Google Cloud Platform

Getting Started with Google BigQuery

Executive Summary

This paper is intended for data analysts and IT professionals who want to understand how to use Google BigQuery within their current IT environment. It provides a basic understanding of how it works, an overview of the end-to-end process from loading data into BigQuery to visualization, and design practices that should be considered when using BigQuery.

BigQuery supports open standards and protocols such as SQL-like query language and a REST-based API. It also supports traditional Extract, Transform and Load (ETL) tools for data ingestion and Business Intelligence (BI) tools for data visualization, making it straightforward to integrate BigQuery into an organization's existing IT environment.

Big Data Landscape

The landscape in data analysis has changed rapidly in the past few years. More and more organizations are using data to drive their decision-making. At the same time, the pace of data generation is accelerating dramatically. This shift towards Big Data within organizations is driven by many factors. Some factors influencing this trend are:

- The dramatic decrease in storage costs that enables organizations to collect and use vast amounts of data more cost effectively than ever before.
- Technological advancements now enable automation of data collection at each point in the process, which has resulted in capturing much larger data sets. The combination of capturing traditional data sets (through processes such as order management) and automated processes driven by technologies (such as RFID tags, sensors, machine logs, mobile device interactions) is fueling huge data growth.
- Innovations in data analysis technologies have made analysis of large scale data sets possible. Technologies pioneered by Google such as MapReduce, BigTable and Dremel are beginning to be used in industry. These technologies allow data analysis to be conducted over longer historical periods and at a much more granular level. This, in turn, provides new business insights not possible in the past.

This shift in the landscape, however, also introduces a new set of challenges. Many businesses are beginning to face problems such as finding a scalable infrastructure and the computing resources required for storing, processing, and analyzing large data sets. Traditional IT infrastructure and processes do not lend themselves well to processing and managing large data sets. For example, with Big Data analysis, a business usually requires a lot of experimentation before durable insights can be developed. Experimenting with data also means that it is hard to identify the Return on Investment (ROI) upfront, making it difficult for IT organizations to plan the capacity required to manage large data sets. Given the pace at which business conditions change, business users want to validate their hypothesis quickly. The traditional Business Intelligence (BI)/Data Warehousing (DW) process, with its long cycle of data modeling, data architecture, and capacity planning, is not meeting their needs.

Google BigQuery

BigQuery is a fully-managed data analysis service offered in the cloud that enables agile business analytics for today's fast changing business environment. Getting started with BigQuery requires no additional capital expenditure (CAPEX) to buy and configure either hardware or software. The BigQuery service is readily available, with a Service Level Agreement of greater or equal to 99.9% uptime [1]. Storage and computing resources automatically scale to the size of the data and analysis requirements. The "pay only for what you use" [2] model, "blazingly fast" [3] performance, a familiar SQL-like query language and ease of provisioning data in BigQuery without the need for complex data architecture, enable organizations to quickly run ad-hoc queries, validate hypotheses, and iterate on analysis to understand the data in an agile and cost effective manner. Query results are obtained in seconds and not hours.

Because the cost and time required for setting up the data analysis infrastructure are no longer barriers, business teams, together with their developers, can rapidly set up and perform analysis projects. The resulting speed enables line of business teams and Chief Marketing Officers to quickly put into place turn-key analytical solutions that help their business to increase revenue.

BigQuery in Action

Businesses have successfully taken advantage of BigQuery's strengths in cost, performance, scale, and uptime availability across a broad range of use cases. Below are several examples that illustrate how BigQuery has been used effectively.

Gmail Log Search [4]:
The Google Apps team uses BigQuery's high performance to power their interactive Gmail Log Search feature in the Administrator Control Panel. They decided to use BigQuery because of its ability to analyze multi-terabyte data sets with billions of headers and deliver precise results in just a few seconds. The Gmail Log Search feature allows the domain app administrator to answer questions such as:

- What happened to an inbound or outbound message?
- Was a message sent to my domain and marked as spam?
- Which user sent or received a specific message?

