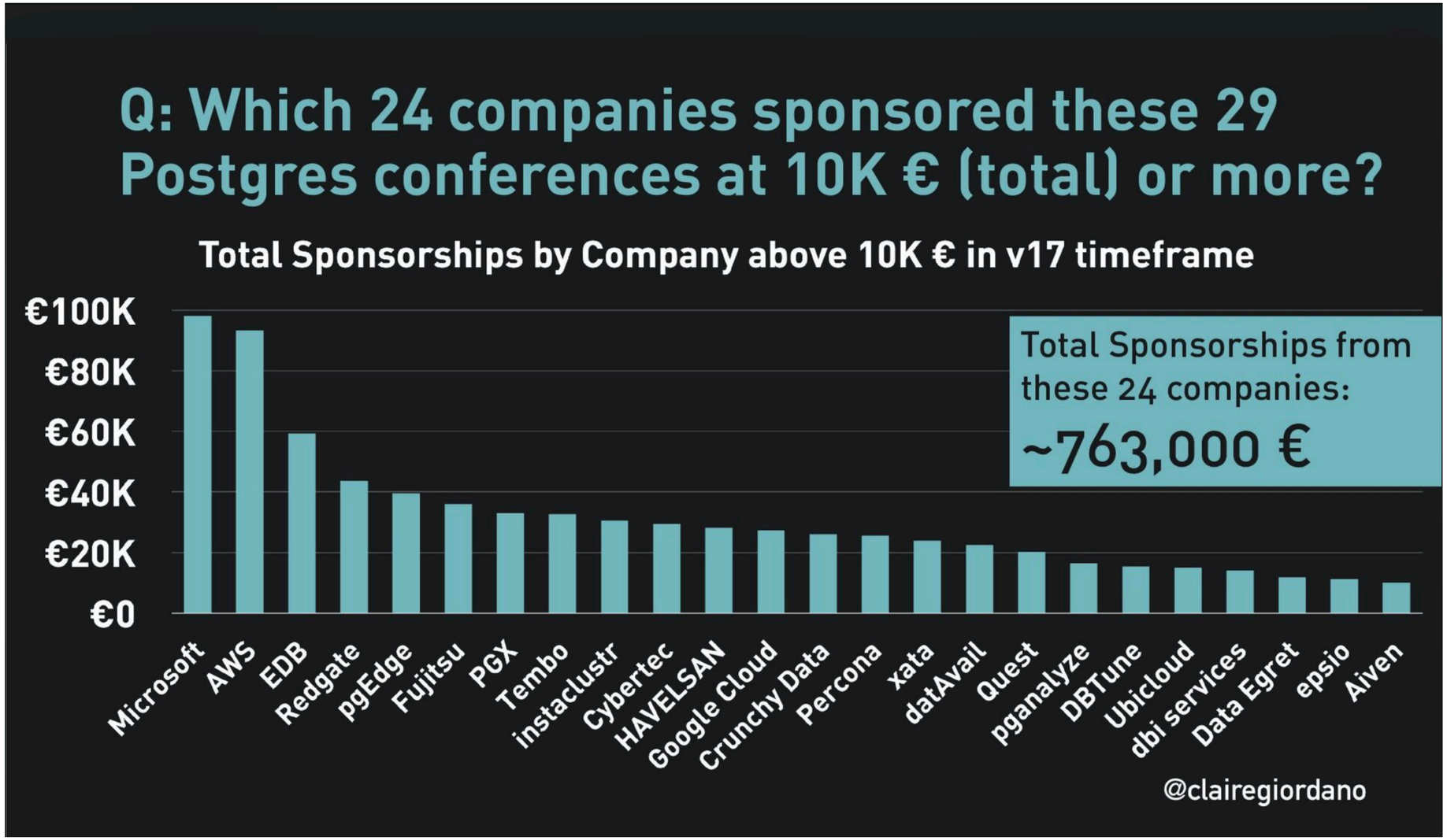
About DBtune

*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

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

Outline

- Introduction on PostgreSQL server parameter tuning
- Quantitative examples
- How do we solve this today?
- Machine learning tuning automation, a.k.a. AI agents for PostgreSQL
- Safety in autotuning
- Same examples of autotuning at DBtune
- Conclusions, user psychology and crossing the chasm

What is database tuning?
And how can it help us deliver against strategic objectives

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 focuse solely on automating PostgreSQL server parameter tuning

Why does it matter?

Technical perspective

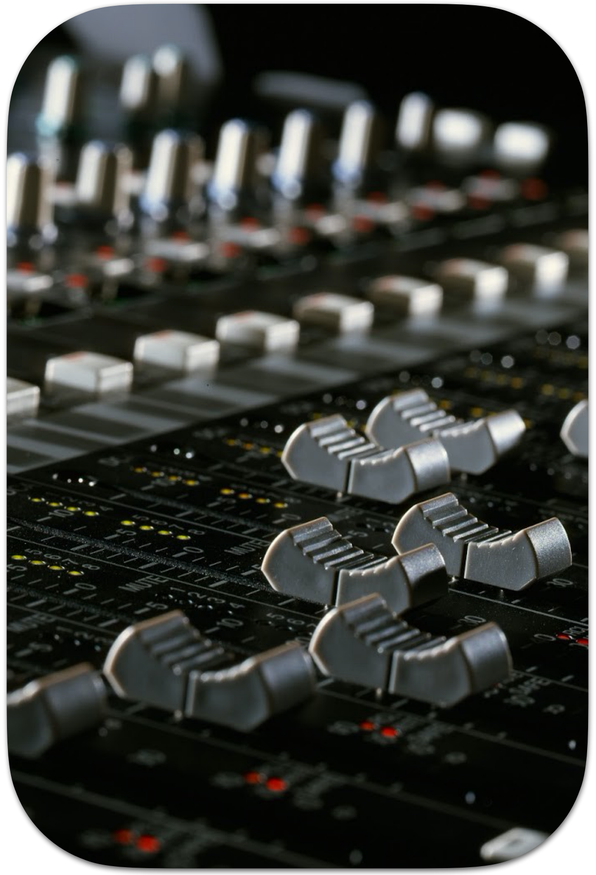
- Impacts system performance
  - Throughput and latency
- Improves scalability / stability / SLA

