2024년 09월 06일

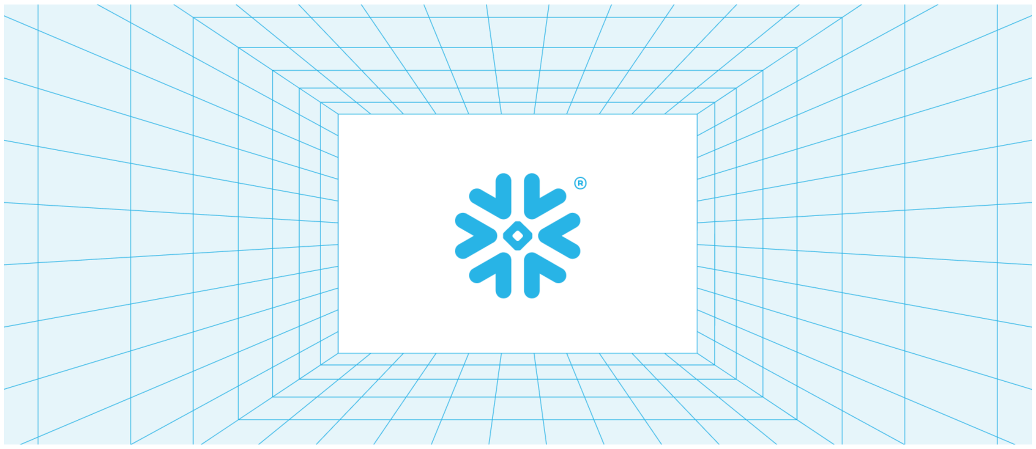
Efficiency at Snowflake: Free Pool Management

Core Platform

AUTHOR

Murray Stokely

Orestis Kostakis



SHARE

Snowflake currently executes more than 4.2 billion queries a day over exabytes of customer data. Given this scale, we spend a lot of time thinking about efficiency to reduce costs, complexity, and carbon-emissions for our customers. By providing a consumption-based service on top of the world's largest public clouds, we are able to combine the best performance and efficiency innovations from both the cloud providers and our own Snowflake application stack.

We work closely with the cloud providers to unlock efficiencies from the latest hardware, network, and data center technologies and partner with them to co-design future technologies for our most demanding workloads. At the same time, our engineers continuously optimize the Snowflake software stack through more efficient data formats, novel query optimizations, automatic clustering and partitioning, and vectorization improvements.

Some of the efficiency work we've done in recent years involves transitioning our virtual warehouses to newer more energy-efficient instance types, launching Query Acceleration Service to provide bursts of computing power for demanding queries, improved support for scanning, filtering, and caching with Unified Iceberg Tables, Vectorized Python UDTFs, aggregation placement, and dozens of other engineering enhancements outlined at the Snowflake Blog and Medium publication. Together, these and similar optimizations have led to a 20% improvement in query duration for stable workloads based on the latest Snowflake Performance Index (SPI) results.

Taking a step back, it is worth looking at the big picture of some of the levers we use to improve the efficiency of a large multi-cloud service like Snowflake:

AUTHOR

Murray Stokely

Orestis Kostakis

Hardware: New hardware architectures, GPUs, Local SSD vs remote block storage.

Financial/Cloud Management: Optimizing usage of Savings Plans, Reserved Instances, and Spot Instances to minimize cost for the needed level of capacity availability guarantees.

System Tuning: Tuning our base software stack for Snowflake workloads: kernel parameters, language runtimes, such as the JDK, and internal Snowflake configurations

Right Sizing: Auto-scaling instance counts, instance sizing, free pool optimization, and deployment sizing and consolidation.

SHARE

Lifecycle management: Utilizing tiered storage for hot and cold data, hibernating/pausing unused compute resources.

Locality: Service placement, data placement, and optimizing network flows across regions and availability zones.

Cost Visibility: Budgets, Resource tagging, alerts, metrics, and transitive cost accounting.

Workload optimization: Time shifting, sharing

More details about these general cloud cost levers can be found elsewhere (AWS, Azure, GCP, Digital Ocean, McKinsey, IBM, etc.).

This article is the first in a series of engineering deep-dives on some of these topics. For the remainder of this article, let's dive into the space of right-sizing cloud VM reservations: "Free Pool Optimization".

