dbtune

Workload Fingerprints

A key to understanding PostgreSQL performance

PGConf US, NYC

September 30, 2025

Dr. Luigi Nardi

Founder & CEO, DBtune

Talk co-written with

dbtune

Dr. Erik Hellsten

Contributors

Raffaello Baluyot

Dr. Miguel González-Duque

Dr. David Edwards

Eddie Bergman

Matthew Ingram

Mohsin Ejaz

Ellyne Phneah

Alexandre Gougeon

Marko Bocevski

Dr. Marc Linster

Magnus Hagander

Kenneth Rugg

About me

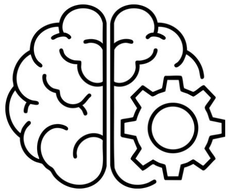
B.Sc and M.Sc. Computer Engineering at La Sapienza — Rome (Italy)

- 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)

dbtune

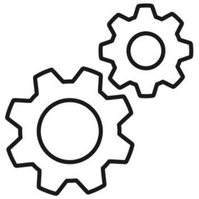
What

DBtune is an AI-powered database tuning service



Where

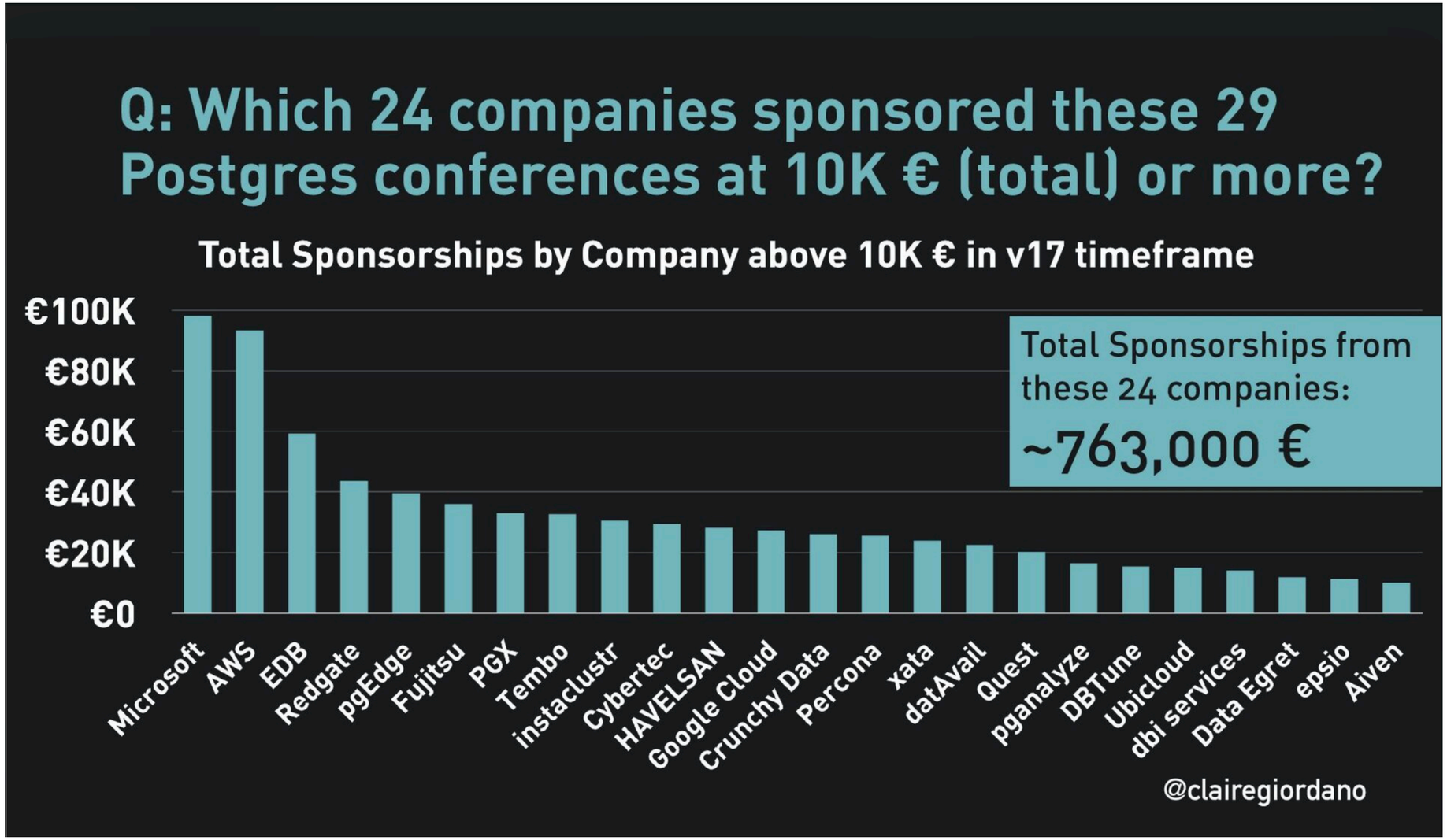
Spun out of research at Stanford University