Business perspective

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

PostgreSQL server parameter tuning

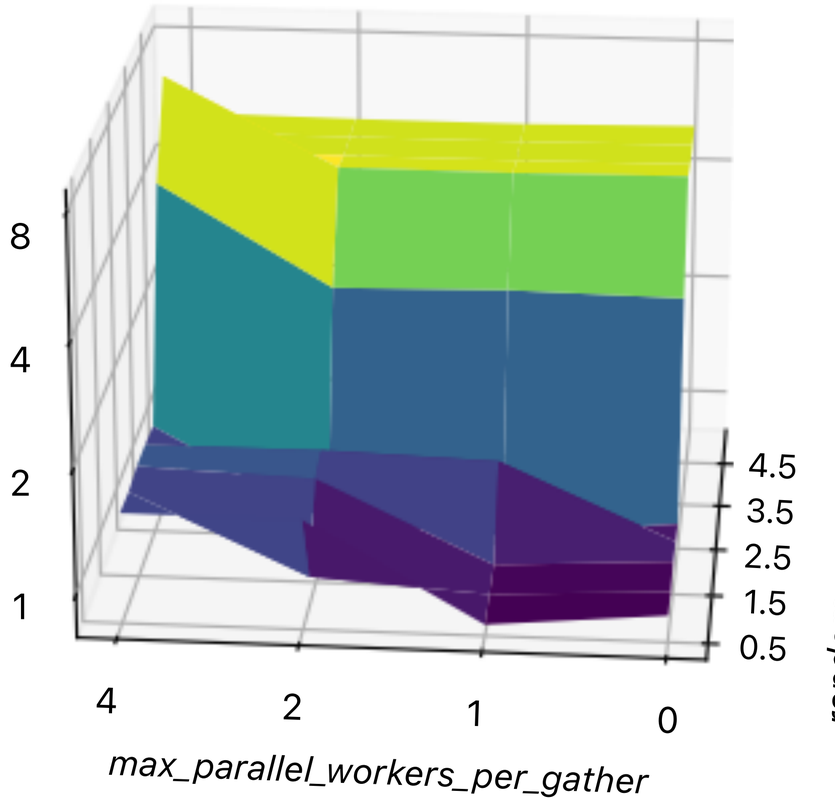
- Adjusting knobs to best fit the workload
- PostgreSQL parameters that are typically important:
  work_mem, *shared_buffers*, *max_wal_size*, etc.
- Example *max_parallel_workers_per_gather*:
  Max # of workers started by a Gather or Gather Merge node
- Example *random_page_cost*:
  Planner's cost of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

