ROCKSET

Increasing the ROI of Real-Time Analytics

Summary:
Achieving the speed and scale required of real-time analytics has been a complex and costly endeavour for many engineering teams. In this whitepaper, we discuss the challenges implementing customer-facing analytics in applications including meeting latency requirements, handling constantly changing semi-structured data and managing distributed data systems. We share how the implementation challenges on existing systems limit the return on investment and offer a new, simplified route to real-time analytics. Learn how engineering teams are increasing their ROI with a real-time analytics solution designed for low latency, data flexibility and minimal operations.

Table of Contents

Introduction

Technology companies are faced with an increasingly challenging predicament- their data infrastructure cannot meet the speed and scale required for analytics in their applications. This is happening at a time when users are demanding more access to data to enhance their productivity and automate decision-making.

Companies with larger budgets attempt to meet the analytics requirements by making sizable investments in data and operations teams. These teams are tasked with the tedious, time-consuming process of structuring their data and queries in a way to afford the greatest performance gains at the lowest possible cost. Not only do these teams invest heavily in performance engineering, they also take on the management of big data solutions that require deep expertise for controlling costs at scale. In the end, each new analytics offering takes considerable time to develop and ever-increasing human expense to maintain. Every new feature or change to an existing feature must meet a high value bar to the end customer in order to justify the expense, limiting the iteration and exploration of new offerings.

To meet the demands of data applications while conserving human energy requires a new approach to data- an approach where speed is paramount, flexibility is ingrained and operations is minimized.

Why Real-Time Analytics is Hard

Challenging to attain sub-second latency queries on terabytes of data

As application load and data size increases, companies are challenged to meet the low latency requirements of complex analytics. This only becomes more unwieldy as applications expand their analytics capabilities, providing ad-hoc slicing and dicing with customer-facing analytics or automating actions using real-time data.

Many teams start by building real-time analytics on their operational databases but encounter performance issues at scale. Operational databases use indexes to speed up queries and there are high costs to building and maintaining indexes when the underlying data is modified. Every new index added slows down the insert performance, so operational databases quickly become limiting as to the set of queries they can run efficiently. Thus, teams walk a tightrope of ensuring that they are indexing just enough but not too much on their operational database.

When analytics starts to hinder database performance, many teams employ read replicas or offload queries to another analytics solution. With read replicas, teams use the existing data model from their primary system. This is not a viable solution if the required queries are not already performing adequately on the primary system. Read replicas are efficient at removing load from the primary system but not in expanding the analytical capabilities of applications.

Some teams will attempt to transition these workloads to a data warehouse but face limitations on speed and costs. Warehouses resort to columnar storage formats and scan based query processing. To execute any query, entire columns need to be scanned. This results in slow queries which usually require spending more compute to accelerate.

In all these alternative systems, you either have to selectively build and maintain indexes, squeeze performance using read replicas or offload queries to another solution not designed for speed. This impedes engineering teams from embracing real-time analytics in their applications.

Data warehouses were not built for low data latency

Data warehouses were built for a batch oriented world, where the goal was to efficiently store large datasets for infrequent trend analysis. In these analytics systems, data is queued and loaded in increments with data delays in minutes to hours.

Reducing the data latency of warehouses is expensive due to how the data is stored. The column oriented storage means that every insertion of a new record needs multiple operations to write to all the separately stored columns, making it an inefficient system for processing updates. This makes it costly in terms of time and compute.

To achieve low data latency, the analytics system must also be designed to handle high velocity data from applications or devices without impacting query performance. In many architectures, ingest compute and query compute pull from the same compute pool, so a spike in new data can cause query performance to suffer and vice versa.

To meet the low data latency requirements, the system has to efficiently process updates and control for sudden bursts in data or load.

Growth in constantly changing, semi-structured data

Applications are generating more and more data in a variety of modern formats including JSON, XML and Parquet.

Despite the growth in semi-structured data, many analytics systems require the data schema to be well defined. Data pipelines are constructed to clean, homogenize and flatten semi-structured data into a structured format or denormalize data so that it can be made queryable by the analytics system. Data pipelines add to data latency. They are also time-consuming, manual processes that data teams need to build and maintain to make the data queryable. Data pipelines also need to be reconfigured every time the data or schema changes to ensure that the query meets the latency required of the application. This ongoing performance tuning impacts development time, elongating the time it takes for teams to iterate on their application.

Real-time analytics requires a system that limits the use of pipelines and provides native support for semi-structured data formats so as to reduce latency and development time.

Complexity of operating distributed data systems

Many engineers who have managed distributed data systems tell us they don't want to do it again. Even so-called fully managed systems still require you to think about servers and clusters.

