Home Credit Default Analysis using Google Big Query

Ananya Varsha (40197013) Saghana Mahesh Sarma (40198979)

Concordia University Concordia University

Montreal, Canada Montreal, Canada

[ananyavarsha28@gmail.com](mailto:ananyavarsha28@gmail.com) [maheshsaghana0@gmail.com](mailto:maheshsaghana0@gmail.com)

Sivakumaran Janardhanan (40155790) Vishanth Surresh (40181942)

Concordia University Concordia University

Montreal, Canada Montreal, Canada

[sivamjsk96@gmail.com](mailto:sivamjsk96@gmail.com) [vishanthsurresh@gmail.com](mailto:vishanthsurresh@gmail.com)

ABSTRACT

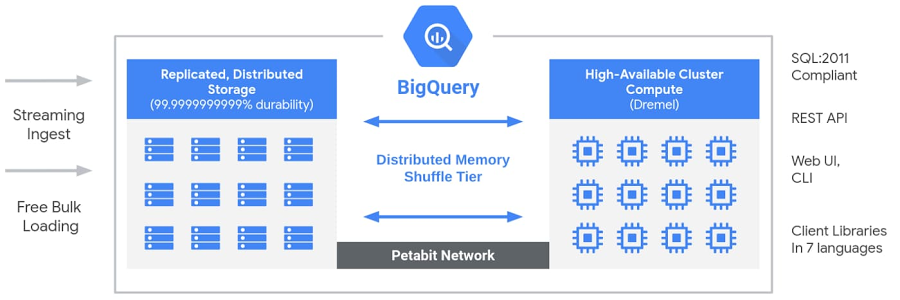
Many people face lot of struggles to get loan due to insufficient or non-existent credit histories. This unfortunately leads people to believe untrustworthy lenders. Thus, opening an opportunity to analyze the Home Credit Default data using Big-Data systems. This project, therefore, aims to analyze this Credit Default data using Google Big Query platform. In this report, we majorly discuss about how distributed system aspects are implemented in Google Big Query, how we used Big Query to analyze the data and the comparisons between single system database and Google Big Query.

1. INTRODUCTION

Due to a lack of or missing credit history, a significant portion of the population has difficulty getting their house loans authorised. This prohibits them from purchasing their ideal homes and, in certain cases, leads them to rely on untrustworthy and high-interest sources of financing. In contrast, deciding which individuals to qualify for housing loans is a huge difficulty for banks and other financial lending institutions. Credit history is not always an adequate tool for making judgments, because even borrowers with a long credit history can default on a loan, and some persons who have a reasonable likelihood of repaying a loan may simply lack a long enough credit history. Google's BigQuery platform is essentially an enterprise-level warehousing system that acts as one of the most suitable cloud-based platforms for querying and visualizing large-scale problems such as Credit Risk. The serverless, fully managed BigQuery platform provides some features that contribute to an imaginative, reliable, and easy-to-use query experience, extracting the most important insights from the data under consideration.

1. ARCHITECTURE

Big Query is a cloud-based data platform that allows users to have complete control over their data. The warehouse enables large-scale economic queries and data volumes that are comparable to those seen on Google to be carried out at Google-like speeds. Business insights are strengthened by taking use of low prices and Google's world-class scalability and security infrastructure. Big-Query is a petabyte-scale data warehouse solution for big-data analysis that is one of the fastest on the market. With no have to manage the infrastructure, customers can specialize in building meaningful insights using familiar SQL without the necessity for a database administrator. it is also economical because you merely procure the processing and storage you utilize [1].



The server-less architecture of Big Query separates storage and computation, allowing for independent scaling as needed. This structure provides customers with both great flexibility and cost control because they don't have to keep running expensive computing resources all the time. On a very high level, big query has two service components – storage service and query service. These are connected by Goggle's Jupiter's high-speed network infrastructure. The storage component manages data with high durability. When a user submits a query, Big Query initiates computing unit on demand, reading data from distributed storage and executing the query. All data is compressed and stored in a columnar storage format called the Colossus File System. This provides durability and availability. Storage service supports bulk data ingestion and streaming ingestion. It works with large amounts of data as well as real-time data streams. The query component uses compute slots to process the SQL queries and after computation returns the result. Big Query uses slots which are basically virtual CPUs that are allocated based on the query size.

If compute isn't being employed, we will power it off and just keep the storage nodes running. It significantly reduces the prices incurred because the customers don’t need to keep the expensive compute resources running all the time. thanks to this, both storage and compute may be scaled differently, without affecting or slowing down one another. If we are streaming in or bulk loading the info into Big Query, while parallelly executing heavy queries at the identical time, both of the operations can run together with no dependency on one another. That’s the major reason why Big Query is more economical and scalable.

Big Query uses four internal components to complete and serve your query results. The primary component, Dremel generates execution trees from SQL queries. The tree's leaves are referred to as slots, and that they are answerable for reading data from storage and doing any necessary calculations. The tree's branches are 'mixers,' which are accountable for aggregation. Within the second component Colossus, all of the information is compressed and stored in a very columnar storage structure. Colossus is answerable for replication, recovery (in the event that drives fail), and distributed administration (so there's no single point of failure). Colossus allows customers to store several of petabytes of knowledge in an exceedingly single location. The third component, Jupiter network allows compute and storage to speak with each other. The fourth component, BORG is to blame for coordinating in order that all of the engine's sections can operate together effortlessly. Borg controls the mixers and slots, further as allocating hardware resources.