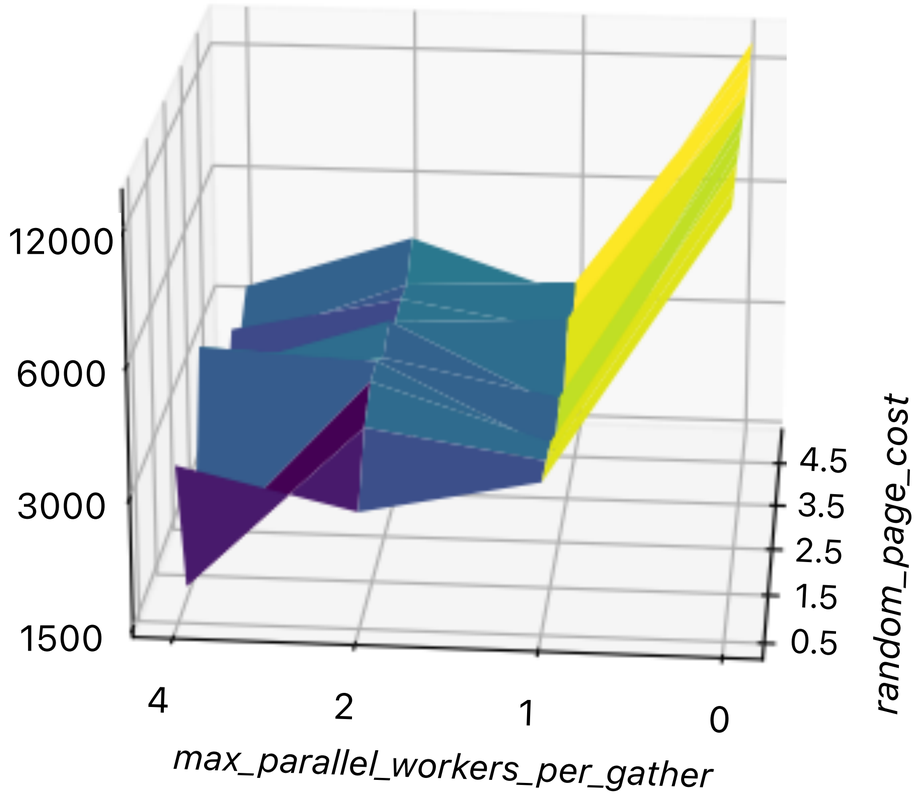
Epinions



Query runtime in ms

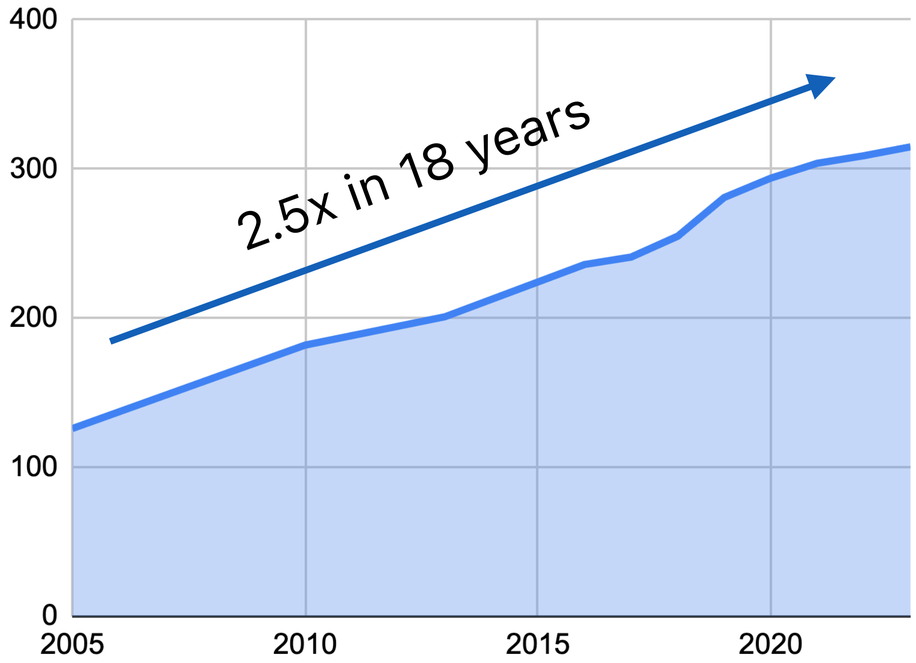
Lower the better

TPC-H



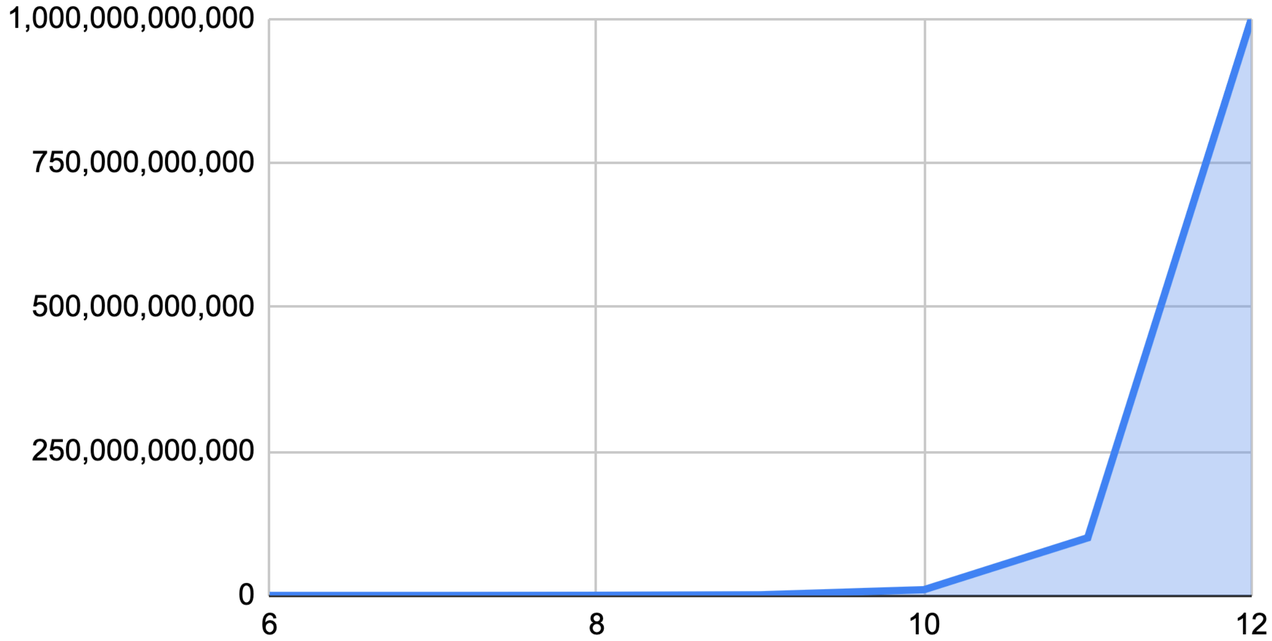
Complexity is growing over time

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

New approach

dbtune

Ideally a solution that learns by observation and autotunes

A solution that adapts to changing workloads

Heuristic-based server parameter tuning

Heuristics

PGTune

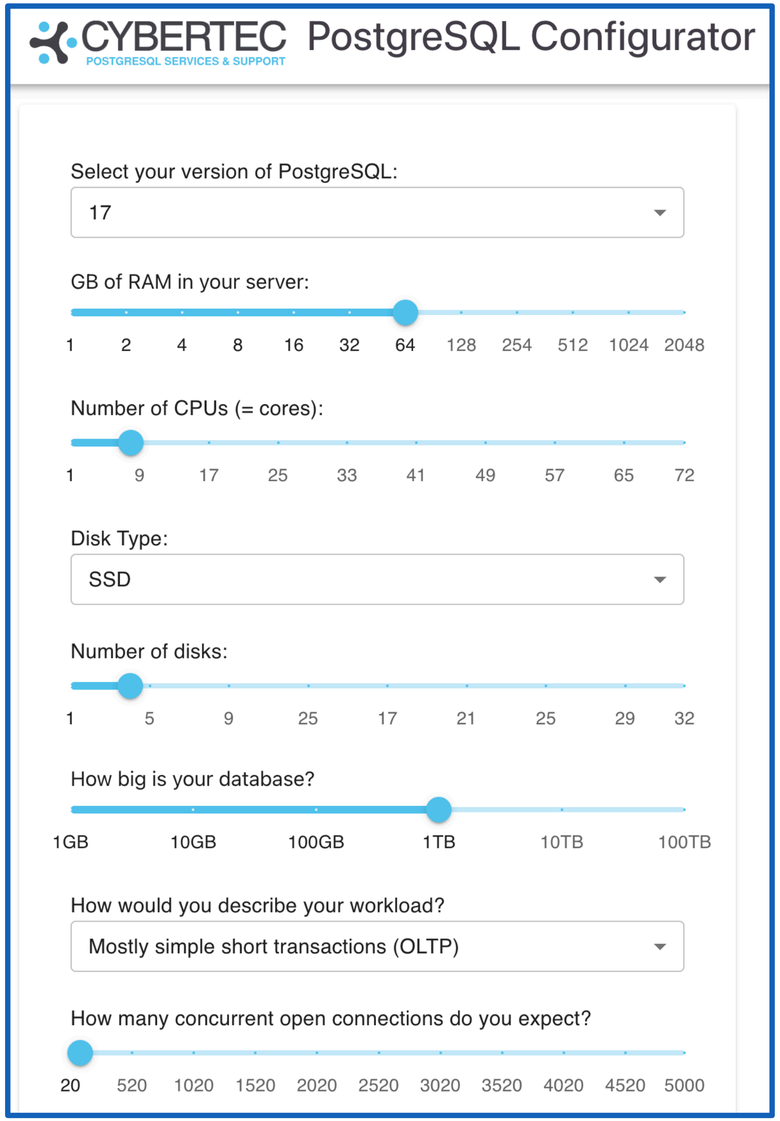
POSTGRESQL CONFIGURATOR

One-size-fits-all
Uses generic rules

Workload agnostic
Not bespoke

Ineffective
Tune again in a week

Inadequate
Seasonal workload



How often do you tune?

