keebo

Using AI to Save Money on Snowflake: A Practical Guide for the Data Team.

© 2024 Keebo. All Rights Reserved.

Use AI to Save Money on Snowflake

Introduction

In mid-2023, Snowflake found itself in the middle of some controversy about Instacart, one of its biggest customers. While the situation was interesting for pundits and those with particular interest in the Snowflake-Databricks rivalry, that's not what caught our eye here at Keebo. Rather, the discussion about reducing Snowflake costs is what drove our interest.

Keebo is the only fully-automated Snowflake warehouse optimizer. Our customers count on us every day to make their Snowflake credits go further as they face ever increasing complexity and demand for data and applications. So when Snowflake suddenly started showing great interest in helping their customers save money, we were intrigued.

Perhaps Snowflake's newfound appreciation for optimization has brought you to this article. If Instacart can save 50% on their Snowflake bill, and other customers are reporting similar results, then why shouldn't you benefit as well?

The problem is that Snowflake's advice on "optimizing costs" is tough for many of their customers to follow, and there should be an easier way to save money on Snowflake.

In this article, we will examine:

1. Snowflake's advice for reducing costs
2. Problems with Snowflake's optimization methodology
3. Using AI to automate Snowflake savings
4. How Keebo's AI works

© 2024 Keebo. All Rights Reserved.

Part 1: Snowflake's Advice for Reducing Costs

Snowflake offers data clouds on a consumption model. This has many advantages, including flexibility, speed, and scalability. Simply put, it is now easier than ever before to spin-up new data warehouses and applications. This is likely why you started using Snowflake in the first place. The issue with consumption models, though, is that you pay more the more you use them. Therefore, the key to optimizing costs is to make sure that you are running Snowflake as efficiently as possible.

Snowflake outlines their advice in a presentation from Snowflake Summit 2023 which is now available for anyone to watch upon registration. We can examine this advice by organizing it into 3 areas:

1. Snowflake optimization capabilities
2. The most important way to reduce costs
3. Snowflake's 3-step process for optimization

Snowflake Cost-saving Capabilities

Snowflake provides three general areas for reducing costs. Let's look at each and some of the practical steps you can take:

Visibility
Measure your usage

- **Account and usage data.** Snowflake has a robust set of usage data you can query. A good place to start is metering_daily_history.
- **Tagging.** Attaching tags on queries helps you attribute costs.
- **Budgets.** Set a spending limit so when you are close to exceeding it, you'll be notified.
- **Warehouse utilization.** Shows warehouse usage as a percentage so you can balance your workloads across warehouses.
- **Per-job cost attribution.** Allows you to understand spending better on a job basis.

To use these, you'll need to create your reports with Snowsight or similar tools. Or check out Snowflake's open-source Streamlit usage app to get started.

Control
Make adjustments

- **Warehouse size.** If your warehouse is too small, your queries can be too slow for your SLAs. But if it is too big, you are paying for power you don't need.
- **Warehouse suspend.** If a warehouse isn't in use, Snowflake can shut it down for you after a time interval you set. But take care you don't shut down too soon, since restarting a warehouse will slow down queries.
- **Multi-cluster.** While increasing warehouse size will help individual queries, increasing the number of clusters improves concurrency. But again, if you make this too big, you are paying for power you don't need. Too small and you might not get the performance your users need.

These three control parameters in particular are excellent candidates for AI and automation, as we will see in section 3 of this document.

Optimization
Use advanced features where possible

- **Materialized views.** Materialized views are a well-known concept that can improve query performance. But the decisions on what to build and when to build them are still left to admins. Requires Enterprise Edition.
- **Storage optimization.** Snowflake auto clustering can optimize storage to better answer queries by tuning how data is clustered.
- **Search optimization.** For specific types of queries, especially those from data science teams, this service can improve performance by essentially making search access paths persistent, allowing the pruning of micro partitions to be faster. Requires Enterprise Edition.

These are part of "serverless" services that, because they are serverless, are easier for Snowflake to automatically tune. But they may not work for every workload or budget.

© 2024 Keebo. All Rights Reserved.

The most important way to reduce Snowflake costs

All of this could be a lot to absorb, so it is helpful to understand the main optimization target. There are lots of things you can optimize in Snowflake, but for those just starting out to reduce Snowflake costs, the priority is clear: setting an optimal warehouse size. In case you aren't familiar with how warehouse sizes work in Snowflake, let's review briefly.

How Snowflake data warehouse sizes work

Size in this case does not refer to storage, but computing power. The larger the warehouse, the faster it can handle a compute task (query). In general, the bigger and more complicated your query workload, the larger your warehouse should be. It should be large enough to process queries in time to satisfy your users' requirements or SLAs. Snowflake currently offers the following "T-shirt" sizes:

Warehouse Size	Credits Per Hour (cost)
X-small	1
Small	2
Medium	4
Large	8
X-large	16
2X-large	32
3X-large	64
4X-large	128
5X-large	256
6X-large	512

Warehouse size and Snowflake cost

Did you notice that your cost will double with each increase in warehouse size? This is why every cost saving strategy prioritizes warehouse size. The basic idea is to pick a warehouse size that will efficiently process most of your queries and meet your response time SLA. Sizing based on your biggest query would mean you are paying too much, since most of the time you will be paying for an underutilized warehouse. On the other hand, if your warehouse is undersized, your queries could spill out of memory and need to access storage, either local (slow) or remote (very slow). The result of spillage is slow or even failed queries and the unhappy users that follow.

