

SNOWFLAKE OPTIMIZATION BEST PRACTICES

A guide to balancing cost and
performance at scale





TABLE OF CONTENTS

GETTING STARTED

- 3 Getting started with Snowflake
- 4 Understanding Snowflake architecture
- 5 Importance of Snowflake warehouse optimization

BEST PRACTICES

- 7 Balancing cost and performance in Snowflake
- 8 Best practices for optimizing Snowflake
- 10 Snowflake features for warehouse optimization

SCALE SNOWFLAKE CONFIDENTLY

- 12 Scale Snowflake confidently with Capital One Slingshot
- 13 Enhancing Snowflake with Slingshot
- 14 Slingshot features for Snowflake optimization



GETTING STARTED WITH SNOWFLAKE

Snowflake's AI Data Cloud is a global network that connects customers to the data, applications and models that are most critical for their businesses. Organizations successfully use Snowflake as a data warehouse to run complex analytics on large volumes of data concurrently without sacrificing performance. Due to its unique architecture, Snowflake allows organizations to scale their data operations while paying for only what they use.

Understanding Snowflake's architecture, the pricing structure and how it can be optimized is vital to maximizing the use of Snowflake for a well-managed data platform, including managing costs and minimizing operational inefficiencies.

Understanding Snowflake architecture: Separation of compute and storage

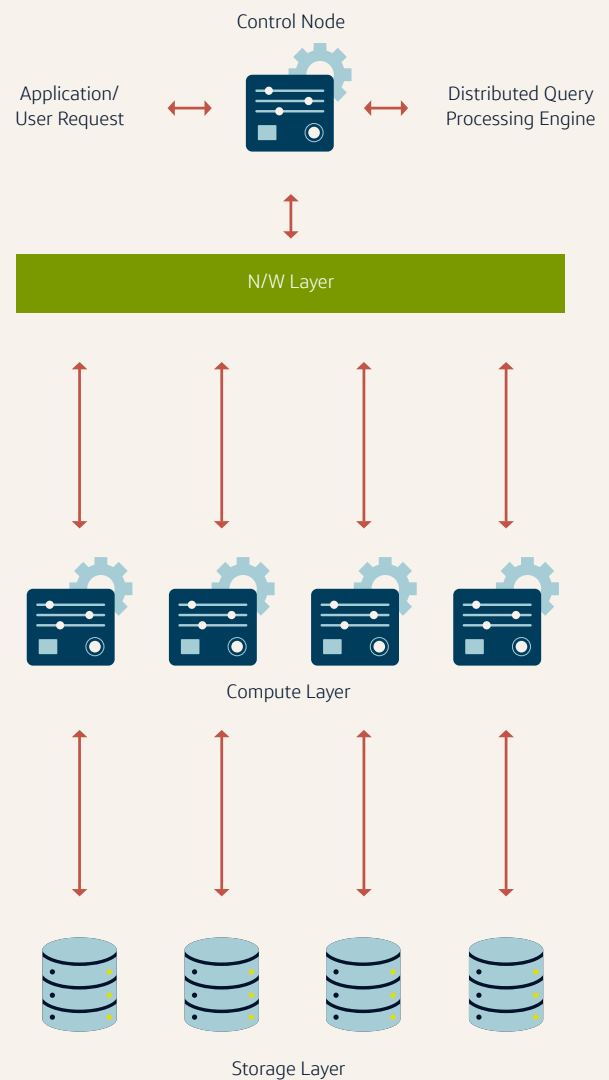
A distinct characteristic of Snowflake's architecture is the separation of processing power (compute) and storage, allowing users to scale each component independently of the other based on need. This decoupling of compute and storage, which have traditionally been tied together in on-premise systems, allows for great flexibility and elasticity such as the near-infinite scaling of data storage.

The decoupling of storage and compute is achieved through a hybrid of the standard shared-disk and shared-nothing database approaches. In a shared-disk architecture, all servers or processors (nodes) in the database cluster have access to centrally located disk storage. Data is stored in a single, common location that is accessible to all the nodes simultaneously. In contrast, a shared-nothing architecture separates data across nodes and each node operates independently with its own disk storage, resources and set of data. Such systems scale horizontally by simply adding more nodes.