Frequent

- Your workload changes — Change queries and application
- Your database grows and 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 ($$$)

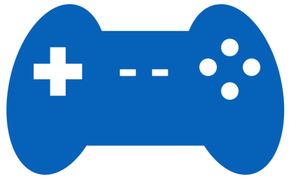
Desiderata for PostgreSQL autotuning

Machine learning approach

Agent that learns to solve workload-specific optimization challenges

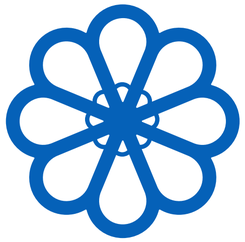
Dynamic adaptation

Autotune a PostgreSQL instance irrespective of size and complexity



Easy to use

Ideally, no need for background in ML or PostgreSQL tuning



Scaleable

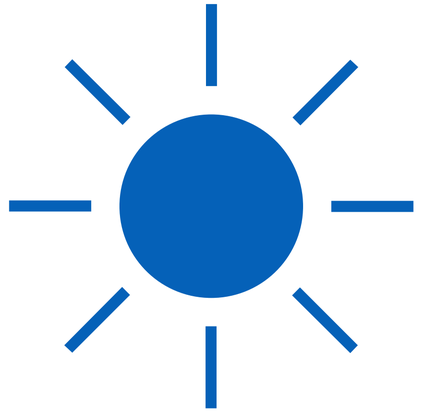
Tune multiple databases in heterogeneous environments

