Challenge: Accelerate data analytics while optimizing costs to meet increasing demand

Executive Summary

Liftopia is a commerce platform that provides pricing, ecommerce and distribution solutions for high volume ticketing businesses, and is used by over 120 ski lift companies, water parks and other attraction providers worldwide. They also operate their own ski portal, Liftopia.com to help their ski clients find customers. In order to manage and analyze their customer data, they use Amazon Redshift to centralize data across multiple sources, and Looker as an analytics solution provider for themselves and their partners. As Liftopia's partner base increased and reporting needs expanded, the performance of their end user experience began to be impacted and Service Level Objectives for report rendering were not being met. With peak ski season looming, they needed help fast to stay ahead of the impending wave of demand so they could deliver performant insights to their partners. Bytecode IO was engaged to analyze and optimize their Redshift implementation and Looker queries, so Liftopia could keep delivering value to their partners in a timely fashion and optimize costs, even as demand increased.

Customer Challenge

The challenges for Liftopia are multifaceted. First, they needed analysis and tuning of Redshift implementation. Do they have sufficient disk space, processing capability and is their workload management set up in the most efficient way? In addition, analysis time needed to be spent on the structure of the data and the Looker queries themselves. With Liftopia developer resources busy focusing on their application, they didn't have the bandwidth needed to tune both Redshift and Looker.

What Liftopia needed was:

- A trusted partner that could come in quickly and get their analytics stack back in shape;
- A partner skilled in Redshift best practices along with a deep understanding of how to increase the efficiency of Looker queries, in order to significantly improve the end user experience.

Why AWS and Bytecode IO

Liftopia moved to AWS early on, as it provided the most scalable, secure and reliable base on which to build their business. When Redshift was released as a low administrative and affordable data warehouse, it was a natural fit on which to build Liftopia's analytics platform.

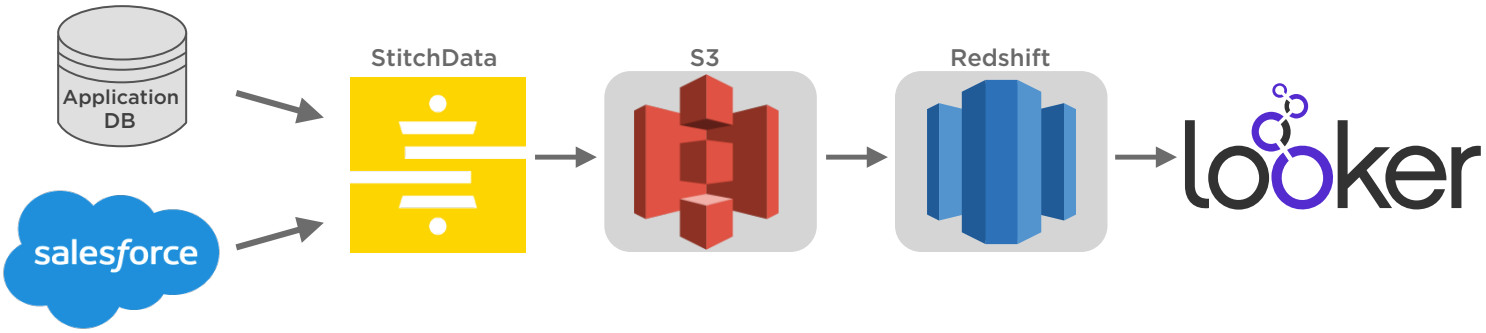
Liftopia leverages several Amazon Web Services:

Simple Storage Service (S3) - A flexible way to store and retrieve data, providing Liftopia with cost optimization, access control, and compliance.

Redshift - Is a scalable and cost effective data warehouse store that has already proven itself as a low administration analytics database for Liftopia.

Bytecode IO was initially brought in to augment Liftopia's DevOps team. As a Looker Services Partner already familiar with Liftopia's data, Bytecode IO was primed to augment Liftopia's analysis team when Looker was implemented.

Bytecode IO brought to Liftopia the prior experience of a dozen Redshift performance tunings, half of which included Looker as the primary BI tool. Bytecode IO was familiar with the areas where performance is often compromised, and had an arsenal of tools to remedy them. Bytecode IO was also engaged with Liftopia in enhancing Looker reports, so it was an opportune time to make adjustments to that code.