Snowflake takes advantage of the strengths of both approaches in its data warehouse architecture. Data is stored in a centralized, cloud storage layer that is accessible from all compute nodes. At the same time, each virtual warehouse is an MPP (massively parallel processing) compute cluster that is independent of other virtual warehouses and locally keeps a subset of the whole data set. A query submitted in Snowflake is executed in parallel with other queries. Processing queries in one virtual warehouse has no impact on the performance of another warehouse. Storage is centralized and shared, while compute resources can scale horizontally and independently. The decoupling allows scaling of virtual warehouses separate from storage, and vice versa.

Customers gain the best of both worlds through this combination, benefiting from the simpler [data management](#) of shared-disk architectures and the scaling advantages of the shared-nothing approach. While queries are running, scaling of compute resources can happen without disruption or need to recalibrate data storage. Separate workloads, such as running SQL queries and data loading, can run simultaneously.

Massively parallel processing (MPP)



Importance of Snowflake warehouse optimization

The [data warehouse](#) is the cornerstone of a Snowflake database. With the separation of compute (warehouses) and storage, the warehouse is exclusively where data processing tasks occur distinct from Snowflake's cloud storage layer. This makes the warehouse the main factor determining Snowflake performance and costs. Understanding the architecture of a Snowflake warehouse and [techniques to optimize warehouses](#) is key to utilizing the platform for high performance while managing costs for the organization.

SNOWFLAKE'S PRICING STRUCTURE

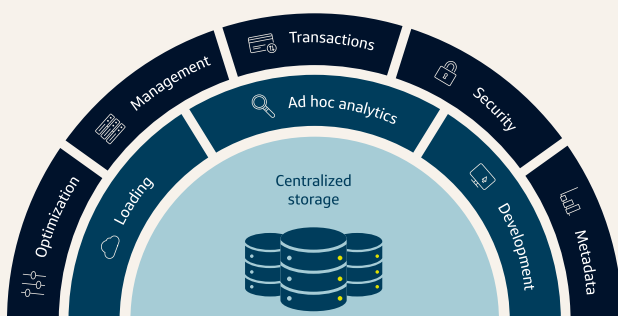
Given Snowflake's consumption-based pricing structure, the bulk of an organization's total costs will come from the compute costs of running virtual warehouses. Other costs come from the usage of storage, serverless features, cloud services and data transfer. Snowflake charges customers based on the amount of compute resources used (measured in credits), which are billed for the time a warehouse is running. Snowflake bills the customer for a minimum of one minute of usage when a warehouse starts or resumes, followed thereafter by billing per second. The number of virtual warehouses used, their sizes and the length of time they run determine how much credit is charged.

SCALING AND CONCURRENCY

Snowflake warehouses are designed for flexibility and elasticity of computational resources, which is reflected in the way it scales and manages concurrent workloads. Snowflake has the ability to dynamically scale both vertically, by resizing warehouses to add more power, and horizontally, by adding clusters to distribute the workload. This combination of scaling gives the platform the ability to handle a wide variety of workloads from small load jobs to complex, large-scale queries. Through dynamic scaling, organizations can scale up instantly when demand is high and scale down for less intensive tasks. Because of the multi-cluster warehouse architecture, Snowflake supports concurrent workloads that run simultaneously without affecting each other's performance.

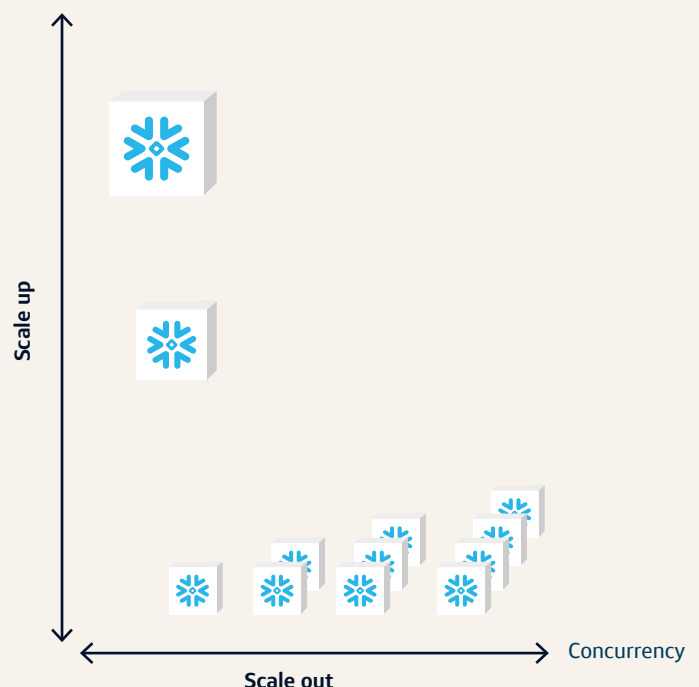
Snowflake architecture: Separation of storage and compute