Managing a distributed data system for real-time analytics like Elasticsearch or Druid requires operational expertise. There are many different levels of the system that need to be managed- the server, operating system, network and software. Maintaining an intricate knowledge of each element of the system, especially as the number of nodes increases, can be challenging to keep up with. And, the chances of something going awry increases at scale.

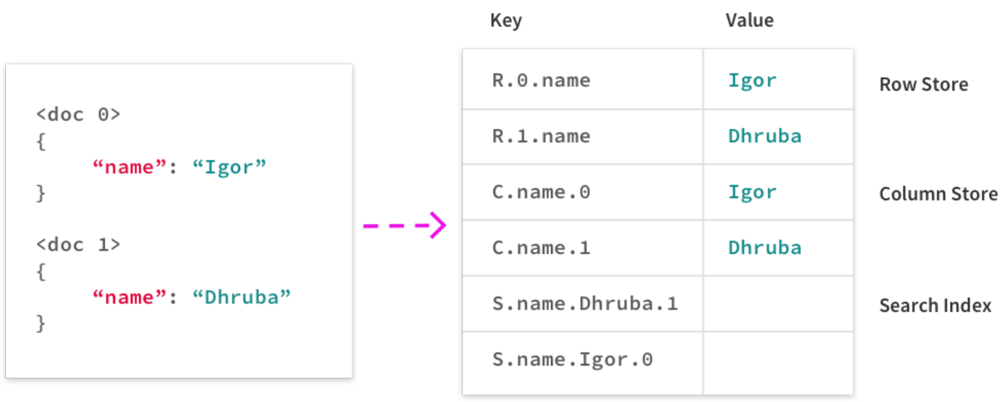
Many teams that use Elasticsearch or Druid will opt for the cloud solution. Cloud offerings of Elasticsearch and Druid still require clusters and nodes to be managed and resources to be provisioned for peak capacity. These systems tightly couple compute and storage resources so you cannot scale resources independently, leading to inefficient resource utilization. As these systems require time for additional resources to be spun up and made available to users, any miscalculations in resource provisioning can negatively impact the user experience or result in unnecessary expense.

The ongoing operations of managing distributed data systems have dissuaded some teams with more limited budgets and headcounts from investing in real-time analytics.

How Rockset Makes It Easy

Index Everything for Compute Efficient Queries

Rockset embraced indexing and introduced the Converged Index, a search index, columnar index and row index on all fields. Rockset employed RocksDB, a write-optimized LSM-based storage engine, to make it possible to index everything cheaply. With Rockset's Converged Index, teams do not need to build or manage any indexes. The indexes can be exploited in parallel to execute fast search, aggregations and joins.



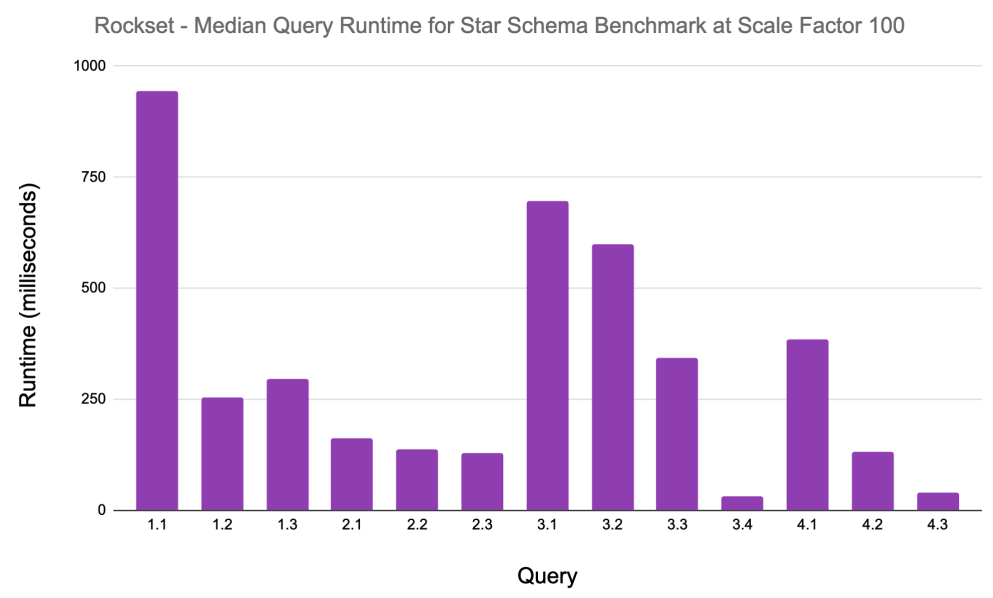
```
<doc 0>
{
    "name": "Igor"
}

<doc 1>
{
    "name": "Dhruba"
}
```

Key	Value
R.0.name	Igor
R.1.name	Dhruba
C.name.0	Igor
C.name.1	Dhruba
S.name.Dhruba.1	
S.name.Igor.0	

Rockset's Converged Index stores every column of every document in a search index, column store and row store for millisecond latency queries.