How

Tunes for a specific workload, use case and machine

DBtune 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

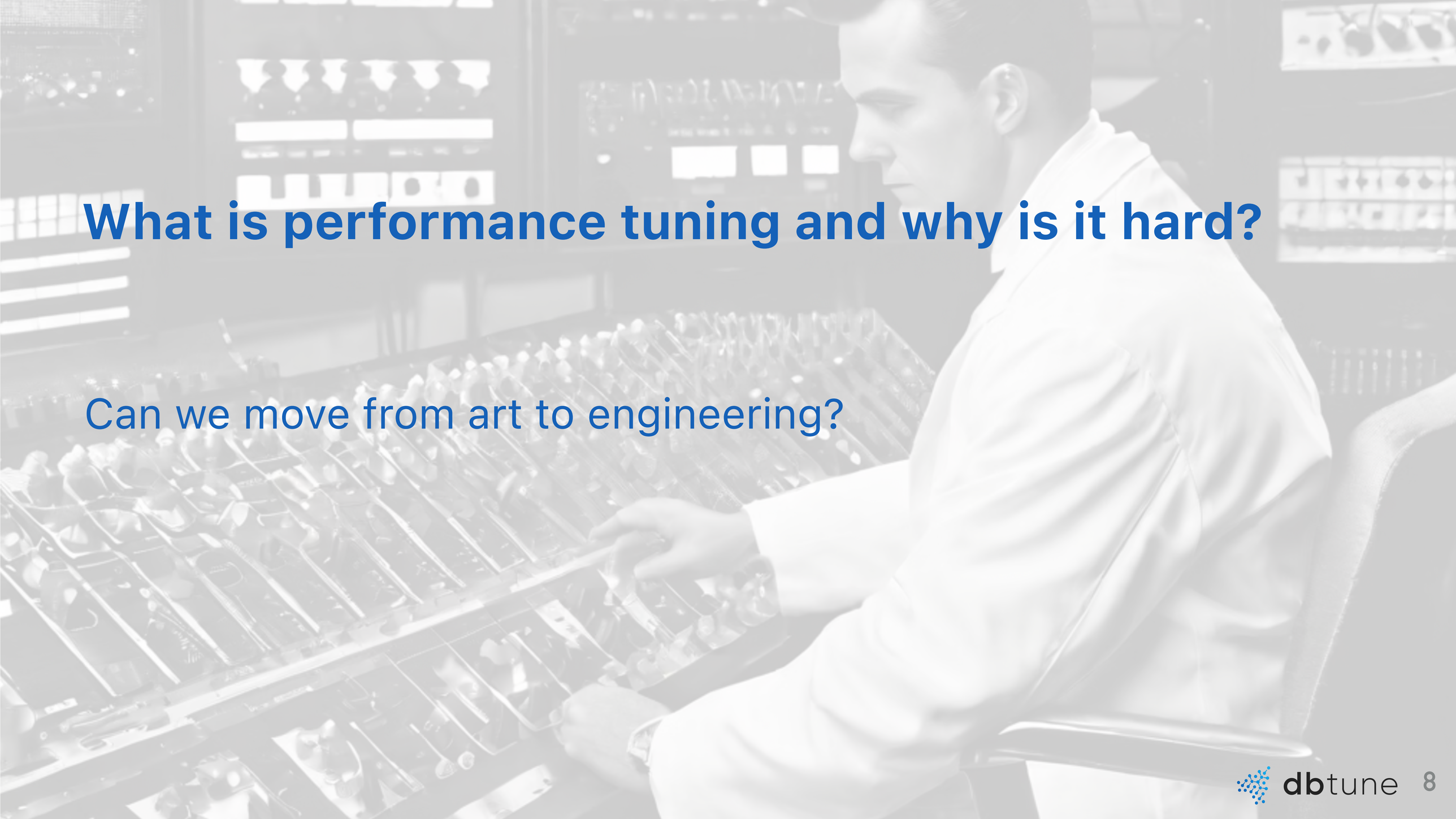
- The group is officially recognized by PostgreSQL Europe
- Regular meetups every 4-8 weeks in Malmö — Top speakers
- We are building a vibrant PostgreSQL community in the region

Outline

- Intro to PostgreSQL performance tuning
- Effect of noise in the performance measurement
- Workload Fingerprint: A robust method for tackling measurement noise
- Automatic recommendation of Workload Fingerprints
- Conclusions and future work

What is performance tuning and why is it hard?

Can we move from art to engineering?



What is PostgreSQL performance 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**
- Performed by a “tuner”: Either a DBA/developer or an AI agent
- Tuning includes query, server parameters, index, OS parameters, etc.

Why does it matter?

Technical perspective

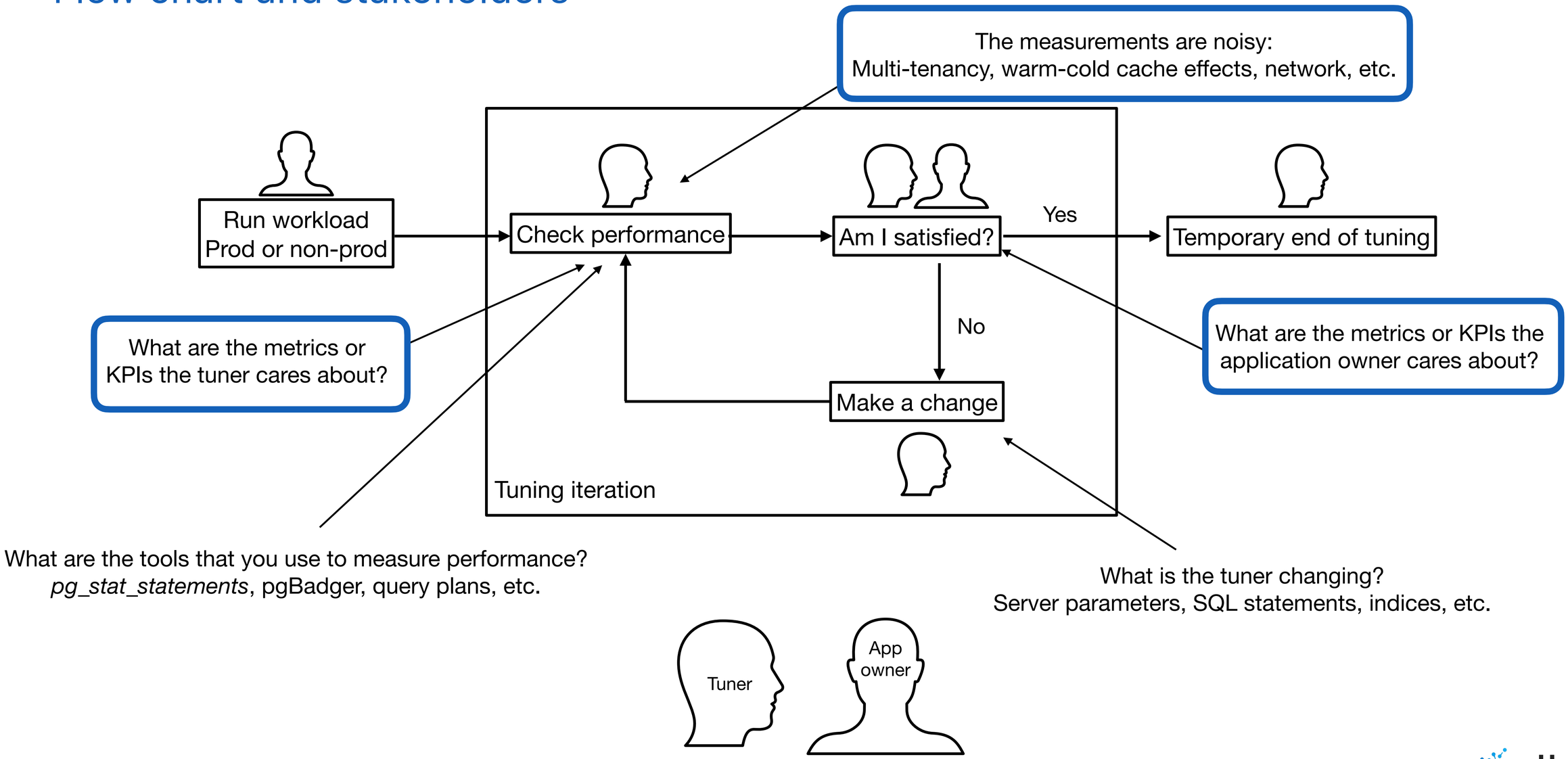
- Impacts system performance (throughput and latency)
- Improves scalability / stability / SLA

Business perspective

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

Tuning process pictorial

Flow chart and stakeholders



Outline of a tuning process

1. Make a change
2. Observe "long enough" (next slide)
3. Calculate performance (next slide)
4. Am I satisfied?
5. Go back to 1 or exit tuning