- Management and metadata**
Independent management services manage metadata, optimization, security.
- Compute**
Multiple, independent compute clusters process queries.
- Storage**
Database data centrally stored in cloud storage.

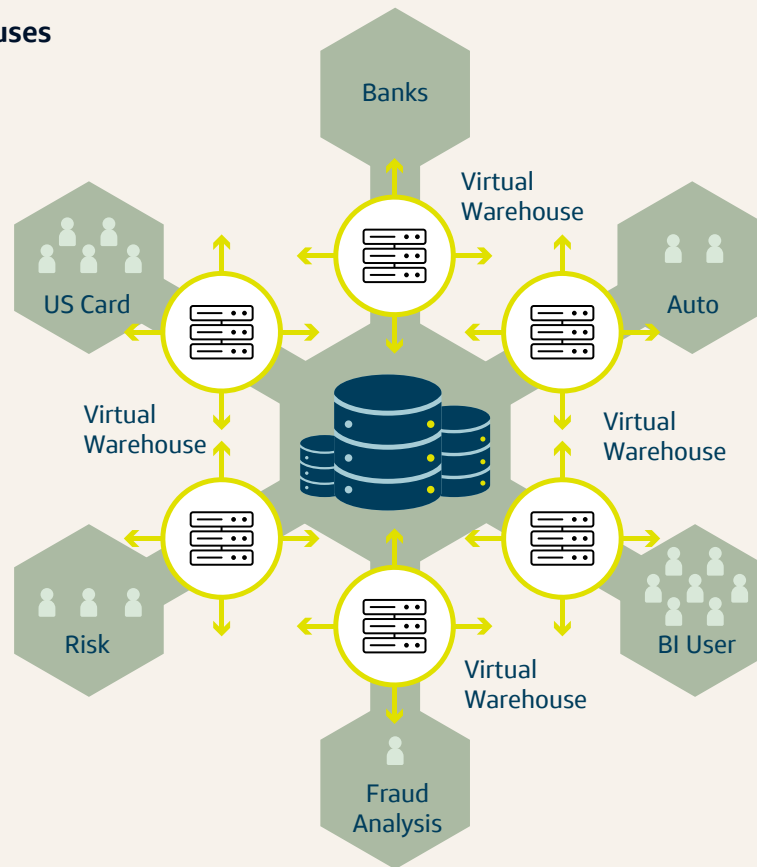


Scaling and concurrency

Query performance



Provisioning warehouses



PROVISIONING WAREHOUSES

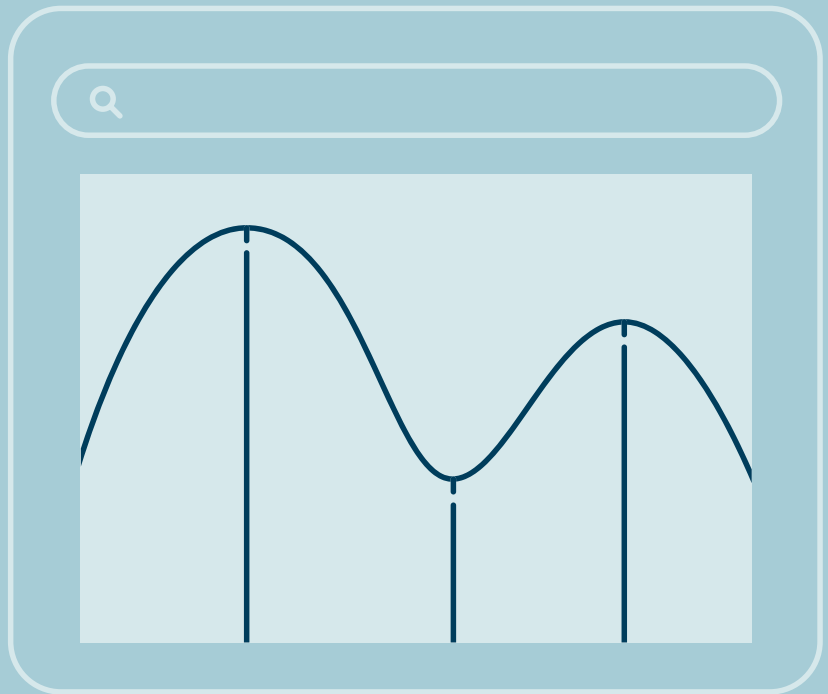
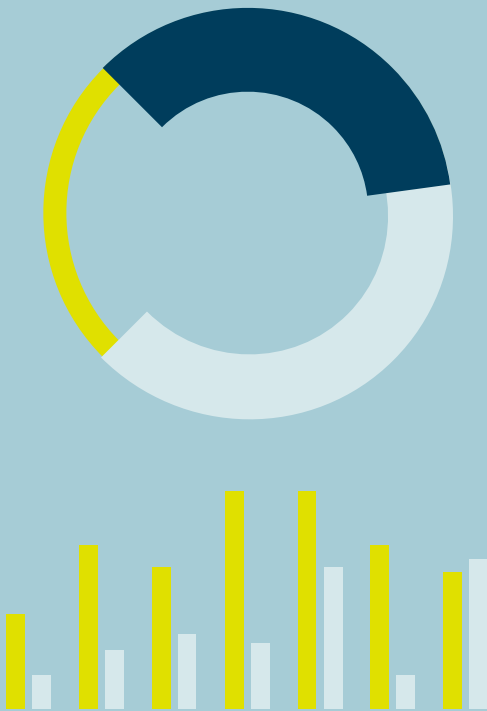
Warehouse provisioning in Snowflake involves different warehouse sizes (x-small, small, medium, large, x-large and so on) with increasing levels of compute power and storage capacity with size. Larger warehouses have more compute resources and can process more tasks simultaneously. Snowflake uses threads, or parallel processing units responsible for executing queries, to process tasks concurrently. In addition to warehouse size, configuring a warehouse with the right maximum and minimum concurrency levels through the number of threads is an important component of optimizing Snowflake warehouses.

Snowflake charges credits per second a warehouse is running, whether or not it's in use. Users can better control warehouse usage by utilizing the maximum concurrency parameter, which limits the number of concurrent queries running in a warehouse, and setting a minimum and maximum number of clusters for a particular warehouse.

ONGOING OPTIMIZATION FOR GREATER ROI

As organizations grow, their data workloads must also scale, for example to ingest greater volumes of customer and sales data or adopt more advanced analytics involving machine learning (ML). With increasing workloads, many organizations encounter challenges [identifying inefficiencies](#) such as poorly written queries and the right times to scale back, which can lead to escalating and unexpected costs. It's crucial for organizations to understand how to optimize their data warehouses by utilizing resources with the right balance between performance and cost, optimizing queries and choosing the right size warehouse to ensure they are not over- or under-provisioning.

Understanding Snowflake's warehouse architecture and how it can be optimized allows organizations to get the most out of their Snowflake investments in performance and efficiency. We will next walk through some of the best ways to optimize Snowflake warehouses through a look at Snowflake features and best practices, including lessons learned from Capital One's own experience using this powerful platform.



BALANCING COST AND PERFORMANCE IN SNOWFLAKE

Data warehouses are crucial for data storage, running queries and data manipulation. Due to its flexibility and scalability, it's easy to under- or over-utilize resources in Snowflake warehouses. Understanding Snowflake optimization techniques and best practices can help teams manage consumption costs while maximizing benefits to the business.

Best practices for optimizing Snowflake

Snowflake's flexibility and power have helped many organizations, [including Capital One](#), scale their data operations in the cloud. Managing costs while optimizing performance on Snowflake is paramount for data-focused organizations as they grow and modernize their workloads. For organizations looking to maximize their use of Snowflake, the following best practices can support more efficient warehouse management.

