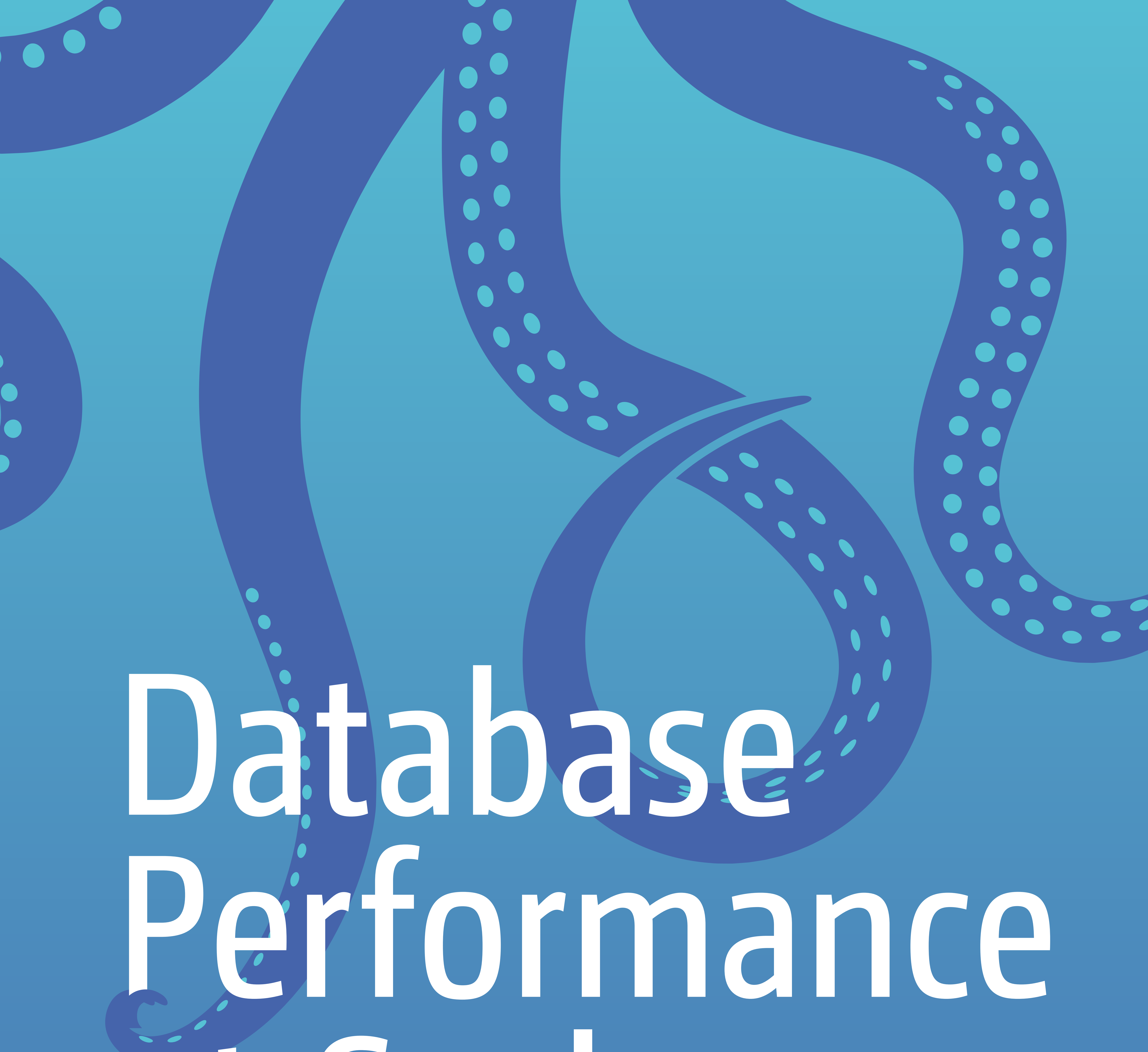

Database Performance at Scale

A Practical Guide

Felipe Cardeneti Mendes · Piotr Sarna
Pavel Emelyanov · Cynthia Dunlop

Apress open

Table of Contents

Introduction

Sisyphean challenge. Gordian knot. Rabbit hole. Many metaphors have been used to describe the daunting challenge of achieving database performance at scale. That isn't surprising. Consider just a handful of the many factors that contribute to satisfying database latency and throughput expectations for a single application:

- How well you know your workload access patterns and whether they are a good fit for your current or target database.
- How your database interacts with its underlying hardware, and whether your infrastructure is correctly sized for the present as well as the future.
- How well your database driver understands your database—and how well *you* understand the internal workings of both.

It's complex. And that's just the tip of the iceberg.

Then, once you feel like you're finally in a good spot, something changes. Your business experiences "catastrophic success," exposing the limitations of your initial approach right when you're entering the spotlight. Maybe market shifts mean that your team is suddenly expected to reduce latency—and reduce costs at the same time, too. Or perhaps you venture on to tackle a new application and find that the lessons learned from the original project don't translate to the new one.

Why Read/Write a Book on Database Performance?

The most common approaches to optimizing database performance are conducting performance tuning and scaling out. They are important—but in many cases, they aren't enough to satisfy strict latency expectations at medium to high throughput. To break past that plateau, other factors need to be addressed.

As with any engineering challenge, there's no one-size-fits-all solution. But there are a lot of commonly overlooked considerations and opportunities with the potential to help teams meet their database performance objectives faster, and with fewer headaches.

As a group of people with experience across a variety of performance-oriented database projects, we (the authors) have a unique perspective into what works well for different performance-sensitive use cases—from low-level engineering optimizations, to infrastructure components, to topology considerations and the KPIs to focus on for monitoring. Frequently, we engage with teams when they're facing a performance challenge so excruciating that they're considering changing their production database (which can seem like the application development equivalent of open heart surgery). And in many cases, we develop a long-term relationship with a team, watching their projects and objectives evolve over time and helping them maintain or improve performance across the shifting sands.