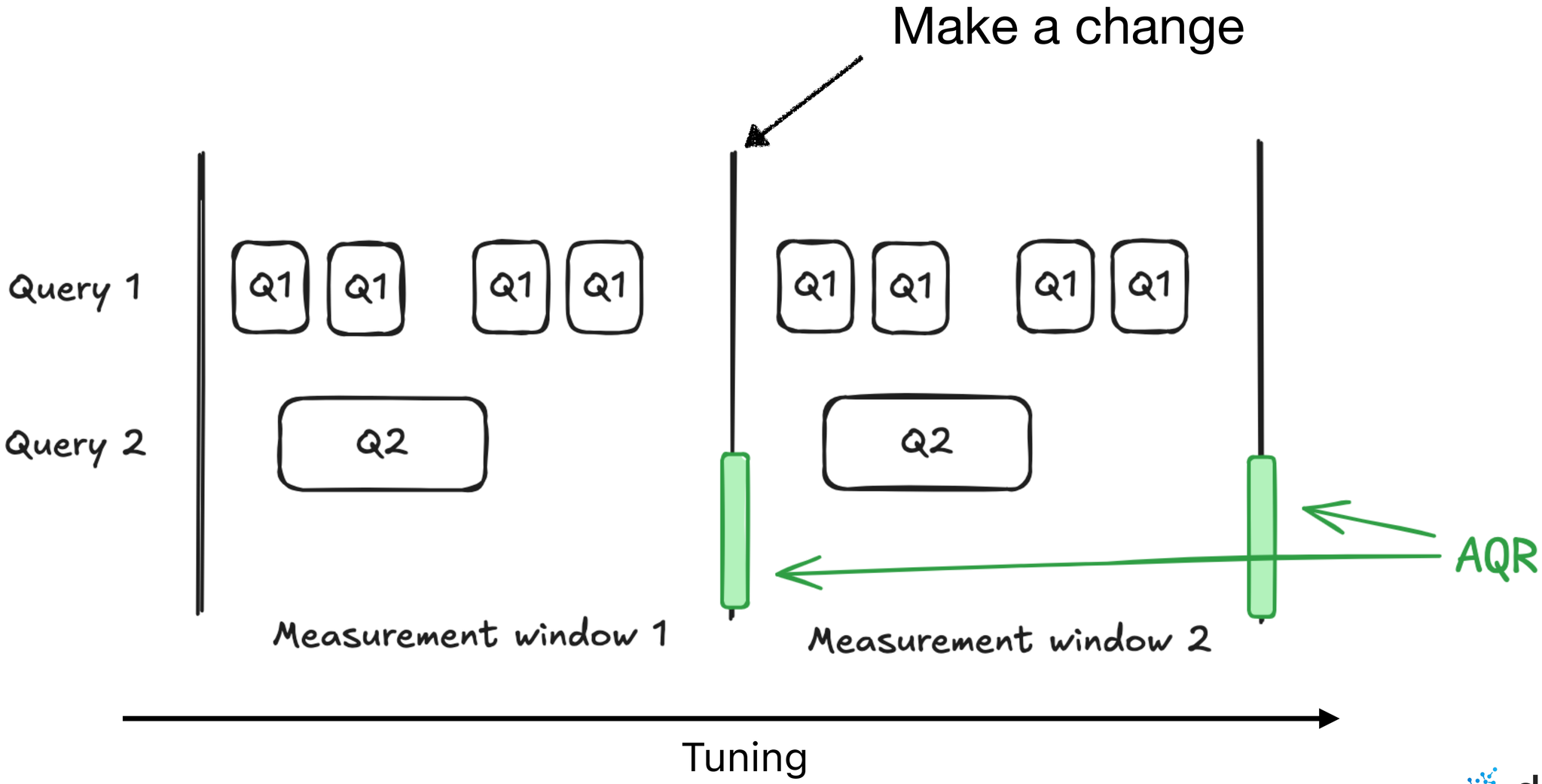
Calculate performance

Common performance metrics

- Latency:
  - ***Average query runtime*** (AQR)
  - Query runtime P99
- Throughput:
  - Transactions per second (TPS)
- Other:
  - Individual query plans
  - Resource usage
  - Etc.

Observe long enough

Measurement windows in a tuning loop



Workload Fingerprint (WF) method to improve the AQR measurement

- Allows a user to focus on what queries they care about
- Creates a robust and comparable measurement — art vs engineering

How?

- Replace AQR over all queries with AQR for the relevant queries
- Define a method that provides the length of the measurement window
- Define a method to measure runtime in the presence of noise

How is it practically done in the industry?

Dr. Marc Linster

“The first thing we have always asked our clients is to give us the 10 most important queries they want to tune”

What makes AQR measurements unstable?

The variability is mainly given by two factors

Problem 1: Variation in query distribution

Problem 2: Variation in individual query runtimes

Problem 1: Variation in query distribution

Average query runtime (AQR) is subject to variability in some workloads

It is measured across all queries run on the system for a given time span

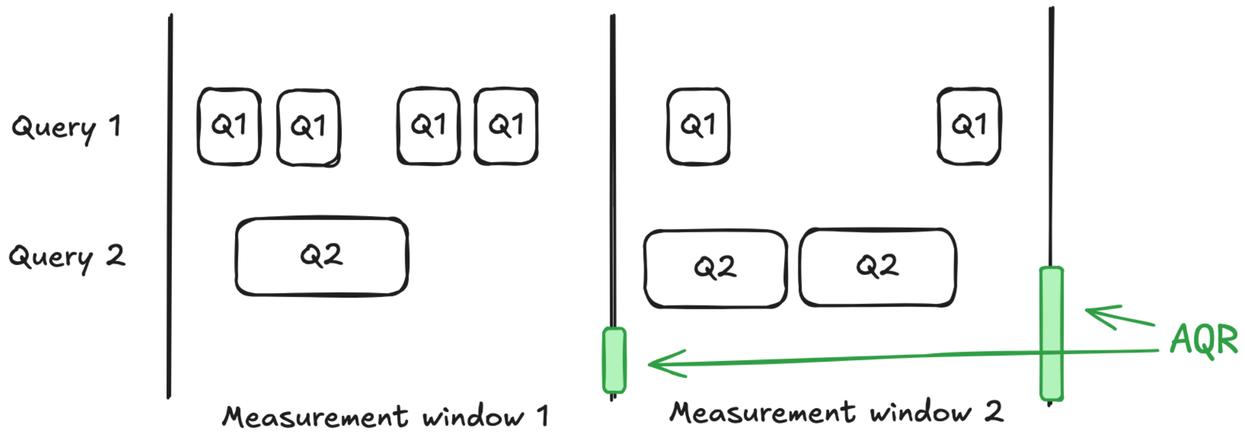
- Many fast queries: low AQR
- Many slow queries: high AQR