On the surface, this seems to be straightforward: increasing your warehouse size will mean queries run faster and hopefully fast enough to make up for the doubling of cost. But in reality there is more nuance. For more information on this, see part 3 of our series on Snowflake Optimization.

Since it is difficult to know exactly what warehouse size to choose, we should look at a process top Snowflake customers use to optimize warehouse size. A repeatable process is critical because setting a parameter like warehouse size is not a one-time thing. Your analytic workloads are dynamic, and the setting that you use today might not work tomorrow.

© 2024 Keebo. All Rights Reserved.

A 3-step process for Snowflake optimization

If you were to watch all the presentations on optimization done by Snowflake customers, you would find some common themes which we can summarize into a 3-step process.

Step 1: Write good queries

If your queries are not optimized to begin with, there is little you can do to optimize Snowflake and save money. It makes sense–an optimized query will require less computing power and thus a smaller, cheaper warehouse. So with an eye on optimizing warehouse size, we can categorize the query optimization advice like this:

- **Process as little data as possible.** We say this at the risk of stating the obvious. But the fewer columns you choose, the better for performance and thus for cost.
- **Avoid common SQL mistakes.** There are many different ways to achieve the same results with SQL. The technique you choose can use a lot of processing time and drive up your costs. We've compiled a list of the top 7 SQL mistakes we see Snowflake customers make.
- **Be careful with clustering.** Snowflake stores your data in micro-partitions. The more partitions you need to scan for your query, the slower and thus costlier the query becomes. By default, Snowflake stores your data in the order it was ingested. This results in a sort of "natural cluster" where all your recent data is grouped together by time. This works well as most queries are narrowing down the data (pruning) by a time period. So you should be hesitant to change this unless you have special circumstances, such as querying by event from streaming data

In addition to these basic categories, some Snowflake customers report great results from materialized views.

Step 2: Monitor key performance parameters

You've probably heard the classic management mantra: you can't manage what you don't measure. The customers that presented at Snowflake Summit in 2023 certainly took this to heart. Each of them created custom dashboards using Snowsight (the online SQL tool for Snowflake) to measure several different performance metrics. You could also use Snowflake's Query Profile that is available in the classic console. Again, with an eye toward warehouse size optimization, the most common query performance metrics are:

Bytes scanned	In general, the higher the bytes scanned to answer a query, the larger the warehouse needed. If most queries scan less than a GB you probably only need a small or extra small warehouse.
Execution time	If this is high, it indicates expensive (long-running) queries. You need to either optimize these queries or scale-up to a larger warehouse.
Query load percent	If this is less than 50 for a query, then it should be moved to another warehouse or you could increase the warehouse size.
Bytes spilled to local and bytes spilled to remote	This indicates the warehouse is undersized. Bytes that can't fit into memory "spill" to disk storage. This will be slow on local storage and even slower on remote storage.
Queued time	A query that is competing with others for resources will have a high queued time. Increase warehouse size or move queries to a less utilized warehouse.

There are many other parameters you can measure, and many customers look at big picture items too, such as credits used (spend) per warehouse and account. Snowflake Budgets is a great way to get started on this.

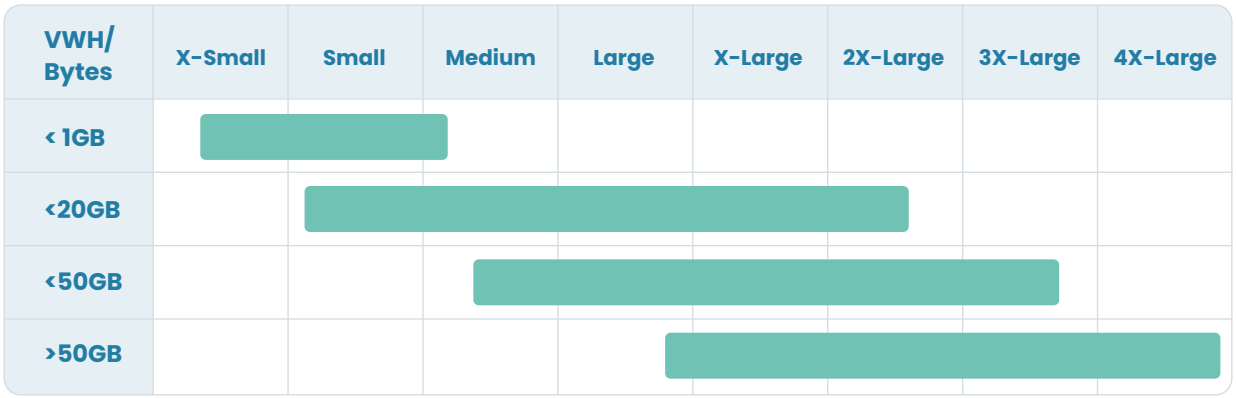
© 2024 Keebo. All Rights Reserved.

Step 3: Make warehouse size optimizations

Once you start tracking performance metrics, you have the data you need to begin making adjustments. Since warehouse size was always brought up as the most important optimization factor, let's look at that as our example here.