[1] https://developers.google.com/bigquery/docs/sla

[2] https://developers.google.com/bigquery/docs/pricing

[3] A sample query to find the top ten revised articles in wikipedia usually completed within 10 seconds. The size of the data set is 36GB, with 313M rows.

[4] For more information, please refer to Gmail log search feature enhances visibility for domain admins.

Online Ad Sales Optimization in Vacation Industry:
Crystalloids, an Amsterdam-based analytics firm, worked with Auto Camper Service International (ACSI), one of Europe's leading camping guide publishers, to improve its online ad sales. Using BigQuery, the ACSI Ads Sales team was able to contrast the average number of reservation requests between campsites that did or did not advertise, given the same region and features. This data enabled them to sell ads more effectively.

Booking and Route Management in Travel [5]:
redBus.in, an online travel agency that introduced internet bus ticketing in India, chose BigQuery to analyze booking and inventory data across their system of hundreds of bus operators serving more than 10,000 routes. They were able to analyze data sets as large as 2 terabytes in less than 30 seconds and spent 80% less than they would have on a Hadoop [6] infrastructure.

[5] For more information, please refer to the Redbus Business Case Study.

[6] Hadoop is a registered trademark owned by the Apache Software Foundation.

[7] For more details, please refer to Dremel: Interactive Analysis of Web-Scale Datasets.

How BigQuery Works

BigQuery is based on Dremel [7], a technology pioneered by Google, and used extensively within Google. BigQuery uses columnar storage and multi-level execution trees to achieve interactive performance for queries against multi-terabyte datasets. The technical implementation is different from that of a relational database management system (RDBMS) and traditional data warehousing solutions. BigQuery supports data inserts, deletion (through dropping of tables), and the ability to add columns to table schema after the data is already loaded. It does not support direct updates to the data stored. Even though direct row-level updates are not supported, many customers are able to accomplish updates by breaking up large datasets into finalized tables and non-finalized tables, and replacing the non-finalized tables as needed, so that the final analysis reflects the results of data updates.

BigQuery's performance advantage comes from its parallel processing architecture. The query is processed by thousands of servers in a multi-level execution tree structure, with the final results aggregated at the root. BigQuery stores the data in a columnar format so that only data from the columns being queried are read. This architecture has significant I/O performance gains over the traditional relational database/data warehousing model because the entire record is not read for each query.

Building A Big Data Solution on BigQuery: Pipeline to Analysis

It is easy to build an end-to-end solution on BigQuery. This paper presents several approaches for ingesting and visualizing the data. It also includes design recommendations for using BigQuery effectively.

Google Cloud Platform products used in this solution are:

- BigQuery to analyze event data
- Google Cloud Storage to store the aggregated event data (optional)
- Google App Engine to run the App Engine MapReduce framework (optional)
- Google Compute Engine to run the Hadoop MapReduce framework (optional)

Reference Architecture Diagram

The following architecture diagram (Fig. 1) illustrates the key components in the pipeline (data loading to visualization) proposed in this paper:

