CelerData

Analytics - A New Revenue Stream for SaaS Applications

Analytics, a new revenue stream for your SaaS applications

Today's SaaS applications sit on a gold mine of insights. While conducting their usual business transactions, SaaS applications collect critical information such as users viewing behavior, searching and filtering patterns, time spent on pages, and more. The level of these details is nearly impossible to be captured in brick and motor stores. For example, a travel booking application can tell exactly what other properties the customer has viewed before a reservation for a certain hotel was made. There is no way for a travel agency to know this. Hence, there is a tremendous amount of data in SaaS applications that can help hotel partners improve operations.

With this information on hand, SaaS applications can offer an add-on product, or an analytics module, to its subscribers and partners. These add-on products can either be billed as an optional module or bundled as part of a higher tier offering. Not only can these new modules bring in extra revenue, they can also set the app apart from its competitors.

Maintaining a Healthy Profit Margin

Before we get too excited about the prospect of these analytics add-ons, we must realize a big challenge facing such modules is to keep it profitable. While the offer is usually billed at a fixed monthly fee, the cost of running these modules, if not carefully planned, can go out of control and cause the app to lose money.

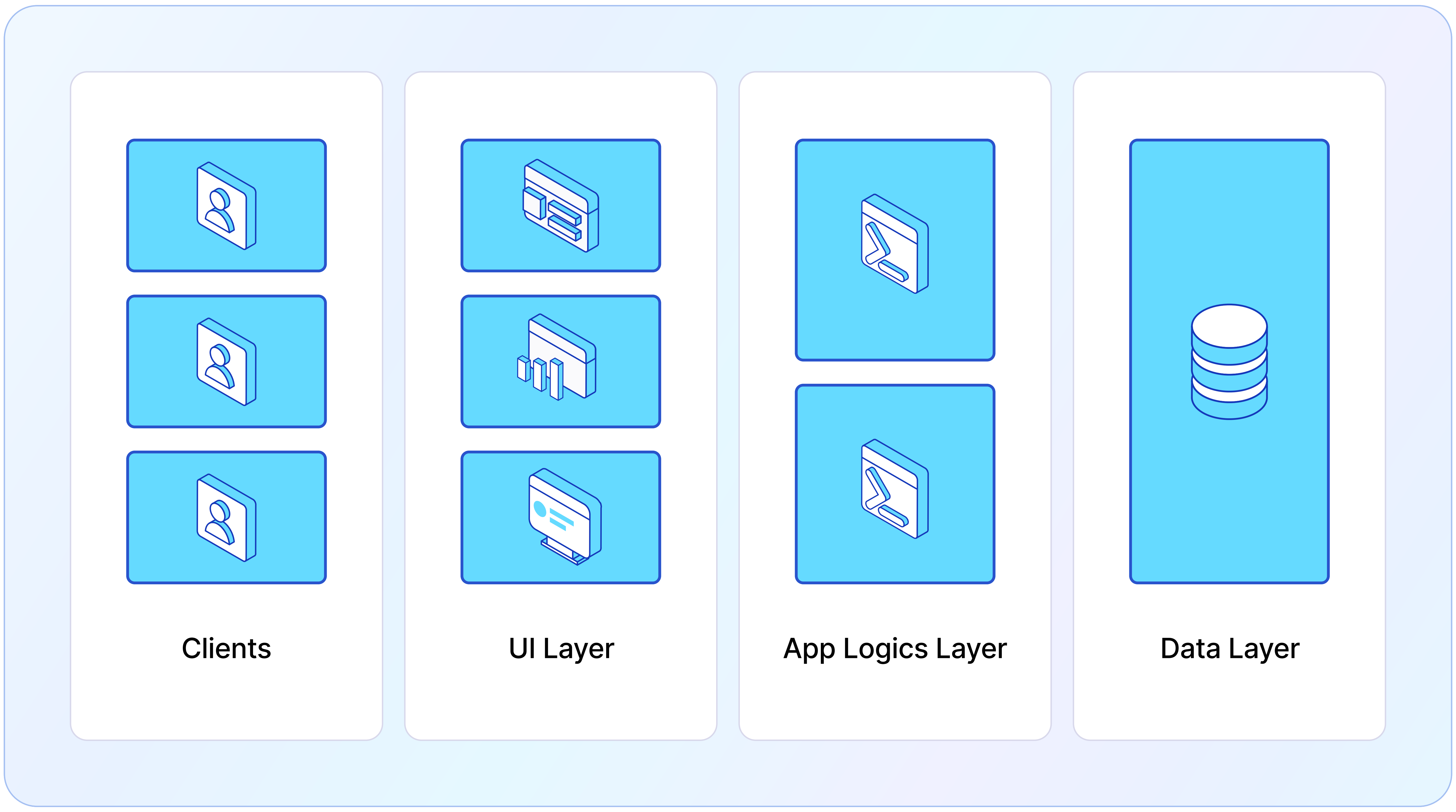
The reasons for the runaway costs lies in the business models of the cloud resources these SaaS run on: the better your analytics module is, the more users are going to use it, and the more it will cost you. It is like opening a buffet restaurant and your customers all have bottomless appetites. The more delicious the food, the faster you'll go out of business!