One of the key parameters to track is bytes scanned. You can then translate that to recommendations for warehouse size. The following chart has been shown in many presentations by Snowflake and their customers:

Warehouse size by bytes scanned recommendation chart



So if your bytes scanned is between 1-20GB, the recommendation is either a small, medium, large, x-large or 2x-large warehouse. Clearly, you are going to get more specific here. Unfortunately, the exact size you need can't be determined since query workloads vary wildly. Even in Snowflake's own documentation, the advice is to experiment until you find the ideal size: "Experiment by running the same queries against warehouses of multiple sizes (e.g. X-Large, Large, Medium). The queries you experiment with should be of a size and complexity that you know will typically complete within 5 to 10 minutes (or less)."

Any Google search on the subject of Snowflake warehouse sizing will reveal essentially the same advice. But you can cut down on the guesswork by building and monitoring a report that will show you key metrics like those mentioned above.

© 2024 Keebo. All Rights Reserved.

Part 2: Problems with Snowflake's Optimization Methodology

Snowflake's recommended optimization method isn't unique to them. Many enterprises, whether they use Snowflake or not, follow a similar methodology for optimizing the query workload across the entire company to reduce the cost of their shared infrastructure. In the IT world, there has always been a reactionary tendency to try and clean up the mess one query at a time and it seems the move to cloud data warehouses hasn't changed this common attitude at all, and indeed Snowflake's own advice starts with tracking queries. This leads directly to four problems.

Ignoring the long tail

A significant portion of Snowflake's approach is focused on identifying the most expensive queries. This tells us something important: even big organizations with massive data teams will find it prohibitive to look at all their queries. They can only focus on a small fraction of the queries by picking the most expensive ones. For example, if you can only afford manually inspecting 1,000 queries, then you obviously want to focus on your 1,000 slowest and most frequent queries. But in a large organization this means ignoring millions of smaller queries whose impact collectively on your overall bill is significant. In other words, the 80-20 rule doesn't always work when it comes to database workloads: there is often a long-tailed distribution of smaller, faster queries that because of their large numbers still count for a large portion of your overall bill. Imagine what the potential could be if you could analyze all the queries and optimize all of them, not just a fraction of the most expensive ones.

Relying on manual effort

Even for the queries that a data team flags as candidates for optimization, there still isn't a magic bullet to optimize them. Most data teams must depend on the users from across the enterprise to fix their problem queries. So essentially the job of optimizing the most expensive queries is now crowdsourced to all these different teams across the company instead of falling on the data team only. In a sense, this is fair and probably a good "culture". After all, these analysts are using a shared resource that is on a pay-as-you-go model, so they need to use it responsibly.

But the question is whether this is a good use of the company's resources, for a number of reasons:

- **Depends on expertise.** Not everyone who needs to use the database needs to be a database expert. Is it wise to ask a supply chain guru or a marketing operations expert to spend their time writing perfect queries, or optimize the queries they wrote months ago? Weren't these individuals recruited for a different skill set to perform a different business function for their company?
- **Hides the reliance on manual effort.** More importantly, distributing the manual work to almost everyone who is writing queries will certainly lessen the burden on any single individual and allows the company to "manually" optimize more queries, but it doesn't make it any less manual. You are still spending a lot of time on hand-tuning SQL queries, one query and one person at a time. That's a lot of money, even indirectly. In some ways this is even worse because now you cannot even track how much of the organization's resources are spent on this manual work. It's all hidden and distributed.

In some ways, the company is still paying the same tax for the manual optimizations, but more people are being taxed and some of them are paying an even higher tax rate because they are not trained, hired, or motivated for this line of work.

© 2024 Keebo. All Rights Reserved.

Not specific enough

A lot of optimization methodology depends on two tasks:

1. Finding the expensive and frequently-run queries that need to be examined by people.
2. Examining the query plan (the query profile in the case of Snowflake) for optimization inspiration.

But all this can do is reveal an expensive query and the expensive operator in that query. The team must examine it long enough to see what can be done to speed it up. But we all know how intimidating real-life queries can look. In many cases you cannot even see your entire query on the screen without having to scroll through multiple pages. Likewise, what if the query profile is simply showing you are scanning several terabytes of data? How do you expect your end user to solve that?

We need to provide better tooling and more specific advice to our end users if we're expecting them to "solve" cost and performance problems. Even still, in many cases it's not even possible to optimize your workload one query at a time. Instead, you should identify the common blocks of computation across many queries and find a way to reuse that computation (one of the things Keebo's query acceleration product does).

Dependent on trial and error

One of the most common ways of optimizing Snowflake (or cloud warehouses in general) is rightsizing your warehouses (see this article for more on this). The problem, as we described earlier in this document, is that even Snowflake's official advice is to choose the optimal size for your warehouse through trial-and-error. Their advice is to track bytes scanned for each query and then choose the appropriate size for your warehouse. Say your query scans less than 20GB. According to Snowflake's own advice, as shown in the table above, the optimal warehouse size for you might be small, medium, large, x-large, or even 2x-large. But which one? Well, you won't know or as it's found via "trial-and-error".

Imagine how many hours of experimentation will be required to find the right size. Even if you manage to get it right, there's no guarantee that it will work tomorrow or a week later when the workload changes. There is also no guarantee that the size that's optimal at 9 pm will also be optimal during the 9 am peak traffic. So what should we do then? Your data team has to keep repeating this tedious and error-prone process over and over again.

© 2024 Keebo. All Rights Reserved.

