Google BigQuery - A data warehouse for large-scale data analytics

SAGAR VORA[1]

[1] *School of Informatics and Computing, Bloomington, IN 47408, U.S.A.*

April 9, 2017

The amount of relational data generated is increasing rapidly since it is generated through a large number of sources. Moreover this data is the information which companies like to explore and analyse quickly to identify solutions to business. Therefore, the need to solve the problem of traditional database management systems in order to support large volumes of data arises Google's BigQuery platform. Google BigQuery is an enterprise data warehouse used for large scale data analytics. A user can store and query the massive datasets by storing data in BigQuery and quering the database. BigQuery runs in cloud using the processing power provided by Google's infrastructure and provides SQL-like queries to perform analysis on masssive quantities of data, providing real-time insights about the data.

Keywords: BigQuery, BigData, Google, Cloud, Database, IaaS, PaaS, SaaS, SQL

https://github.com/cloudmesh/sp17-i524/raw/master/paper2/S17-IR-2041/report.pdf

1. INTRODUCTION

Nowadays, the amount of data being collected, stored and processed continues to grow rapidly. Querying this massive datasets can be time consuming and expensive without the right hardware and infrastructure. BigQuery[1] solves this problem by providing super-fast, SQL-like queries, using the processing power of Google's infrastructure. Google BigQuery[2] is a cloud web service data warehouse used for large-scale data processing. It is suitable for businesses that cannot afford to spend a huge amount of investment in infrastructure to process a huge amount of information. This platform allows to store and retrieve large amounts of information in near real time as well as providing some important analysis of the data which is stored.

2. GOOGLE BIGQUERY

To solve the architectural problems faced by Hadoop[3] MapReduce[4], Google developed Dremel[5] application in order to process large volumes of data. Dremel was designed to deliver high performance on data which was spread across a huge range of multiple servers and SQL support. But in 2012 at Google I/O event, it was announced that they would no longer support Dremel and this led to the beginning of BigQuery which became then the high-performance cloud offering of google. BigQuery makes use of SSL (Secure Sockets Layer) to take care of the security concerns related to cloud management.

2.1. System Architecture and Internal Structure

Google BigQuery platform is a Software As a Solution (SaaS) model in the cloud. As seen in figure 1 data which is generated from a variety of sources like event logs, relational databases, IoT devices like sensors, actuators, social media websites, Informatica[6] applications, etc is loaded in the databases which are created in BigQuery.

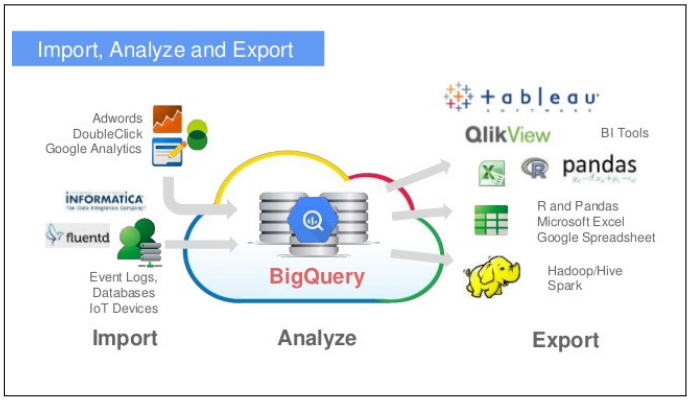

Fig. 1. [7] System Architecture of BigQuery

This data can then be processed and anaylsed using some algorthmic logic or applying some processing tool. Finally the data can then be represented and exported using Tableau[8], Qlikview[9], MS Excel[10] and other BI tools. It can also be ex-

ported on Hadoop system for parallel processing.
Now to store data in BigQuery, you need to create Projects. Projects[11] in BigQuery act as top-level containers which store the BigQuery data. Each project is referenced by a name and unique ID. Tables in BigQuery store the actual data where each table has a schema which describes the field name, types and other information. In BigQuery each table must belong to a dataset.Datasets help to organise the tables and control the access to it.