To solve this dilemma, we need to understand the technical architecture of the application and its analytics module.

Technical architectures behind the scenes

A SaaS application typically has 3 layers of components:

- A User Interface layer that handles page rendering and user interactions.
- A business logics layer that guides the page flow based on business rules.
- A data layer that stores data, transaction information, and application status.



On the surface, the analytics module is very similar to the main application module: a dashboard, which is part of the UI layer, translates the mouse clicks into a series of database queries. These queries are then passed on to the data layer to retrieve necessary information and update the dashboards.
But there are a couple of noteable differences between the normal business operations and the analytics modules:

- **Transaction on hand vs. historical data**: Normal business operations focus on maintaining the integrity of a single business transaction, such as making or canceling a hotel reservation. On the other hand, analytics modules focus on getting insights from a large number of historical transactions, i.e., average nightly spending of hotel customers across the last 6 months.
- **More complex queries**: While your core business logic is mainly involved in creating and updating a specific business transaction, the analytics module tends to run more complex queries in order to derive insights.

To run these complex queries over a large amount of data requires a lot more CPU and memory resources than your typical operation queries. For a SaaS application that pays for every CPU clock cycle and every byte in memory, these queries may drive up your cloud costs significantly and you end up getting a nasty surprise at the end of the billing cycle.

Why is it so hard to control the cost of analytics modules?

With these differences in mind, let's examine why it's so hard to control the costs of these analytics modules.

1) Predictable low latency

It is not uncommon for an analytical query to comb through millions of records and include complex, sometimes multi-step, calculations.

These queries need to finish within a second or two to give user a 'snappy' experience (system architects also need to make sure page rendering and other tasks are not slow).

To make things even more challenging, these modules are customer facing, the queries need to be consistently fast, even occasional slowness is not acceptable. To finish these queries in a timely manner (so that your users don't lose patience and give up on page refreshes) requires enormous amount of compute power. The efficiency of the query engine has a direct impact to the resource consumed by these queries.

2) Large numbers of concurrent users

A SaaS application typically has thousands to hundreds of thousands of users. Supporting even a fraction of its user base to run analytics simultaneously can be a daunting task for many query engines. Most analytical query engines were designed to support a selective group of 'business analysts' and it becomes financially prohibitive for them to deal with 1000s or more users running analytics at the same time, in the cloud.

3) Multi-tenant support

While providing services to many users, it is critical to leverage economies of scale by supporting many or even all clients in one analytics environment. This type of multi-tenant systems face two common challenges:

- How to protect the data of each tenant from being leaked?
- How to appropriately allocate compute resources to all tenants without over provisioning?

Addressing these challenges often leads to significant increase in hardware resources and costs.

4) High availability

Just like the application itself, the add-on module also needs to be highly available. A truly highly available system can be very difficult to implement and manage.

