Snowflake Performance Challenges & Solutions (Part 1)

Slim Baltagi Follow 29 min read · Nov 8, 2023

Disclaimer: The opinions in this two-part blog series are entirely mine and do not necessarily reflect my employers (past, present, or future) opinions. This series is neither a critique nor a praise of the Snowflake Data Cloud. I am attempting to cover a complex topic, Snowflake Performance Challenges & Solutions, with the hope of paving the way to open, honest, and constructive related discussions for the benefit of all.

For your convenience, here is the table of contents so you can quickly jump to the item or items that interest you the most.

TABLE OF CONTENTS

1. Misperceptions and confusion from some Snowflake marketing messages
2. Inherent Snowflake performance limitations

2.1. Performance was not originally a major focus of Snowflake

2.2. Out-of-the-box limited concurrency

2.3. Heavy reliance on caching

2.4. Data clustering limitations

2.5. Vertical scaling is a manual process

2.6. Initial poor performance of ad-hoc queries

2.7. No exposure of functionality like query optimizer hints

2.8. Rigid virtual warehouse sizing leads to computing waste and overspending

2.9. Missing out on performance optimizations from Open File Formats

2.10. No uniqueness enforcement

2.11. No full separation of storage and compute

2.12. Heavy scans

2.13. Out-of-the-box concurrent write Limits

2.14. Data Latency

2.15. Inefficient natural partitioning as the data grows

2.16. Shared Cloud Services layer performance degradation

2.17. No Snowflake query workload manager

2.18. Significant delay in getting large output Resultset in Snowflake UI

3. Forced spending caused by built-in Snowflake service offerings

4. Not enough performance tuning tools and services

5. Overall performance degradation due to data, users, and query complexity growth

6. Performance limitations due to cloud generic hardware infrastructure

7. Dependency on cloud providers and risk of services downtime

8. Limitations in the Query Profile

9. Lack of documentation of error codes and messages

III. Snowflake Ecosystem (3)

1. Inefficient integration with Snowflake
2. Best practices knowledge required by third-party tools and services
3. Piecemeal information about Snowflake performance optimization and tuning

IV. Snowflake Users (4)

1. Snowflake users need to get familiar with Snowflake's ways of dealing with performance
2. Snowflake users take shortcuts for solving performance issues
3. Snowflake users lack the knowledge about performance optimization and tuning
4. Snowflake users face a steep learning curve of Snowflake performance optimization and tuning

4.1. Long-running queries

4.2. Complex queries

4.3. Inefficient queries

4.4. Storage Spillage

4.5. Concurrent queries

4.6. Outlier queries

4.7. Queued queries

I. Introduction

You might not be satisfied with the performance of your Snowflake account and need to optimize it for one or all of the following reasons:

1. Data & Users Growth: As the underlying data grows or the number of users querying the data increases, queries might start seeing performance issues.
2. Better end-user experience: Users of executive dashboards and data applications powered by Snowflake would require a better experience.
3. Cost: Striking a balance between performance and cost might be one of your goals unless you have an unlimited budget! Tuning long-running queries, often executed, help reduce cost.
4. Service Level Agreement (SLA): Specific use cases might require to meet SLA, otherwise the business can be negatively impacted. For example, a specific query should return results in less than 10 seconds.
5. Critical path: Queries rely on the results of other queries. For example, queries transforming data or reading data would depend on queries loading data.

You might then need some help with understanding Snowflake's performance challenges and related solutions. That's why I am writing this 2-part blog series!

In this first part, I focus on the performance challenges of Snowflake based on a multidimensional approach that addresses Snowflake technology, Snowflake ecosystem, and Snowflake users. As users, let's not blame the technology if we are misusing or abusing Snowflake! As vendors, let's not blame the technology if our Snowflake integration is not efficient or optimal!

In the second part, I will propose solutions for Snowflake performance tuning and optimization with a unique step-by-step approach based on needs, symptoms, diagnoses, and remediations.

II. Snowflake Technology (6)

1. Misperceptions and confusion from some Snowflake marketing messages

As a SaaS, Snowflake does not require the management of physical hardware, the installation of software, and related maintenance. The Snowflake data platform is continually updated in the background without the need for user involvement. Under the hood, Snowflake takes care of many performance-related aspects that are usually the responsibilities of the customers in other data warehouses and data platforms. Examples include horizontal or vertical data partitioning to specify, data shards for even distribution across nodes, vacuuming, data statistics collection and maintenance, and distribution Keys.

Many misperceptions and confusions about Snowflake performance tuning and optimization are due to claims such as:

- 'With the arrival of the cloud-built data warehouse, performance optimization becomes a challenge of the past'. This is claimed by Snowflake Inc. in this white paper titled 'How Snowflake Automates Performance in a Modern Cloud Data Warehouse' and published on October 16, 2019.
- 'Using Snowflake, everyone benefits from performance automation with very little manual effort or maintenance'. This is claimed by Snowflake inc. in this white paper titled 'How Snowflake Automates Performance in a Modern Cloud Data Warehouse' and published on October 16, 2019.
- Insights into Snowflake's Near-Zero Management a recorded presentation published on January 23, 2020. Here is an example of the reaction of a Snowflake customer to a such statement of 'Zero Management or Near

Zero Management', as reported by a Snowflake employee in his blog: "I was recently working for a major UK customer, where the system manager said 'Snowflake says it needs Zero Management, but surely that's just a marketing ploy'.

- Automatic Query Optimization. No Tuning!, a blog by Snowflake Inc. published on May 19, 2016. Nevertheless, Snowflake offers a USD 3,000 Snowflake Performance Automation and Tuning 3-Day Training!
- Snowflake Data Management — No Admin Required, a recorded presentation by Snowflake Inc. published on January 13, 2020. Nevertheless, Snowflake Inc. offers role-based USD 3,000 'Administering Snowflake Training' and USD 375 'SnowPro Advanced Administrator Certification!
- Pacific Life- Busting Bottlenecks for Data Scientists With 1,800x Faster Query Performance, A case study from Snowflake Inc.
- Scale a near-infinite amount of computing resources, up or down, with just a few clicks. Actually, you might run a query and get the error: "Maximum number of servers for the account exceeded". See the article from Snowflake: Query Failed with Error: Max number of servers for the account exceeded

Such statements not backed by facts from Snowflake Inc. would be considered marketing fluff and do a disservice to Snowflake Data Cloud technology due to customers not being satisfied with them and competition exploiting them.