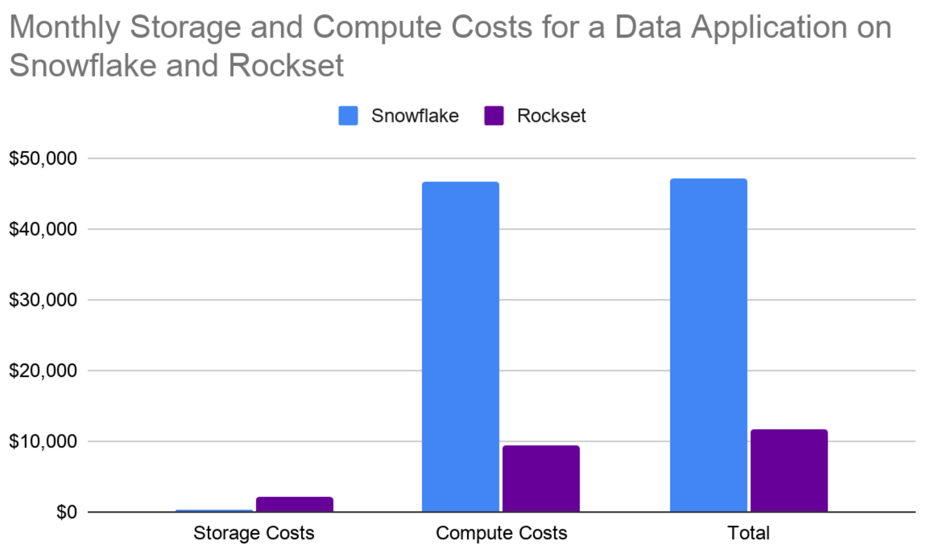
Rockset's indexing approach allows teams to meet the sub-second latency required of data applications. On the Star Schema Benchmark, an industry-standard benchmark for analytical applications, Rockset was able to deliver every query in under one second latency.



Graph of the query runtimes when running the Star Schema Benchmark on Rockset at a scale factor 100

Indexes are also compute efficient. One customer reduced their overall bill by 75% by building their data application on Rockset as compared to Snowflake. While each index requires upfront storage and compute costs, the subsequent queries actually require less compute to execute. That's because the query hits the index rather than scanning the data which is compute intensive. While Rockset requires more storage space for indexes, it consumes significantly less compute on query execution. When dealing with high query volume, as is the case in applications with hundreds to thousands of users, building indexes actually decreases your overall compute bill.

One customer **reduced their overall bill by 75%** by building their data application on Rockset as compared to Snowflake.



A customer's storage and compute spend for their data application on Snowflake and Rockset. While the customer saw their storage costs on Rockset increase from $295.40/mo to $2,141.65/mo, the customer's compute costs decreased from $46,720/mo to $9,509.81/mo. This is for a data application that runs 24x7.

Supporting real-time analytics is first order for Rockset so the system has been designed to index everything efficiently to minimize query latency at scale.

Ship faster. Enable new types of queries without restructuring your data.

Rockset makes it easy for developers to iterate on their analytics capabilities over time; they can introduce new data or queries and still see the same performance.

Rockset ingests semi-structured, structured, geo and time series data and applies a schema based on the fields and types present. In most SQL based systems, type is associated with each column header, prohibiting mixed data types like strings, objects and arrays in the same column. By contrast, Rockset associates type with each column value to accept mixed types and null values but its SQL processing is strongly typed and uses advanced vectorization techniques so that query performance is not compromised.

Collection

```
1: {"name": "Marie", "age": 40, "zip": 94542}
2: {"name": "Irene", "age": 21, "zip": "91126"}
3: {"name": "Dorothy"}
4: {"name": "Frances", "zip":94110.0}
5: {"name": "Ada", "age":35, "zip": "94020"}
6: {"name": "Donna", "age":44, "zip": "94210"}
```

Smart Schema

Field	Type	Occurrences	Total
age	int	4	6
name	string	6	6
zip	float	1	6
zip	int	1	6
zip	string	3	6

An example of a smart schema for a collection, or table, in Rockset. The schema is recorded based on the exact fields and types present in the ingested data. The schema is based on the entire dataset, not just a sample of the data.