5) Data ingestion pipeline

To improve query performance, it has been a common practice to pre-process data for commonly used queries.

- **One Big Table (OBT) Model** - One technique commonly used to accelerate analytical queries is to break the normalization principles and flatten data into one big table to avoid the most expensive operation in database queries - Joins. For many analytical engines, OBT is the only way to get great query performance. But building an OBT model requires dedicated hardware resources and added complexity to the overall architecture.

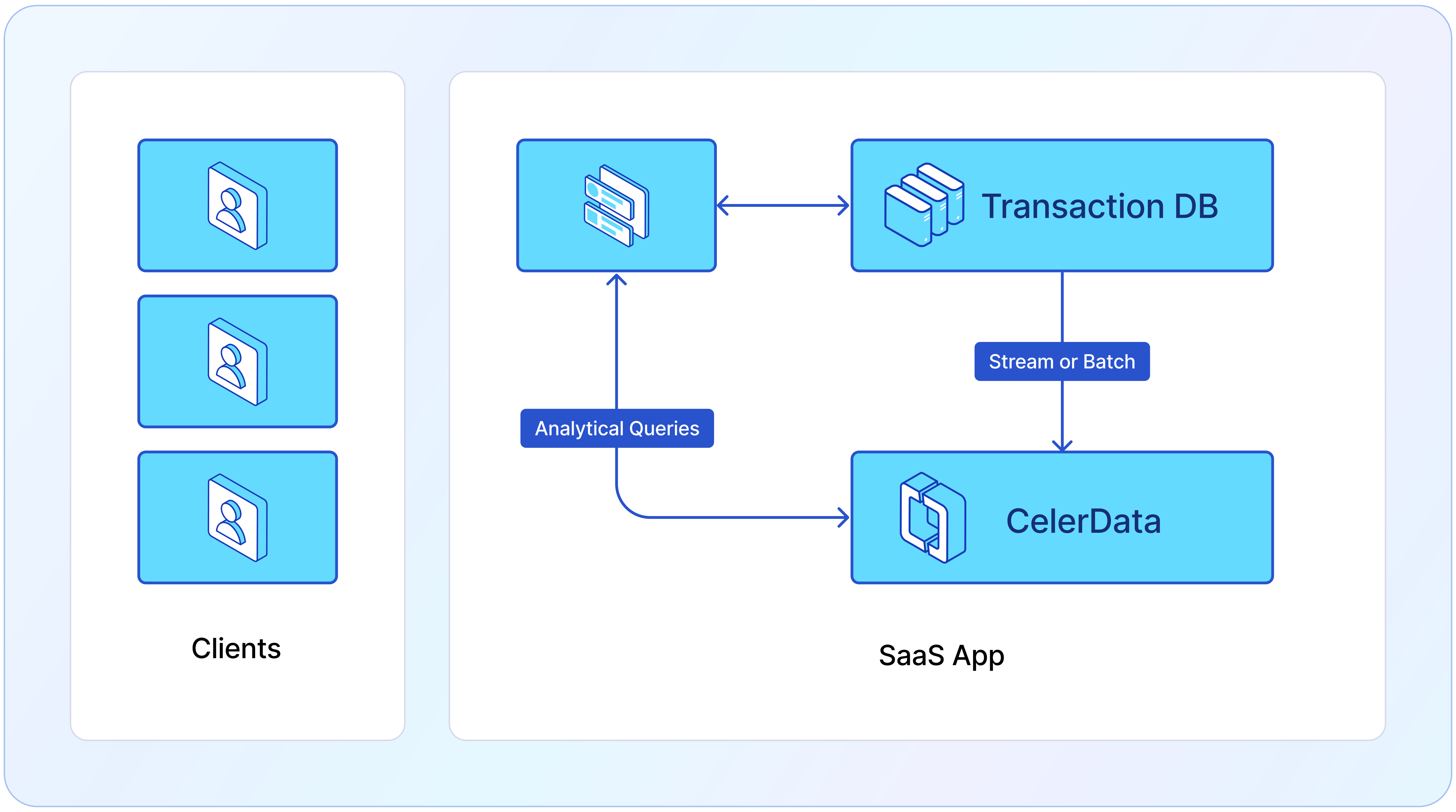
CelerData Cloud solution

CelerData Cloud, built on the highly performant open source project StarRocks, is a cloud lakehouse engine. It delivers unparalleled query performance for a large number of concurrent sessions. It can also scale out to support 1000s of tenants, all at just a fraction of the cost of other cloud data warehouses.