1 RIGHTSIZED WAREHOUSES

Determining the correct size for warehouses is one powerful way organizations can strike the right balance between cost and performance while reducing waste. A business wants to use compute only when necessary and reallocate costs from poorly optimized configurations. In general, the goal is configuring the smallest warehouse size that will complete queries while meeting the organization's performance needs.

How does a business determine the right size for the needs? There are many factors to consider in selecting the right warehouse size that minimizes waste while improving the performance overall. Key metrics to consider include query size, data spillage, query load and queued queries.

Taking an inventory of **query sizes** that run in a warehouse will help with understanding the warehouse size necessary, and could reveal the organization has been paying for an XL warehouse when all that was needed was a medium. Viewing patterns in **data spillage** will help identify the queries that lead to disk spillage and the actual amount of disk storage needed to complete requests. **Query load**, or the time it takes for a query to run in a warehouse, is also an important consideration in the trade off between sizing up (decreasing query load times) and increasing costs. Lastly, examining the number of **queued queries** due to the load on the warehouse will help determine the right scaling configurations for clustering.

An important principle in optimizing warehouses is that one warehouse size does not fit all. One warehouse size should not be running at all times and should be scheduled to change based on the query load on a certain day of the week or at a given time of day.

2 KEEP DIFFERENT TYPES OF WORKLOADS SEPARATE

Each team will have its own requirements and the warehouse size should match their specific goals and needs. Rightsize warehouses by type of workload, such as machine learning and ETL, and keep them distinct. Also, keeping small and large queries separated in different warehouses is one best practice for avoiding unnecessary costs. For example, if users are running small queries in a 4XL warehouse along with large query jobs, performance for those small queries will not improve while keeping the warehouse running for longer, leading to large and unnecessary costs for the business.

3 SET THE RIGHT AUTO-SUSPEND TIME

An idle warehouse consumes Snowflake credits even if there are no queries or workloads running. Snowflake's auto-suspend setting helps minimize the cost of unused warehouses by suspending idle warehouses and is set at 10 minutes by default. At Capital One, we've found that setting the [auto-suspend](#) times in smaller increments, for example reducing the auto-suspend time from 10 minutes to 2 minutes, decreases idleness while saving costs. Some cache is lost with this method, but we found an initial decrease in performance of queries is made up for by the warehouse running at optimal levels subsequently.

“

An important principle in optimizing warehouses is that one warehouse size does not fit all.”

4 CHOOSE THE RIGHT SCALING POLICY

In Snowflake, you can configure the multi-cluster feature to automatically scale compute resources and support the concurrency of users and queries. Configuring multi-cluster in Snowflake with the right scaling policy is important to ensure the environment handles the demands of the workload while managing costs. There are two types of scaling policies: **standard**, which automatically adds or removes clusters based on query demand, and **economy**, which fully utilizes current clusters and avoids spinning new clusters to save credits. In our experience, we have found if the workload could afford a delay of more than five minutes, there are significant cost savings with choosing the economy setting.

5 IMPROVE QUERY PERFORMANCE

Queries are instrumental to the health and success of data-driven organizations, but they can also be costly. Inefficient queries can strain any business with more use of resources like CPU and memory and slowdowns in performance that affect teams across the organization. By taking the time to improve the quality of queries through [query optimization](#) companies can gain significant cost advantages.

There are a number of ways to optimize queries for cost savings and better performance in Snowflake. Companies should have **alerting mechanisms** in place that give notice when a query is running when it should not be. Alerts provide an opportunity for intervention to avoid an unwanted large bill at the end of the month. At Capital One, we've also invested in **education on**

“

The bulk of an organization's total costs will come from the compute costs of running virtual warehouses.”

queries so that users can avoid writing bad queries from the outset or identify these queries before they run. A few of the mistakes in inefficient queries we have identified and educate users on to avoid include:

- Single row inserts
- Selecting * from a large table
- Cartesian product joins
- Deep nested views
- Performances that spill to disk

6 SET UP MONITORING AND REPORTING

Snowflake users need the ability to monitor their Snowflake usage continuously and to be notified of credit usage as it happens. Snowflake's powerful elasticity and by-the-second charging structure means costs can quickly rise and get out of hand, and the business may not find out until the monthly bill. These events can occur through faulty configurations or inefficient queries. Identifying these errors and inefficiencies and then remediating can be time-consuming and costly to do manually, pulling valuable developer resources away from strategic tasks. Putting in place alerts and limits will bring greater visibility and control over Snowflake spend so the organization is not overspending and blindsided.

