Google Cloud Platform

IDS 521 Project Report - Presented by

Siva Ganesh Subramaniyan

Vigneshwaran Giri Velumani

Yokesh Vishwanathan Suresh Kumar

Table of Contents

Introduction to Google BigTable

In a world, where data is becoming the breath of a business process and every decision that is made as part of it. We are looking for different ways of managing the information, different ways that will make it easy to store, process and retrieve data. [1] With technologies like Internet of Things, analytics data and with critical information like financial data, the necessity of high speed, high load processing of data have hit the sky and that gave birth to cloud technologies like BigTable, Google's NoSQL database service. Google itself uses BigTable for many of its services like Google Analytics, Gmail, YouTube, Maps, Google Search and so on.

So, what is big about BigTable? Well, the answer lies in the question itself. BigTable is a database system, that can handle massive workloads with consistent low latency and high throughput as the mentioned in the article about their product. This enables scaling of information to billions of rows and thousands of columns, which leads to storage of even petabytes of information. To dig deeper into what this platform offers, their limitations, how efficient it is in terms of cost effectiveness as well as in terms of performance, the following flow of information will discuss them in detail.

Features of BigTable

Performance – When you talk about performance of a database system, it merely means how good it is in terms of data processing like storage and retrieval of data, and BigTable has higher performance under high load than alternative products. This means that large applications are faster, more reliable and efficient in BigTable.

Security & Permissions – All the data that goes through BigTable are encrypted and the platform also enables the Project Admins to add different permission levels to control the access of data stored in Cloud BigTable.

Scalability – BigTable enables auto rebalancing of data stored in their system, enabling the users to dynamically add or remove clusters based on their usage strategy, with the system up and running all the time.

HBase Compatible - Cloud BigTable provides an HBase-compatible interface enabling portability of applications between HBase and BigTable.

Global Availability – This platform offers its service across the world in various regions, which enables the customers to place their service and data based on their necessity.

Low Latency Storage – BigTable enables single-digit millisecond latency at the 99th percentile, compared to that of which are more than 50 times the value with other products.

BigTable Architecture

[1] Cloud BigTable stores information in massive scalable table, each of which is sorted key/value map. Basically, in a table, a row represents an entity for which there will be multiple columns, which again represents variety of data corresponding to that entity. A row key is used to index the rows in the table and comes with that, the columns, which contains multiple data related to a row. These columns are identified by column family – qualifier combination. [1] Cloud BigTable tables are sparse, empty cells in a table does not occupy any space. The following block represents the overall architecture of Cloud BigTable.