3. FEATURES OF BIGQUERY

Google BigQuery presents some characteristics like velocity, simplicity, and multiple access methods.

3.1. Velocity

BigQuery can process millions of information in seconds because of its columnar storage, and tree-like architecture.

3.2. Columnar Storage and Tree Architecture

The data instead of being stored in terms of rows like in standard SQL, is stored as columns and thus storage is oriented as shown in figure 2. This not only results in fast access to data but also results in scanning only the required values, which largely reduces latency. This also results in a higher compression ratios. Google reports[12] that "BigQuery can achieve columnar compression ratios of 1:10 as opposed to 1:3 when storing data in a traditional row-based format".

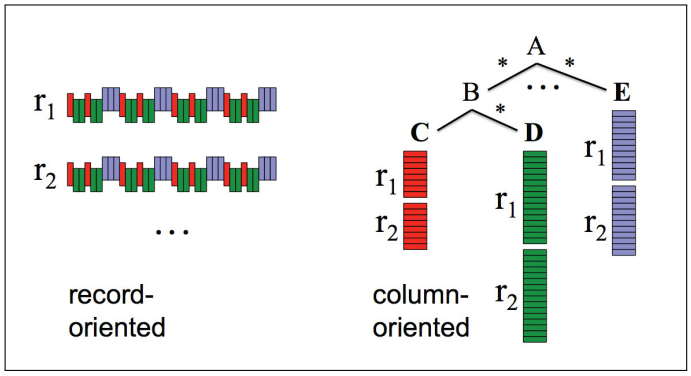

Fig. 2. [12] Row-oriented vs Column-oriented Storage in tree architecture

Moreover the tree-like architecture is used for processing queries and aggregating results across variety of different nodes. The tree-like structure also helps in retrieving data faster.
Let us see this with the help of 2 examples.
Considering the column-oriented storage in tree architecture format in the figure 2, lets refer node A as our root server, node B being an intermediate server and C, D, E being leaf servers having local disk storage space.

Example 1: Fast-retrieval
Statement: Find out all the customer names whose name starts with 'A'.
Assuming node C contains customer names. Hence, it is as simple as traversing A -> B -> C and looking at the datasets Cr1 and Cr2 for names starting with 'A'. One need not look at the paths from A -> B -> D and A -> E. Hence, in this simple scenario, A query may not scan the entire storage structure of the BigQuery and hence speeds up retrieving the information. Here the reason why the query looks only in the path A -> B -> C because BigQuery knows that all the customer names starting with A have been placed in the local disk storage at node C itself.

Example 2: Parallel Processing
Statement: Count all the customer whose names starting with 'A'.

1. Root node A sends out the query to node B which in turn translates the query to sum the names of all the customers starting with 'A'.
2. Now after translating, node B passes this query to leaf node C which has data stored in columnar format in the form of r1 and r2 tables.
3. node C accesses these tablets in parallel, and counts the number of customers whose names start with 'A' and passes the count of Cr1 and Cr2 to node B which sums the result and passes it to the root Aas the output of the query.

So this not only increases the speed of retriving the information but also the goal of parallel processing which is the need in quering huge datasets is also achieved.

3.3. Simplicity

Figure 3 shows the simple user interface provided by BigQuery. The dataset stored in BigQuery can be easily queried using SQL-like queries.