Part 3: Using AI to Optimize Snowflake

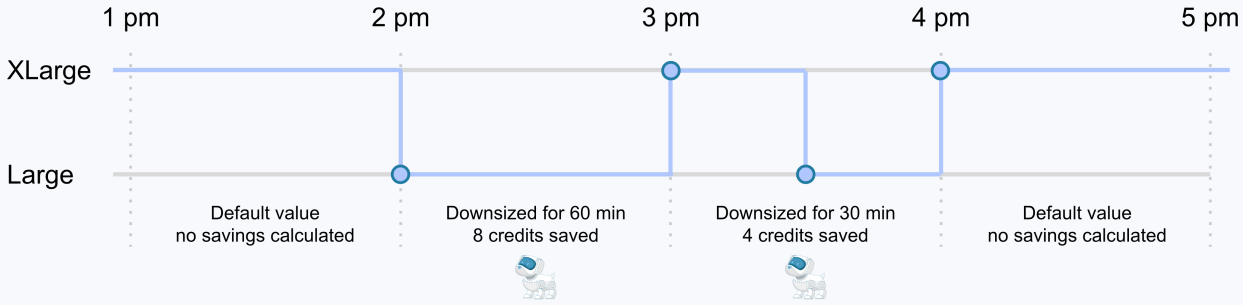
Even if you want to try and overcome all these problems and build a team to optimize Snowflake, you still face another challenge: you've only optimized to the workload you have now. When your workload changes, the optimizations your team spent so much time on could become irrelevant instantly.

How Machine Learning can Optimize Snowflake

The way to solve the resourcing issue and the challenge posed by a dynamic environment is to enlist your own army of robots. At Keebo, our mascot is a friendly robot dog, because this is what we do 24/7/365: optimize Snowflake warehouses. There are 3 factors you should consider when choosing a "robot" to optimize Snowflake for you:

1. It must always be watching.
2. It must make actual optimizations, not merely suggestions.
3. It should align with your goals, and be priced based on your savings.

Let's examine how Keebo works and see how it can fulfill all of these requirements. Recall from earlier in this document that the most important parameter to optimize is warehouse size. So let's begin with that. Take as an example, a Snowflake warehouse during several horse of use with a default size of extra large:



With our patented algorithms, Keebo will constantly use over 70 different parameters to determine what the most efficient warehouse size should be. In this example, at 2 pm, Keebo determined that the warehouse could be downsized from extra large to large for 60 minutes. Then Keebo actually made that change to the warehouse size. At 3 pm, an increasing volume of queries (or size of queries) meant that Keebo increased the size back to extra large so users would not have a performance slowdown. Then again a downsize at 3:30 pm and so on.

In total, 12 credits were saved between 1 and 5 pm. To calculate this, we use the following formula:

Default size - downsize * (hours downsized) = savings

Remember from the table earlier in this document that an extra large warehouse costs 16 credits per hour and a large warehouse is only 8 credits per hour. If we plug this into our formula, we get:

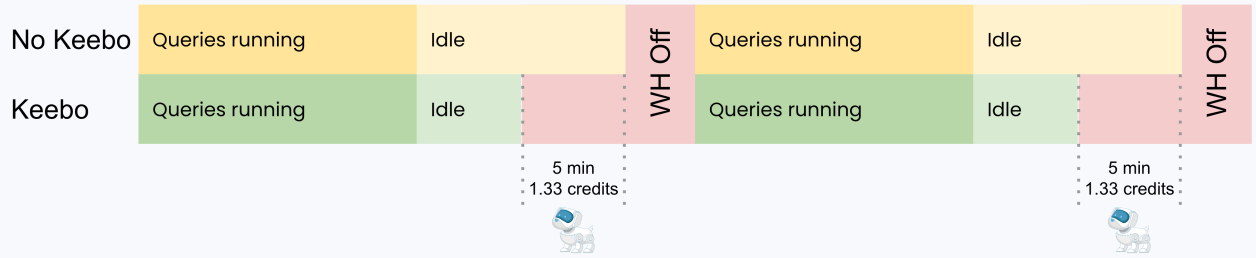
16 - 8 * (1.5) = 12

© 2024 Keebo. All Rights Reserved.

So, could a person accomplish the same thing? In theory, yes. You could assign someone to watch the warehouse size of this particular warehouse and then make the adjustment at 2 pm. But can this person evaluate over 70 parameters to get exactly the right setting at the right time? Not likely. And what if this person made a mistake or wasn't around when query traffic increases at 3 pm? Hopefully they can be reached. Keebo's patented machine learning, on the other hand, will notice and make the adjustment.

When you consider all this, it is easy to see why the solution you choose for optimization should be capable of always watching and can actually make the optimization settings automatically. There are many products on the market today that claim to do automatic Snowflake optimization, but all they really do is send you reports and recommendations. They still depend on a person to make the actual changes and then be present to fix any issues. And to this point, we've only been talking about a 4 hour stretch of time for one single warehouse. Imagine this happening 24x7 and for dozens of warehouses. You simply must have a solution that operates automatically.