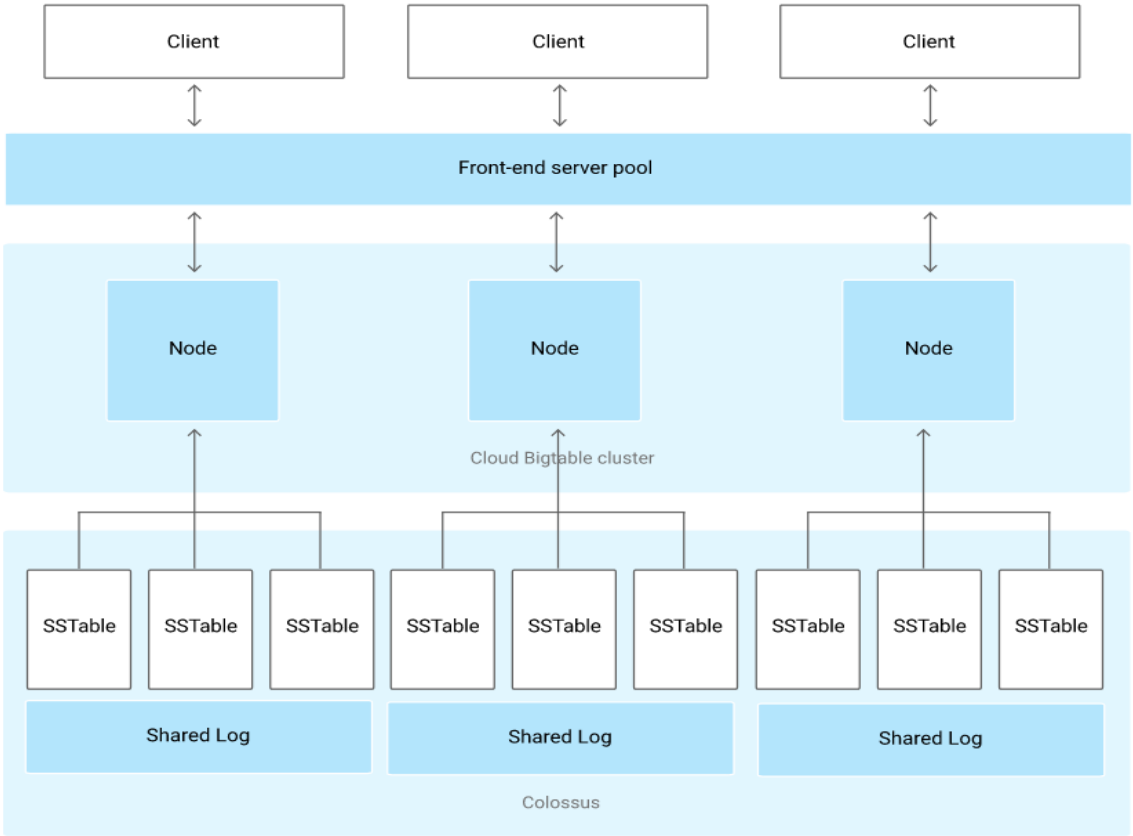


Image Source - https://cloud.google.com/bigtable/docs/overview

As represented in the diagram, the clients basically interact through front-end server which they basically call it Tablet server and the next layer comprises of the different set of nodes, which can be configured based on the project requirements. These are together organized within a cluster which belongs to an instance of a project and these nodes are used for splitting of loads based on the incoming requests. These so-called Tablets are stored in Googles File system, in SSTable format. The SSTable provide appropriate map with keys to the values, stored as string bytes. In addition, all write requests are stored in Shared Log which improves the durability of the system. Data is never stored in nodes, instead it represents pointers to tablets that are stored on Colossus.

MapReduce Operation

[8] MapReduce is a programming model and an associated implementation for processing and generating large data sets. The workflow is such that, the user specifies a map function that will process a key-value pair which in turn generated intermediate key-value combination, and then a reduce function that will merge all these intermediate maps back together once the data is processed. Many of the programming models are following this MapReduce model in which multiple processes are parallelized and they are executed on different nodes available on the system, and then they are combined back together based on the generated intermediate maps created. All these functions are done internally by the MapReduce system itself and that makes this model an immediate success in the market. This allows programmers from various aspects, to easily access the resources of a largely distributed system. These are majorly used in aggregating related information from external sources, application logs, etc.

Limitations of BigTable

Pricing – Google, like its other open source products, has not made it available for its customers to use BigTable as an open source product. To calculate a normal usage of BigTable for a month, one node in a cluster costs $0.65 per hour and its mandatory to use three such nodes in an instance. So, 3*0.65*24*30 comes to $1404 per month for a basic system with minimal load and on top of this, the storage is charged separately which comes to $0.17 GB/Month for SSD and $0.026 GB/Month for HDD.

Data Usage – BigTable is suitable for business that requires massive transformation of data, real time processing that requires multiple transactions which all comes down to one single point, that it is apt for large scale applications. [10] Also, querying can be done only in two regions as of present [us-central1 and europe-west1]

Quotas & Limits – Multiple limits are set on the usage of Cloud BigTable. For example, only 10 operations per second are allowed at the maximum, and considering real time process of data for a global level application, this might not just be enough. [1] Even limits are sets on operations on daily basis like 5000 operations per day. And if a company requires to transfer data from a node in Region 1 to a different server in Region 2, then there will be separate cost for the transfer of data between regions.

Lacks ACID - Google Cloud BigTable misses the crucial property of a relational system, which is ACID - Atomicity, Consistency, Isolation, Durability. So, simultaneous transactions might not be successful always.

