





Database Performance at Scale

A Practical Guide

Felipe Cardeneti Mendes · Piotr Sarna
Pavel Emelyanov · Cynthia Dunlop

Apress
open

Table of Contents

About the Authors	xiii
About the Technical Reviewers	xv
Acknowledgments	xvii
Introduction	xix
Chapter 1: A Taste of What You're Up Against: Two Tales	1
Joan Dives Into Drivers and Debugging	1
Joan's Diary of Lessons Learned, Part I	3
The Tuning	3
Joan's Diary of Lessons Learned, Part II	5
Patrick's Unlucky Green Fedoras	6
Patrick's Diary of Lessons Learned, Part I	7
The First Spike.....	8
Patrick's Diary of Lessons Learned, Part II	8
The First Loss	9
Patrick's Diary of Lessons Learned, Part III	9
The Spike Strikes Again	10
Patrick's Diary of Lessons Learned, Part IV	11
Backup Strikes Back	11
Patrick's Diary of Lessons Learned, Part V	12
Summary.....	13
Chapter 2: Your Project, Through the Lens of Database Performance	15
Workload Mix (Read/Write Ratio)	15
Write-Heavy Workloads	16
Read-Heavy Workloads.....	17

TABLE OF CONTENTS

Mixed Workloads	19
Delete-Heavy Workloads	20
Competing Workloads (Real-Time vs Batch).....	21
Item Size	23
Item Type.....	24
Dataset Size	26
Throughput Expectations	27
Latency Expectations	29
Concurrency	31
Connected Technologies	32
Demand Fluctuations	33
ACID Transactions	34
Consistency Expectations	36
Geographic Distribution	38
High-Availability Expectations.....	39
Summary.....	40
Chapter 3: Database Internals: Hardware and Operating System Interactions	41
CPU	42
Share Nothing Across Cores	42
Futures-Promises	43
Execution Stages.....	45
Memory.....	47
Allocation.....	47
Cache Control	50
I/O	51
Traditional Read/Write	51
mmap	52
Direct I/O (DIO).....	52
Asynchronous I/O (AIO/DIO)	53
Understanding the Tradeoffs	54
Choosing the Filesystem and/or Disk	57

Filesystems vs Raw Disks	57
How Modern SSDs Work.....	58
Networking.....	61
DPDK.....	62
IRQ Binding.....	62
Summary.....	63
Chapter 4: Database Internals: Algorithmic Optimizations	65
Optimizing Collections	66
To B- or Not to B-Tree	66
Linear Search on Steroids.....	68
Scanning the Tree	69
When the Tree Size Matters	70
The Secret Life of Separation Keys.....	72
Summary.....	74
Chapter 5: Database Drivers.....	77
Relationship Between Clients and Servers.....	78
Workload Types.....	79
Throughput vs Goodput	81
Timeouts	83
Client-Side Timeouts	83
Server-Side Timeouts	84
Contextual Awareness.....	86
Topology and Metadata	86
Current Load.....	87
Request Caching.....	88
Query Locality	91
Retries.....	94
Error Categories.....	94
Idempotence.....	95
Retry Policies.....	97

TABLE OF CONTENTS

Paging	100
Concurrency	101
Modern Hardware	102
Modern Software	104
What to Look for When Selecting a Driver	105
Summary	107
Chapter 6: Getting Data Closer	109
Databases as Compute Engines	109
User-Defined Functions and Procedures	110
User-Defined Aggregates	117
WebAssembly for User-Defined Functions	124
Edge Computing	126
Performance	127
Conflict-Free Replicated Data Types	127
Summary	129
Chapter 7: Infrastructure and Deployment Models	131
Core Hardware Considerations for Speed at Scale	132
Identifying the Source of Your Performance Bottlenecks	132
Achieving Balance	133
Setting Realistic Expectations	134
Recommendations for Specific Hardware Components	135
Storage	135
CPUs (Cores)	144
Memory (RAM)	145
Network	147
Considerations in the Cloud	148
Fully Managed Database-as-a-Service	150
Serverless Deployment Models	151
Containerization and Kubernetes	152
Summary	155

Chapter 8: Topology Considerations	157
Replication Strategy	157
Rack Configuration	158
Multi-Region or Global Replication	158
Multi-Availability Zones vs. Multi-Region	159
Scaling Up vs Scaling Out	160
Workload Isolation	162
More on Workload Prioritization for Logical Isolation	163
Abstraction Layers	167
Load Balancing	169
External Caches	170
An External Cache Adds Latency	170
An External Cache Is an Additional Cost.....	171
External Caching Decreases Availability.....	171
Application Complexity: Your Application Needs to Handle More Cases.....	172
External Caching Ruins the Database Caching.....	172
External Caching Might Increase Security Risks	172
External Caching Ignores the Database Knowledge and Database Resources	172
Summary.....	173
Chapter 9: Benchmarking.....	175
Latency or Throughput: Choose Your Focus	176
Less Is More (at First): Taking a Phased Approach.....	180
Benchmarking Do's and Don'ts.....	182
Know What's Under the Hood of Your Database (Or Find Someone Who Knows).....	182
Choose an Environment That Takes Advantage of the Database's Potential	183
Use an Environment That Represents Production	183
Don't Overlook Observability	184
Use Standardized Benchmarking Tools Whenever Feasible	184
Use Representative Data Models, Datasets, and Workloads	185
Exercise Your Cache Realistically	187
Look at Steady State	187

TABLE OF CONTENTS

Watch Out for Client-Side Bottlenecks	188
Also Watch Out for Networking Issues	189
Document Meticulously to Ensure Repeatability	189
Reporting Do's and Don'ts	189
Be Careful with Aggregations	190
Don't Assume People Will Believe You	191
Take Coordinated Omission Into Account	193
Special Considerations for Various Benchmarking Goals	194
Preparing for Growth	194
Comparing Different Databases	195
Comparing the Same Database on Different Infrastructure	195
Assessing the Impact of a Data Modeling or Database Configuration Change.....	195
Beyond the Usual Benchmark.....	196
Benchmarking Admin Operations	196
Testing Disaster Recovery	196
Benchmarking at Extreme Scale	197
Summary.....	199
Chapter 10: Monitoring.....	201
Taking a Proactive Approach.....	201
Tracking Core Database KPIs	203
Database Cluster KPIs	203
Application KPIs.....	207
Infrastructure/Hardware KPIs	209
Creating Effective Custom Alerts	210
Walking Through Sample Scenarios	211
One Replica Is Lagging in Acknowledging Requests.....	211
Disappointing P99 Read Latencies	213
Monitoring Options.....	217
The Database Vendor's Monitoring Stack.....	217
Build Your Own Dashboards and Alerting (Grafana, Grafana Loki).....	218

Third-Party Database Monitoring Tools	218
Full Stack Application Performance Monitoring (APM) Tool.....	218
Summary.....	219
Chapter 11: Administration	221
Admin Operations and Performance	221
Looking at Admin Operations Through the Lens of Performance.....	222
Backups	224
Impacts.....	225
Optimization	226
Compaction	227
Impacts.....	227
Optimization	229
Summary.....	231
Appendix A: A Brief Look at Fundamental Database Design Decisions	233
Index.....	249

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.

INTRODUCTION

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>.