7 STANDARDIZE PROCESSES AND AUTOMATE GOVERNANCE

As a business scales, the traditional way of managing data through a central data team breaks down in the face of exponentially growing volumes of data and requests, leading to bottlenecks and inefficiencies. Ensuring data is managed responsibly becomes critical. By standardizing data processes and automating [data governance](#) across the organization, the business can empower teams to perform tasks independently, such as provisioning warehouses on their own, in a secure and responsible way. At Capital One, for example, we created a self-service portal that equips teams with the data they need while ensuring they are following data security and governance standards. A built-in traceability solution enables approval workflows for remediation and retention use cases.

Snowflake features for warehouse optimization

Snowflake provides built-in features for efficient warehouse management and optimal performance: Snowpark-optimized warehouses and caching.

SNOWPARK-OPTIMIZED WAREHOUSES

Choosing the right warehouse type and size for each workload is key to [optimizing Snowflake costs](#). In Snowflake, there are two warehouse types that determine the memory-to-CPU ratio: **Standard** and **Snowpark-optimized** warehouses. There are also multiple sizes (e.g., XS, S, M, L ... 6XL), which set the total amount of CPU and memory available.

Snowpark is a data framework that allows developers to write code using Python, Scala or Java directly onto Snowflake. Developers can use Snowpark to execute UDFs (user-defined functions) in their chosen language, which opens up the platform to new use cases and complex workloads. One of these is the ability to build and deploy machine learning models directly on Snowflake, which requires memory-intensive workloads on large data sets.

For these workloads with large memory requirements, such as ML model training and complex data processing, Snowpark-optimized warehouses are recommended for better performance and speed. A Snowpark-optimized warehouse can provide up to [16 times more memory per node and 10 times more local cache](#) compared to a standard warehouse. Running Snowpark UDFs with intensive tasks on Snowpark-optimized warehouses ensures the right amount of resources to handle the demands of the workload.

Snowpark-optimized warehouses are not ideal for simple data retrieval and analysis. In this case, standard warehouses are usually the most cost-effective and efficient. Snowpark-optimized warehouses have a different pricing structure from standard warehouses and require a longer time to start up. Examining the requirements and purpose of a workload can help determine if the additional memory and performance enhancements are worth the additional costs. For example, a pattern in which the majority of queries spill bytes to local and remote storage may mean the warehouses could benefit from sizing up or changing to a Snowpark-optimized warehouse.

“Understanding Snowflake’s architecture, the pricing structure and how it can be optimized is vital to maximizing the use of Snowflake for a well-managed data platform...”

SNOWFLAKE CACHING

Query performance, or the time to execute a query and return results, can suffer in traditional data warehouses when complex queries require large datasets and heavy resources. In Snowflake, where using resources efficiently is necessary to manage costs, caching is a key performance tuning feature that allows quick access to frequently used data. Caching enhances query performance by reducing the time it takes to get results while using fewer resources, which in turn leads to cost savings. Let's take a look at two types of caches in Snowflake: warehouse cache and query cache.

WAREHOUSE CACHE

The warehouse cache, also referred to as the “local disk cache,” helps optimize performance within a particular virtual warehouse. Snowflake's warehouse cache, housed within Snowflake's compute layer where queries execute across the nodes of virtual warehouses, optimizes query performance through the storage of frequently accessed data in-memory. This storage in-memory reduces the time to retrieve data and leads to faster query processing. Each time a warehouse runs, it keeps a local cache of data that can be accessed by future queries. Reading from the cache instead of the tables that live in the cloud storage layer leads to better query performance. The cache drops when a warehouse is suspended, which is a key factor in determining whether to suspend a warehouse (saving credits) or keep it running (maintaining the cache).

QUERY CACHE

Snowflake's query cache, also known as the “result cache,” lives in the cloud services layer and holds the results of every query executed in the last 24 hours. These results are available to all virtual warehouses so that any user in the system that executes the same query has access to the query results. Snowflake caches and persists every result from queries, which leads to a great reduction in the time it takes to return an answer. Query results are reused if certain conditions are met:

- The new query matches the old query syntactically
- The underlying query results table data has not changed, and previous query results are still available
- Every time the query result is reused, Snowflake refreshes the 24-hour retention duration for up to 31 days from the query's execution

One important thing to note about this feature: when performing benchmark testing, the user should disable query cache because it can skew the results of query performance.