Basic Queries

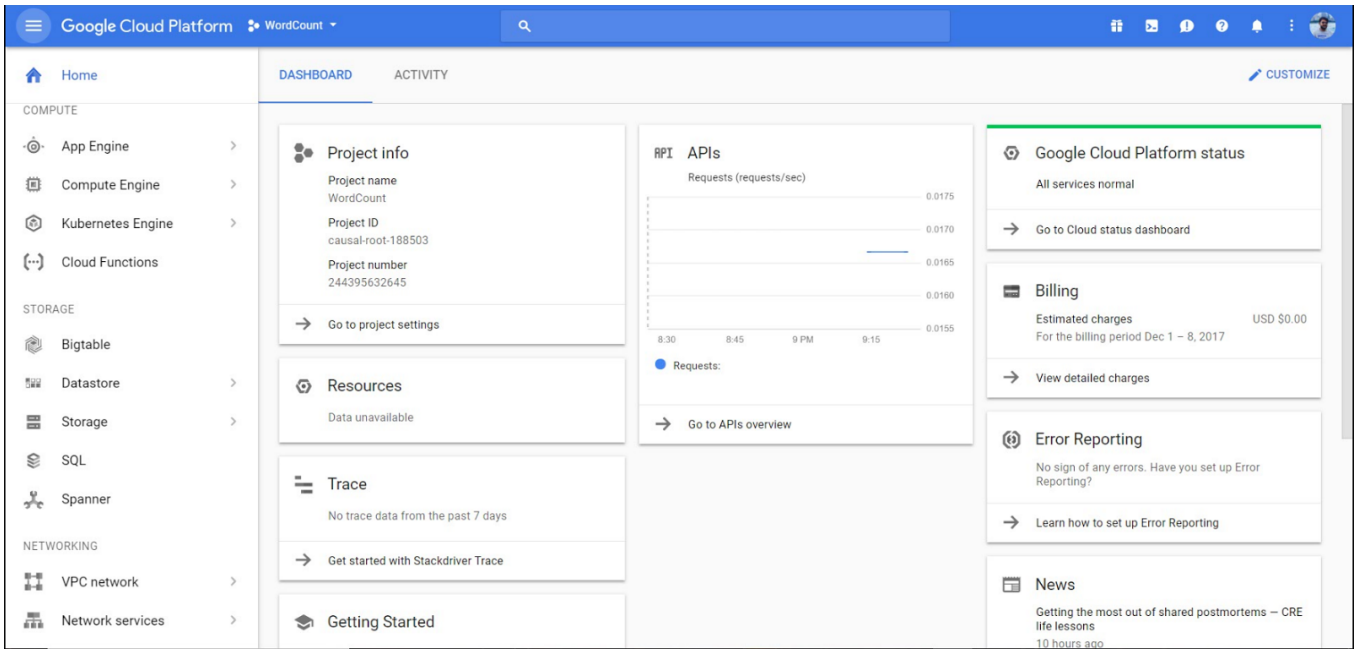
Tool	Function	Query
HBase Shell	Create Table	create 'my-table', 'cf1'
	Insert values into table	put 'my-table', 'r1', 'cf1:c1', 'test-value'
	Get values from table	scan 'my-table'
	Delete Table	drop ‘my-table’
CBT Tool	Create Table	cbt createtable <table_name>
	Set Values of cell in Table	set <table_name> <row_key> family:column=val[@ts]
	Get values from table	cbt read <table_name> start=<row_no> end=<row_no> count=<n>
	Delete Row	cbt deleterow <table_name> <row>
	Delete Table	cbt deletetable <table_name>

Implementation

The implementation explains how to load data to Cloud BigTable using DataProc API on Google Compute Engine. DataProc API uses Wordcount data from GitHub and using Map/Reduce, it is loading this data into Cloud BigTable. Initially an instance of the project is created and data from GitHub is extracted and cloned into the storage buckets. With DataProc API running in the background, using maven, we build the MapReduce function and create clusters to run the MapReduce function in its nodes and finally load the data table into Cloud BigTable.

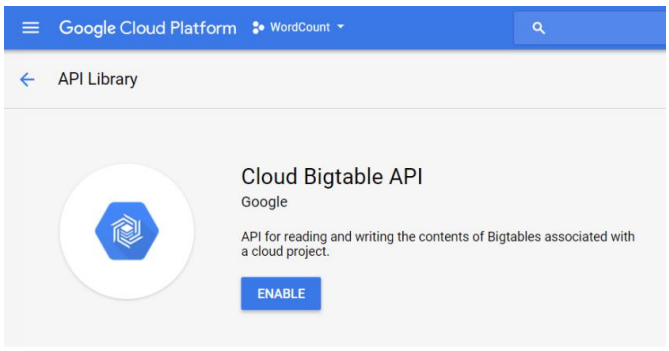
The following steps need to be executed to implement the Cloud BigTable MapReduce Example.

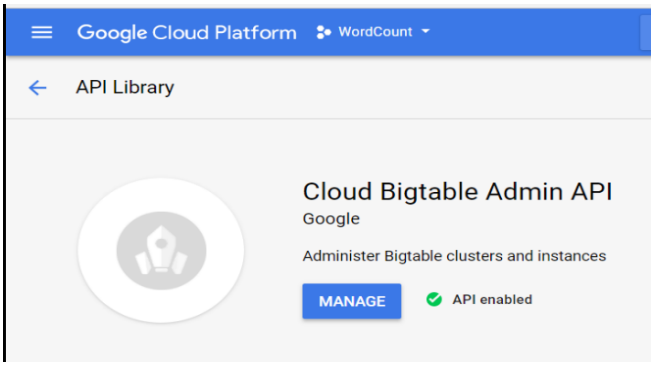
Step 1 – *Create a new Project* – 'WordCount' after setting up Billing for the new Google Cloud Platform Account.



