Tuning the Snowflake Data Cloud

Optimizing Your Data Platform to Minimize Cost and Maximize Performance

Andrew Carruthers

Apress®

Tuning the Snowflake Data Cloud

Optimizing Your Data Platform to Minimize Cost and Maximize Performance

Andrew Carruthers

Apress®

Tuning the Snowflake Data Cloud: Optimizing Your Data Platform to Minimize Cost and Maximize Performance

Andrew Carruthers
Birmingham, UK

ISBN-13 (pbk): 979-8-8688-0378-9 ISBN-13 (electronic): 979-8-8688-0379-6
https://doi.org/10.1007/979-8-8688-0379-6

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Shaul Elson
Development Editor: Laura Berendson
Editorial Project Manager: Gryffin Winkler

Cover designed by eStudioCalamar
Cover image by Silke from Pixabay

Distributed to the book trade worldwide by Apress Media, LLC, 1 New York Plaza, New York, NY 10004, U.S.A. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub (https://github.com/Apress). For more detailed information, please visit https://www.apress.com/gp/services/source-code.

If disposing of this product, please recycle the paper.

For Diane, Esther, Josh, Verity, Evan, Violet, Jordan, and Beth

Table of Contents

About the Author

Andrew Carruthers is the director for Snowflake distribution at the London Stock Exchange Group (LSEG). In this role, Andrew delivers several Snowflake accounts supporting Refinitiv “final mile” data product content delivery via Snowflake Marketplace, Private Listings, and Data Shares. He leads their Center For Enablement (C4E) in developing tooling, best practices, and training.

Previously, Andrew was responsible for the Snowflake Corporate Data Cloud at LSEG, which comprises two Snowflake accounts supporting an ingestion data lake and a consumption analytics hub and services a growing customer base of more than 7,000 end users. He also developed the Snowflake Landing Zone for provisioning Snowflake accounts conforming to both internal standards and best practices.

Andrew has more than 30 years of hands-on relational database design, development, and implementation experience starting with Oracle in 1993. Before joining the London Stock Exchange Group, he operated as an independent IT consultant, predominantly with major European financial institutions. Andrew is considered a visionary and thought leader within his domain, with a tight focus on delivery. Successfully bridging the gap between Snowflake technological capability and business usage of technology, he often develops proofs of concepts to showcase benefits leading to successful business outcomes.

Since 2020 Andrew has immersed himself in Snowflake and is considered a subject-matter expert. He is CorePro certified, contributes to online forums, and speaks at Snowflake events on behalf of LSEG. In recognition of his contribution to implementing Snowflake at LSEG, Andrew received the Snowflake Data Driver award, which recognizes a technology trailblazer who has pioneered the use of the data cloud within their organization.

Andrew has two daughters, both of whom are elite figure skaters. He has a passion for Jaguar cars, having designed and implemented modifications for them, and has published articles for Jaguar Enthusiast and Jaguar Driver. Andrew enjoys 3D printing and has a mechanical engineering workshop with a lathe, milling machine, and TIG welder, to name but a few tools, and enjoys developing his workshop skills.

About the Technical Reviewer

Nadir Doctor is a database and data warehousing architect and a DBA who has worked in various industries with multiple OLTP and OLAP technologies. He has also worked on primary data platforms, including Snowflake, Databricks, CockroachDB, DataStax, Cassandra, ScyllaDB, Redis, MS SQL Server, Oracle, Db2 Cloud, AWS, Azure, and GCP. His major focus is health-check scripting for security, high availability, performance optimization, cost reduction, and operational excellence. He has presented at several technical conference events, is active in user group participation, and can be reached on LinkedIn.

Thank you to Andrew and all the staff at Springer. I'm grateful for the immense support of my loving wife, children, and family during the technical review of this book. I hope that you all find the content enjoyable, inspiring, and useful.

—Nadir

Acknowledgments

Thanks to the Apress team for the opportunity to deliver this book. Specifically, to Nirmal Selvaraj, Shaul Elson, and Mark Powers: thank you for your patient guidance, help, and assistance. Also, Nadir Doctor, thank you for delivering a comprehensive technical review. Your input provided more than just insight and valuable comments: I learned some new things, too. For those unknown to me, including editors, reviewers, and production staff, please take a bow. You are the unsung heroes who have made lightning strike three times (this is my third book with Apress).

To my very dear friend Andy McCann: I am more indebted to you than I can say. Your patience, insight, encouragement, and pragmatic approach provided much needed help and guidance. This book would not be anywhere near as complete or consistent without your input. I owe you more than a few beers.