Free Pools and Instant Scalability of Snowflake Warehouses

One of the benefits of Snowflake over traditional data warehouses is the separation of storage from compute. This enables our customers to instantaneously scale their compute resources on-demand based on their workloads. We allow Snowflake customers to scale up their compute resources in less than a second. However, cloud providers have some latency before new virtual machines (VMs) can be provisioned, which can be up to several minutes. Figure 1 highlights the observed latency of new VM instances provisioned across three major cloud providers on a typical day.

AUTHOR

Murray Stokely

Orestis Kostakis

Figure 1: Observed provisioning latency by hour of day across all three cloud providers on a typical day.

To insulate customers from this latency, we closely track consumption and latency of VM provisioning with the cloud providers. Based on this information, we maintain a free pool of ready-to-go VMs that can be immediately assigned to a customer’s Virtual Warehouse when needed. Fre ol often also called “warm pools”. Their use is prevalent in the industry and finding ways to efficiently manage them is an active area of research across public and private clouds.

Since servers in the free pools are not performing useful work for customers, they represent a cost to Snowflake. Therefore, our goal is to satisfy customer demand while minimizing the size of the free pool. This is formulated as an optimization problem to minimize free pool server minutes while maintaining our tight internal latency targets for warehouse creation and resumption.

We also model customer workloads to ensure that we maintain an optimal free pool size. Economies of scale mean that with more customer workloads on Snowflake, we are better able to absorb different types of spikes at low cost, since those spikes are usually uncorrelated. However, there are still cases where requests tend to be batched up in quick succession. For example, many customers run hourly jobs at the top of the hour or daily jobs at midnight and this can lead to a large spike of requests for more resources as can be seen over an example week in Figure 2. This problem is exacerbated by the fact that other customers of the cloud providers often have similar affinity for these time boundaries and the associated latency metrics spike at exactly the time we need them the most.

AUTHOR

Murray Stokely

Orestis Kostakis

Figure 2: Number of VMs requested per minute across an example day on an example deployment.

With these two figures in mind, we forecast the required size of the free pool with a time horizon of up to 1 hour into the future. As mentioned above, we must size our free pools to satisfy the customer demand but also be cost-efficient. So, we define our loss or cost function c(t) as:

th nalty we assign to each over-provisioned server

p_u is the penalty for each under-provisioned server

$\hat{y}_t$ is the pool size we set

d_t is the customer-demand of servers for a period of time t

Our goal is to forecast the value $\hat{y}_t$ at each time period so that it minimizes our cost function c(t).

Given this formulation of the cost function and the relevant inputs, we use a bespoke time series forecasting method to generate optimal free pool sizes, for each cloud provider deployment and for each time window. Our cloud provisioning systems then use this to request the VMs and prepare them for customer workloads before the expected customer demand requests them. By better predicting the required number of servers, our customers can scale their compute-usage in under a second, thus dramatically speeding up their query performance. The results of this prediction for an example deployment are shown in Figure 3.

AUTHOR

Murray Stokely

Orestis Kostakis

Figure 3: Comparison of static free pool and predicted free pool sizing for measured net demand of an example deployment.

Lessons Learned

The predictive free pool provisioning work described here has been in production for three years, and there are many lessons we've learned along the way:

> Forecasting the right free pool size is only part of the problem. Any practical solution needs to be responsive to the different VM provisioning times required by different clouds and regions.
>
> Minimizing the end-to-end time for provisioning new VMs allows for more frequent updates to the pool-size, which ultimately allows a closer fit to the actual customer demand.
>
> ıaı edictability increases as usage of our product grows, and thus, allows us to be more cost-effective.
>
> Many customers have large demand-spikes during the top of the hour from automated jobs. Our need to pre-provision the servers to be ready to satisfy that demand often pushes the public cloud providers to their limits and exposes bottlenecks in their infrastructure.
>
> Regional growth varies due to factors such as customer demand in each geographic location and customer cloud preferences. Working closely with our cloud providers to give them a long-term forecast of our resource demands across regions greatly eases long-term capacity constraints.
>
> In Virtual Private Snowflake (VPS) deployments, our top-tier offering that consist of a completely separate Snowflake environment for use by a single customer and its partners, the usage pattern tends to be reflective of the business hours in that location. This is significantly less pronounced in our public deployments.

Conclusion

This is just one example of the many types of efficiency work we do for optimizing cloud costs described at the beginning of this post. We have teams across the company focusing on many different levers to continue improving the efficiency and scalability of Snowflake for our customers. To learn more about Snowflake's current engineering efforts, follow us on our blog and on Medium.

SHARE

AUTHOR

Murray Stokely

Orestis Kostakis