Based on our experience with performance-focused database engineering as well as performance-focused database users, this book represents what we think teams striving for extreme database performance—low latency, high throughput, or both—should be thinking about. We have experience working with multi-petabyte distributed systems requiring millions of interactions per second. We've engineered systems supporting business critical real-time applications with sustained latencies below one millisecond. Finally, we're well aware of commonly-experienced "gotchas" that no one has dared to tell you about, until now.

What We Mean by Database Performance at Scale

Database performance at scale means different things to different teams. For some, it might mean achieving extremely low read latencies; for others, it might mean ingesting very large datasets as quickly as possible. For example:

- **Messaging:** Keeping latency consistently low for thousands to millions of operations per second, because users expect to interact in real-time on popular social media platforms, especially when there's a big event or major news.

- **Fraud detection:** Analyzing a massive dataset as rapidly as possible (millions of operations per second), because faster processing helps stop fraud in its tracks.

- **AdTech:** Providing lightning fast (sub-millisecond P9999 latency) responses with zero tolerance for latency spikes, because an ad bid that's sent even a millisecond past the cutoff is worthless to the ad company and the clients who rely on it.

We specifically tagged on the "at scale" modifier to emphasize that we're catering to teams who are outside of the honeymoon zone, where everything is just blissfully fast no matter what you do with respect to setup, usage, and management. Different teams will reach that inflection point for different reasons, and at different thresholds. But one thing is always the same: It's better to anticipate and prepare than to wait and scramble to react.

Who This Book Is For

This book was written for individuals and teams looking to optimize distributed database performance for an existing project or to begin a new performance-sensitive project with a solid and scalable foundation. You are most likely:

- Experiencing or anticipating some pain related to database latency and/or throughput
- Working primarily on a use case with terabytes to petabytes of raw (unreplicated) data, over 10K operations per second, and with P99 latencies measured in milliseconds
- At least somewhat familiar with scalable distributed databases such as Apache Cassandra, ScyllaDB, Amazon DynamoDB, Google Cloud Bigtable, CockroachDB, and so on
- A software architect, database architect, software engineer, VP of engineering, or technical CTO/founder working with a data-intensive application

You might also be looking to reduce costs without compromising performance, but unsure of all the considerations involved in doing so.

We assume that you want to get your database performance challenges resolved, fast. That's why we focus on providing very direct and opinionated recommendations based on what we have seen work (and fail) in real-world situations. There are, of course, exceptions to every rule and ways to debate the finer points of almost any tip in excruciating detail. We'll focus on presenting the battle-tested "best practices" and anti-patterns here, and encourage additional discussion in whatever public or private channels you prefer.

What This Book Is NOT

A few things that this book is *not* attempting to be:

- A reference for infrastructure engineers building databases. We focus on people working with a database.
- A "definitive guide" to distributed databases, NoSQL, or data-intensive applications. We focus on the top database considerations most critical to performance.
- A guide on how to configure, work with, optimize, or tune any specific database. We focus on broader strategies you can "port" across databases.

There are already many outstanding references that cover the topics we're deliberately not addressing, so we're not going to attempt to re-create or replace them. See Appendix A for a list of recommended resources.

Also, this is not a book about ScyllaDB, even though the authors and technical reviewers have experience with ScyllaDB. Our goal is to present strategies that are useful across the broader class of performance-oriented databases. We reference ScyllaDB, as well as other databases, as appropriate to provide concrete examples.

A Tour of What We Cover

Given that database performance is a multivariate challenge, we explore it from a number of different angles and perspectives. Not every angle will be relevant to every reader—at least not yet. We encourage you to browse around and focus on what seems most applicable to your current situation.

To start, we explore challenges. Chapter 1 kicks it off with two highly fictionalized tales that highlight the variety of database performance challenges that can arise and introduce some of the available strategies for addressing them. Next, we look at the database performance challenges and tradeoffs that you're likely to face depending on your project's specific workload characteristics and technical/business requirements.

The next set of chapters provides a window into many often-overlooked engineering details that could be constraining—or helping—your database performance. First, we look at ways databases can extract more performance from your CPU, memory, storage, and networking. Next, we shift the focus from hardware interactions to algorithmic optimizations—deep diving into the intricacies of a sample performance optimization from the perspective of the engineer behind it. Following that, we share everything a performance-obsessed developer really *should* know about database drivers but never thought to ask. Driver-level optimizations —both how they're engineered and how you work with them—are absolutely critical for performance, so we spend a good amount of time on topics like the interaction between clients and servers, contextual awareness, maximizing concurrency while keeping latencies under control, correct usage of paging, timeout control, retry strategies, and so on. Finally, we look at the performance possibilities in moving more logic into the database (via user-defined functions and user-defined aggregates) as well as moving the database servers closer to users.

Then, the final set of chapters shifts into field-tested recommendations for getting better performance out of your database deployment. It starts by looking at infrastructure and deployment model considerations that are important to understand, whether you're managing your own deployment or opting for a database-as-a-service (maybe serverless) deployment model. Then, we share our top strategies related to topology, benchmarking, monitoring, and admin—all through the not-always-rosy lens of performance.

After all that, we hope you end up with a new appreciation of the countless considerations that impact database performance at scale, discover some previously overlooked opportunities to optimize your database performance, and avoid the common traps and pitfalls that inflict unnecessary pain and distractions on all too many dev and database teams.

Tip Check out our GitHub repo for easy access to the sources we reference in footnotes, plus additional resources on database performance at scale: `https://github.com/Apress/db-performance-at-scale`.