To my friends at Snowflake who continue to both inspire and spur me on to bigger and better things: Jonathan Nicholson, Will Riley, Cillian Bane, James Hunt, Ben Conneely, and Adrian Randle. Keep on pressing forward; Snowflake is in good hands. Also, thanks to Jiaqi Yan and Minzhen Yang, whose inspiring talk sparked the idea that led to this third book in what has become a series. Little did I know back then just how hard this book would be to investigate, test, and write!

To John Ryan (`https://www.analytics.today/`), who put shape to my thoughts and inspired a section within this book, thank you.

To all my colleagues at London Stock Exchange Group (LSEG), specifically:

- **Corporate Data Cloud:** Nitin Rane, Srinivas Venkata, Matt Willis, Dhiraj Saxena, Bally Gill, Ramya Purushothaman, Radhakrishnan Leela, and Rajan Babu Selvanamasivayam
- **Snowflake Landing Zone:** Nareesh Komuravelly, Nathan Hawes, and Ravi Singh
- **Enterprise Data:** Kevin Whitchurch, Mike Frayne, Sahir Ahmed, Kalpesh Parekh, Matt Adams, Rajen Pather, R.Senthil Kumar, Prosenjit Chattoraj, and Chaitanya Kadiyam

Take a bow. Thank you for your confidence, contribution, support, and help delivering world-class data products into LSEG distribution venues.

To my very dear friends Marco Costella, Martin Cole, Mike Sutherland, Lavkumaar Pandey, and Steve Loosley: keep on doing what you do best. If it isn't broken, don't fix it.

To my family, Esther and Josh; Verity, Evan, and baby Violet; and also Jordan and Beth: thank you. And to my wonderful girlfriend Diane, who continues her Snowflake journey: your smile brightens my day, and your presence makes me whole.

Will there be a fourth book in the series? Possibly. For now, it's time to rest and recharge. Eight months of preparation went into this book. I am not committing to writing a fourth book about Snowflake, though I do have enough material for half a book along with a title. And who knows what will happen after Snowflake Summit 2024?

CHAPTER 1

Tuning the Snowflake Data Cloud

This book continues from where both *Building the Snowflake Data Cloud* (Apress, 2022) and *Maturing the Snowflake Data Cloud* (Apress, 2023) left off. In this new volume, I deep dive into tuning Snowflake queries to deliver blisteringly fast performance along with a concurrent focus on cost-reduction efforts.

I unpack the core principles of how to approach performance optimization from several perspectives.

- Developers migrating existing applications to Snowflake must understand the pitfalls and “gotchas” that await the unwary.
- Cost management in an on-demand environment is a perpetual challenge, and squeezing every drop of performance from Snowflake is imperative.
- Optimizing warehouse size can reduce costs and improve throughput but often treats the symptoms and not the root cause of performance issues.
- Reducing micro-partition churn also reduces both storage and replication costs with the further benefit of reducing propagated data set latency, and I show you how.
- Remediating performance issues and refactoring production code to optimize performance involves trade-offs; there are no silver bullets!
- Updating existing Snowflake implementations to take advantage of new techniques is dependent upon understanding emerging product capabilities.

A. Carruthers, *Tuning the Snowflake Data Cloud*, https://doi.org/10.1007/979-8-8688-0379-6_1

In this book you will learn to develop tools and techniques based upon sound, proven, real-life scenarios. I use these tools and techniques daily, and as you become familiar with them, I hope you will too.

Performance tuning needs to be a continual activity. Data profiles change over time, and `INSERT`, `UPDATE`, and `DELETE` operations can cause skewed data where the distribution of data within a table or database becomes increasingly imbalanced or uneven. The impact of data skew over time can be significant, particularly when it comes to query performance.

All the examples used within this book were developed using a Snowflake trial account available at `www.snowflake.com`. Click the Start For Free button, and enter a few details to start a 30-day free trial account.

For those operating within a corporate environment, select Business Critical Edition because it is most likely the version used by your organization.

All the code samples in this book have been tested using Business Critical Edition and are believed to work with lower editions. You can find further details on Snowflake editions at `https://docs.snowflake.com/en/user-guide/intro-editions`.

I also assume you are familiar with the Snowflake user interface SnowSight (though the examples should work using SnowSQL or Visual Studio configured for Snowflake). You can find further details on SnowSight at `https://docs.snowflake.com/en/user-guide/ui-snowsight`. And for those starting their Snowflake journey for the very first time, start here: `https://docs.snowflake.com/en/user-guide-getting-started`.

I have attempted to divide this book content into readily consumed thematic chapters, and for the curious, the last chapter of this book on "gotchas" summarizes best practices. Before you jump straight to the end of this book, though, please read the intervening chapters as they will give you helpful context.