The flexibility of the system to changes in the data model and the full SQL functionality has reduced development time for teams**.** One customer was able to decrease their time to market from 6 months to a single weekend by switching from Elasticsearch to Rockset for real-time analytics. The team was freed from data preparation, performance engineering and operations and they didn't have to learn the intricacies of Elasticsearch to migrate over their existing workloads.

One customer was able to decrease their time to market from **6 months to a single weekend** by switching from Elasticsearch to Rockset for real-time analytics.

We find that even teams with experience using Elasticsearch will spend on average 10 more days building the new data application as well as 2 days a week maintaining the system. These numbers would be larger if we're talking about moving an existing production application to Elasticsearch.

Time to Develop

Setup	elasticsearch	ROCKSET
Capacity planning	days	not applicable
Provisioning and configuring resources	days	not applicable
Implementing monitoring and alerting system	days	not applicable

Development Time	elasticsearch	ROCKSET
Configure data connector	hours	minutes
Configure the indexes	do it yourself	not applicable
Conduct performance testing	same	same
Data denormalization	days	not required
Develop queries	learn new language	use SQL

Time to Maintain

Ongoing Maintenance	elasticsearch	ROCKSET
Monitoring & tuning the cluster	1 day a week	not applicable
Scaling the cluster	2 weeks a year	not applicable
Handling software upgrades	2-3 days a month	not applicable
Triaging & error resolution	1/2 day a week	not applicable

Additional Time to Develop & Maintain Elasticsearch
Engineering Time
~10 business days to setup
~2 days per week to manage

An estimate of the time to build a new data application on Elasticsearch and Rockset. This assumes that the application is not operating at scale and that the team has prior experience managing Elasticsearch.

Go serverless to minimize time spent managing clusters

Rockset exploits the elasticity of the cloud. With a disaggregated architecture, Rockset can scale ingest compute, storage and query compute independently for optimal price-performance.

Rockset's Virtual Instances are compute and memory resources for your application that can be scaled at the push of a button or an API call, making a massively distributed system simple to operate. One customer was able to handle an unexpected load spike in the middle of the night with the push of a button, increasing the resource allocation to meet the usage demands of the application.

"Rockset is pure magic. We chose Rockset over Druid, because it requires no planning whatsoever in terms of indexes or scaling. In one hour, we were up and running, serving complex OLAP queries for our live leaderboards and dashboards at very high queries per second. As we grow in traffic, we can just 'turn a knob' and Rockset scales with us."

Yaron Levi, Chief Architect at Rumble

Virtual Instances are mini distributed clusters that work like one big compute node. Queries are automatically broken down into hundreds to thousands of fragments to be executed over the distributed cluster quickly. The total time it takes for the SQL query to be compiled, optimized and distributed over the cluster takes 1.2 milliseconds. In most other warehouse and big data solutions, the process can take 500+ milliseconds. Rockset's execution framework is reactive, event-driven and runs in a volcanic execution model with built-in flow controls. This enables queries to be executed swiftly without any buffer queues or queuing delays.

The total time it takes for the SQL query to be compiled, optimized and distributed over the cluster takes **1.2 milliseconds.** In most other warehouse and big data solutions, the process can take 500+ milliseconds.

On-premise big data systems have not been built from the ground up to take advantage of cloud economics. Many systems continue to couple compute and storage, making it challenging to handle changing workloads cost-effectively. They are also a pain to manage—one of our customers estimated that it would take 1 full-time employee to manage Elasticsearch because of the intricacies provisioning resources, capacity planning, scaling and sharding to support a user-facing application.

With Rockset, you achieve the same scalability as other massively distributed systems at incredibly low latency without any of the operational overhead of datacenter era technologies.

One of our customers estimated that it would take **1 full-time employee** to manage Elasticsearch

Conclusion

The existing approaches to real-time analytics have not been designed to address the challenges of latency, scale and operational complexity. Data warehouses are too costly, operational databases cannot scale and real-time analytics solutions Elasticsearch and Druid introduce significant operational complexity, all of which erode the ROI of real-time analytics. With Rockset, teams are meeting the sub-second latency requirements at a fraction of the compute costs, decreasing their engineering development time with a flexible data model, and reducing their operational costs of managing distributed data systems. As Rockset has significantly decreased the cost of building and maintaining real-time analytics in production, teams are now seeing an increase in their ROI. Analytical capabilities are rapidly expanding in applications to simplify and streamline decision-making, positively impacting user adoption and retention.