2. Inherent Snowflake performance limitations

2.1. Performance was not originally a major focus of Snowflake: This is a quote from Snowflake founders in their paper The Snowflake Elastic Data

Warehouse by Snowflake Computing: '… Snowflake has only one tuning parameter: how much performance the user wants (and is willing to pay for). While Snowflake's performance is already very competitive, especially considering the no-tuning aspect, we know of many optimizations that we have not had the time for yet. Somewhat unexpected though, core performance turned out to be almost never an issue for our users. The reason is that elastic compute via virtual warehouses can offer the performance boost occasionally needed. That made us focus our development efforts on other aspects of the system.'

Although Snowflake kept adding new services to improve performance such as Materialized Views, Auto Clustering, Query Acceleration Service, Search Optimization and a few transparent enhancements such as data compression rate for new data loaded in Snowflake, ability to eliminate joins on key columns, … Most either came at additional costs or did not solve many of its inherent performance limitations as evidenced in the below list.

2.2. Out-of-the-box limited concurrency: 8 concurrent queries per warehouse by default. Autoscaling up to 10 warehouses. On a single-cluster virtual warehouse, you might hit a limit of eight concurrent queries. 8 is the default value of the Snowflake parameter MAX_CONCURRENCY_LEVEL that defines the maximum number of parallel or concurrent statements a warehouse can execute. See also the article from Snowflake Knowledge Base Warehouse Concurrency and Statement Timeout Parameters, published on August 16, 2020. Such a low query concurrency limit forces either increasing the size of a single-cluster virtual warehouse or starting additional clusters in the case of a multi-cluster virtual warehouse (in Auto-scale mode). In both cases, this forces you to burn even more credits compared to what default concurrency you get out of the box from Snowflake!

This is an answer from the Snowflake product team that I am posting as is:" We don't have a maximum of 8 concurrent queries: Check out this documentation for more details: Parameters — Snowflake Documentation Note that this parameter does not limit the number of statements that can be executed concurrently by a warehouse cluster. Instead, it serves as an upper-boundary to protect against over-allocation of resources. As each statement is submitted to a warehouse, Snowflake allocates resources for executing the statement; if there aren't enough resources available, the statement is queued or additional clusters are started, depending on the warehouse.

The actual number of statements executed concurrently by a warehouse might be more or less than the specified level:

- Smaller, more basic statements: More statements might execute concurrently because small statements generally execute on a subset of the available compute resources in a warehouse. This means they only count as a fraction towards the concurrency level.
- Larger, more complex statements: Fewer statements might execute concurrently."

2.3. Heavy reliance on caching: Heavy reliance of Snowflake on caching can result in unpredictable and non-optimal performance. At its core, Snowflake is built on a caching architecture which works well on small scale data sets or repetitive traditional queries. This architecture starts to fall down as data volumes expand or the workload complexity increases. In the emerging space of advanced analytics, where machine learning, artificial intelligence, graph theory, geospatial analytics, time-series analysis, adhoc analysis, and real-time analytics are becoming predominant in every enterprise — data sets are typically larger and workloads are becoming much more complex.

This is recognized by Snowflake Inc! “Since end-to-end query performance depends on both cache hit rate for persistent data files and I/O throughput for intermediate data, it is important to optimize how the ephemeral storage system splits capacity between the two. Although we currently use the simple policy of always prioritizing intermediate data, it may not be the optimal policy with respect to end-to-end performance objectives.” Excerpt from this presentation and related paper titled ‘Building An Elastic Query Engine on Disaggregated Storage’ and published in February 2020. PDF (15 pages), Video (19' 57")

2.4. Data clustering limitations: Time-based data is loaded by Snowflake in natural ingestion order and helps gain performance benefits, by eliminating unnecessary reads through the combination of automatically creating micro-partitions to hold data in them and automatically capturing statistics, without any further action required from the user. Such behavior is not guaranteed to happen if you are loading your data in a random sequence or using multiple parallel load processes.

Oftentimes, the natural ingestion order is not the optimal physical ordering of data for customer workload patterns. The user can define a set of key(s) to create a clustered table where the underlying data is physically stored in the order of a user-defined set of key(s). Selecting proper clustering keys is critical and requires an in-depth understanding of the common workloads and access patterns against the table in question. Once the user selects the proper keys, he can benefit from performance gains through the Snowflake automatic clustering service.

Snowflake automatic clustering comes with some limitations. Examples include cost-ineffectiveness for tables that change frequently and clustering

jobs that are not always smart. Snowflake is still improving and optimizing the automatic clustering service but did not publish the related roadmap.

2.5. Vertical scaling is a manual process: Vertical scaling or scaling up and down by resizing a virtual warehouse is a manual process. It is also a common misconception to think that the only solution available to improve query performance is to scale up to a bigger warehouse!

2.6. Initial poor performance of ad-hoc queries: When there is a need to answer questions not already solved with predetermined or predefined queries and datasets, users create Ad-hoc queries. For example, analysts might write ad-hoc queries for immediate data exploration needs that tend to be heavy. Most of the time, analysts might not know what virtual warehouse size to use, how to tune these queries, or whether it does make sense to tune them.

Related best practices would be to:

- Isolate such ad-hoc queries by using a separate virtual warehouse to prevent them from affecting the performance of other workloads.
- Run such ad-hoc queries on bigger warehouses! They will end up running faster and the cost would be the same compared to running slower in smaller warehouses.
- Use Snowflake Query Acceleration Service (QAS), a feature built into all Snowflake Virtual Warehouses, to improve Ad-hoc query performance by offloading large table scans to the QAS service.