Last but certainly not least, you can find the Snowflake documentation at `https://docs.snowflake.com/en/`. Reading this book will definitely improve your learning curve; however, there are times where there is no substitute for reading official documentation (which is actually rather good); I will highlight some of it later, but for now, at least you know where it is.

Setting the Scene

I began writing this book in July 2023, a week after Snowflake Summit ended. My head was full of ideas, buzzing with the prospect of writing this book to impart my perspective and available wisdom on performance tuning Snowflake to a wider audience. What struck me was that, in just four years, Snowflake had transitioned from the cloud data warehouse of choice to a much richer and hard-to-define platform encompassing a wide variety of tooling, data formats, and capabilities.

Within this book I do not dive into the ever-expanding Snowflake product capabilities, instead preferring to focus on what some describe as the "black art" of performance tuning. By now, plenty of organizations have both ported applications to Snowflake and/or developed applications on Snowflake from scratch. The time is right for a book on Snowflake performance tuning to extract maximum value from these investments.

It would be too easy to cover what has already been described at an overview level by many vendors, some of whom are offering solutions that treat the symptoms and not the root cause. Conversations supported by Microsoft PowerPoint is one thing; practical techniques supported by hard and fast empirical evidence is entirely another. I prefer to demonstrate pragmatic approaches to resolving performance issues while developing tools to both educate and deliver a firm foundation for you to later build upon.

Snowflake is designed from the ground up to deliver optimal query performance with minimal user intervention. The "out-of-the-box" developer and user experience is truly exceptional, delivering astounding results for both data warehousing applications and, increasingly, much wider use cases including AI/ML applications.

In contrast to a provision-based model where you are constrained by your deployed infrastructure, Snowflake implements a consumption-based model: you pay for what you consume. Typically, provision-based infrastructure is idle for an average of 70 percent to 80 percent of the time, with occasional activity or, more commonly, overloaded activity peaks. In contrast, consumption-based models scale according to demand, providing performance elastically.

But this flexibility comes at a price: scalability and performance cost real money. you must therefore reconsider your approach in a consumption-based model and focus on reducing cost wherever possible. Costs are incurred when you execute code where you consume CPU and memory. In Snowflake parlance, CPU and memory are encapsulated within warehouses. You also incur costs for storage on a per-terabyte basis. At the time of writing, this is a direct pass-through cost from your cloud service provider (CSP). You also incur costs when you replicate data across regions and when you egress data from one CSP to another external location.

Unlike legacy products, Snowflake provides few levers and switches to influence system behavior and application performance, instead preferring to hide complexity to enable developers to focus on delivering business benefit. You might be lulled into a false sense of security by the ease with which you can port your applications into Snowflake, but this can be an expensive mistake.

Tuning the Snowflake Data Cloud is a project-oriented book with a hands-on approach to identifying migration and performance issues with experience drawn from real-world examples. As you work through the examples, you will develop the skill, knowledge, and deep understanding of Snowflake tuning options and capabilities while preparing for later Snowflake features as they become available. Your Snowflake platform will cost less to run and will improve your customer experience.

It is important to note that Snowflake is a constantly evolving product, and therefore best practices will change over time. You should not expect the advice, hints, and tips in this book to be static; this book offers what I know right now, with both eyes on the future.

Regardless of your relational database management system (RDBMS) experience, it's safe to say some of your performance tuning skill, knowledge, and expertise is directly transferable. Equally, some prior learning is not transferable; a degree of unlearning will be required, and for those working on both legacy RDBMS and Snowflake, the operating paradigms are distinctly different.

I next discuss some common themes.

Use Cases for Snowflake

Fundamentally, the underlying CSP storage (whether S3, Azure Blob, or Google Cloud storage) and Snowflake's immutable storage policy dictate the supported transaction style, with data warehousing preferred over online transaction processing (OLTP).

As a general rule of thumb, Snowflake prefers high-volume bulk-load operations supporting analytics workloads. Low-latency, high-volume transactions are not yet common workloads for Snowflake.

The forthcoming Unistore workload joining transactional and analytical data via hybrid tables may change this perception.

Hybrid tables are not yet generally available.

You can find further details on Unistore at https://www.snowflake.com/en/data-cloud/workloads/unistore/.

Rapid data ingest options via data streaming requiring low latency for low-data volume is another common use case. I recommend the "Tour of Ingest" at https://quickstarts.snowflake.com/guide/tour_of_ingest/index.html.

For those looking to understand a much wider suite of Snowflake use cases, please investigate all the various quick starts at https://quickstarts.snowflake.com/.