AQR depends on the distribution of the queries observed

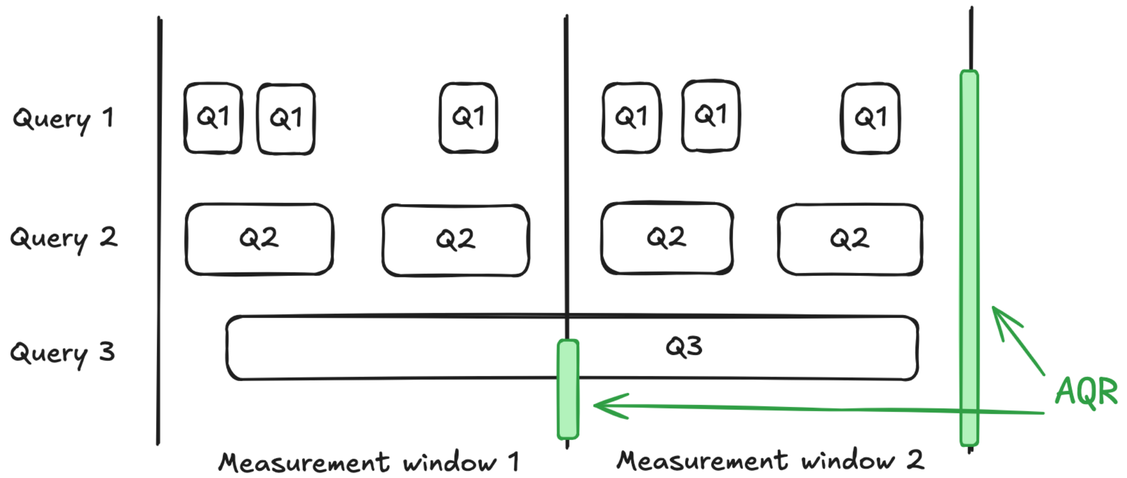
This leads to measurement inaccuracies — examples in the next slide

Problem 1: AQR performance measurement inaccuracies

Example 1: Query distribution has changed between measurement windows



Example 2: Measurement window is too short to capture the query distribution



Problem 1: Workload Fingerprint method when tuning

Old algorithm

1. Make a change
2. Observe "long enough"
3. Calculate AQR
4. Am I satisfied?
5. Go back to 1 or exit tuning

Problem 1: Workload Fingerprint method when tuning (step 1)

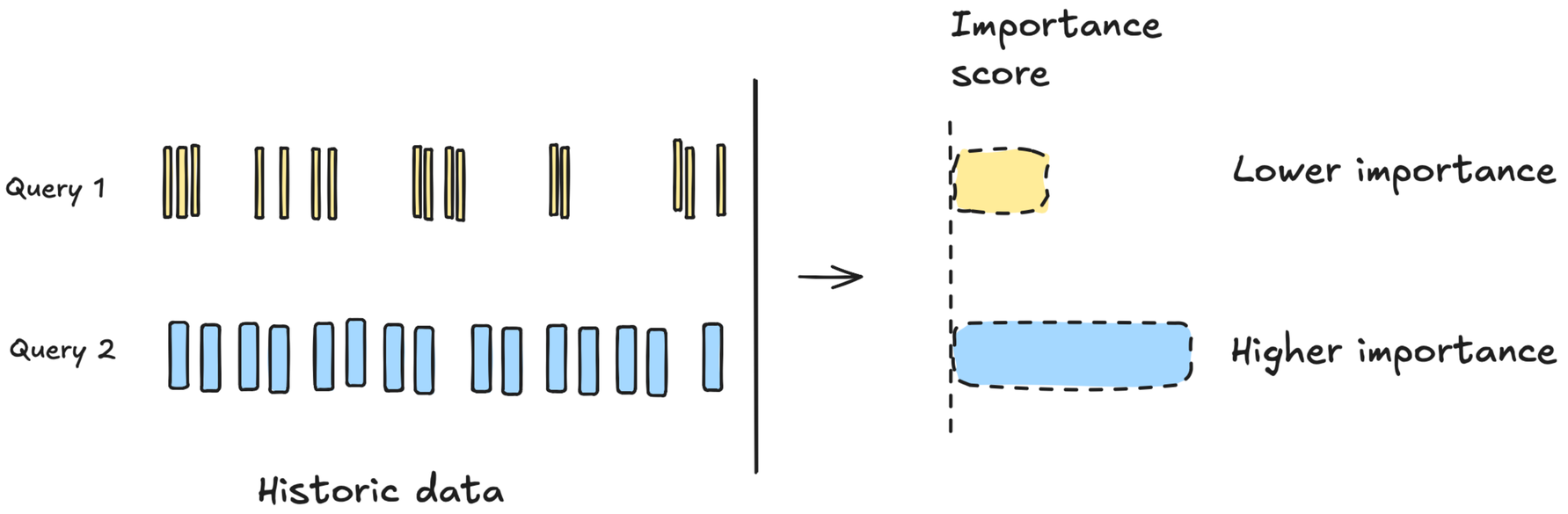
1. Make a change
2. Observe "long enough"
3. **Calculate Workload Fingerprint AQR (next slide)**
4. Am I satisfied?
5. Go back to 1 or exit tuning

Problem 1: How do we calculate the Workload Fingerprint AQR?

Step 1: Define a target workload and the importance of the different queries

Step 2: Calculate the AQR for every query individually

Step 3: Combine the different AQRs according to their importance globally

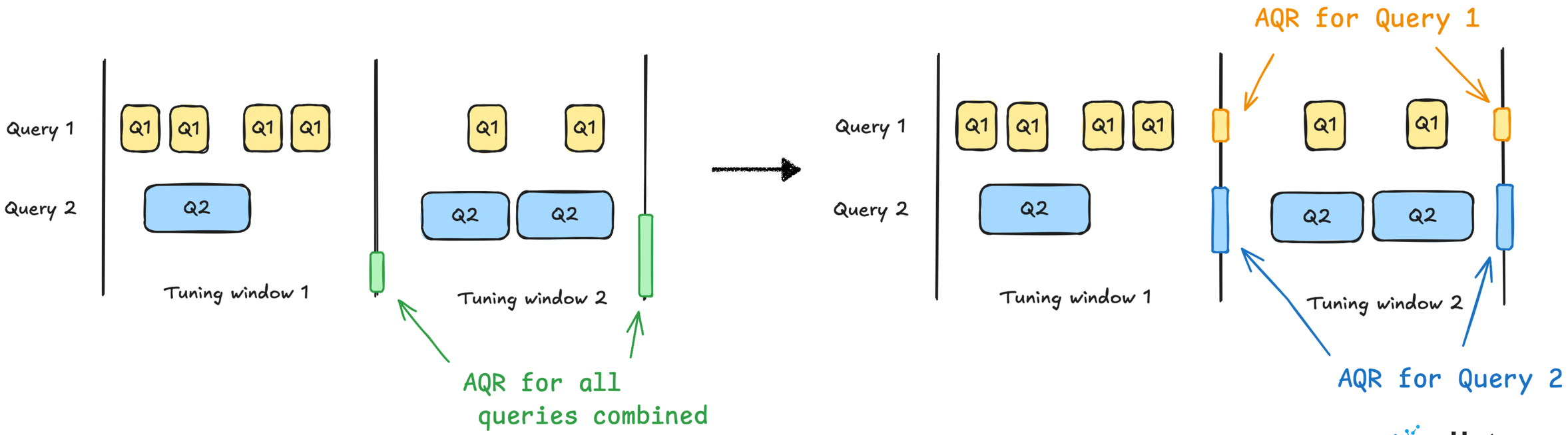


Problem 1: How do we calculate the Workload Fingerprint AQR?