2.7. No exposure of functionality like query optimizer hints: Snowflake does not expose functionality like query optimizer hints that are found in other

databases and data warehouses to control the order in which joins are performed for example.

In some situations, it is not possible for Snowflake optimizer to identify the join ordering that would result in the fastest execution. You might need to rewrite your query using an approach that guarantees that the joins are executed in your preferred order. See this Snowflake Knowledge Base article that is published on May 28, 2019 and titled How To: Control Join Order

This is an answer from the Snowflake product team that I am posting as is: "This is not a limitation but a design of Snowflake, to avoid common problems associated with join order hints such as the query (including joining many tables) are going to have their join order forced and this render the query very brittle and fragile. if the underlying data changes in the future, you could be forcing multiple inefficient join orders. Your query that you tuned with join order could go from running in seconds to minutes or hours."

Update: Join elimination is a new feature in Snowflake that takes advantage of foreign key constraints. In a way, this is the first optimizer hint in Snowflake! When two tables are joined on a column, you have the option to annotate those columns with primary & foreign key constraints using the RELY property. Setting this property tells the optimizer to check the relationship between the tables during query planning. If the join isn't needed, it will be removed entirely.

2.8. Rigid virtual warehouse sizing leads to computing waste and overspending: Snowflake virtual warehouses come in fixed sizes that must be manually scaled to the next instance doubling size and cost to match query complexity. For example, a Large virtual warehouse (8 nodes) would

need to go to an X-Large virtual warehouse (16 nodes), even if meeting current query complexity demands would require only one more node. Relying on fixed warehouse sizing that doubles warehouse sizes and costs every time a little more query performance is needed leads to computing waste and overspending.

2.9. Missing out on performance optimizations from Open File Formats: Snowflake users might miss out on performance optimizations that common Open file formats such as Apache Arrow, Apache Parquet, Apache Avro, and Apache ORC offer. Snowflake can import data from these Open formats to its proprietary file format (FDN: Flocon De Neige) but can not directly work with these open file formats.

With the announcement of the new Snowflake feature called Iceberg table format, Snowflake is adding support for the Apache Parquet file format. Iceberg Tables are in a private preview as of the publishing date of this article and are not publicly available to Snowflake customers yet. Snowflake announced on January 21, 2022, expanded support for Iceberg via External Tables. At the Snowflake Summit on June 14, 2022, Snowflake announced a new type of Snowflake table called Iceberg Tables: “In this 6 minutes demo, Snowflake Software Engineer Polita Paulus shows you how a new type of Snowflake table, called an Iceberg Table, extends the features of Snowflake’s platform to Open formats, Apache Iceberg and Apache Parquet, in storage managed by customers. You can work with Iceberg Tables as you would with any Snowflake table, including being able to apply native column-level security, without losing the interoperability that an open table format provides.”

You might wonder what is the difference between support of Snowflake of Apache Iceberg via External Tables or Iceberg Tables. External tables are

read only while Iceberg tables allow read, insert and update.

2.10. No uniqueness enforcement: There is no way to enforce uniqueness in inserted data. If you have a distributed system and it writes data on Snowflake, you will have to handle the uniqueness yourself either on the application layer or by using some method of data de-duplication.

Snowflake announced Unistore, not yet in public preview. This means Snowflake has a new Hybrid Table Type that allows Unique, Primary, and Foreign Key constraints. Here's a 6-minute youtube demo.