User value propositions

Reduce cloud / infrastructure costs

Make your service radically faster

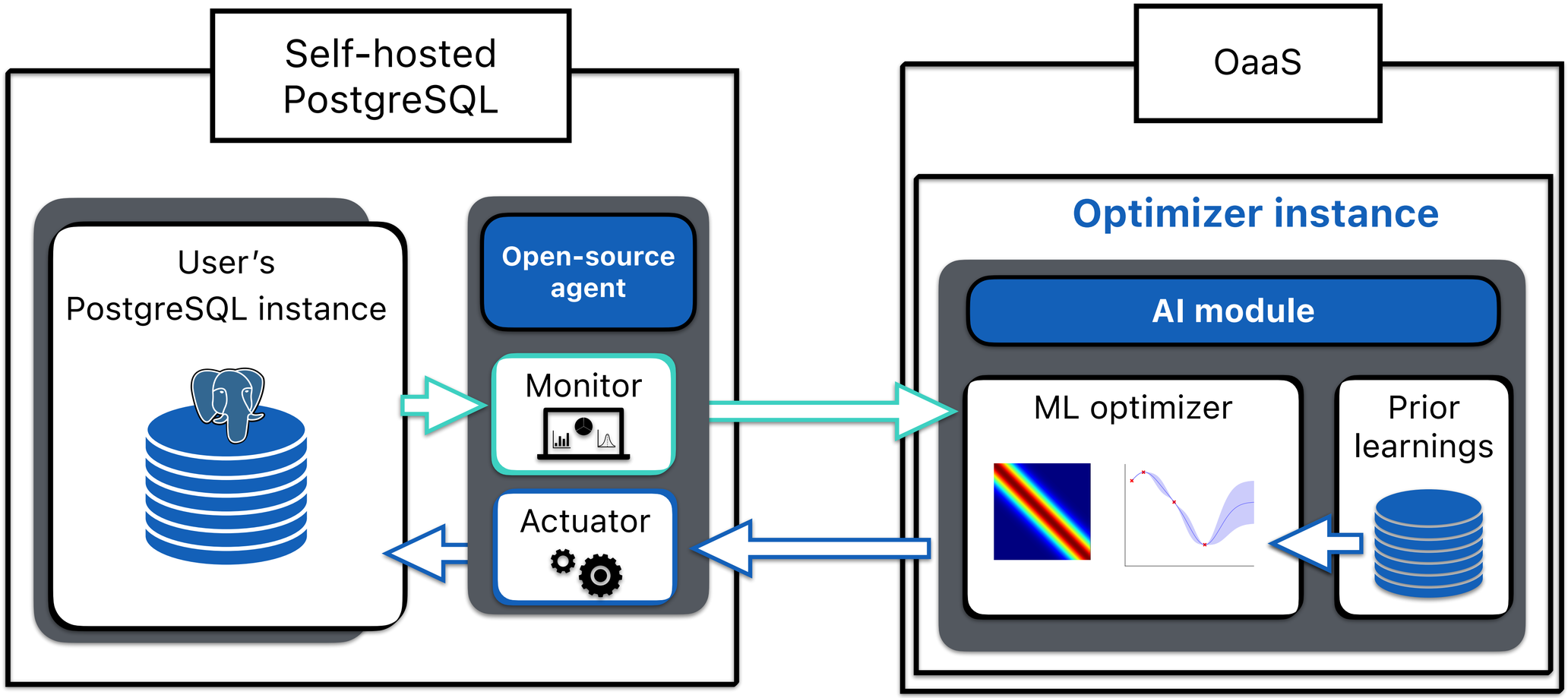
Free up your DBAs



Reduce energy consumption

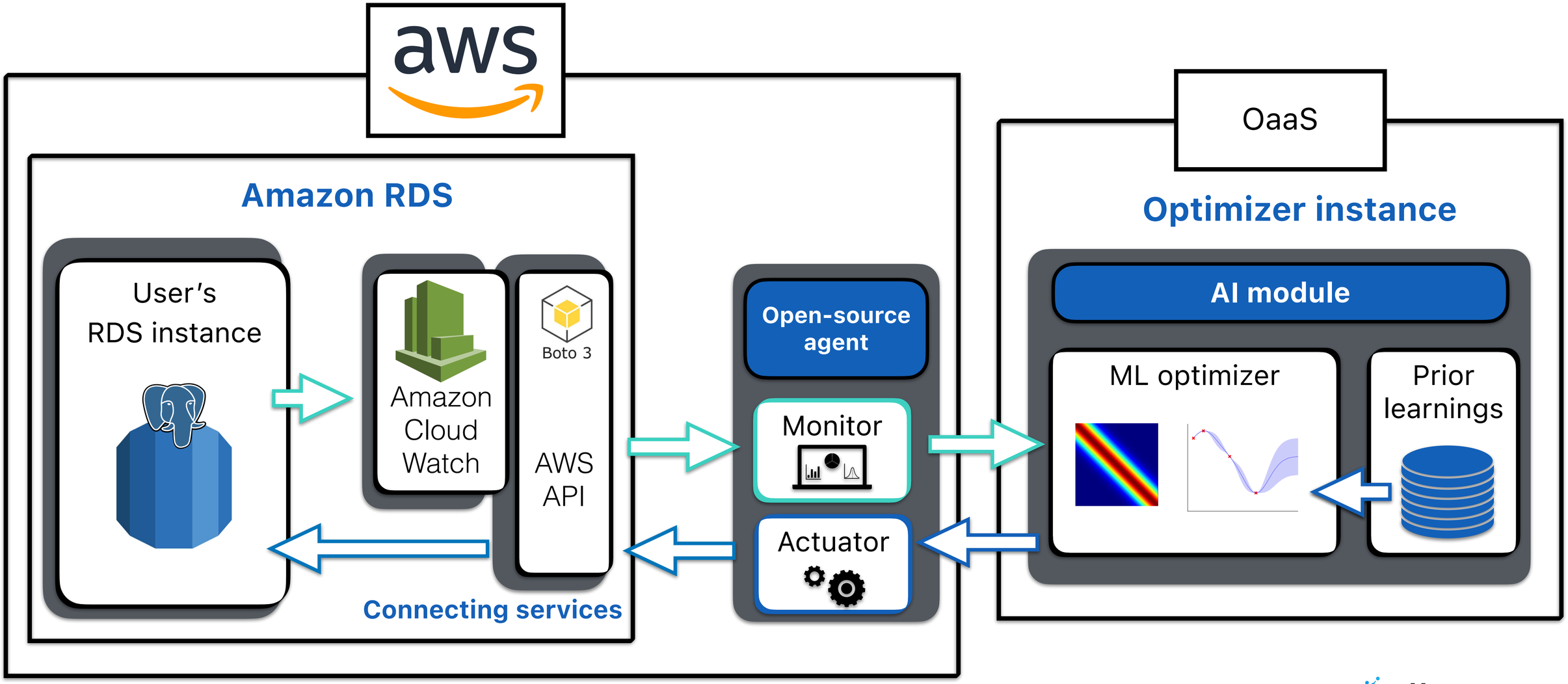
PostgreSQL Optimizer-as-a-Service (OaaS)

High-level architecture view for self-managed PostgreSQL



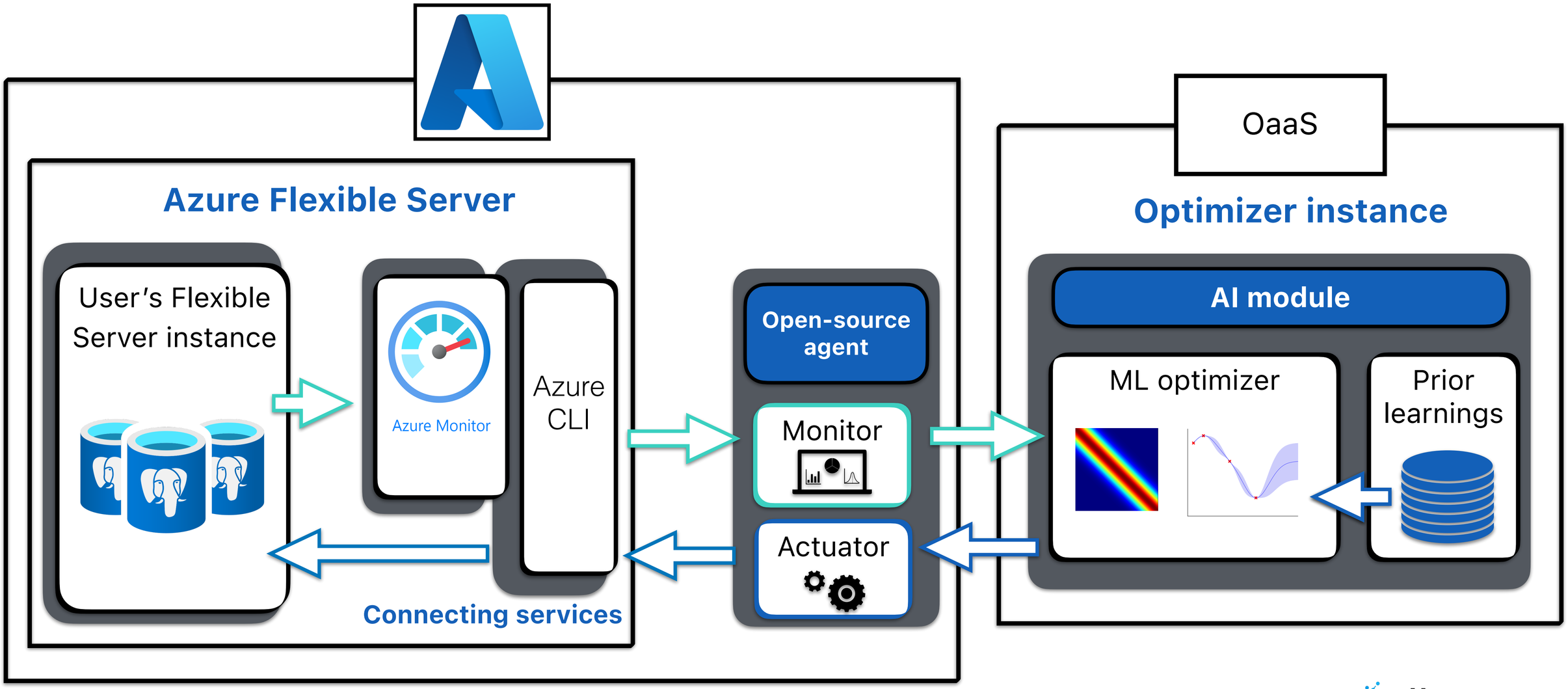
Autotuning architecture for Database as a Service (DBaaS) (2)