Step 2 – *Enable API* – Enable APIs in the Developers Console

- Cloud BigTable API
- Cloud BigTable Table Admin API
- Google Cloud DataProc API





Step 3 – *Create Instance* –

- Give Instance Name: 'wordcountinstance1' and select Instance type as 'Production'
- specify zone as the same zone as the user is using the application - 'us-central-1c'
- Specify no of nodes to '3'
- Select Storage type - 'SSD' for low latency and higher read QPS.

Step 4 – *Create Bucket* –

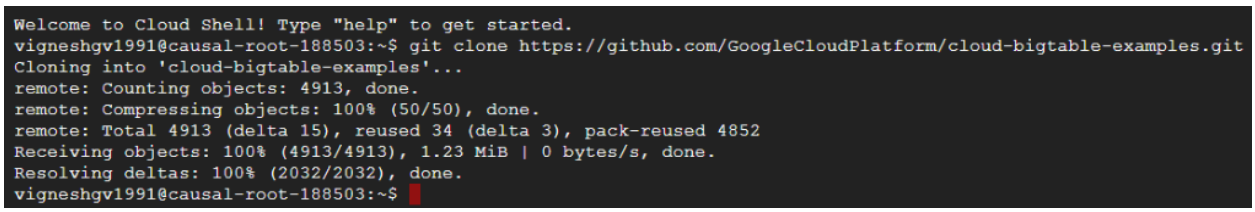
- Create a bucket in the same name as the project ID, since it must be unique
- Give Bucket Name: 'casual-root-188503bucket1'
- Default storage class: 'Regional' since we are accessing data only in the US.
- Regional Location should be us-central1

Step 5 – *Clone GitHub wordcount project* –

- The DataProc wordcount project is available on the GitHub which needs to be cloned to be instance of our project.
- GitHub Link - https://github.com/GoogleCloudPlatform/cloud-bigtable-examples.git
- Open Shell to execute following command

Shell Commands –

git clone https://github.com/GoogleCloudPlatform/cloud-bigtable-examples.git



Step 6 – *Build the Jar File* –

- Navigate to the project folder in the shell command line using change directory command.
- Build the MapReduce function using the command –

mvn clean package exec:java -Dbigtable.projectID = causal-root-188503 -Dbigtable.instanceID = wordcountinstance1

Step 7 – *Creating Cluster* –

- Create a cluster with default name 'dp' in the created bucket 'causal-root-188503bucket1'
- Specify the master machine type n1 and standard to 2 as higher numbers are allocated to premium Google BigTable users.
- Run the below command to create the cluster.

gcloud dataproc clusters create dp --bucket causal-root-188503bucket1 --num-workers 3 --zone us-central1-c --master-machine-type n1-standard-2 --worker-machine-type n1-standard-2

Step 8 – *Starting the Job* –

- Ensure the user is in the same folder as cluster.sh then execute the code: *./cluster.sh start dp*

Step 9 – *Watch your results* –

- [7] The content of table cannot be viewed directly from BigTable server. We need to implement HBase API to view the available data.

Unzip QuickStart –

- *curl -f -O https://storage.googleapis.com/cloud-bigtable/quickstart/GoogleCloudBigtable-Quickstart-1.0.0-pre2.zip*
- *unzip GoogleCloudBigtable-Quickstart-1.0.0-pre2.zip*
- *Go to the quickstart folder using cd command then use: ./quickstart.sh*

Step 10 – *Scan the table* –

[7] This is the HBase command to scan first 100 lines of Table-Name:

scan 'Table-Name', {LIMIT => 100}

```
=> ["WordCount-1512792378"]
hbase(main):003:0>
hbase(main):004:0* scan 'WordCount-1512792378', {LIMIT => 100}
```