2.11. No full separation of storage and compute

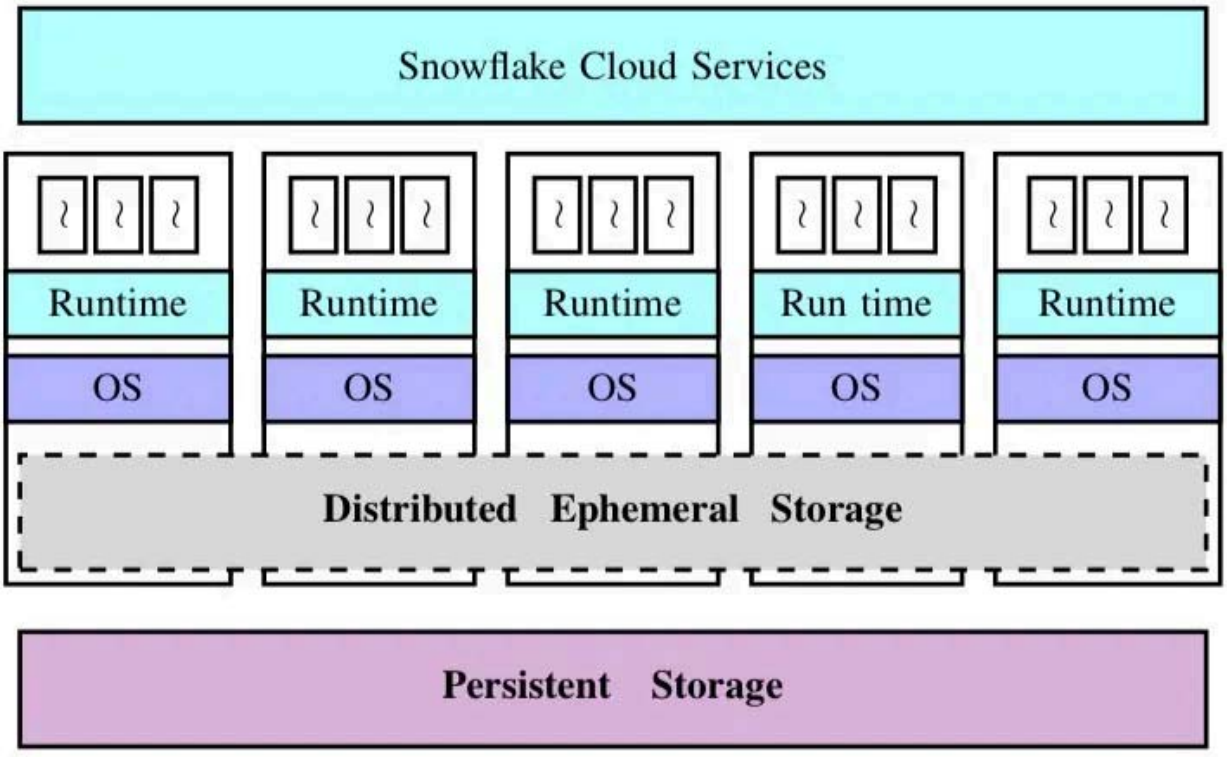


Figure 1: **Snowflake (Virtual) Warehouse Architecture (§2.2)**

“Since end-to-end query performance depends on both cache hit rate for persistent data files and I/O throughput for intermediate data, it is important to optimize how the ephemeral storage system splits capacity between the two. Although we currently use the simple policy of always prioritizing intermediate data, it may not be the optimal policy with respect to end-to-end performance objectives.” Excerpt from this presentation and related paper titled ‘Building An Elastic Query Engine on Disaggregated Storage’ and published in February 2020. PDF (15 pages), Video (19' 57")

2.12. Heavy scans: Before copying data, Snowflake checks that files have not already been loaded. This will affect load performance. To avoid scanning terabytes of files that have already been loaded, you can simply partition your staged data files! This will help maximize your load performance.

2.13. Out-of-the-box concurrent write Limits: Snowflake has a built-in limit of 20 DML statements that target the same table concurrently, including COPY, INSERT, MERGE, UPDATE, and DELETE. Snowflake users might not be aware of such a concurrent write limit as this is not in Snowflake documentation. The root cause seems to be related to constraints in the Global Service Layer’s metadata repository database. You need to be aware that Snowflake is designed for high-volume high-concurrency reading and not writing.

To overcome such a performance challenge of concurrent write limits in Snowflake, you need to architect around it as explained in this article published on July 26, 2019.

2.14. Data Latency: Latency issues in Snowflake can be due to many reasons. Examples include Snowflake and other data services not in the same cloud region. To avoid such latency issues, you’d need to have Snowflake on the

same public cloud and the same or closer cloud region where your data, users, and other services reside.

2.15. Inefficient natural partitioning as the data grows: By default, Snowflake auto clusters your rows according to their insertion order. Because insertion order is often correlated with dates, and dates are a popular filtering condition, this default clustering works quite well in many cases. However, when you have too many DML statements after your initial data load, your clustering may lose its effectiveness over time: more overlap means fewer pruning opportunities. Using SYSTEM$CLUSTERING_DEPTH system function helps with seeing how effective your micro-partitions are in terms of a specified set of columns. It returns the average depth as an indicator of how much your micro-partitions overlap. The lower this number, the less overlap, the more pruning opportunities, and the better your query performance when filtering on those columns.

2.16. Shared Cloud Services layer performance degradation: Snowflake shared Cloud Service Layer in a particular region can become overloaded.

This is a real-world interaction with Snowflake Customer Service with their analysis of why our metadata queries either were extremely slow or timed out: “Our team has reviewed the information provided. We found on September 1, 2022 there was an issue with our Cloud Services servers encountering heavier than normal usage. As a result, some queries had timeouts reading metadata resulting in incidents. This issue with the servers was identified by the Snowflake team during the event and steps were taken to resolve the issue. We apologize for the inconvenience. Thanks”. A potential solution is for Snowflake to predictively scale Cloud Services clusters prior to hitting any resource limits.

In addition, an outage of the shared Cloud Services will affect multiple customers. Getting technical support will become a big challenge.

2.17. No Snowflake query workload manager: Unlike other data warehouses, Snowflake lacks a resource manager to assign resources to queries based on their importance and overall system workload.

2.18. Significant delay in getting large output Resultset in Snowflake UI: You might experience query slowness due to fetching a large output resultset.

You can set this parameter UI_QUERY_RESULT_FORMAT to ARROW at the session/account/user level and test if you are getting the results faster:

alter session set UI_QUERY_RESULT_FORMAT='ARROW';

This post, How To: Resolve Query slowness due to large output published on March 28, 2022, provided the solution to avoid query slowness due to fetching large output resultset.

3. Forced spending caused by built-in Snowflake service offerings

The use of performance-related additional services from Snowflake forces a cost increase for users and challenges them to strike a balance between performance and cost:

3.1. Automatic Clustering Service: Reorganize table data to align with query patterns. Clustering in Snowflake relates to colocating rows with other similar rows in a micro partition. Data that is well clustered can be queried faster and more affordably due to partition pruning that allows Snowflake to

skip data that does not pertain to the query based on statistics of the micro partitions. When clustering is defined, the Automatic Clustering service, in the background, rewrites micro-partitions to group rows with similar values for the clustering columns in the same micro-partition.

Snowflake users might make some clustering choices that don't have any performance gains and are a waste of spending:

DO NOT: Cluster by a timestamp when most predicates are by DAY.

DO NOT: Cluster on a VARCHAR field that has low cardinality prefix

DO NOT: Try to compensate for starting character cardinality with a HASH function

DO NOT: Change clustering in production on a table w/o a manual rewrite

This quick write-up Tips for Clustering in Snowflake will help you understand what clustering is on Snowflake, Why it is important, How to pick cluster keys, and what not to do.

3.2. Materialized Views: To boost query performance for workloads that contain a lot of the same queries, you might create materialized views to store frequently used projections and aggregations. Although the results from related materialized views queries are guaranteed to be up-to-date, there are extra costs associated with those materialized views. As a result, before you create any materialized views, think about whether the expenses will be compensated by the savings from reusing the results frequently enough.

3.3. Search Optimization Service: To quickly find the needle in the haystack to return a small number of rows on a large table, you might enable search optimization on that table. The maintenance service begins constructing the table's search access paths in the background and might massively parallelize related job. This might result in rapid increase in spending.