Bytecode IO's Performance Solutions for Liftopia

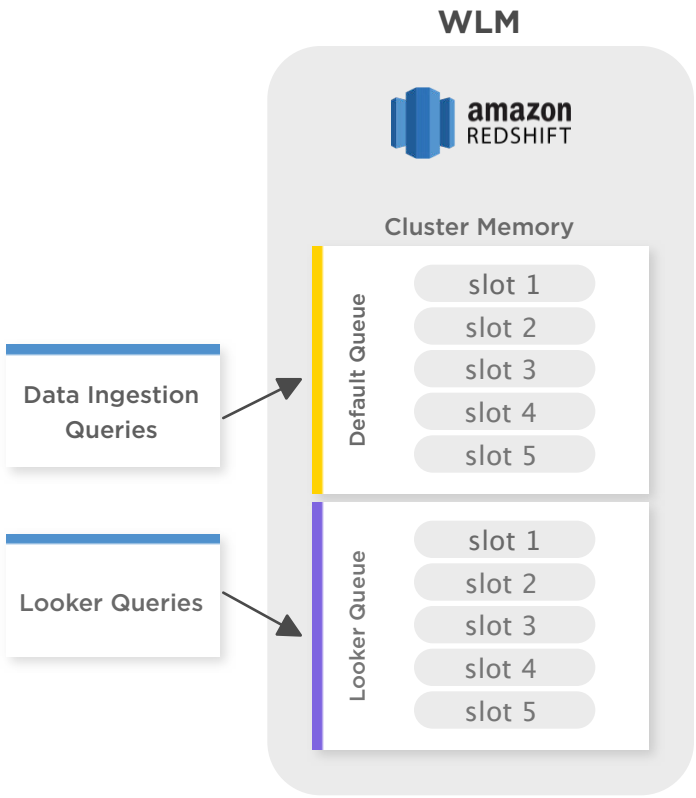
Bytecode IO's approach to taking on complex challenges like these is methodical, logical and data driven. Starting with the analysis of the Redshift implementation and then working forward to the data and queries, no opportunity for optimization is missed.

1 Bytecode IO's first step in a Redshift performance evaluation is to examine the cluster configuration for right-sizing to ensure performance efficiency. In sizing a cluster, we start with an estimated need of storage capacity, since the amount of storage available per node of the cluster is a fixed amount. AWS guidelines and user experience shows that performance can suffer when space becomes tight (greater than 80%), since disk can be used for temporary tables that don't fit in memory.

When sizing the cluster for a specific capacity, an extra 20% will need to be tacked onto calculations. In Liftopia's configuration, the cluster consists of four dc2.large with an average disk space utilization of 40%, with only very occasional temporary spikes up to 60%. Considering that data growth is around 3% per month, that gives Liftopia a full year of headroom, even with no data archive policy. So, this analysis showed that Liftopia's cluster has enough space and is not contributing to the slow down that end users are experiencing.

2 For sizing analysis, Bytecode IO also considers performance. Through the UI, we saw the CPU utilization averaged around 20% utilization, but compute nodes spiked to 100% hourly. While performance issues can be mitigated by other approaches described below, we determined the cluster could benefit from the parallelization of adding another node. In discussion with Liftopia, we found that their Service Level Objective (SLO) was to have all queries in Looker return within 20 seconds. At the end of our engagement, after we had applied the other approaches, we were still seeing between 20-25% of queries going over 20 seconds of run time (execution plus queue time); so we made the recommendation that another node be added. The four existing nodes were not reserved. If another node was added to the cluster and they reserved them all, the overall cluster cost would remain about the same. During our analysis, we also discovered that usage trends were largely seasonal, with daily activity starting to rise in mid November. An extra node or two could be added then, and removed when reporting activity dies down in April. The extra node could be removed until the following November or December to optimize cost while ensuring optimal performance.

3 Once it is determined that the cluster size and performance capability is correct, Bytecode IO's approach becomes iterative and data driven; implementing logical changes, analyzing results and taking appropriate steps from there. For example, the next configuration element considered was the efficiency of the WorkLoad Management (WLM) system. Right away, it was observed that the WLM configuration was set to the default with a single queue having 5 concurrency slots. In querying the STL_WLM_QUERY table, we noticed that there was a huge amount of waiting for slots to become available that far exceeds the actual execution time of queries. So as a first step, the WLM mode was changed to automatic, with Short Query Acceleration enabled. The impact was immediate, with an average of 25% (~6 seconds) reduction in query times. This change was allowed to settle in so that a useful range of usage metrics could be collected as work continued in other areas. Coming back to take a deeper look, it was found that almost all of the queueing was associated with Looker queries.



The WLM mode was switched back from automatic mode to manual to give more control, and Looker was given its own queue, letting everything else fall under the default queue.