High-level view RDS PostgreSQL/Aurora



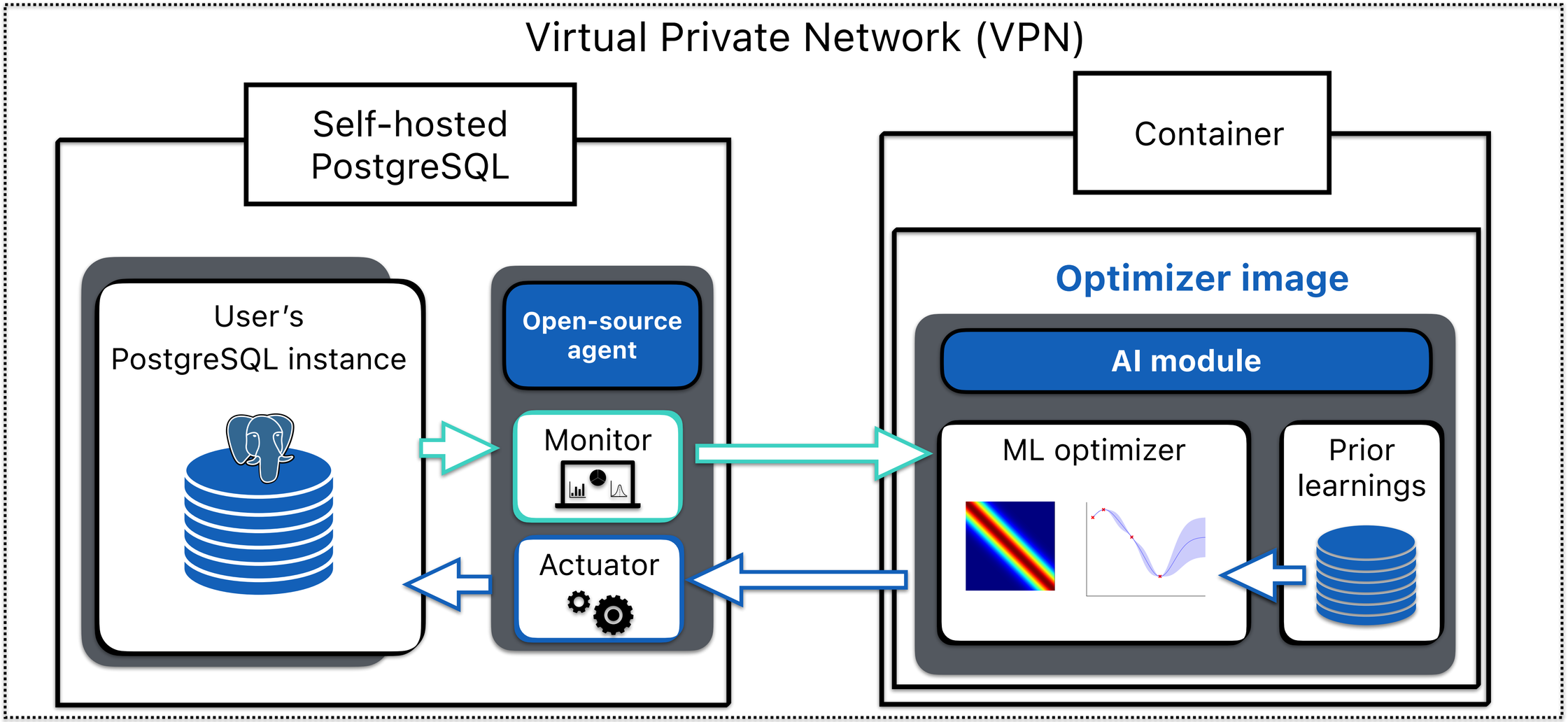
Autotuning architecture for Database as a Service (DBaaS) (3)

High-level view Azure Flexible Server



Autotuning architecture for instances that are offline (4)

High-level view



Safe tuning in production environments

System guardrails to avoid unsafe configurations

Constrained optimization

Parameters have safe upper / lower limits in place

Memory monitoring guardrail

Real-time system memory monitoring to revert from potentially unsafe configurations

E.g. configuration that uses too much RAM — Triggered at 90% of RAM

Performance degradation early exit condition

Optimization space may result in configuration with worse performance than the user default

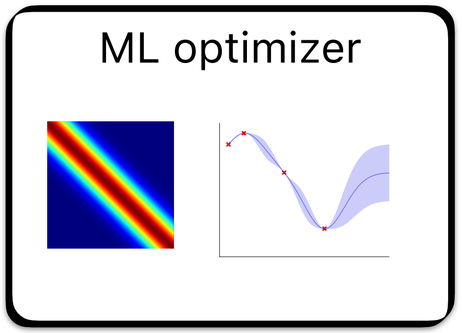
This triggers early exit from existing configuration and move to next iteration

Safe tuning in production environments (2)