Recall from earlier in this document that the second most important parameter for Snowflake optimization is warehouse suspend. If a warehouse isn't in use, Snowflake can shut it down for you (suspend) after a time interval you set. But take care you don't shut down too soon, since restarting a warehouse will slow down queries. So let's consider another example of an extra large warehouse, this time using the automatic suspend interval:



Let's focus on the top line first, labeled "No Keebo." Here the default suspend parameter is 10 minutes. In other words, after a 10 minute period with no activity, Snowflake will shut down the warehouse. But what if, as is shown in the second line, Keebo can determine that the suspend interval can safely be set to 5 minutes? In this case you would save 2.66 credits from that extra time. Here's the formula we use to calculate this:

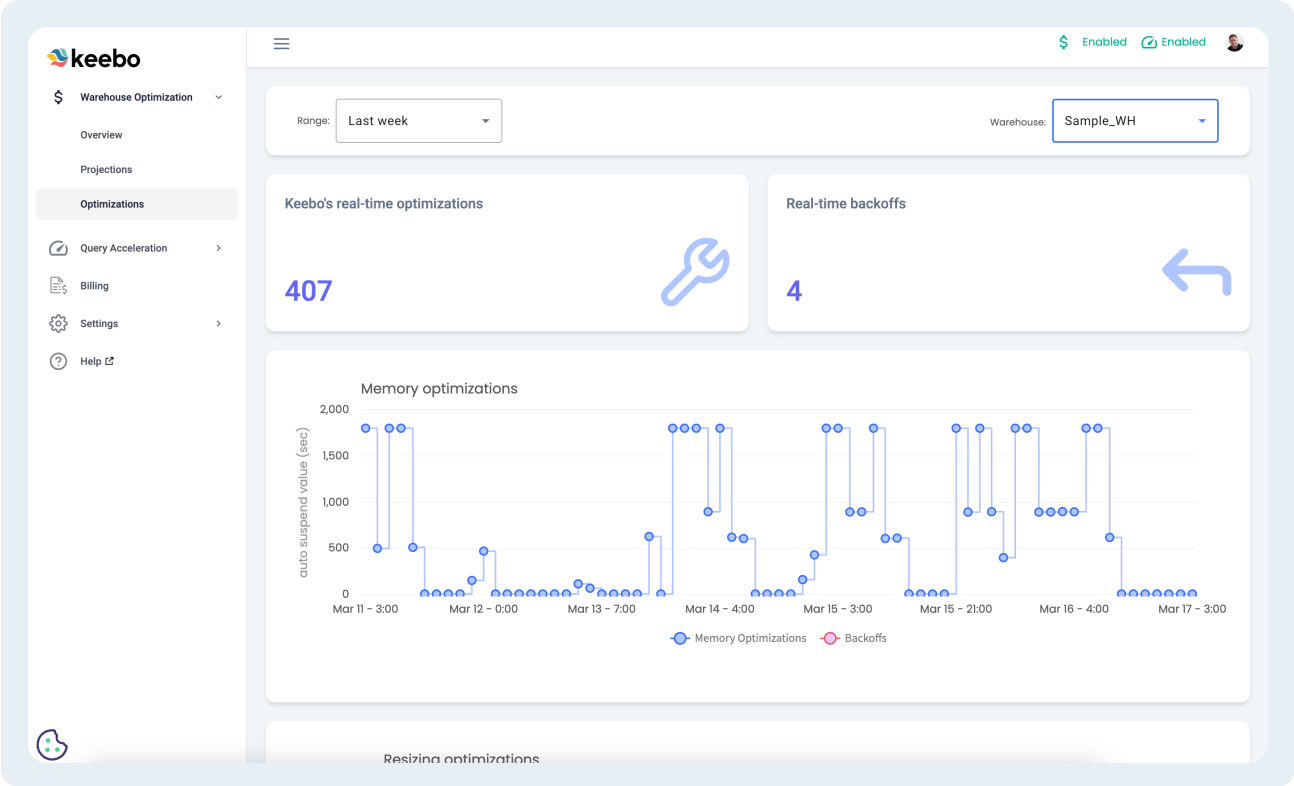
Warehouse cost in credits * (extra minutes of suspend time / 60) = savings

Since an extra large warehouse costs 16 credits per hour to run, we can then compute:

16 * (10 / 60) = 2.66

© 2024 Keebo. All Rights Reserved.

Initially, a dozen credits here and there doesn't seem like a big deal. But added up over time and across all warehouses, the savings could be substantial. Keebo customers have reported savings of 20% and more. Again, a person devoted to this task could theoretically achieve these savings. But as we've already discussed, this is not sustainable over a long period of time and for many warehouses. Keebo is working constantly to identify and implement cost-saving measures, and you can always check on what it is doing. Here is an actual customer screenshot showing optimization of warehouse suspend:



Even though a machine can do this work more efficiently and accurately than a person, it may not be worth it if the machine is too expensive. This is where the third requirement comes into play: the optimization solution must be priced in a way that is aligned with your goals. Keebo's pricing model is unique. Rather than charge a flat fee, or base pricing on your Snowflake spending, Keebo charges a percentage of savings. Saving you more money is good for both of us.

© 2024 Keebo. All Rights Reserved.