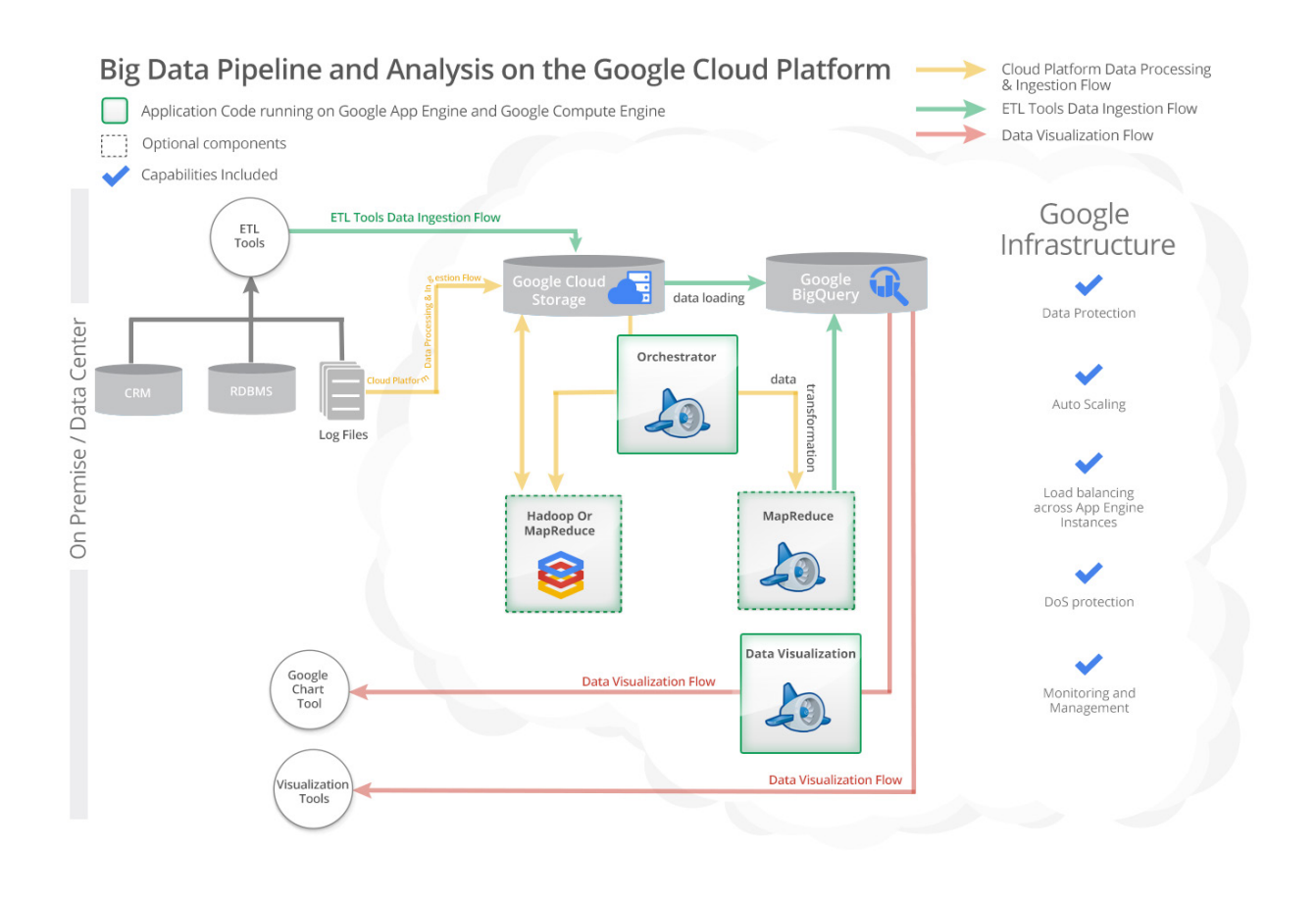

Figure 1. Components of the Big Data Pipeline to Analysis For BigQuery

Pipeline Walkthrough

There are two main flows for data analysis, as depicted in Figure 1:

- Data Pipeline - transforming and loading data into BigQuery
- Data Visualization - performing data analysis on BigQuery and visualizing the results

Data Pipeline

The process of loading of data into BigQuery follows these steps:

- Extract source data from the originating system.
- Denormalize the data since BigQuery performs better with fewer JOINs (optional).
- Transform the data into comma separated values (CSV) [8] file or Javascript Object Notation (JSON) file.

[8] The field delimiter is not restricted to comma. BigQuery supports any single-byte character as delimiter.

- Upload the CSV or JSON file into Google Cloud Storage. This can be accomplished by using Google Cloud Storage's gsutil command line utility. Some third-party ETL tools can upload directly into BigQuery.
- Load the CSV or JSON file from Google Cloud Storage into BigQuery using either the bq [9] command line utility or the BigQuery REST API.

[9] https://developers.google.com/bigquery/docs/cli_tool

Using Google Cloud Platform For Data Pipeline

The process of using the Google Cloud Platform to upload data into BigQuery involves uploading the CSV files or Javascript Object Notation (JSON) files to Google Cloud Storage before loading the data into BigQuery. Both Cloud Storage and BigQuery provide easy-to-use command line tools to upload data. Alternatively, REST API can also be used to provide programmatic integration into the current computing environment.