Before enabling search optimization on a large table, you can get an estimate of the spending using SYSTEM$ESTIMATE_SEARCH_OPTIMIZATION_COSTS so you know what you're in for.

3.4. Query Acceleration Service: Elastic scale out of computing resources without changing the size of virtual warehouses. See What is Snowflake Query Acceleration Service?

4. Not enough performance tuning tools and services

Despite already celebrating its 10th anniversary, Snowflake is considered relatively new and evolving fast compared to legacy data warehouses and platforms that have matured over decades and provided fine-grained tuning tools that DBAs typically use.

Although Snowflake features for investigating slow Snowflake performance issues, such as query profile, query history page, and warehouse loading charts, offer valuable data and insights, there is a need for more tuning tools and services. For example, some additional tools that might be helpful would be Clustered Tables Explorer & Analyzer, Warehouse Idle Time Alerter, …

5. Overall Performance degradation due to data, users, and query complexity growth

Queries might start seeing performance issues as underlying data grows or the number of users querying data increases. Although performance degradation is not specific to Snowflake, users find it difficult to fix it in Snowflake with limited tuning options. They lack features, such as indexes and workload managers, they are used from other data platforms and tools to investigate and fix Snowflake performance issues.

6. Performance limitations due to cloud generic hardware infrastructure

Snowflake runs on generic and general-purpose hardware infrastructure available in AWS, GCP, or Azure. Such cloud hardware limits Snowflake's performance as it is not optimized for data warehousing workloads and other workloads that Snowflake supports.

In addition, unlike competitors, Snowflake does not offer any performance tuning through hardware selection. Snowflake does not disclose its hardware specs to allow customizing performance through hardware. Such lack of transparency of Snowflake about its hardware specs reduces the flexibility in customizing Snowflake performance through hardware beyond simply selecting a virtual warehouse size without knowing underlying details or having any other hardware-related choices.

At its 2022 Summit, Snowflake announced 'Transparent engine updates': "For those of you running on AWS, you will get faster performance for all of your workloads. We've optimized Snowflake to take advantage of new hardware improvements offered by AWS, and we are seeing 10% faster compute on average in the regions already rolled out. No user intervention or choosing a particular configuration is required for this latest performance enhancement." See blog post published on July 27, 2022 and titled Snowflake's New Engine and Platform Announcements

Snowflake announced at its 2022 summit, large memory instances for ML workloads to enable the training of models inside Snowflake. See demo titled 'Train And Deploy Machine Learning Models With Snowpark For Python' and published on June 14, 2022

7. Dependency on cloud providers and risk of services downtime

Snowflake depends on many services provided by cloud providers such as AWS, GCP, and Azure. When such services suffer a performance degradation or go down, this directly affects your Snowflake account.

Downtime is a disadvantage of cloud computing as cloud services, including Snowflake, are not immune to such outages or slowdowns. That is an unfortunate possibility and can occur for any reason. Your business needs to assess the impacts of an outage, slowdown, or any planned downtime from Snowflake, follow best practices for minimizing impact or slowdowns and outages, and implement solutions such as Business Continuity and Disaster Recovery plans. You can subscribe to Snowflake status updates here.

8. Limitations in the Query Profile

Snowflake offers Query Profile to analyze queries. In Snowflake SnowSight UI, in the Query Profile view, there is a section called Profile Overview where you can see the breakdown of the total execution time. It contains statistics like Processing, Local Disk I/O, Remote Disk I/O, Synchronization etc. At this time, there is no way in Snowflake to access those statistics programmatically instead of having to navigate to that section for each query that you want to analyze. This is a frequent request from customers and it might get offered one day!

Meanwhile, a couple workarounds are to use the open source project 'Snowflake Snowsight Extensions' that enables manipulation of Snowsight features from command-line. You can also wait for GET_QUERY_STATS, system function that returns statistics about the execution of one or more queries, available in the query profile tab in Snowsight, via a programmatic interface. It is still in private preview as of the date of publication of this article. Update: GET_QUERY_OPERATOR_STATS(), in public preview available to all accounts, is a system function that returns statistics about individual query operators within a query. You can run this function for any query that was executed in the past 14 days. Such statistics, available in the query profile tab in Snowsight, are now available via a programmatic interface. See blog from Snowflake Inc. published on December 20, 2022 and titled Analyze Your Query Performance Like Never Before with Programmatic Access to Query Profile.

Update: 'Programmatic Access to Query Profile Statistics, in public preview soon, which will let customers analyze long-running and time-consuming queries more easily. This view will also identify (and let users resolve) performance problems such as exploding joins before they impact the end user.' See blog from Snowflake published on November 8th, 2022: A Faster, More Efficient Snowflake Takes the Stage at Snowday

9. Lack of documentation of error codes and messages

Snowflake might display cryptic error messages that are not available in Snowflake documentation. You need to contact Snowflake technical support or dig into source code of connectors and drivers in GitHub repositories. This does not help with all parts of Snowflake data platform. Although Snowflake is made aware of this issue, it does not have any plan on adding related documentation at this time!

III. Snowflake Ecosystem (3)

1. Inefficient integration with Snowflake

Code generated by third-party tools to connect and integrate to Snowflake could be inefficient from a performance perspective. For Example, metadata-related queries generated as introspection code can be extremely slow

2. Best practices knowledge required by third-party tools and services

Almost every third-party tool and service from the Snowflake ecosystem comes with an exhaustive list of best practices to follow! The burden is on Snowflake users to be aware of such best practices and implement them when integrating with Snowflake.

3. Piecemeal information about Snowflake performance optimization and tuning

Some information about Snowflake's performance, available piecemeal here and there, lacks structure, needs consolidation, and might be outdated or misleading.