SCALE SNOWFLAKE CONFIDENTLY WITH CAPITAL ONE SLINGSHOT

At Capital One, we found we needed additional ways to manage our Snowflake costs and streamline work processes at scale across our teams. To minimize bottlenecks and streamline governance, we developed internally what would later become [Capital One Slingshot](#), a cloud-based data management solution that optimizes Snowflake usage and helps businesses maximize their Snowflake investment. For an organization that uses both, Snowflake is the data warehouse platform for storing and analyzing data sets while Slingshot is the tool that builds on Snowflake to enhance its capabilities for businesses.

Enhancing Snowflake with Slingshot

While Snowflake provides the scalable and flexible foundation for storing and analyzing data, Slingshot's focus is on optimizing Snowflake usage, increasing visibility into costs and streamlining workflows. Organizations get greater visibility into their cloud costs and self-service capabilities for greater efficiency and fewer manual dependencies. Let's unpack the main ways Slingshot enhances and builds on top of Snowflake.



COST VISIBILITY

Visibility into Snowflake usage is crucial to optimizing spend in the ways that most benefit the business. With the ability to view usage at a granular level through tagging and attribute spending to specific teams and business units, Slingshot users can gain greater insight and understanding into Snowflake costs and usage. Additionally, while Snowflake provides information by account, Slingshot's Cost Breakdown Report is valuable for organizations looking to deepen their understanding with a view of costs across accounts.



MINIMIZING RESOURCE CONSTRAINTS AND MANUAL EFFORT

Using Snowflake, customers can configure warehouses to automatically allocate resources based on changing workloads, but this is largely a manual effort that becomes time consuming and complex when trying to fine tune optimization across many warehouses. The dynamic scaling of data warehouses based on real-time workloads ensures optimization of resources while minimizing manual work to meet cost and performance goals. Dynamic scheduling based on custom schedules can ensure your business is incurring costs only for what is needed. Managed workflows also allow business teams to manage warehouses without involving database administrators. As a result, the organization can reinvest those savings into the business for new workloads and use cases. At Capital One, we prevented 50,000 hours of manual effort through use of our automated tools.



SCALABILITY AND FEDERATION

A key strength of Snowflake is its data sharing abilities, which opens up data access across the organization and allows multiple teams to execute queries and run workloads concurrently without a degradation in performance. As businesses scale and data volumes and business requirements grow, bottlenecks and inefficiencies become barriers to growth and speed. [Federated data management](#) places ownership of data in the hands of domain teams that produce the data. Unlike a traditional reliance on a centralized data team, this approach disperses data responsibilities into lines of business who are the experts on a particular set of data and equips them with self-service tools, which frees technical resources from a backlog of data requests. Slingshot provides the tools that allow lines of business to manage their own data and compute while following best practices and adhering to proper governance. Organizations can grow their use of Snowflake without dependencies and bottlenecks that come from relying on a central data team.



STREAMLINING PROVISIONING FOR GREATER EFFICIENCIES

Streamlined workflows allow organizations to reduce time spent on provisioning their warehouses so teams can get to using Snowflake more easily. Automating common workflows in a self-service tool while enjoying built-in governance controls can lead to savings and efficiency gains across the board. At Capital One, these gains included [reducing inefficient query patterns](#) and decreasing our cost per query.

Slingshot features for Snowflake optimization

Snowflake gives organizations the flexibility and architecture to scale quickly without sacrificing performance. Slingshot accelerates adoption of Snowflake in the organization, while giving greater control and transparency through key cost and performance optimization features:



COST VISIBILITY

Organizations gain deep visibility into cost drivers and line-of-business allocations within Snowflake with detailed dashboards of all the key metrics impacting cost, performance and usage of Snowflake. Tagging enables a granular level of visibility, breaking down costs by user, accounts, warehouses, lines of business and queries. The [Cost Breakdown Report](#) allows admins to further customize the way they attribute spending to specific teams or business units, enabling chargebacks.