ETL tools can be run on Google Compute Engine to transform source data into a format ready to be imported into BigQuery.

The App Engine MapReduce framework and running Hadoop on Google Compute Engine are two options for performing ETL. The following outline suggests how an App Engine application can be used to orchestrate either approach.

1. Upload the source data into Google Cloud Storage. This can be accomplished by using the gsutil command line tool or using the Cloud Storage REST API.
2. The Google App Engine application auto-detects the presence of such files. Auto-detection can be implemented in a couple of ways:
   - **A.** Use the App Engine Cron Service to periodically poll the Cloud Storage bucket, using the REST API to detect newly uploaded objects.
   - **B.** Use the Cloud Storage Bucket Change Notification API. This is a publish/subscribe model where Cloud Storage acts as the publisher and the application acts as the subscriber. Upon startup of the application, a callback method is registered to receive messages when there are new or updated files.
3. The App Engine application can invoke either one of the following MapReduce pipelines:
   - **A.** Using the App Engine MapReduce framework
   - **B.** Initiating a process to run Hadoop on Google Compute Engine. The application uses Google Compute Engine's REST API to set up the instances and start the MapReduce job. The processed files are written back to Cloud Storage.
4. The application can then upload the processed log data into BigQuery using the REST API.
5. The App Engine Mail API may be used during any part of this process to generate a job completion notification.

Using Third-Party Providers For Data Pipeline

In the current marketplace, there are many third party vendors that provide Extract, Transform and Load (ETL) tools for BigQuery. These tools provide an easy to use "drag and drop" user interface for transforming and de-normalizing data and have the capability of loading data directly into BigQuery. They essentially perform all the steps above in a seamless fashion. These tools are especially useful if

there is enterprise data originating from multiple sources. Please refer to the Third-party Tools page for the current list of companies that have integrated their software with BigQuery.

Data Visualization

BigQuery supports multiple interfaces for performing queries. For interactive querying, a web interface and a command line tool are available. Likewise, the REST API can be used programmatically. Because queries are expressed in a simplified subset of SQL, the learning curve is lowered for anyone who is already familiar with standard SQL.

Using Google Cloud Platform Technologies For Visualization

A custom, web-based dashboard can be built on Google App Engine using the BigQuery REST API to execute the queries and using Google Chart Tools to visualize the results. The following steps briefly explain how this can be done:

1. Create an App Engine application.
2. Create a Google API Project and enable the BigQuery API.
3. Create an OAuth 2.0 Client ID for the application to access BigQuery.
4. Write a Google App Engine application that uses the BigQuery API and Charts API to create the dashboard.

For further details on how to build a custom dashboard, please refer to the following codelab at developers.google.com: "Codelab: Creating a BigQuery Dashboard".

Using Third Party Visualization Providers

Google has a list of companies that have already integrated BigQuery into their products. These products provide easy to use "drag and drop" interfaces to query the data and build reporting dashboards, without having to specify any SQL statements. Please refer to the Third-party Tools page for the current list of companies that have integrated their software with BigQuery.

Design Practices

When loading data into BigQuery, there are some design practices that should be considered early on to ensure a successful deployment of the Big Data analysis project.

Consider Quota Policies When Architecting A Solution

There are various quota policies that should be considered. One example is with data ingestion, where it's important not to exceed the maximum file size per load job. Knowing this limitation is important because if a file exceeds the limit, then the file will need to be split into smaller files and loaded separately. The most current quota policies for BigQuery are documented on the Google BigQuery Developers Site.

Data Denormalization Prior to Data Loading

The technical implementation of BigQuery is different from an RDBMS. BigQuery uses columnar storage and multi-level execution trees [10] to achieve the scale and performance. Appendix 1 explains the process of de-normalizing tables in more detail.

[10] For more details, please refer to Dremel: Interactive Analysis of Web-Scale Datasets

JSON vs CSV

BigQuery supports loading of data in both CSV and Java Script Object Notation (JSON) format. JSON

supports nested/repeated structure and represents a more object-oriented view of the data compared to CSV. JSON also removes the need for duplication of data required by the flattening of records into CSV.