1. **ROLE OF DISTRIBUTED SYSTEMS IN BIG QUERY**

BigQuery is based on Colossus, the most recent version of Google's distributed file system. Each Google Data Center possesses a Colossus cluster, which is furnished with an adequate number of discs to offer each BigQuery client with great many devoted drives simultaneously. Data in Big Query is automatically compressed, encrypted, replicated, and distributed. Colossus guarantees sturdiness utilizing deletion encoding where the information is deteriorated into parts and encoded with repetitive pieces of information and put away on multiple physical disks. Replication, Recovery and Distributed management is handled by Colossus.

Following are the 5 different features offered by Google's BigQuery that makes it more reliable:

1. Concurrency

It's nearly impossible to steal resources from BigQuery, and as the concurrency requirements grow, BigQuery easily adapts to those requirements. Slots to queries are dynamically allocated when in need keeping up with decency for concurrent queries from various clients.

1. Scalability

It's vital to take note of that the BigQuery engineering isolates the ideas of storage (Colossus) and compute (Borg) and permits them to be scaled freely. Colossus and compute functions are managed independently of each other, so you can switch to power-off mode when there are no tasks. This makes BigQuery more affordable and versatile than its partners. Also, Colossus enables users to scale to dozens of petabytes of data stored seamlessly. Storage and Compute can grow per your requirements and you don’t have to acquire any unused capacity.

1. Fault Tolerance and Replication

Fault tolerance is a characteristic that allows a system to function properly even if an error occurs during operation. Failures can be characterized as either transient(soft) failures or persistent(hard) failures. Transient failures are failures that occur temporarily then disappear upon rolling back and/or retrying/rebooting. For example, frozen keyboard or window, race conditions and deadlocks. The failures that are persist until explicitly repaired are called persistent failures. Data is not lost in either of the cases when used with BigQuery [9].

BigQuery distributes data across multiple data centers and considers multiple elements of replication to provide the user with maximum data durability and service time up. It also doesn’t provide cloud-provided replication of data [10]. BigQuery continues to execute with only a few milliseconds of delay in the event of a machine-level failure. All currently running queries will continue to be processed. No data is lost in the event of a transient or persistent failure. However, the currently running query can fail and must be resubmitted. Soft zone failure, like Power outage, transformer destruction, or network partition is a trial and error method and will be fixed automatically within minutes. Soft regional failure, for example, if a network connection is lost in a region, availability will be lost until the region is brought back online, but no data will be lost. Serious regional obstacles, such as disasters that destroy the entire region, can lead to the loss of data stored in that region. BigQuery does not automatically provide backups or replicas of data in different geographic regions. Cross-region dataset copies can be used to improve the disaster recovery of the dataset.

1. Security

BigQuery data is automatically encrypted during storage or transfer. BigQuery also provides the ability to isolate jobs and manage the security of multitenant activities. BigQuery is tightly integrated with the security features of other GCP products, giving enterprises a complete picture of data security. Users can share records with Google Cloud Identity and Access Management (IAM). Administrators can set permissions for individuals and groups to access records, tables, and views. Google's Virtual Private Cloud Policy Control prevents anyone outside your organization from accessing your data or attempting to export it to unauthorized third parties. IAM and VPC work across Google Cloud Platform to bridge security gaps between products.

1. **METHODOLOGY**

**Diagram

Description automatically generated**

1. **RESULTS**

6. CITATIONS

Some examples of references. A paginated journal article [2], an enumerated journal article [7], a reference to an entire issue [6], a monograph (whole book) [15], a monograph/whole book in a series (see 2a in spec. document) [13], a divisible-book such as an anthology or compilation [10] followed by the same example, how-ever we only output the series if the volume number is given [9](so Editor00a’s series should NOT be present since it has no vol. no.), a chapter in a divisible book [17], a chapter in a divisible book in a series [8], a multi-volume work as book [14], an article in a proceedings (of a conference, symposium, workshop for example)(paginated proceedings article) [3], a proceedings article with all possible elements [16], an example of an enumerated proceedings article [11], an informally published work [12], a doctoral disserta-tion [5], a master’s thesis [4], an finally two online documents or world wide web resources [1, 18].

ACKNOWLEDGMENTS

This work was supported by the [...] Research Fund of [...] (Number [...]). Additional funding was provided by [...] and [...]. We also thank [...] for contributing [...].

REFERENCES

[1] https://cloud.google.com/blog/products/data-analytics/new-blog-series-bigquery-explained-overview

[2] Sten Andler. 1979. Predicate path expressions. In *Proceedings of the 6th. ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL '79)*. ACM Press, New York, NY, 226-236. DOI:https://doi.org/10.1145/567752.567774

[3] Ian Editor (Ed.). 2007. *The title of book one* (1st. ed.). The name of the series one, Vol. 9. University of Chicago Press, Chicago. DOI:https://doi.org/10.1007/3-540-09237-4.

[4] David Kosiur. 2001. *Understanding Policy-Based Networking* (2nd. ed.). Wiley, New York, NY..

Conference Name:ACM Woodstock conference

Conference Short Name:WOODSTOCK’18

Conference Location:El Paso, Texas USA

ISBN:978-1-4503-0000-0/18/06

Year:2018

Date:June

Copyright Year:2018

Copyright Statement:rightsretained

DOI:10.1145/1234567890

RRH: F. Surname et al.