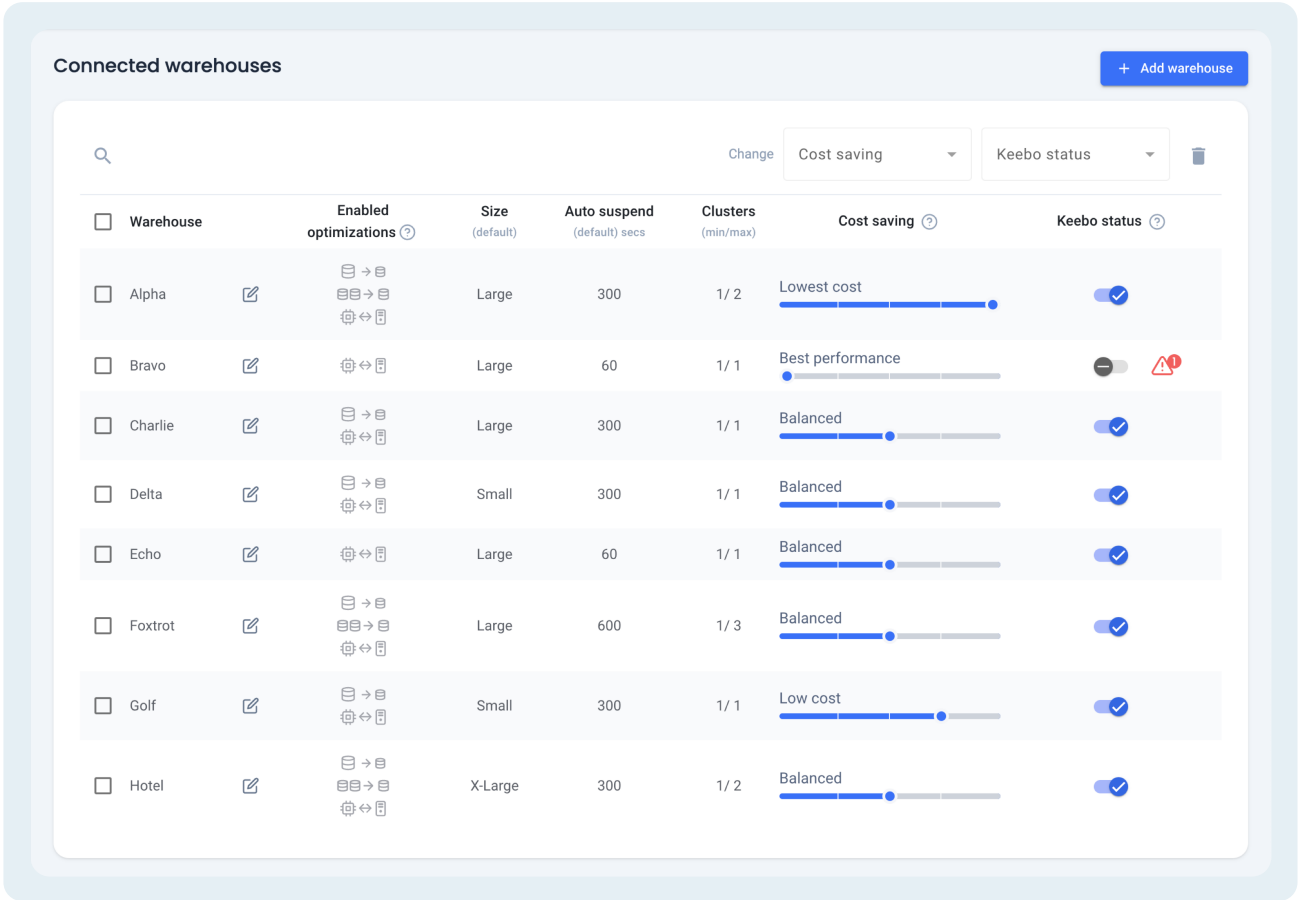
Getting Started with Keebo

If enlisting our patented machine learning technology is interesting, you can get started very quickly. Keebo is cloud-based and it only requires access to your usage metadata, so there is no need for an extensive security review. The basic steps to set up Keebo take only about 30 minutes:

1. Create a Snowflake account for Keebo that has the ability to modify warehouses.
2. Create a schema for Keebo that contains only the usage metadata we need to see. This means that Keebo can only see this metadata and never any user data.
3. Add members of your data team to the Keebo portal so you can all track optimizations and savings.

After a couple weeks of learning and optimizing, we will know how much we can save you in direct Snowflake costs.

Keebo is automatic and adjusts to changing conditions in real time, so there is no need to monitor it. But this does not mean you don't have any control. In fact, you can adjust our algorithms for complex scenarios and schedules, and Keebo is fully auditable (see this article for details). This screenshot shows the various controls you have per warehouse, including our slider bar to help balance cost savings with query performance:



© 2024 Keebo. All Rights Reserved.

Part 4: How Keebo's AI Works

You might be wondering how this all works. After all, we are making some big promises here: fully automated optimization that requires no human involvement and is priced based on savings. Perhaps you've seen other "optimizers" that claim they are fully automated. In this final section, we'll describe how our AI works–a look behind the scenes so you can see exactly why Keebo is unique.

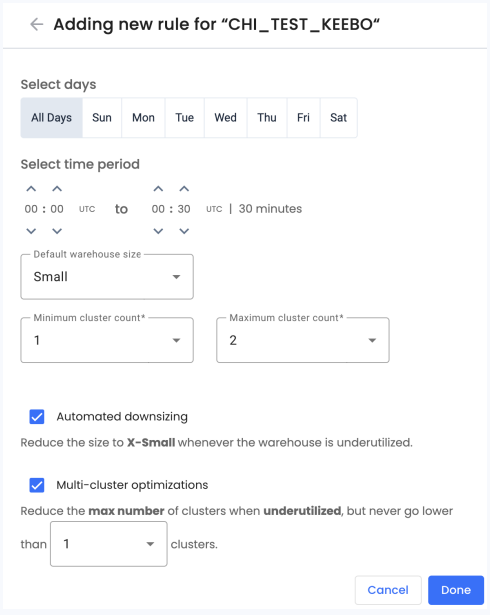
Core Components

At the core of Keebo is data learning, our patented machine learning which continuously trains specific models for warehouse optimization. Specifically, we train our models on performance telemetry data provided by Snowflake (the 70+ fields mentioned earlier), such as the number of queries, their arrival, queuing, completion time, number of bytes scanned, etc. This means we do not train our models on any customer or business data, nor do we even need to see the query text itself. Most importantly, smart models improve over time as they constantly train on additional data and learn from how past actions impacted cost and performance. This means Keebo can "backoff" or autocorrect if we notice a change that is intended to save costs but might hurt query performance too much.