BigQuery provides special SQL functions, FLATTEN and WITHIN, to support querying of repeated and nested data. Not all BI tools support these special SQL functions, so it is a good practice to test out a small subset of the data with the intended third-party tools for JSON format.

Table Partition

Partitioning of a table means splitting it into one or more tables with the same schema for specific purposes.

Static versus Changing Data

As mentioned earlier, BigQuery does not support updates to data that is already loaded. By separating data that is more likely to change from data that will not, updates can achieved by simply deleting a table and re-populating it with the updated data, avoiding the need to reload the entire dataset.

Cost Efficiency

The pricing for both interactive and batch queries is based on the amount of data processed. Costs can be optimized by partitioning the tables according to the usage pattern. For example, if most of the queries focus on monthly trends, then partitioning the table by month will reduce query costs. If querying data across multiple months is needed, it can be supported by table unions as shown in Figure 2.

If table unions are needed, there is no limit on the number of tables that can be specified, but there is a query length limitation, which is currently 10 KB.

Figure 2.
Example of partitioned tables

```
// Find suspicious activity over several months
SELECT FORMAT_UTC_USEC(event.timestamp_in_usec) AS time, request_url
FROM [applogs.events_201205], [applogs.events_201206],
[applogs.events_201207]
WHERE event.username = 'root' AND NOT event.source_ip.is_internal;
```

Storing Data in Google Cloud Storage As Backup

After loading the data into BigQuery, consider leaving the data in Google Cloud Storage as data into BigQuery.

Join Operations

BigQuery supports two types of JOIN operations. The use of each operation is determined by the size of the right-side table of the JOIN, while there is no size limit on the left-side table. The basic JOIN statement requires that the right-side table contains less than 8 MB [11] of compressed data. This JOIN operation is designed to be fast hence the maximum size limit for the right-side table.

When both tables are extremely large (tens of gigabytes to multiple terabytes), the JOIN EACH operation can be used. JOIN EACH causes BigQuery to perform an internal operation that breaks down both large tables into smaller chunks that are more efficiently processed in parallel. This step incurs additional processing than that of a JOIN clause.

[11]
Google is constantly making improvements. Check the online documentation on the JOIN clause for the latest limit.

Performance Optimization Using Subselect

In some cases, a subselect statement can be used to reduce the amount of data from the right-side table to fit within the 8 MB limit without having to use JOIN EACH, as shown in Figure 3.

Figure 3.
Example of using subselect to reduce data size

```
// Limit number of fields and rows.
SELECT purchases.amount, users.email_address
FROM PurchaseData AS purchases
AS users
ON purchases.user_id = users.user_id
WHERE purchases.amount > 500
```

Avoid Data that is Highly Skewed

When JOIN EACH is used, and a query response appears to be taking an unusually long time, it is an indication that the data is unevenly distributed. The uneven distribution causes some data chunks to be disproportionately larger than others, holding up the entire query execution. Figure 4 can be used to illustrates this point. It designed to analyze the effectiveness of an email campaign.

Figure 4.
Example of a query that potentially contain uneven distributed data

```
select date_of_visit, page, sum(time_spent_on_page)
from weblog_20130301 w
join each
(select user_id from email_campaign) e
on w.user_id = e.user_id
group by date_of_visit, page
```

However, if a large portion of the weblog contains traffic from users who are not logged in, and the system uses an anonymous user_id, then this query will take longer than necessary processing the anonymous users. A simple filter, as shown in Figure 5, would improve the querying performance:

Figure 5.
Example of using filter to eliminate data that caused uneven distributed data

```
select date_of_visit, page, sum(time_spent_on_page)
from weblog_20130301 w
join each
(select user_id from email_campaign) e
on w.user_id = e.user_id
where w.user_id != '<anonymous_user_id>'
group by time_of_visit, page
```

Sample Applications

Below are two sample applications that accompany this paper that will be available for download from http://cloud.google.com/resources. Subscribe to the Cloud Newsletter for future announcements

Automated BigQuery Loader

This sample Java App Engine application demonstrates how to automate data loading from Google Cloud Storage to BigQuery. This application uses Cloud Storage Object Notification API to receive notifications that files have been uploaded to a Cloud Storage bucket. It then uses the BigQuery API to load the data from the files into BigQuery.