Solution data flow

The following diagram shows a typical data flow of the analytical module.

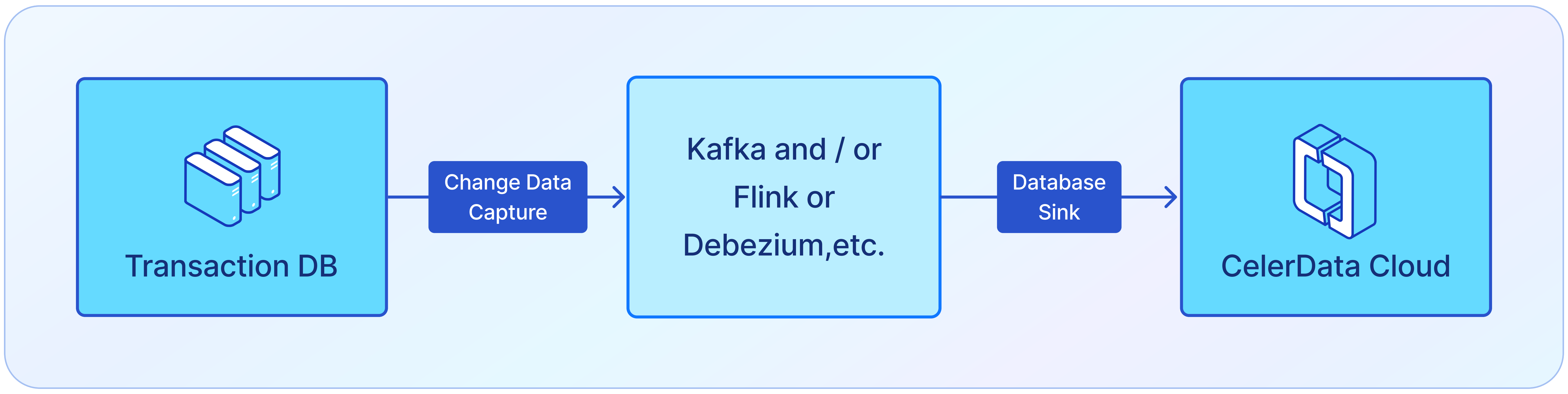
- Daily business transaction data is stored in a transaction processing (TP) database such as Oracle, Postgres, MySQL, or MongoDB.
- Data in the TP database is moved to an Analytical Processing (AP) database, which is CelerData Cloud.
- Dashboards of the analytical module query CelerData Cloud.