System guardrails to avoid unsafe configurations

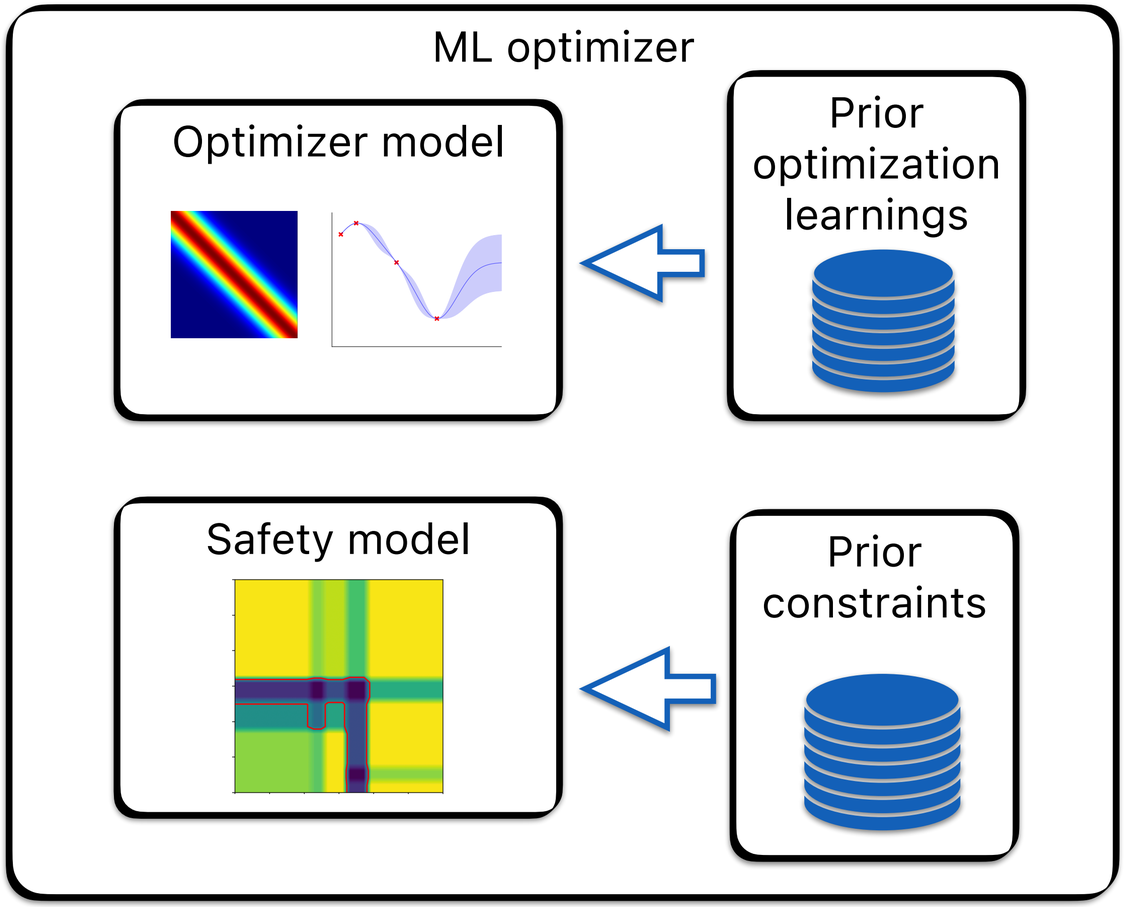
When the guardrails are triggered a safety model is trained

This model learns feasibility constraints



More precisely

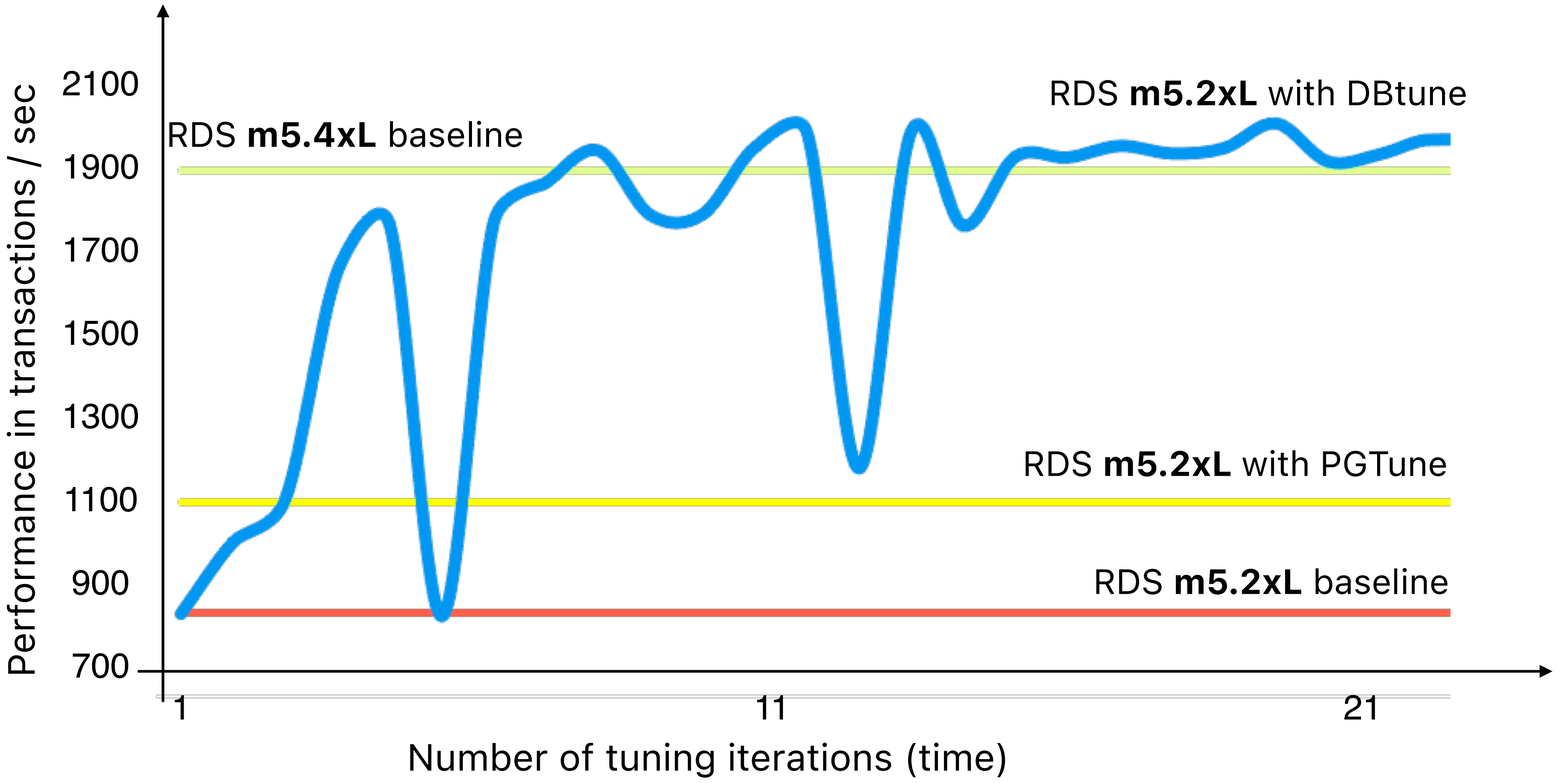
=



Performance tuning results

Doubling the performance of PostgreSQL Amazon RDS

Performance impact of tuning RDS m5.2xLarge cloud instance on the TPC-C benchmark



On the smaller instance type it can achieve a level performance in excess of that achieved by an instance twice the size