Deploy ETL in a Box

This sample script provisions a virtual machine instance in Google Compute Engine and installs the necessary software for running an open source ETL tool. Another related document provides a step by step guide on how to use the ETL tool straight out of the box.

Please visit cloud.google.com to check when the downloadable sample applications will become available.

Conclusion

BigQuery is a fully-managed hosted service that is well suited for agile business analytics. By following the approach described in this paper, data can be easily loaded and immediately queried to gain business insights. This can be achieved without incurring additional capital expenditure for purchasing expensive software and setting up the computing infrastructure. Existing investments in technology and skill sets are leveraged because BigQuery provides a standard SQL-like query language and growing ecosystem of third-party tools.

Additional Resources

- An Inside Look to Google BigQuery
- Google BigQuery Case Studies
- Google BigQuery In Action - A Tour of BigQuery
- Google BigQuery Browser
- Third-party Tools for Google BigQuery
- BigQuery Best Practices
- Google BigQuery API Reference
- Google Cloud Storage gsutil Tool
- Google BigQuery bq Tool

Appendix 1 - Data Denormalization

BigQuery uses columnar storage and multi-level execution trees [12] to achieve the scale and performance for interactive queries. The technical implementation is different from an RDBMS. BigQuery performs best when the number of table JOINs is minimized. In some cases, tables should be denormalized before they are loaded into BigQuery.

[12] For more details, please refer to Dremel: Interactive Analysis of Web-Scale Datasets

The following example, a simple ordering system, illustrates the process of denormalization. A typical RDBMS database contains the following RDBMS tables: order, items, customer, and product. An order consists of a foreign key reference to the customer table, a number of items from the items table, and the items themselves, each of which reference a product in the product table. This relationship is depicted in the following Enhanced Entity Relationship diagram (Fig. 6).

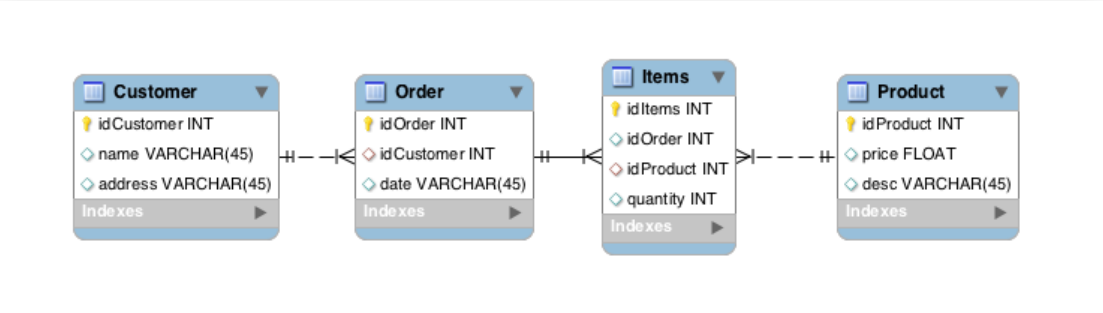

Figure 6. Enhanced Entity Relationship diagram

Figure 7 provides an example of how tables are denormalized into a single table with the following layout:

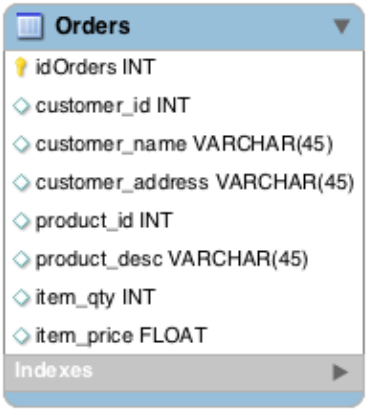

Figure 7. Tables denormalized into a single table