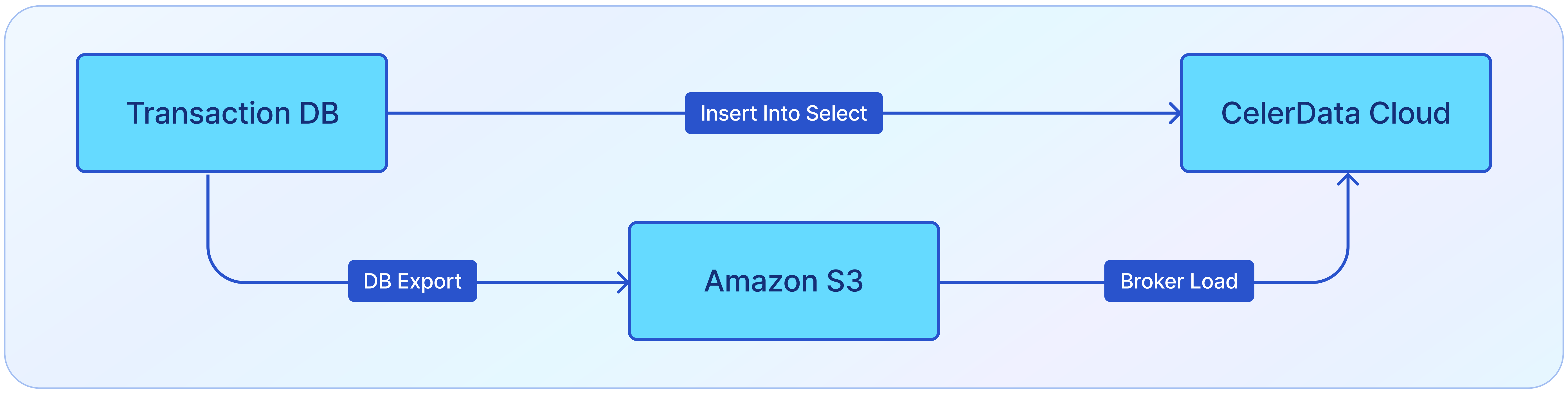
1) Data Ingestion

Data can be ingested in Stream or Batch mode, depending on the applications' technology stack.

- **Stream Ingestion**: any changes in the TP database are detected and extracted immediately through a technology called Change Data Capture (CDC), which is commonly available on all major databases. The extracted data will be applied to CelerData.



- **Batch Ingestion**: Batch ingestion happens periodically, such as daily/nightly. Data can be directly ingested into CelerData or landed on a staging area such as an S3 bucket, as depicted in the following diagram. The latter can be part of a data lake strategy to support versatile workloads.



2) Data Storage

- **Storage format**: Data can be stored either in StarRocks' internal columnar storage format or in a standard open format such as Apache Parquet. The former gives us better performance while the latter improves interoperability. If stored in an open file format, a table format layer can be added through Iceberg or similar technologies.
- **Storage location**: Data can be stored on either local disks attached to the servers (Shared Nothing Architecture), or on shared object storage such as S3 (Shared Data Architecture). With the right cache setup, Shared Data Architecture can achieve the same performance as the Shared Nothing architecture while being more cost effective for large amounts of data. Shared Nothing Architecture involves less admin work and is the original architecture of StarRocks.

3) Data Model

CelerData provides different data models to support different use cases.

- **Duplicate Key Model**: This is the default model where all historical records are preserved, but modification to those records is not allowed. This model is suitable for scenarios like user activity tracking and log analysis.
- **Primary Key Model**: If users need to make changes to historical records, such as returns or exchanges in an e-commerce app, then a Primary Key Model may be a better option. Primary Key Model supports true updates in place.
- **Aggregate Model**: For certain use cases where only aggregated results matter, such as total viewership of a program, the Aggregate Model may be the right choice. In this model, data will be aggregated while being ingested.

4) Distributed Queries

Queries will be processed by a cluster of nodes. Data is distributed into partitions and buckets to make sure queries are processed by all nodes in a balanced fashion to maximize resource efficiency.

- **Partition by time dimension**: Data is most commonly distributed into different partitions based a timestamp column. It is also possible to partition data by other columns, such as locations.
- **User ID or Tenant ID**: Having a Tenant ID column can greatly reduce the amount of data scanned in query execution. User ID or Tenant ID is often used as a sort key during ingestion to improve query performance.

5) Query Execution

- **SQL syntax and MySQL Protocol**: CelerData supports standard SQL syntax and is fully compatable with MySQL protocol, which means you can use any familiar SQL and BI tools to interact with CelerData.
- **Workload management**: CelerData's Resource Group and Multi-warehouse capabilities can effectively isolate the workloads of each tenant from others, protecting the privacy of each tenant and the stability of the application.

How can CelerData keep your cloud cost at bay?

1) Highly efficient query engine reduces cloud resource cost

CelerData’s (StarRocks) query engine leverages modern computing architecture such as a vectorized processing engine and MPP architecture to improve the parallel processing of analytical queries in a distributed fashion. Its C++ based engine is naturally much more efficient compared to a lot of Java based query engines. Many customers have reported that to get to the same performance, CelerData's resource consumption is only 1/3 of other technologies.