Step 1: Define a target workload and the importance of the different queries

Step 2: Calculate the AQR for every query individually

Step 3: Combine the different AQRs according to their importance globally

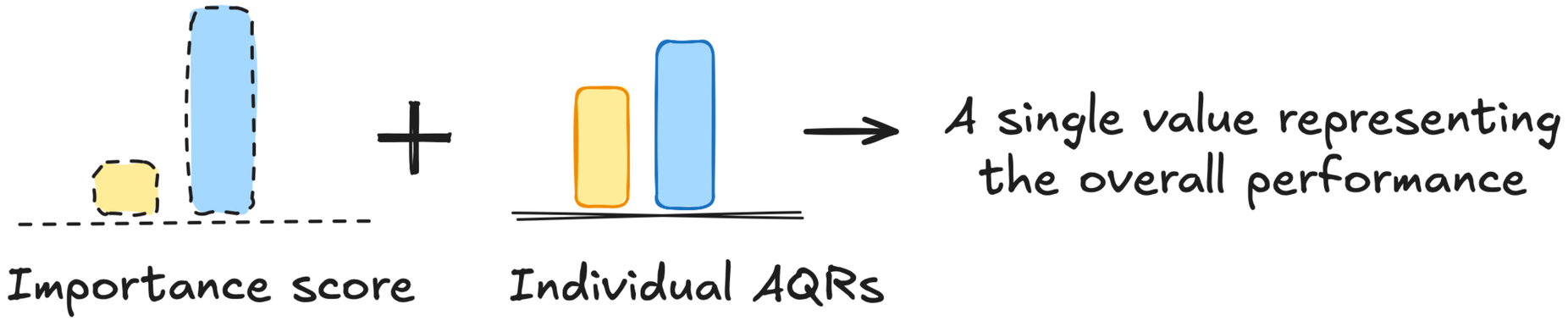


Problem 1: How do we calculate the Workload Fingerprint AQR?

Step 1: Define a target workload and the importance of the different queries

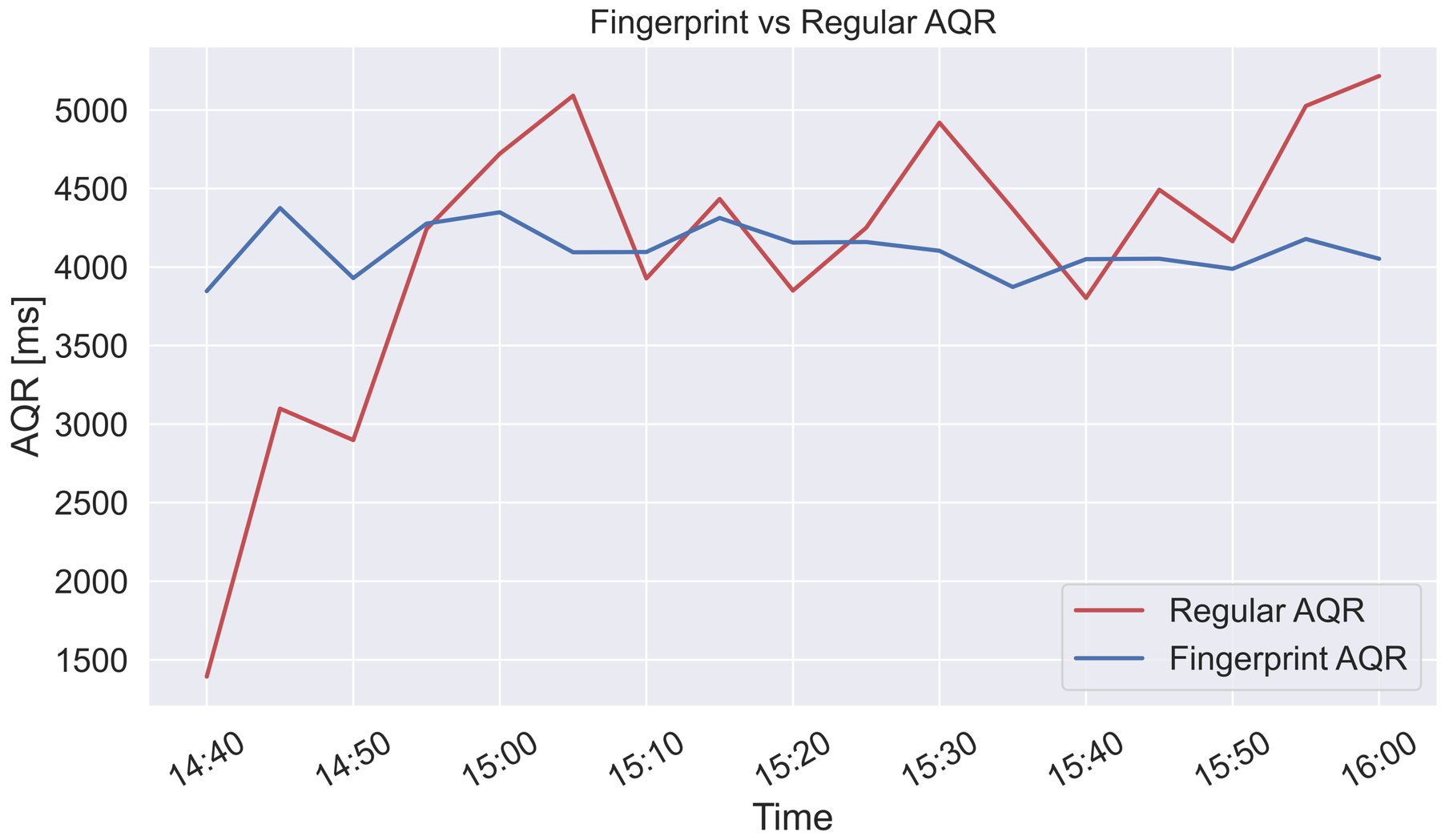
Step 2: Calculate the AQR for every query individually