IV. Snowflake Users (3)

1. Snowflake users need to get familiar with Snowflake's ways of dealing with performance

Users familiar with their legacy systems and migrating to Snowflake using a lift and shift approach might encounter performance issues in their Snowflake account. Some of the old ays they used to with their legacy

systems such as indexes and won't work in Snowflake! They would need to tune their Lift and Shift approach and take a fresh look at their performance problems.

2. Snowflake users take shortcuts for solving performance issues

Snowflake makes it incredibly easy for users to try and fix slow workloads by simply throwing more compute resources at them. Snowflake charges for services such as automated clustering that help optimize performance when used properly.

Instead of identifying the root causes of performance problems, following known Snowflake performance best practices, and avoiding anti-patterns, Snowflake users take shortcuts. To solve some of their performance problems, they opt for brute force and the ease of use of features such as scaling up, scaling out, and automatic clustering at the cost of unnecessarily computing costs!

3. Snowflake users lack the knowledge about performance optimization and tuning

Snowflake users are a broad mix of business and technical users with varying levels of Snowflake proficiency. As a result, they might have inefficient queries and processes of data ingestion, transformation, and consumption.

4. Snowflake users face a steep learning curve of Snowflake performance optimization and tuning

Like any new change or experience, it can be jarring to transition to Snowflake and learn more about its performance optimization and tuning.

In addition, as Snowflake keeps quickly evolving and adding new features and functionality, you need to keep yourself up-to-date to get the most out of your Snowflake account. Snowflake does not remove the need for highly skilled technical resources to operate and tune a Snowflake account.

Here are some examples of Snowflake performance aspects you need to keep learning about and related challenges you need to solve:

4.1. Long-running queries: You might need to seek and destroy long running queries that are result of mistakes, are eating up compute resources and are not adding any value! This blog posted on September 5th 2022 and titled Snowflake: Long Running Queries Seek & Destroy shows an example of a long running query and 2 ways on how to solve this problem.

Using QUERY_HISTORY, you can accurately identify long running queries by looking at their EXECUTION_TIME. Please note, that the TOTAL_ELAPSED_TIME includes the time the query spent sitting on queue as the warehouse is being provisioned or overloaded. See this blog published on January 10, 2022 and titled Long Running Queries in Snowflake: QUERY_HISTORY function.

4.2. Complex queries: If a query is spending more time compiling compared to executing, perhaps it is time to review the complexity of the query. See article Understanding Why Compilation Time in Snowflake Can Be Higher than Execution Time

4.3. Inefficient queries: Although tuning and optimizing SQL queries is a big topic, some known guardrails could help you pick some low-hanging fruits of performance improvement. Examples include

- Select *: When fetching required attributes, avoid using the SELECT * statement as it conveys all the attributes from the storage to the warehouse cache. This slows down the process and fills the warehouse cache with unwanted data. Forget about SELECT * queries! Snowflake is a columnar data store, explicitly write only the columns you need or use or specify the columns that should be excluded from the results using EXCLUDE within a SELECT * EXCLUDE (col_name, col_name, ...). See Selecting All Columns Except Two or More Columns

- Exploding joins: This happens when joining tables without providing a join condition (resulting in a "Cartesian product"), or providing a condition where records from one table match multiple records from another table. For such queries, the Join operator produces significantly (often by orders of magnitude) more tuples than it consumes. This is a common mistake to avoid when writing code to join tables and to identify using the Query Profile. See related Snowflake documentation "Exploding" Joins and the Snowflake Knowledge Base article published on January 15, 2019 and titled How To: Recognize Row Explosion

- UNION without ALL: In SQL, it is possible to combine two sets of data with either UNION or UNION ALL constructs. The difference between them is that UNION ALL simply concatenates inputs, while UNION does the same, but also performs duplicate elimination. A common mistake is to use UNION when the UNION ALL semantics are sufficient. These queries show in Query Profile as a UnionAll operator with an extra Aggregate operator on top (which performs duplicate elimination). The best practice os to use UNION ALL instead of UNION when deduplication is not required (or if we know that the rows being combined are already distinct). See related Snowflake documentation UNION Without ALL.

- Not using ANSI join: "There can be potential performance differences between non-ANSI and ANSI join syntax as the parser does not currently

transform/rewrite the join using the ANSI syntax after parsing. As a result, Non-ANSI syntax doesn't benefit from the subsequent steps of transformation and optimization by the compiler that eventually generate a good plan." See Snowflake Knowledge Base article, published on June 18, 2022, and titled Performance Implications of using Non-ANSI syntax versus ANSI join syntax.

- Using String instead of Date or Timestamp data type: Date/Time Data Types for Columns: "When defining columns to contain dates or timestamps, Snowflake recommends choosing a date or timestamp data type rather than a character data type. Snowflake stores DATE and TIMESTAMP data more efficiently than VARCHAR, resulting in better query performance. Choose an appropriate date or timestamp data type, depending on the level of granularity required." . See Snowflake Table Design considerations

People might learn SQL on the job and make their fair share of mistakes, including similar queries to the above ones. They can learn how to write SQL more efficiently. by taking an online SQL class or a training that focus on SQL for Snowflake.

4.4. Storage Spillage: For some operations (e.g. duplicate elimination for a huge data set), the amount of memory available for the compute resources used to execute the operation might not be sufficient to hold intermediate results. As a result, the query processing engine will start spilling the data to the local disk. If the local disk space is not sufficient, the spilled data is then saved to remote disks. Disk drives are a lot slower than ram. This spilling can have a profound effect on query performance, especially if a remote disk is used for spilling. By using a larger sized warehouse, you also get more ram. Thus, the query or load can complete so much faster that you are saving

more than the extra cost of being large. See Snowflake Knowledge Base article published on February 5th 2019 and titled How To: Recognize Disk Spilling

4.5. Concurrent queries: If a user experiences a performance issue for a query, one should find out the queries running concurrently with a problematic query in the same warehouse. Concurrent queries utilizing the same virtual warehouse can cause the sharing of resources such as CPU, Memory, and other key resources. This may potentially impact the response time for concurrent queries.