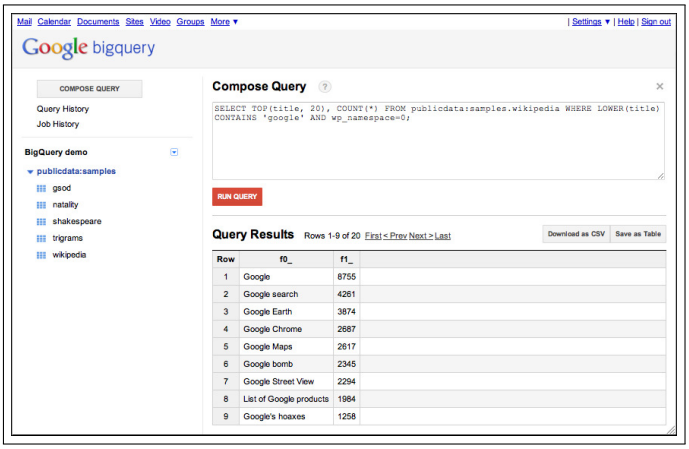

Fig. 3. [13] Big Query Sample User Interface

You can enter your query in the text area provided and the results will be displayed below. At the left side of the interface, it lists all the tables related to a particular database which are referenced as projects in BigQuery. In figure 3, the left-hand side consists a list of all the tables related to the 'publicdata:samples' project. The query is fired in the text area below the 'Compose Query' text and its results are displayed below in tabluar foramt.

3.4. Multiple access methods

You can access the BigQuery service in 3 different ways. We can either use a BigQuery browser tool, a bq command-line tool or a REST-based API.

1. BigQuery Browser Tool: It allows to easily browse, create tables, run queries, and export data to Google Cloud Storage.

2. bq command-line Tool: This is a Python command - line tool which permits to manage and query the data.

3. REST API: We can access BigQueryeven by making calls to the REST API using a variety of client libraries such as Java, PHP or Python.

4. COMPARISON OF BIGQUERY WITH IMPALA, SHARK, HIVE, REDSHIFT AND TEZ

In [14], the performance[15] of BigQuery is compared against Impala[16], Shark[17], Hive[18], Redshift[19] and Tez[20] by using Intel's Hadoop benchmark[21] tools . This benchmark has 3 different sizes of datasets - tiny, 1node and 5nodes. The 'rankings' table has 90 million records and 'uservisits' table has 775 million records. The data is taken using Common Crawl[22] document corpus. The two tables have the following schemas:

Rankings table schema: (lists websites and their page rank)
- pageURL (String)
- pageRank (Integer)
- avgDuration (Integer)

Uservisits table schema: (Stores server logs for each web page)
- sourceIP (String)
- destURL (String)
- visitDate (String)
- adRevenue (Float)
- userAgent (String)
- countryCode (String)
- languageCode (String)
- searchWord (String)
- duration (Integer)

The benchmark measures response time on 3 different types of queries. One being a basic scan query, one being an aggregation query, and one join query.

1. Scan Query

   Figure 4 shows a general scan query on the rankings table.

```
SELECT
    pageURL,
    pageRank
FROM
    benchmark.rankings
WHERE
    pageRank > X
```

Fig. 4. [14] Scan Query

This query has been fired using different values for 'X'. Query 1A means X's value is 1000, query 1B means X's value is 100 and 1C means X's value is 10.

2. Aggregation Query

   Figure 5 shows an aggregation query on the uservisits table. The aggregation is performed using the sum function along with the substr function.

```
SELECT
    SUBSTR(
        sourceIP, 1, X)
AS srcIP,
    SUM(adRevenue)
FROM
    benchmark.uservisits
GROUP EACH BY
    srcIP
```

Fig. 5. [14] Aggregation Query

This query has been fired using different values for 'X'. Query 2A means X's value is 8, query 2B means X's value is 10 and 2C means X's value is 12.

3. Join Query

   Figure 6 shows a join query between the rankings table and uservisits table on some given condition.

```
SELECT
    sourceIP,
    sum(adRevenue) AS totalRevenue,
    avg(pageRank) AS pageRank
FROM
    benchmark.rankings R
JOIN EACH(
        SELECT
            sourceIP,
            destURL,
            adRevenue
        FROM
            benchmark.uservisits UV
        WHERE
            UV.visitDate > "1980-01-01"
            AND
            UV.visitDate < X
        ) NUV
ON
    (R.pageURL = NUV.destURL)
GROUP EACH BY
    sourceIP
ORDER BY
    totalRevenue DESC LIMIT 1
```

Fig. 6. [14] Join Query

Like other queries earlier, this one also has different values

for 'X'. Query 3A means X's value is '1980-04-01', query 3B means X is '1983-01-01' and 3C means X is '2010-01-01'.