Step 3: Combine the different AQRs according to their importance globally



Problem 1: A more robust signal outcome against workload variations

AQR vs Workload Fingerprint stability in a prod environment



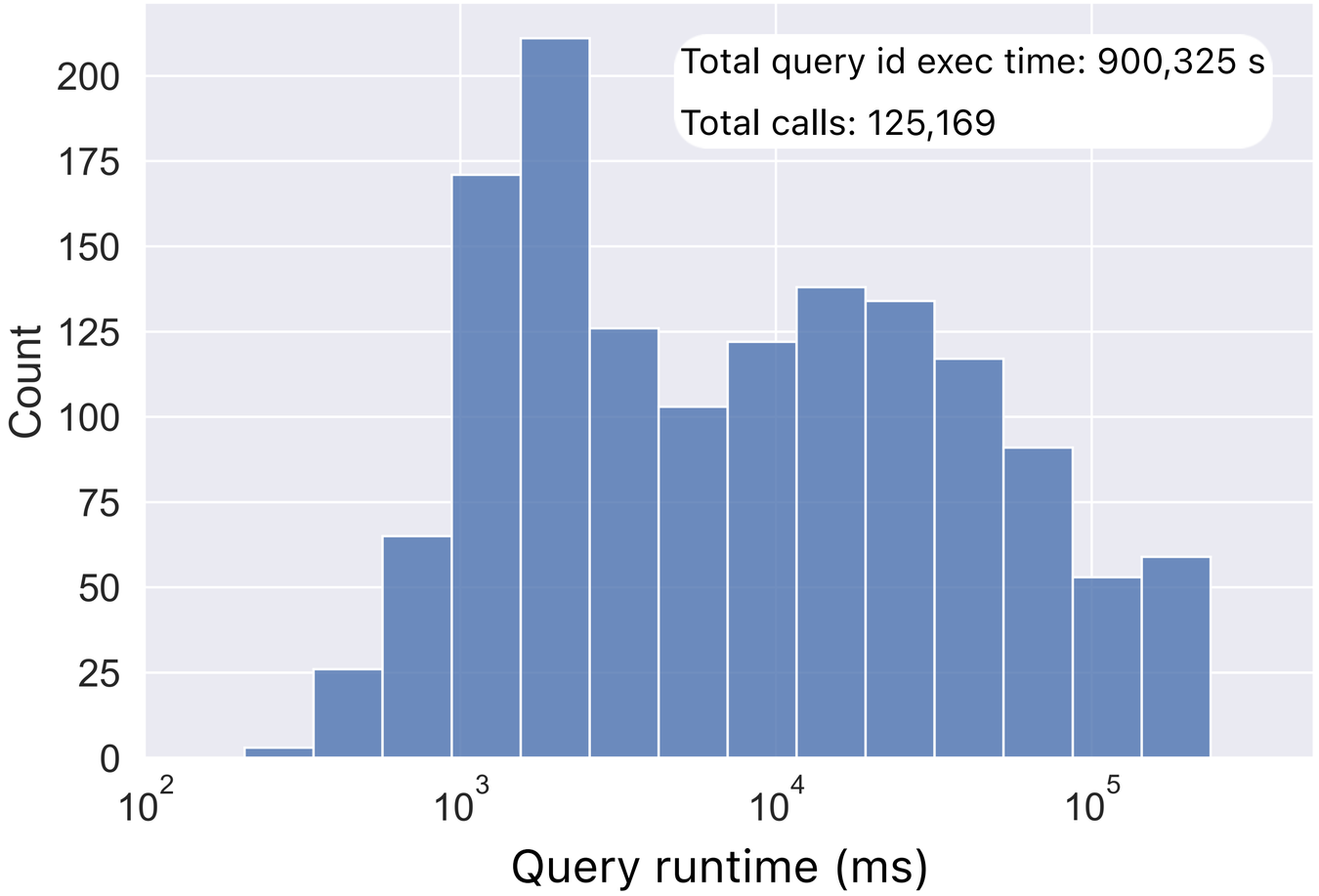
Problem 2: Variation in individual query runtimes

Individual queries also vary in runtime

- Multi-tenancy
- Warm-cold cache effects
- Network effects
- SQL statement bindings (e.g., $1, $2)

To get statistically significant measurements: Observe a query many times

Problem 2: Individual query query runtime distribution



Environment

- production system
- time window = 2h
- unique query IDs = 787;
- total workload exec time = 1,808,598s

Problem: variability

Solution: measure for "long enough" to secure robust statistics

Problem 2: Workload Fingerprint method when tuning (step 2)

1. Make a change

2. Observe ~~"long enough”~~ until Standard Error approaches 0 (next slide)

3. Calculate Workload Fingerprint AQR (step 1)

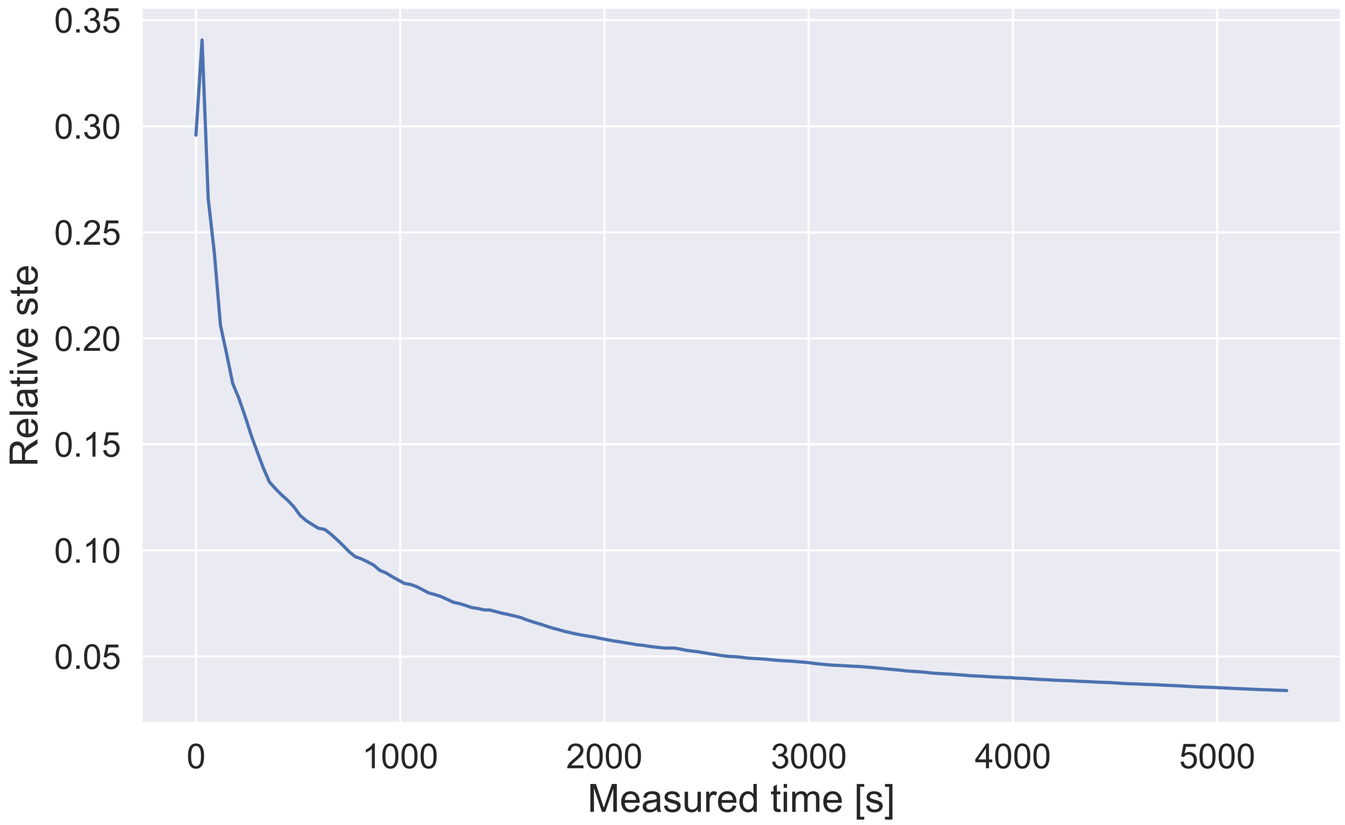
4. Am I satisfied?