Each customer warehouse gets a unique smart model which makes real-time decisions and actions for that warehouse. In addition to the historical behavior learned from the telemetry data, smart models take three additional inputs into account when deciding what action, if any, to take next:

1. Cost model. The predicted impact of each decision on cost and performance.
2. Constraints and trade-offs. These are set by the customer using the slider bar control shown earlier and other business requirements set by rules, such as prohibiting warehouse downsizing during specific days or times.
3. Real-time feedback. As we monitor in real-time, Keebo considers increased latencies, new query patterns, and sudden spikes in the workload.

Here is the dialog in Keebo where data engineers can set a rule constraining the actions of smart models:



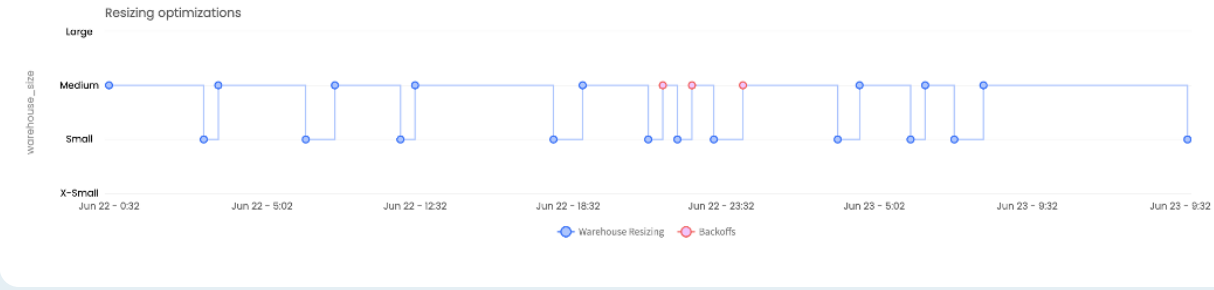
© 2024 Keebo. All Rights Reserved.

Keebo constantly monitors warehouse performance metrics (such as query latency and queue times) to assess the impact of actions taken and provide feedback to the smart models so they adjust their next actions accordingly. This means smart models are aware of sudden spikes and changes to workload, something nearly impossible for humans to track all the time.

All this learning, modeling, and monitoring ultimately leads to an action. This is where Keebo truly distinguishes itself from any other product that claims to "optimize" Snowflake. Keebo's actuator translates the decisions of the smart model into the API of Snowflake. Most actions turn into an ALTER WAREHOUSE command. For example, the following query changes a warehouse called "SALES_WH" to medium size:

```
ALTER WAREHOUSE SALES_WH
SET WAREHOUSE_SIZE=MEDIUM;
```

Every action taken by the actuator is recorded and then displayed in the Optimizations tab of Keebo so you can always keep track of what is happening:



In this example, taken from an actual customer system, we see resizing happening from medium to small and back again, informed by all the data learning that has taken place. Notice the three backoffs (indicated by a red dot) where Keebo determined that a sudden change in workload meant the size should be reset to default, something that a busy data engineer could never keep up with on their own, especially during off hours. This is a benefit of a solution that can take actions rather than just make recommendations.

© 2024 Keebo. All Rights Reserved.

Warehouse Cost Model

Recall that you should look to meet three requirements when choosing a Snowflake optimization solution:

1. It must always be watching.
2. It must make actual optimizations, not merely suggestions.
3. It should align with your goals, and be priced based on your savings.

The previous section of this document describes how our smart models accomplish the first two requirements. Keebo's warehouse cost model is how we meet the third requirement. This is not easy to accomplish, which is why every other "optimization" product can only make recommendations for which they charge a flat fee or percentage of some other easy to identify factor, like overall Snowflake spend.

Keebo, on the other hand, compares the state of the optimized warehouse to its prior unoptimized state. We call this with-Keebo and without-Keebo, respectively. This allows Keebo to present the difference between the two states as savings for the customer, and base our pricing on that. To accomplish this, our cost model replays queries to arrive at a savings estimation for each warehouse.

Keebo runs a *what if analysis* replaying the queries in the workload and comparing the without-Keebo and with-Keebo costs. The with-Keebo costs are easy to get because we can obtain that directly from the Snowflake billing data for the period Keebo was active. Since Snowflake bills hourly for per-second usage, our query replay iterates over all queries run during each hour and computes the number of seconds the warehouse was active in that hour. At each step during the replay, we consider the customer's original settings of auto-suspend, warehouse size (and thus hourly rate), and cluster scaling to compare with Keebo's optimized state.

This is an advanced analysis, recognizing that merely changing a warehouse parameter may not affect how many seconds a warehouse is active. So we consider query arrival times and parallelism in our estimations as well. Further, we scale the execution time of each query according to the historical behavior we have observed for that query using our regression model.