Provision or Consumption Model

Performance tuning in a provision-based model has fixed constraints; you cannot simply pop down to the data center and plug in more memory or replace your hardware with faster devices. Without preplanned system downtime for upgrades along with the service disruption caused, you are limited to eking out every small performance increment from your existing hardware using any and all levers provided by your operating system vendor, RDBMS vendor, network tooling, and storage vendor. And all of these require deep subject-matter experts (SMEs) in each topic to interact and define optimal patterns for repeatability. Well, that's the intent, but as you all know, reality does not always match expectations.

In sharp contrast, a consumption-based model such as Snowflake removes many historically familiar tuning options and levers; no longer are you able to tune the operating system and change the RDBMS kernel settings. Instead, Snowflake implements a managed service where you pay for what you consume, and this brings about totally different challenges. Gone are the provision-based constraints, but leaving aside the shift to a security focus, which a consumption-based model requires, you replace the provision-based hardware constraints with two new major challenges: cost and performance optimizations.

There is one crucial but often overlooked benefit to adopting a consumption-based model. Snowflake performance has steadily improved since reported performance metrics were first established in August 2022, for two reasons.

- Optimizer performance has steadily been enhanced over time, realizing tangible benefit to overall query execution times.
- CSP hardware replacement programs for obsolete or end-of-life hardware utilize the latest hardware automatically providing performance uplifts.

In August 2022, Snowflake began to record these zero-cost performance benefits. Figure 1-1 illustrates the Snowflake Performance Index, which can be found at https://www.snowflake.com/en/data-cloud/pricing/performance-index/.

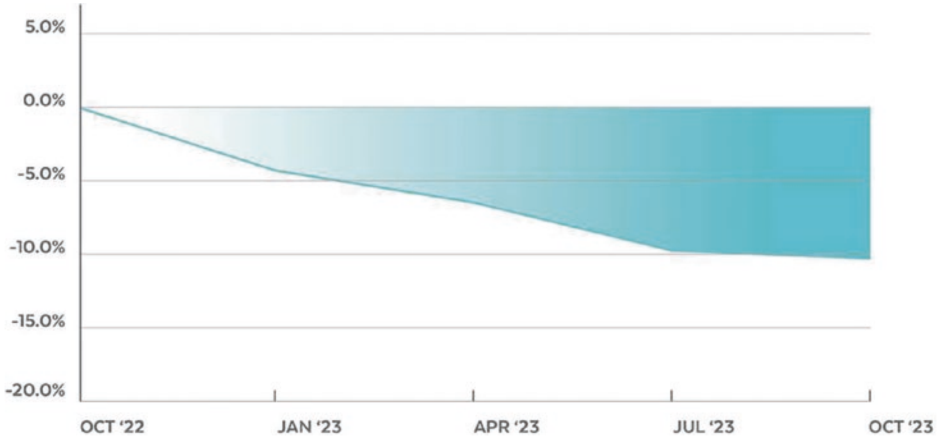

***Figure 1-1.** Snowflake Performance Index*

The trend is set to continue as Snowflake is committed to improving its code base and CSPs periodically replace hardware due to their preventative maintenance policies.

The key takeaway is to periodically monitor your system performance for improvement or degradation over time and take into consideration the probability of Snowflake and CSP changes positively affecting your consumption costs.

Still with me? Good, let's explore common Snowflake starting points (although these are not exhaustive, and your steps may differ).

Refactor or Redesign

Refactoring is the process by which you simplify an existing code base while retaining the original functionality. You might choose to refactor code to take advantage of new performance enhancements, implementing both common design patterns and code structures while improving the overall implementation. Regardless of the rationale for refactoring, the aim is to preserve the original functionality; there should be no discernable behavior differences from the original. Thus, retesting should be as simple as re-running the original test cases utilizing the same inputs.

Refactoring is not intended to address software flaws. It is perfectly valid (and desirable) when refactoring code to improve performance and scalability while preserving the original functionality.

In contrast, redesign may not preserve the original functionality and often modifies, extends, or otherwise improves the functional utility of the component in accordance with the design specification.

Redesign is intended to address software flaws. It is perfectly valid (and desirable) while redesigning code to improve performance and scalability.

Within this book I will use the previous definitions; however, as you will see later, sometimes the boundaries are blurred.