5. Go back to 1 or exit tuning

Problem 2: Nailing the single query distribution by measuring long enough

Standard error (Ste) of the mean as a stop criterion

Ste close to 0 then sample mean is a good estimate of the true population mean



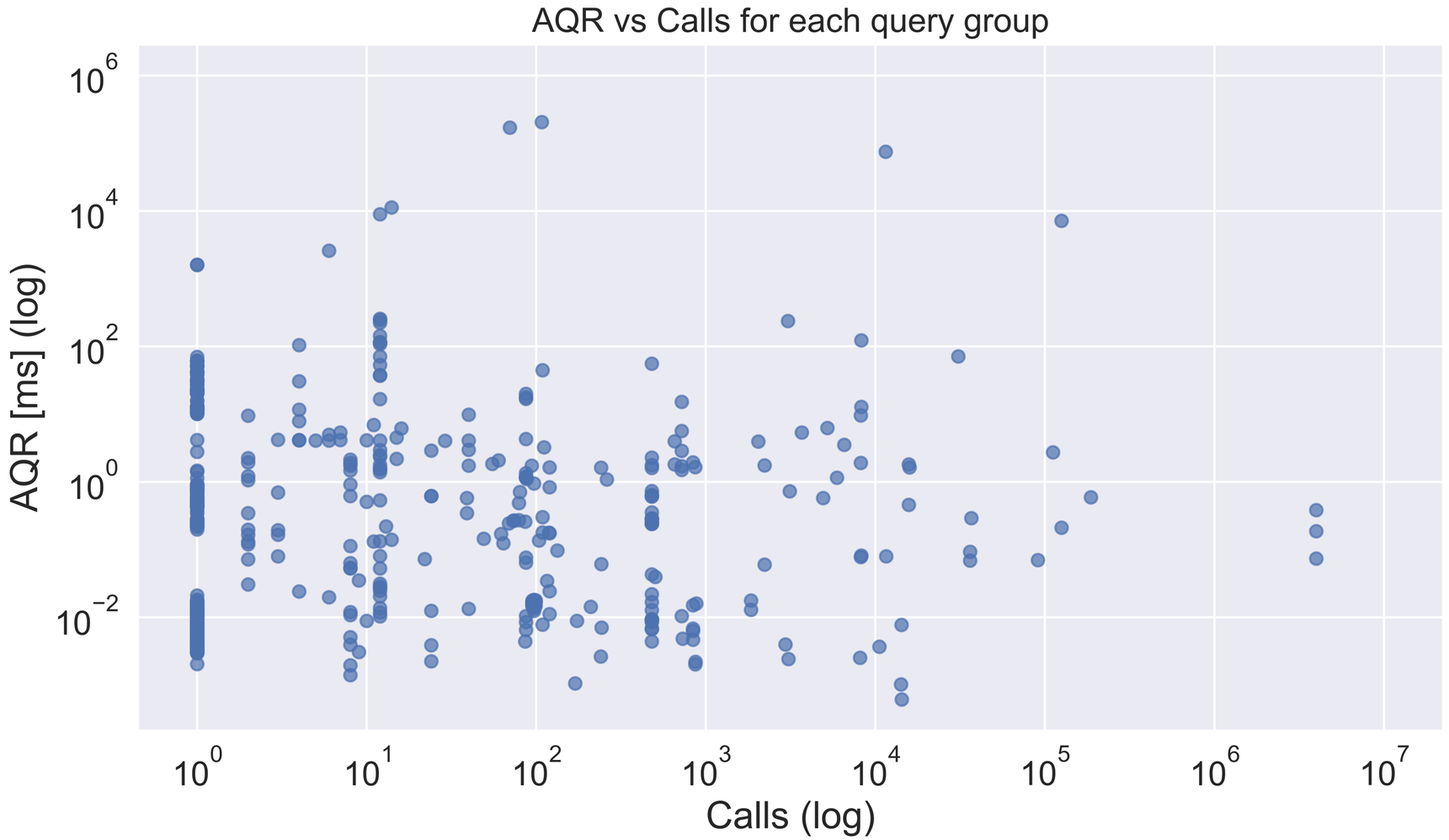
Changing gears:
What is the set of queries we should include in the WF?

What is the agreement between the tuner and the user?

What if you have thousands of queries?