4.1. Comparison results

Figure 7 shows the results.

	Query 1A	Query 1B	Query 1C
Redshift	2.49	2.61	9.46
Impala (Disk)	12.015	12.015	37.085
Impala (Mem)	2.17	3.01	36.04
Shark (Disk)	6.6	7	22.4
Shark (Mem)	1.7	1.8	3.6
Hive	50.49	59.93	43.34
Tez	28.22	36.35	26.44
BigQuery	4.6	14.6	11.4

	Query 2A	Query 2B	Query 2C
Redshift	25.46	56.51	79.15
Impala (Disk)	113.72	155.31	277.53
Impala (Mem)	84.35	134.82	261.015
Shark (Disk)	151.4	164.3	196.5
Shark (Mem)	83.7	100.1	132.6
Hive	730.62	764.95	833.3
Tez	377.48	438.03	427.56
BigQuery	15.1	24.4	11.4

	Query 3A	Query 3B	Query 3C
Redshift	33.29	46.08	168.25
Impala (Disk)	108.68	129.815	431.26
Impala (Mem)	41.21	76.005	386.6
Shark (Disk)	111.7	135.6	382.6
Shark (Mem)	44.7	67.3	318
Hive	561.14	717.56	2374.17
Tez	323.06	402.33	1361.9
BigQuery	9.3	9.1	11.2

Fig. 7. [14] Comparison of BigQuery with other storage platforms

In this comparison experiment, each query was executed atleast 10 times and the results which are displayed are the average values of the response time in seconds. From 7, it shows that only in Query 1A, 1B and 1C BigQuery took more time than Redshift, Impala (on memory) and Shark (on memory) but the results related to query 2 and 3 are much better. For all the different values of X in query 2 and 3 (A,B and C), BigQuery executed them faster than other platforms. This shows that BigQuery can produce amazing results when processing complex queries on large datasets. As the number of processing records grew, BigQuery's response time was less as compared to the other storage platforms.

5. USE CASES OF BIGQUERY

5.1. Safari Books Online

Safari Books Online[23] uses BigQuery to find trends in customer purchase, manage its negative feedback related to customer service, improve the sales team's effectiveness etc. They chose BigQuery over other technologies because of its retrieval speed by quering the datasets using a familiar SQL-like language, and the lack of additional required maintenance.

5.2. RedBus

Online travel agency RedBus[24] introduced internet bus ticketing in India incorporating thousands of bus schedules into a single booking operation. Using BigQuery, redBus was able to manage the terabytes of booking and inventory data quickly and at a lower cost than other big-data services. BigQuery also helped its engineers fix glitches quickly, minimize lost sales and improve customer service as well.

6. CONCLUSION

Google BigQuery is a cloud-based database service which enables to process large data sets quickly. BigQuery allows to run SQL-like queries against multiple gigabytes to terabytes of data in a matter of seconds. It is suitable for ad-hoc OLAP/BI[25] queries that require results as fast as possible. As a cloud-powered parallel query database it provides extremely high full-scan query performance and cost effectiveness compared to other traditional data warehouse solutions and appliances.

ACKNOWLEDGEMENTS

I would like to thank my professor Gregor von Laszewski and all the associate instructors for their constant technical support.