The Liftopia cluster is also hit fairly hard by data loading processes from data integrators. Since other user queries outside of Looker are pretty minimal, those other users could just use the same default queue as the ETL processes, minimizing the siloing of memory and concurrency resources and maximizing memory per slot. This approach was validated by followup monitoring of the WLM queueing. It was also noted that there were only a very small percentage of disk based queries where there wasn't enough memory in the slot and some temporary data gets written to disk. This reinforces us seeing only small disk usage spikes. Since adding a new queue and increasing the overall concurrency cuts the memory per slot, Liftopia was informed that we would need to monitor and adjust as required.

Subsequent followup showed that there was still headroom which allowed us to increase the Looker queue concurrency even more. At the end of the adjustments, it was observed that disk based queries in the Looker queue were up to around 11%, which leads to the conclusion we should be hesitant to further increase the concurrency. Since the DC nodes are using SSDs, the disk based issue is not as big a problem compared to using DS nodes, but it still can degrade performance as data is moved in and out of memory.

After applying WLM changes and addressing some table structure and query issues, queueing was still an issue. Queries executing over 20 seconds were spending 1/4 of their time waiting in-queue. Redshift recently rolled out concurrency scaling where an additional cluster that handles queued queries gets spun up during times of high concurrency. It was considered as an option to alleviate queuing issues occuring during high usage spikes over the course of the day. Redshift credits a free hour of the concurrency cluster for each hour the regular cluster runs. However, there was hesitation to enable this configuration at Liftopia because there was queuing such that concurrency scaling usage could top 30 hours per month, and there was no room in the budget for increased database cost. There was a way to test this scenario out without a risk of added cost by enabling it for a week while monitoring it for usage. Adding query monitoring rules to the Looker queue could also trim the concurrency cluster usage to service only the most egregious of SLO offenders. Since the configuration change is dynamic, these tests could be done without interruption to users.

With Redshift configurations understood and recommendations in place and being monitored, the next step is to turn to the structure of the data. Performance optimization will often come down to tuning tables and queries. Detailed analysis brought to light that very few tables had sort or distribution keys. Bytecode IO then identified the top 30 costliest queries in terms of total seconds running and total count, noting that the top five dominated cluster resources by an order of magnitude. Structural rebuilds of the tables in those queries were executed, adding sort and distribution keys as well as column encodings. A 15% reduction in query execution times resulted from this effort.

The next step of analysis was to focus on Looker and how the Looker models need to be tuned to be performant on Redshift. In this case, there were key changes made in the way Looker queries Redshift. There were several widely used derived views which consist of a query of a single table or more. The query gets written by the Looker SQL compiler as a Common Table Expression (CTE), which may also then be embedded within nested queries. Given that Liftopia only needs a data refresh rate of

15 minutes, these CTEs don't need to be queried in real time, but can be pre-run. Looker allows this ETL type solution to be performed right within the BI tool using a feature called Persistent Derived Tables (PDT), a quick and easy mechanism to make those changes without a large developer effort. The PDTs can also be configured in Looker to create appropriate sort and distribution keys. The end result of this effort reduced query times that used these derived views over 20%.

Although progress had been made on the efficiency of individual queries, the number of queries had steadily increased 5% per week. Improvements were being overshadowed during peak hours. A survey of the top 30 costliest queries showed that when tested individually, most run within the SLO of 20 seconds. When multiple queries run together, as occurs when a Looker dashboard is refreshed, there is substantial resource contention which causes them to run beyond that 20 second window. Additional analysis showed that more than half of the 30 queries, and about 1/3 of all queries run against Redshift, had joins that could be made 30-50% more performant by denormalization of 5 tables into a single wide table. Using a PDT, we can schedule the creation of this flat table (which takes around 30 seconds) to occur every 15 minutes. The query to join these tables would be run only 100 times per day instead of 1000 times per day, thus relieving a lot of pressure on the database. This recommendation is currently being implemented, and Bytecode IO will monitor its effectiveness and take the next data driven decision to continue implementing improvements.

The Benefits

Bytecode IO worked with Liftopia specifically on performance for six weeks, honing queries, tuning tables, setting up WLM and adjusting it, and testing the performance. With each step of the engagement, we were able to impart understanding of the tuning steps and best practice patterns within both Redshift and Looker to the Liftopia team. Bytecode IO has architected the next steps which should enable Liftopia to reach their 20 second goal while minimizing cost for the additional performance. Bytecode IO's DevOps group and Looker developers will continue to work with Liftopia, providing the documentation needed to move forward with those steps and help Liftopia continue to successfully grow their business and deliver positive results.