Application Migration to Snowflake

Migration from legacy RDBMS to Snowflake is a common driver to unleash huge performance benefits while moving to CSP infrastructure. I do not discuss in detail "how" to migrate applications to Snowflake nor leverage CSP infrastructure within this chapter, but note these steps are typically performed:

- **Planning:** Developing a project plan incorporating scope, funding, resources, and timeline.
- **Code conversion:** Writing SQL statements, Data Definition Language (DDL), user-defined functions, stored procedures syntax, language conversion.
- **Entitlements:** Refactoring the legacy application security model to use a Snowflake role-based access model (RBAC) model.

- **Data migration:** Porting the application data into Snowflake and establishing ingestion pipelines and processes.
- **Data consumption:** Re-engineering the application outbound data consumption processes.
- **Platform security:** Adding security. I cover this point in great detail in the *Maturing the Snowflake Data Cloud* book.
- **Performance:** Optimizing Snowflake is the core subject matter of this book.
- **Testing:** Perform back-to-back testing to ensure equivalent outputs for known inputs are delivered, along with the all-important and expected performance benefits.
- **Documentation:** No migration activity is complete without exhaustive documentation.

The most time-consuming and difficult step to determine is code conversion; no two applications have the same profile or migration objectives. Migrating an application for archive legacy purposes to retain data for a specific period will be very different from migrating an active, in-use application.

Refactoring code is expensive, and finding empirical metrics is hard. As a rough guideline, you can expect refactoring costs to be at least four times the cost of developing code from scratch. This guesstimate includes understanding the original code; you can substitute your own multiplication factor taking into consideration the availability of experienced resources and detailed documentation.

Another considerable challenge is ensuring your migrated application functionality matches the source application. I call this out as source applications are not typically fixed in time; enhancements and bug fixes cause divergence that must be considered when porting to Snowflake.

Migration Guides

Snowflake offers a number of legacy RDBMS guides to help you port applications to Snowflake. Some of these are listed here and may require your contact information before access is enabled:

- https://www.snowflake.com/wp-content/uploads/2020/05/oracle-to-snowflake-technical-migration-guide.pdf
- https://www.snowflake.com/resource/microsoft-sql-server-to-snowflake-migration-reference-manual/
- https://www.snowflake.com/wp-content/uploads/2020/08/teradata-to-snowflake-migration-guide.pdf
- https://www.snowflake.com/resource/spark-to-snowflake-migration-guide/

Aside from these product specific listings, other migration guides and additional related information are available at https://www.snowflake.com/en/resources/?tags=content-type%2Fmigration-guide&searchTerm=migration.

Migration Options

In this section I will identify some options to migrate applications to Snowflake and later focus on performance and cost optimization.

> Character set conversions require special attention outside of the scope of this book.

SnowConvert

In January 2023 Snowflake acquired SnowConvert from Mobilize.net, a toolkit for migrating customer workloads from legacy RDBMS to Snowflake. SnowConvert automates schema and functional component conversion to Snowflake from a variety of legacy RDBMSs.

Since the Snowflake acquisition, SnowConvert has become the Snowflake Professional Services (PS) tool of choice for application migration. Naturally, you do not have access to SnowConvert directly, but you can find further information at https://www.mobilize.net/.

Manual Schema Conversion

Depending upon your requirements and perceived application code complexity, it is possible to convert schema objects to Snowflake syntax relatively easily. One successfully used approach involves the use of the shell scripts `awk` and `sed` to refactor Data Definition Language to Snowflake syntax. Note this approach does not address performance tuning concerns but does provide a baseline from which to start.

Manual schema migrations are relatively straightforward; however, there are some caveats.

Identifying source character sets can be challenging. Sometimes character set corruption has occurred before data was ingested within the application to be ported; therefore, reconciliation when converted to the Snowflake default UTF-8 character set is impossible.

User-defined types must be reconciled back to their base data types, which in most scenarios will be the supertype rather than subtype. For example, declare `FLOAT`, `DECIMAL`, `MONEY`, `NUMBER` with or without precision, etc.

Some objects do not lend themselves to direct conversion; for example, this nonexhaustive list of Oracle to Snowflake migration challenges will require remediation:

- Snowflake does not support `ROWID`.
- Within tree walks, Snowflake does not explicitly support `LEVEL`.
- Complex materialized views are not directly supported; dynamic tables are an equivalent, but at the time of writing this feature is not generally available.
- Snowflake `NULL` treatment is ANSI compliant; Oracle `NULL` treatment is not.
- Embedded documents are often encoded, encrypted, or compressed using proprietary algorithms.
- Snowflake doesn't have synonyms and relies upon `search_path`.

Likewise, SQLServer to Snowflake migration challenges may be found by doing the following:

- Resolving user-defined types and platform-specific data types to their equivalent Snowflake supertypes
- Using SQL Server syntax that diverges from the ANSI standard