REFERENCES

[1] "What is bigquery," Web Page, accessed: 2017-3-24. [Online]. Available: https://cloud.google.com/bigquery/

[2] S. Fernandes and J. Bernardino, "What is bigquery?" in *Proceedings of the 19th International Database Engineering & Applications Symposium*. New York, NY, USA: ACM, 2014, pp. 202–203. [Online]. Available: http://doi.acm.org.proxyiub.uits.iu.edu/10.1145/2790755.2790797

[3] "Apache hadoop," Web Page, accessed: 2017-3-25. [Online]. Available: http://hadoop.apache.org/

[4] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008, published as an Article. [Online]. Available: http://doi.acm.org/10.1145/1327452.1327492

[5] S. Melnik, A. Gubarev, J. J. Long, G. Romer, S. Shivakumar, M. Tolton, and T. Vassilakis, "Dremel: Interactive analysis of web-scale datasets," in *Proc. of the 36th Int'l Conf on Very Large Data Bases*, Sep. 2010, pp. 330–339. [Online]. Available: http://www.vldb2010.org/accept.htm

[6] "What is informatica," Webpage, accessed: 2017-4-07. [Online]. Available: https://www.informatica.com

[7] K. Sato, "Fluentd + google bigquery," Web Page, Mar. 2014, accessed: 2017-3-24. [Online]. Available: https://www.slideshare.net/GoogleCloudPlatformJP/google-for-1600-kpi-fluentd-google-big-query

[8] "What is tableau," Webpage, accessed: 2017-4-07. [Online]. Available: https://www.tableau.com/

[9] "Qlik," Webpage, accessed: 2017-4-07. [Online]. Available: http://www.qlik.com/us/

[10] "Microsoft excel," Webpage, accessed: 2017-4-07. [Online]. Available: https://products.office.com/en-us/excel

[11] "What is bigquery? bigquery documentation google cloud platform," Web Page, accessed: 2017-3-23. [Online]. Available: https://cloud.google.com/bigquery/what-is-bigquery

[12] V. Agrawal, "Google bigquery vs mapreduce vs powerdrill," Web Page, accessed: 2017-3-23. [Online]. Available: http://geeksmirage.com/google-bigquery-vs-mapreduce-vs-powerdrill

[13] J.-k. Kwek, "Google bigquery service: Big data analytics at google speed," Blog, Nov. 2011, accessed: 2017-3-23. [Online]. Available: https://cloud.googleblog.com/2011/11/google-bigquery-service-big-data.html

[14] V. Solovey, "Google bigquery benchmark," Blog, Jun. 2015, accessed: 2017-3-23. [Online]. Available: https://www.doit-intl.com/blog/2015/6/9/bigquery-benchmark

[15] "Amp lab big data benchmark," Webpage, accessed: 2017-4-08. [Online]. Available: https://amplab.cs.berkeley.edu/benchmark/

[16] M. Kornacker and J. Erickson, "Cloudera impala: Real-time queries in apache hadoop, for real," Blog, Oct. 2012, accessed: 2017-4-07. [Online]. Available: http://blog.cloudera.com/blog/2012/10/cloudera-impala-real-time-queries-in-apache-hadoop-for-real/

[17] "Apache spark," Webpage, accessed: 2017-4-07. [Online]. Available: https://spark.apache.org/sql/

[18] "Apache hive," Webpage, accessed: 2017-4-07. [Online]. Available: http://hive.apache.org/

[19] "Amazon's redshift," Webpage, accessed: 2017-4-07. [Online]. Available: https://aws.amazon.com/redshift/

[20] C. Shanklin, "Announcing stinger phase 3 technical preview," Blog, Dec. 2013, accessed: 2017-4-08. [Online]. Available: https://hortonworks.com/blog/announcing-stinger-phase-3-technical-preview/

[21] "Intel's hadoop benchmark," Git Repo, accessed: 2017-4-07. [Online]. Available: https://github.com/intel-hadoop/HiBench
[22] "Common crawl," Webpage, accessed: 2017-4-07. [Online]. Available: http://commoncrawl.org/
[23] D. Peter, "How safari books online uses bigquery for business intelligence," Webpage, accessed: 2017-4-08. [Online]. Available: https://cloud.google.com/bigquery/case-studies/safari-books
[24] "Travel agency masters big data with google bigquery," Webpage, accessed: 2017-4-08. [Online]. Available: https://cloud.google.com/customers/redbus/
[25] "OLAP," Webpage, accessed: 2017-4-08. [Online]. Available: http://olap.com/olap-definition/