Proof of cost reduction: Detailed cost analysis

Doubling the performance of PostgreSQL Amazon RDS

Hardware				Cost / Year		
AWS RDS Instance Type	Cores	RAM	IOPS	Instance	EBS	Total
db.m5.4xlarge	8	64 GBs	4000	$12,475	$4,800	$17,275
db.m5.2xlarge	4	32 GBs	2000	$6,237	$2,400	$8,637

Per instance savings: $8,638

- It halves RDS cost (50% saving)
- Matches 4xLarge performance on a 2xLarge instance
- Medium and large companies use hundreds* of RDS instances

*A16z article: "The Cost of Cloud, a Trillion Dollar Paradox"

Example of PostgreSQL parameters tuned by DBtune

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

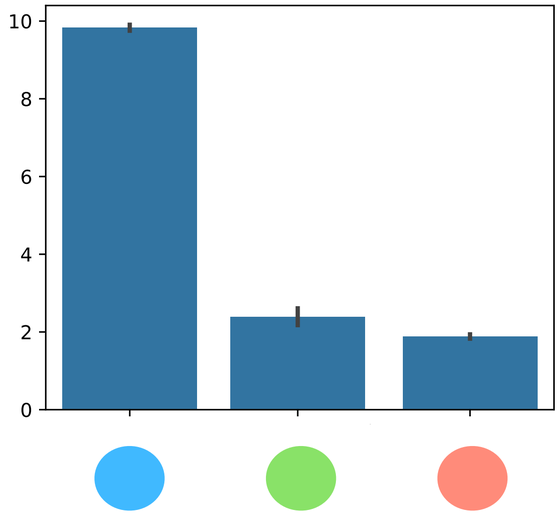
- shared_buffers = 25%
- *max_worker_processes ~ vCPU*

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

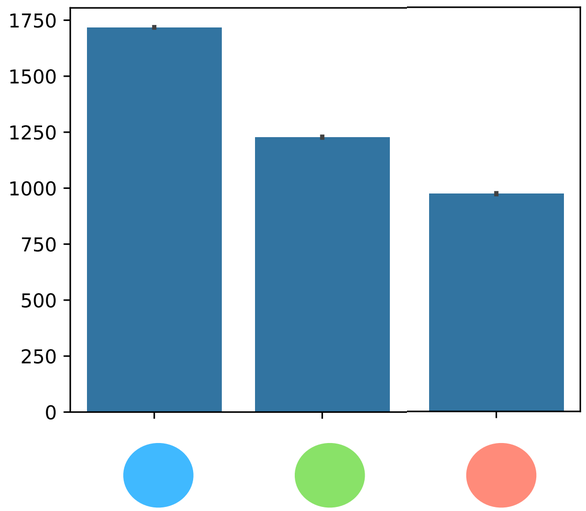
Performance downside of non-restart (reload-only) strategy

Average query runtime

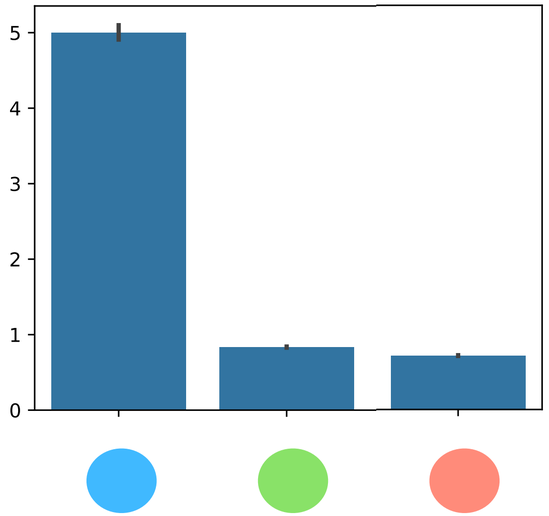
ResourceStresser



TPC-H



Epinions



Default PostgreSQL configuration & no autotuning

shared_buffers=25% & reload-only & autotuning

Restarts allowed & autotuning

Limitations and notes on AI agent PostgreSQL tuning

- On autotuning cloud provider DBaaS
  - Cloud provider APIs need to be comprehensive and flexible
- Model Context Protocol (MCP)
  - MCP could help other agents to interact with the tuning agent in an agentic world
- Performance improvements are somewhat ill-defined
  - Agreement between the user and the agent on what to optimize (workload fingerprint)
- Restarts vs reloads
  - *shared_buffers* will be dynamic soon in future PG releases (see hackers mailing list)
  - No sign of changes for *max_worker_processes* so far

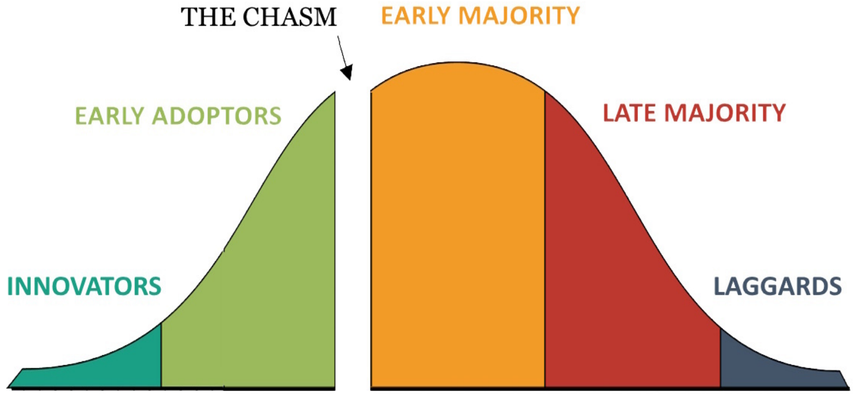
User psychology and society readiness level for PostgreSQL agents

- The system described in this presentation is similar to a Waymo car

  It takes you autonomously from A to B: "Waymo take me from Palo Alto to Menlo Park"

- Would you put your PG production system in the hands of a software agent?

  - Waymo took ~15 years
  - Genetic Query Optimization (GEQO) is an example in PostgreSQL
  - Technology adoption life cycle



TECHNOLOGY ADOPTION LIFECYCLE

- Debate-style discussion on autotuning at PGDay Lowlands on Sep 12

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

luigi@dbtune.com

dbtune


MANUAL
TUNING
DBTUNE