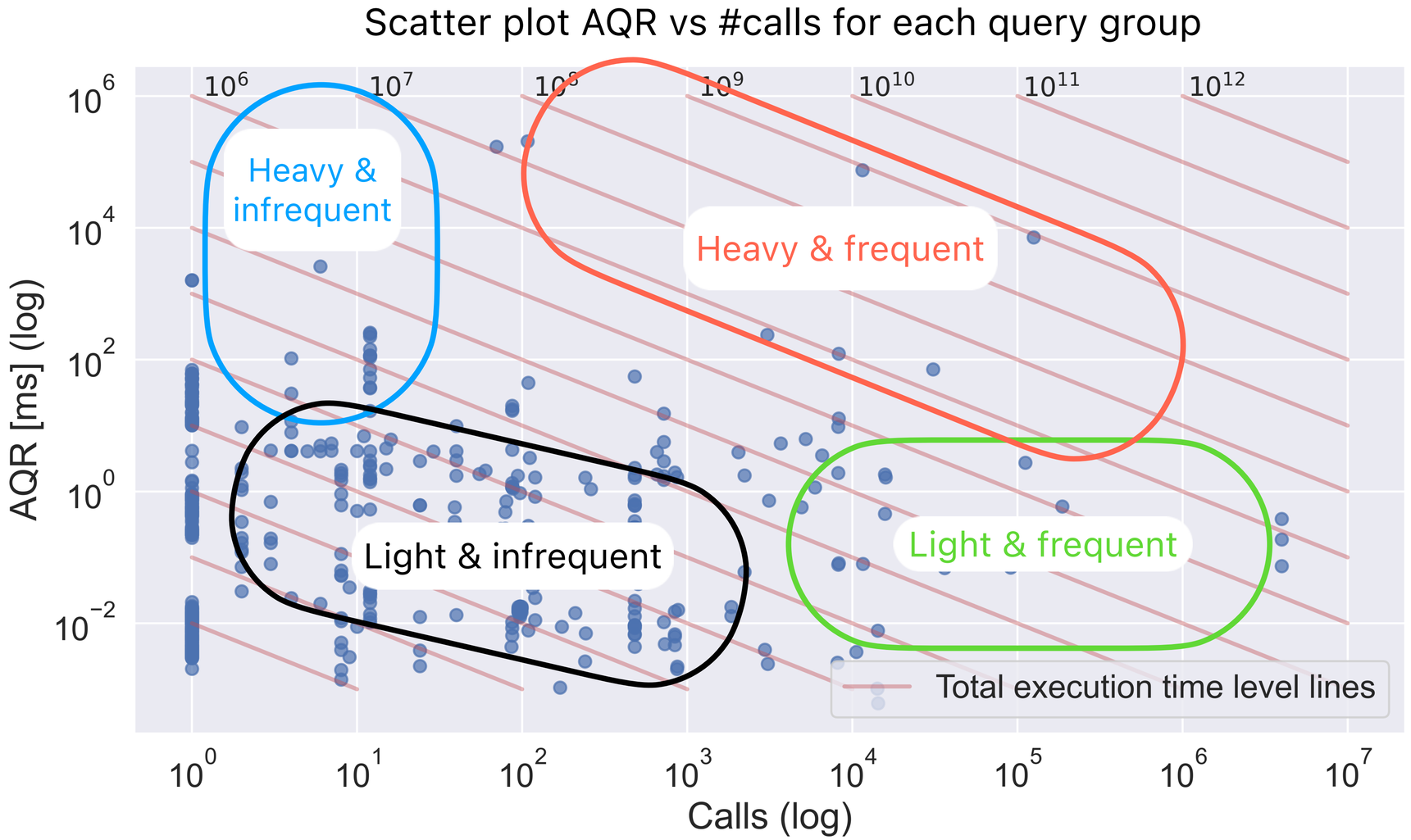
How do we recommend a WF with 100s of query groups?

RDS prod system, time window = 2h; unique query IDs = 787; total exec time = 1,808,598s



How do we recommend a WF with 100s of query groups?

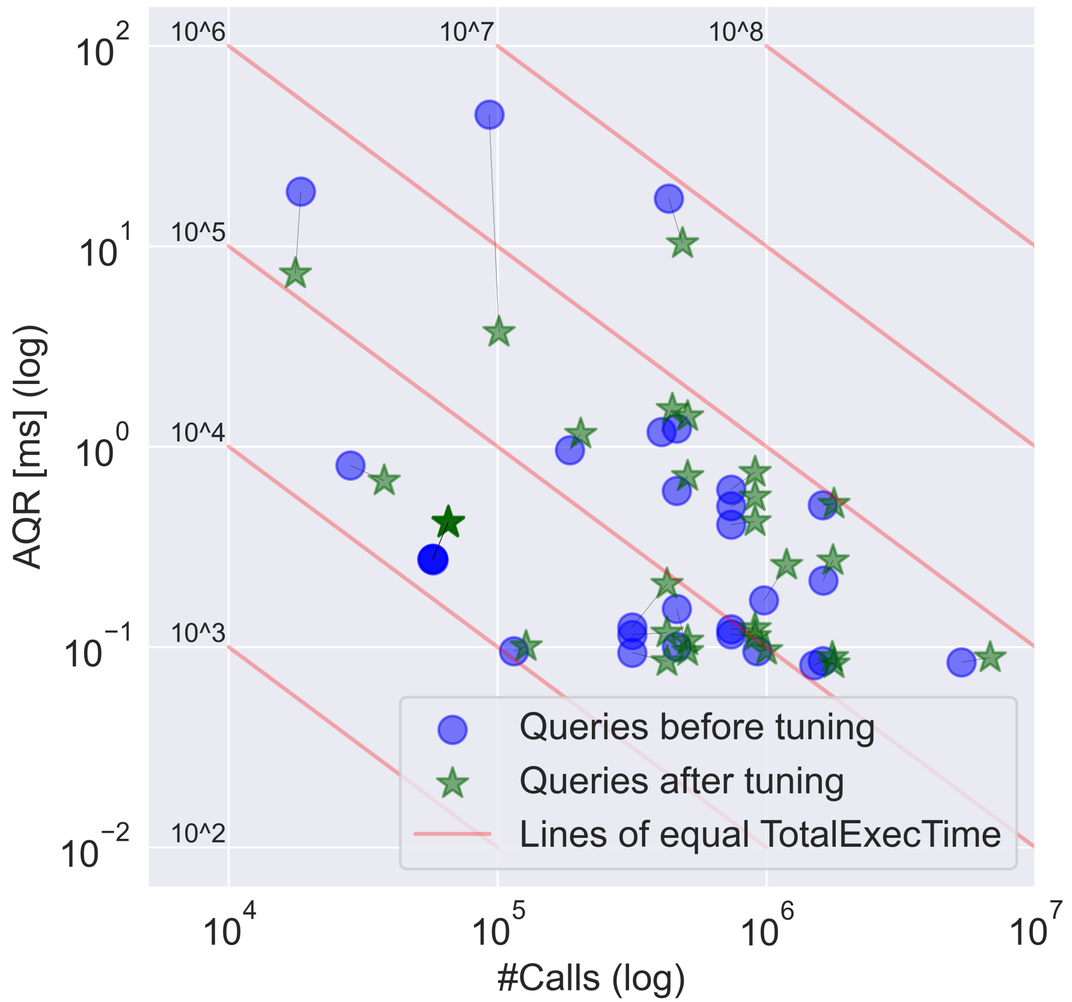
RDS prod system, time window = 2h; unique query IDs = 787; total exec time = 1,808,598s



Server parameter tuning example

Effect of tuning the Workload Fingerprint AQR on the SEATS benchmark

Scatter plot AQR vs #calls for each query group



Time window = 30 minutes

Unique query IDs = 29

Pre-tuning

Total exec time = 17,000 s

Post-tuning

Total exec time = 12,000 s

Putting it all together: The Workload Fingerprint method

- Recommend a subset of queries based on target workload historical data
- Select a subset of queries (using additional app owner recommendations)
- Reduce the uncertainty on the individual query group distribution (using Ste)
- Define the weights based on the target workload
- Aggregate the measurement in a weighted average: WF AQR
- Use WF AQR as a KPI to compare two performance measurements

Conclusions

- We need a better way to measure performance in PostgreSQL
- This presentation translates the art of performance tuning into engineering
- We propose a new method called Workload Fingerprint
- Workload Fingerprint leads to better signal stability
- It defines a clear agreement between the tuner and the app owner

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