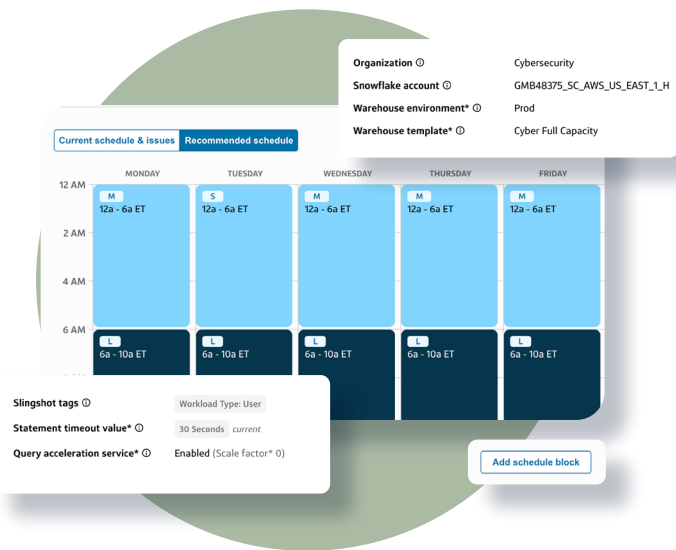
The form is used to provision new warehouses. The fields and options are:

- Organization: Sales
- Warehouse environment: QA
- Warehouse template: Sales Low-Capacity
- Snowflake account: ACME_QA_AZURE_EASTUS2_NH
- Warehouse name: Sales_QA_LowCap

Buttons: Cancel, Create schedule

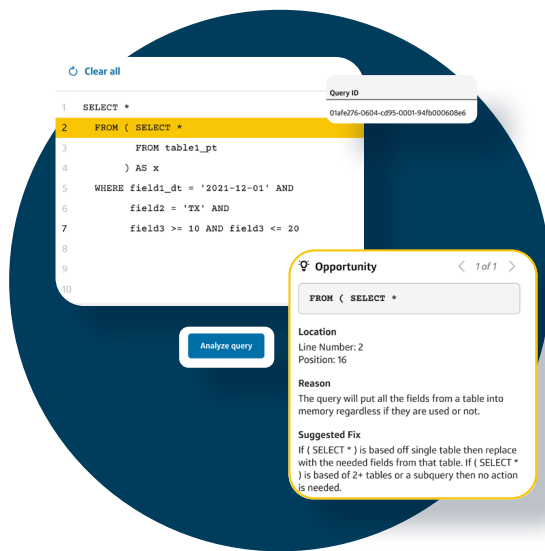
WAREHOUSE GOVERNANCE

Users can provision new warehouses using pre-configured templates and fine tune size and scaling policies to dynamically adjust warehouses based on need. Users can adjust warehouse sizes based on time and day of the week, allowing organizations to [optimize Snowflake warehouses](#) to run as efficiently as possible. Once a warehouse request is complete, Slingshot sends it to the right owner for approval.



WAREHOUSE OPTIMIZATION

For organizations looking for guidance on warehouse usage, Slingshot provides [data-driven recommendations to right-size warehouses](#) and dashboards that track warehouse performance. Slingshot analyzes historical usage patterns and warehouse metadata to determine the best schedule and sizing for balancing cost and performance, taking the guesswork out of optimization.



QUERY OPTIMIZATION

Based on best practices for writing efficient queries, [Query Advisor](#) analyzes queries to identify inefficiencies. It surfaces the costliest queries and the most frequently executed queries for analysis. Then the tool provides opportunities to improve the query for better performance. Slingshot also provides the query impact before and after so users can confidently apply the recommendations. At Capital One, this tool helped us decrease cost per query by 43%.

Maximize your investment with Snowflake best practices

The Snowflake platform provides a unique architecture that supports multiple data types, workloads, languages and runtimes on a common data foundation—connecting your business globally practically at any scale, through a single platform. As organizations grow in the use and management of data in the cloud, proactively optimizing Snowflake for both cost and performance will be instrumental to the success of realizing the full benefits of the platform for the business. In leveraging best practices, along with the right tools, to maximize your Snowflake investment, organizations can ensure they are using their resources efficiently while reaching their performance objectives for stakeholders across the business.



Visit capitalone.com/software/solutions to learn how
Slingshot can help you optimize Snowflake and
request a demo today.