The result is that query replay bases our cost estimation on the actual query processing and billing behavior of the underlying warehouse, thereby scaling as workloads change over time. All of the activity is presented to the customer to make it transparent and explainable. Overall, you get a bill for optimization that is based on what we save you, and this keeps us aligned with your goals.

© 2024 Keebo. All Rights Reserved.

Data Learning Algorithm

All of this is wrapped-up in our data learning algorithm, based on query and billing history (the telemetry data). You can see how all of the items described above, especially slider positions, constraints, actions, and savings estimations coming together:

Algorithm 1 Data Learning Algorithm

```
1:  Input
2:  aggr : The slider position (i.e., aggressiveness)
3:  wh : warehouse name
4:  WCM: warehouse cost model trained for wh
5:  UC : user constraints
6:  T : frequency of collecting detailed telemetry data
7:  T_realtime : frequency of checking real-time performance
8:  Initialization
9:  D ← ReadTelemetryData(last 90 days)
10: action[0] ← null, reward[0] ← null
11: i ← i null
12: while true do
13:     if T hours have elapsed since last training then
14:         D ← D ⋃ ReadTelemetryData(last T hours)
15:         M ← TrainSmartModel(D, wh, aggr, WCM)
16:     end if
17:     if T_realtime mins have elapsed since last action then
18:         feedback[i-1] ← Monitoring.RealTimeState()
19:         action[i] ← M.nextAction(UC, WCM, feedback[i-1])
20:         Actuator.apply(wh, action[i])
21:         i ← i + 1
22:     end if
23:     savings ← cm.estimateSavings(action[], feedback[])
24:     report(action[], feedback[], savings)
25: end while
```

There is more to the story than this, but it is beyond the scope of this paper to dive into the details of the deep reinforcement learning we use (you can see all the details if you'd like in our peer-reviewed paper). For now, let's summarize how data learning works in Keebo.

Keebo's deep reinforcement learning is designed to:

1. Decide on workload-specific optimizations
2. Quickly react to unexpected changes in the workload

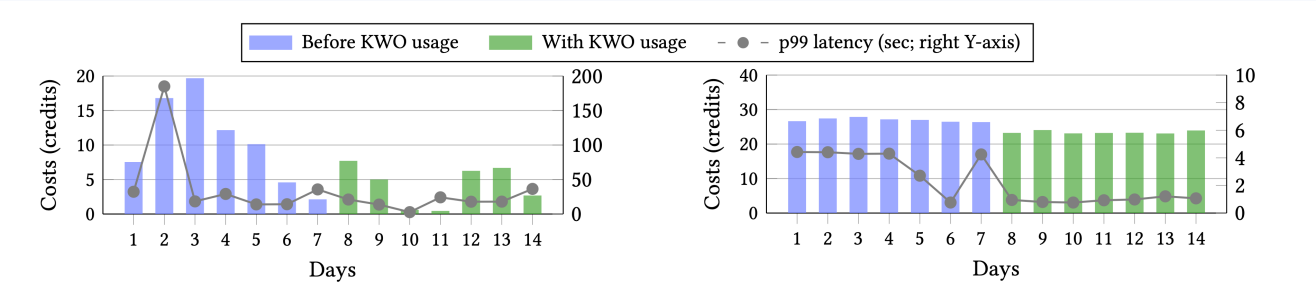
As shown in lines 13-16 of the algorithm, we periodically retrain our neural network using the last 90 days of data. This is augmented with the estimated impact of various actions from our cost model (line 4) and the slider position (line 2) to achieve a balance between costs and performance. In other words, for each action, Keebo decides if it is worth taking that action right now. If we suspend a warehouse too soon, for example, there may be a performance penalty experienced by subsequent queries as those queries wait for the warehouse to "spin up" again. But if a customer wants to prioritize cost savings, they can set the slider bar to be more aggressive, meaning Keebo will suspend a warehouse even if there is a slight risk of a performance hit.

This illustrates the key difference between Keebo and other "optimizers" on the market, or trying to do the optimization yourself. When you consider that Keebo is constantly reinforcing its models, constantly considering the cost of each action, and then making the appropriate decision (and backing off that decision if necessary), you can see how Keebo achieves the goals you have better than any other way. Keebo is the first and only application of machine learning in pursuit of fully automatic Snowflake optimization.

© 2024 Keebo. All Rights Reserved.

Results

This all sounds innovative and interesting. But does it work? You can read the results of our own testing here, but to summarize, Keebo saves Snowflake credits for both predictable and unpredictable workloads without increasing query latency. In fact, the more unpredictable the workload, the better the savings.



While the results above are from lab testing, we see similar and even better results with real-world customers.

DR.SQUATCH — Soap Co. —

34% Snowflake savings, no query slowdown

BARSTOOL SPORTS®

34% Snowflake savings, 70% overall application savings

TeamVelocity®

36% Snowflake savings, first results in 24 hours

PAYJOY

21% Snowflake savings, even with 37% increase in queries

(h[s])® HYPERSCIENCE

50% Snowflake savings with no effort from the data team

Of course, there is nothing like seeing results with your own Snowflake warehouses. Keebo offers a free, two week trial. It takes less than an hour to set up, and you'll start seeing results in a couple days. Contact us for a personalized demonstration and to set up your free trial.

© 2024 Keebo. All Rights Reserved.