2) Multi-table joins eliminates the need to build OBT tables

Thanks to its Cost Based Optimizer and sophisticated Join strategies, CelerData is able to provide highly performant Join operations. Customers eliminate the majority, if not all, of the cloud resources dedicated to building the One Big Table model (sometimes referred to as denormazliation), resulting in cloud cost reduction.

3) Materialized Views reduce query execution costs

Materialized views can pre-compute some queries and have the results handy when these queries are executed. With careful design and planning, you can dramatically cut the cost to run those 'expensive' queries. Yes, in the cloud era, an 'expensive' query literally translates to expensive cloud costs.

4) Separation of storage and compute reduces cloud storage costs

Data in CelerData can be stored in an object storage layer instead of in the servers' local drives. By leveraging an object storage layer such as S3, Azure ADLS, or GCP's GCS, you don't need to store all your data in expensive SSDs anymore.

While there are many more features that enable great query performance at a fraction of other vendors' costs, the above four features probably have the most impact. In the following section, we will share a couple of case studies where customers saved significant costs after migrating to CelerData.

Case Studies

Pinterest (source: Pinterest Engineering Blog)

Pinterest is a visual discovery platform where people can find ideas like recipes, home and style inspiration, and much more. The platform offers its partners shopping capabilities as well as a significant advertising opportunity with 500+ million monthly active users.

Pinterest offers its shopping and advertisement users an analytics module to analyze real-time ads performance. This module supports thousands of concurrent users and provides real-time insights across tens of TB of data. Pinterest had been using Druid to power this module. As their data volume grew, this analytics module struggles with:

- Latency of the partner and advertiser dashboards
- Costs maintaining Druid clusters
- Freshness of the data landing on the end user dashboards

After switching to StarRocks as its real-time analytics platform, Pinterest was able to:

- Reduced latency by 50%
- Deliver data fresh with only 10 seconds lag before data lands on the analytics platform
- Achieve the above goals with only 1/3 of the cloud resource cost

Demandbase (source: Demandbase and StarRocks joint blog)

Demandbase is an account-based marketing (ABM), advertising, sales intelligence and data company. Its products provide B2B companies sales and marketing support that help them to discover, manage, and measure target audiences.

Dashboards are calculated across TB-scale data, with each dashboard requiring dozens of queries, each with sub-second latency requirements.

Data must also be available for various slice-and-dice operations so clients can better understand user behavior.

Demandbase moved to StarRocks with the following results:

- **Cluster Costs**: ClickHouse's 49 3-node clusters were replaced with a single 45-node StarRocks cluster, resulting in a 60% reduction in hardware costs.
- **Storage Costs**: By dramatically reducing denormalized data, Demandbase reduced storage costs by 90%.
- **ETL Costs**: Eliminating the need for heavy ETL pipelines to maintain denormalized data saved millions of dollars per year.
- **Product Improvements**: With denormalization no longer required, DemandBase now has additional opportunities to deliver even more real-time analytics and insights to their customers.

Additional considerations

- **Pricing Model**: The culprit behind the soaring cloud resource cost is the "Pay Per Use" pricing model of the cloud databases. Users need to be aware of the potential cost risk of this model. It is important to have a cap on resource consumption. Otherwise your customers are like those buffet patrons with bottomless appetites.
- **Data Privacy**: Whether the aggregated data is stored in your own cloud environment or a vendor's cloud environment can raise concerns for some companies. The decision is a trade off between operational simplicity and data leakage risk. CelerData Cloud is deployed in your cloud environment where CelerData has very limited access to only configurations and runtime metadata for administration purposes. The CelerData operations team doesn't have access to user data.
- **Future expansion and vendor-lock-in prevention**: With the advancements of AI technologies and data lake technologies, users need to be assured that they are not locked in with a specific vendor. CelerData is committed to fully supporting today's open data formats and open data catalog technologies.