See the article from Snowflake Knowledge Base How To: Query to find concurrent queries running on the same warehouse, published on July 27, 2020.

4.6. Outlier queries: Example of outlier queries include those that use more resources than other queries in the same Snowflake virtual warehouse. See upcoming Snowflake Query Acceleration Service, in public preview as of the writing date of this article, that would help with a related solution. You can add QUERY_ACCELERATION service via Snowsight when creating a compute cluster to make complex queries run faster by injecting additional CPU horsepower from Snowflake serverless pools while the remaining simpler queries execute using only the regular nodes that are in your cluster. No need to increase the size of the entire cluster & waste money just to speed up some of the more complex queries when most other queries do not need that extra CPU power.

4.7. Queued queries: See Snowflake Knowledge Base article, published on September 9, 2020 How To: Understand Queuing about explaining of the different types of queuing and what to do when it occurs

4.8. Blocked queries: A blocked query is attempting to acquire a lock on a table or partition that is already locked by another transaction. Account administrators (ACCOUNTADMIN role) can view all locks, transactions, and session with: SHOW LOCKS [IN ACCOUNT]. For all other roles, the function only shows locks across all sessions for the current user. References:

- Snowflake Knowledge Base article titled How To: Resolve blocked queries and published on May 18, 2017.
- Blog 'Transaction Locks in Snowflake' blog by Sachin Mittal published on March 15, 2022.

4.9. Queries against the metadata layer: A query might taking more time on Metadata Operations. The Information Schema views are optimized for queries that retrieve a small subset of objects from the dictionary. Whenever possible, maximize the performance of your queries by filtering on schema and object names. For more usage information and details, see the Snowflake Information Schema related documentation.

4.10. Inefficient Pruning: 'Snowflake collects rich statistics on data allowing it not to read unnecessary parts of a table based on the query filters. However, for this to have an effect, the data storage order needs to be correlated with the query filter attributes. The efficiency of pruning can be observed by comparing Partitions scanned and Partitions total statistics in the TableScan operators. If the former is a small fraction of the latter, pruning is efficient. If not, the pruning did not have an effect. Of course, pruning can only help queries that filter out a significant amount of data. If the pruning statistics do not show data reduction, but there is a Filter operator above TableScan which filters out a number of records, this might signal that a different data organization might be beneficial for this query.' See related Snowflake documentation on Inefficient pruning and

Understanding Snowflake Table Structures. See also Snowflake Knowledge Base article published on January 24, 2019 and titled How To: Recognize Unsatisfactory Pruning

4.11. Inefficient data model: An inefficient data model can negatively impact your Snowflake performance. Often overlooked, optimizing your data model will improve your Snowflake performance. Snowflake supports various modeling techniques such as Star, Snowflake, Data Vault and BEAM. Let your usage patterns drive your data model design. Think about how you foresee your data consumers and business applications leveraging data assets in Snowflake.

4.12 Connected clients issues: You might be connecting to Snowflake using clients that are out of Snowflake support and not benefiting from fixes, performance and security enhancements, and new features. You will need to periodically check the versions of your connectors to Snowflake and upgrade. Snowflake sends a quarterly email on behalf of Support Notifications titled 'Quarterly Announcement — End of Support Snowflake Client Drivers: Please Upgrade!'

4.13. Lack of sub-query pruning: Pruning is not happening when using Sub-query. See this Snowflake Knowledge Base article

4.14. Queries not leveraging results cache: When a query is executed, the result is persisted (i.e. cached) for a period of time. At the end of the time period, the result is purged from the system. For persisted query results of all sizes, the cache expires after 24 hours. If a user repeats a query that has already been run, and the data in the table(s) hasn't changed since the last time that the query was run, then the result of the query is the same. Instead of running the query again, Snowflake simply returns the same result that it

returned previously. This can substantially reduce query time because Snowflake bypasses query execution and, instead, retrieves the result directly from the cache. To lear more check Using Persisted Query Results.

4.15. Queries doing a point lookup: You might need to improve the performance of selective point-lookup queries that returns only one or a small number of distinct rows from a large table. Search Optimization Service (SOS) is a feature that optimises all supported columns within a table. It is a table-level property, once added on a table, a maintenance service creates and populates search access path that are used to perform lookups.

4.16. Queries against very large tables: You might have a very large table and most of your queries are selecting only the same few columns, have about the same filters and doing aggregates on the same column. Creating a Materialized View (MV) with these repeating patterns can greatly help these queries. The idea is that, the Materialized View already holds the resulting data of these repeating patterns ready for retrieval rather than performing them over and over against the table.

Over time, as the data changes in a very large table as a result of DML transactions, the distribution of the data becomes more and more disorganized. Automatic Clustering enables users to designate a column or a set of columns as the Clustering Key. Snowflake uses this Clustering Key to reorganize the data so that related records are relocated in the same micro-partitions enabling more efficient partition pruning. Once a Clustering Key is defined, the table will be reclustered automatically to maintain optimal data distribution. Automatic Clustering will then help with queries against large tables that uses range or equality filters on the Clustering Key.

4.17. Latency issues

4.18. Heavy Scanning

4.19. Slow data load speed: It can be due to many reasons, such as not partitioning staged data files to avoid scanning files that have already been loaded or not breaking up single large files into the appropriate size to benefit from Snowflake's automatic parallel execution.

4.20. Slow ingestion of streaming data

What else would you like to add to the above Snowflake performance challenges? I much appreciate your comments and feedback.

Please, spread the word, and don't forget to give this article a 'Like' if you find it helpful so that your LinkedIn buddies can read it and comment on it too. I hope this 2-part blog series will pave the way to further constructive discussions about Snowflake performance challenges and solutions for the benefit of all.

Thank you

Snowflake | Snowflake Data Cloud | Snowflake Computing | Optimization | Performance

Written by Slim Baltagi

67 followers · 81 following

Follow

AI enthusiastic. Data & Analytics Thought Leader. 30K+ Followers on LinkedIn
https://www.linkedin.com/in/slimbaltagi/

Responses (1)

Omar Essam he/him

What are your thoughts?

yugandhara saste
Jun 3

Question:

A Data Engineer is using Snowpark to process large datasets in Snowflake and wants to:

Minimize data distribution across nodes

Maximize compute resource utilization

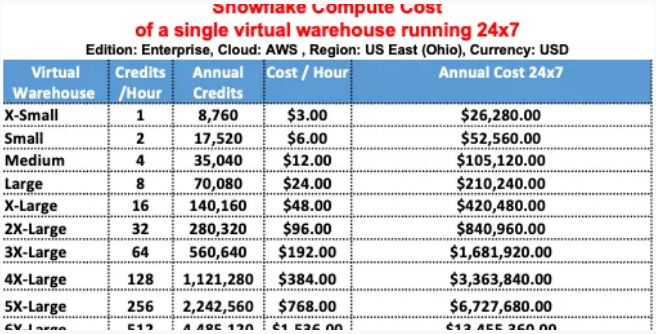
Which approach best meets these requirements?

Options:

A) Load with... more

Reply

More from Slim Baltagi



Snowflake Compute Cost
of a single virtual warehouse running 24x7
Edition: Enterprise, Cloud: AWS , Region: US East (Ohio), Currency: USD

Virtual Warehouse	Credits /Hour	Annual Credits	Cost / Hour	Annual Cost 24x7
X-Small	1	8,760	$3.00	$26,280.00
Small	2	17,520	$6.00	$52,560.00
Medium	4	35,040	$12.00	$105,120.00
Large	8	70,080	$24.00	$210,240.00
X-Large	16	140,160	$48.00	$420,480.00
2X-Large	32	280,320	$96.00	$840,960.00
3X-Large	64	560,640	$192.00	$1,681,920.00
4X-Large	128	1,121,280	$384.00	$3,363,840.00
5X-Large	256	2,242,560	$768.00	$6,727,680.00
6X-Large	512	4,485,120	$1,536.00	$13,455,360.00

Slim Baltagi

Snowflake Cost Intelligence & Optimization (Part 1)

Disclaimer: The opinions in this three-part blog series are entirely mine and do not...

Jan 29, 2024 13

Slim Baltagi

24 FREE Snowflake Optimization Apps for Cost & Performance

Slim Baltagi

Apr 9, 2024 9

In Snowflake Builders Blog: Data Engine... by Slim...

A novel approach for your Snowflake Health Check

Disclaimer: This short thought leadership article is my contribution to the Data...

Jun 22, 2022 7

Slim Baltagi

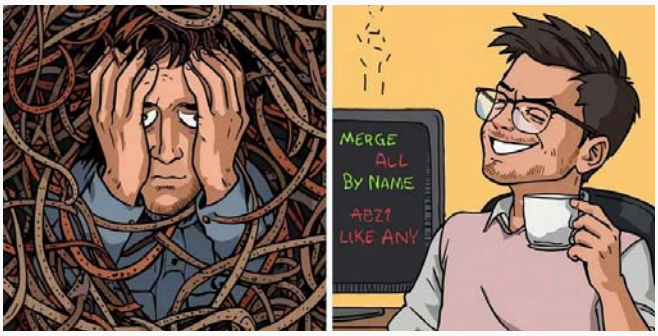
20 Snowflake Inefficiencies To Avoid and Save Cost!

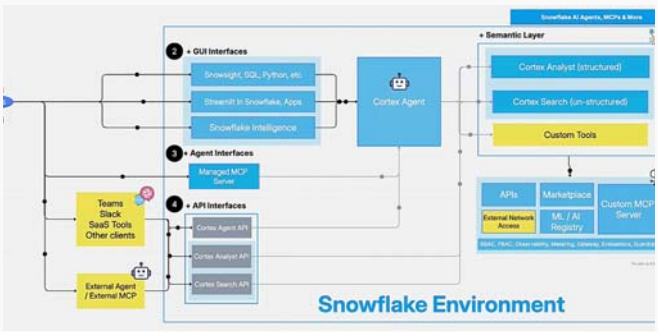
Disclaimer: The opinions in this article are entirely mine and do not necessarily reflect...

Jul 9, 2023 15

See all from Slim Baltagi

Recommended from Medium





Vishal Kaushal

Top 10 Snowflake SQL Functions Every Data Professional Should...

Are you spending too much time writing and maintaining long SQL queries in Snowflake?...

Oct 16 34 1

In Fru.dev by Fru

5 Strategic Pillars to Understanding Snowflake's AI...

A Complete Architecture Overview

Sep 16 23

John Ryan

Snowflake Gen-2 Warehouses: Faster Performance or Just Highe...

Introduction

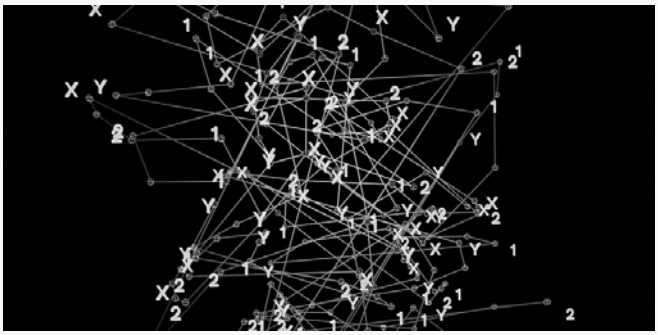
Oct 1 6

Data Engineering Simplified

Apache Iceberg in Practice: Beyond The Docs

A few months ago, I spoke with a data engineering team at a large enterprise. They...

Sep 9 26



Javier Martinez

AWS S3 - Snowflake Data Pipeline with Snowpipe

A step-by-step guide on how to build a data pipeline between your S3 bucket and...

Kamalakannan R

Snowflake

— use role sysadmin; —use warehouse compute_wh; —create or replace database...

Aug 13 2 Aug 20

See more recommendations