Output table – 'WordCount-1512792378' – Displays three columns cf:count, timestamp and value. The value column displays the number of times the specific character has repeated in hexadecimal value.

```
=> ["WordCount-1512792378"]
hbase(main):003:0>
hbase(main):004:0* scan 'WordCount-1512792378', {LIMIT => 100}
ROW                                             COLUMN+CELL
 !),                                            column=cf:count, timestamp=1512792443998, value=\x00\x00\x00\x01
 !=                                             column=cf:count, timestamp=1512792435611, value=\x00\x00\x00\x09
 !=),                                           column=cf:count, timestamp=1512792439311, value=\x00\x00\x00\x01
 !=,                                            column=cf:count, timestamp=1512792446688, value=\x00\x00\x00\x01
 !probationary</pre></div></td>                 column=cf:count, timestamp=1512792435716, value=\x00\x00\x00\x01
 ""                                             column=cf:count, timestamp=1512792440749, value=\x00\x00\x00\xF4
 ""                                             column=cf:count, timestamp=1512792441022, value=\x00\x00\x00\x01
 "%5$s"</code>}                                 column=cf:count, timestamp=1512792441023, value=\x00\x00\x00\x01
 "&#8230;&#8203;number                          column=cf:count, timestamp=1512792441243, value=\x00\x00\x00\x01
 "&lt;html&gt;&#8230;&#8203;"</p></td>          column=cf:count, timestamp=1512792444203, value=\x00\x00\x00\x06
 "'My                                           column=cf:count, timestamp=1512792435279, value=\x00\x00\x00\x01
 "'The                                          column=cf:count, timestamp=1512792444211, value=\x00\x00\x00\x01
 "'Tis                                          column=cf:count, timestamp=1512792439442, value=\x00\x00\x00\x0A
 "'Tuque                                        column=cf:count, timestamp=1512792435411, value=\x00\x00\x00\x01
 "(PrefixFilter                                 column=cf:count, timestamp=1512792441024, value=\x00\x00\x00\x02
 "*hadoop-core*7070*SNAPSHOT.jar"`              column=cf:count, timestamp=1512792444211, value=\x00\x00\x00\x01
 "+clusterId                                    column=cf:count, timestamp=1512792444212, value=\x00\x00\x00\x01
 "-i                                            column=cf:count, timestamp=1512792435717, value=\x00\x00\x00\x01
 "0000000000000000"                             column=cf:count, timestamp=1512792441025, value=\x00\x00\x00\x01
 "1000MB/s",                                    column=cf:count, timestamp=1512792444212, value=\x00\x00\x00\x01
 "9                                             column=cf:count, timestamp=1512792439442, value=\x00\x00\x00\x01
 "<a                                            column=cf:count, timestamp=1512792447517, value=\x00\x00\x00\x02
 "<strong>*/IntegrationTest*ClassX</strong>".   column=cf:count, timestamp=1512792444212, value=\x00\x00\x00\x01
 "@"                                            column=cf:count, timestamp=1512792444212, value=\x00\x00\x00\x01
 "A                                             column=cf:count, timestamp=1512792439443, value=\x00\x00\x00\x0B
 "AES"                                          column=cf:count, timestamp=1512792441501, value=\x00\x00\x00\x01
 "AEneas                                        column=cf:count, timestamp=1512792448242, value=\x00\x00\x00\x01
 "AEneid,"                                      column=cf:count, timestamp=1512792435717, value=\x00\x00\x00\x01
 "AS-IS".                                       column=cf:count, timestamp=1512792439444, value=\x00\x00\x00\x01
 "Abstain                                       column=cf:count, timestamp=1512792444212, value=\x00\x00\x00\x01
 "Accept                                        column=cf:count, timestamp=1512792441025, value=\x00\x00\x00\x01
 "Accept,                                       column=cf:count, timestamp=1512792444212, value=\x00\x00\x00\x01
 "Accept:                                       column=cf:count, timestamp=1512792441026, value=\x00\x00\x00\x02
 "Accursed                                      column=cf:count, timestamp=1512792435717, value=\x00\x00\x00\x01
 "Achilles                                      column=cf:count, timestamp=1512792435412, value=\x00\x00\x00\x01
 "Achilles!                                     column=cf:count, timestamp=1512792447173, value=\x00\x00\x00\x01
 "Achilles'                                     column=cf:count, timestamp=1512792435717, value=\x00\x00\x00\x01
```

Step 11 – *Delete the cluster* –

- Every minute the cluster is in active state, BigTable is going to charge you separately for it. So, once the program is executed, delete the cluster at the earliest. Run the code - *./cluster.sh delete dp*

```
vigneshgv1991@causal-root-188503:~/cloud-bigtable-examples/java/dataproc-wordcount$  ./cluster.sh delete dp
Waiting on operation [projects/causal-root-188503/regions/global/operations/7d118b06-8bc9-45fb-8886-c35e135bea8d].
Waiting for cluster deletion operation...
....
```

Step 12 – *Delete the Project* – Its recommended to delete the project once the implementation is done, as there are various types of charges that the user will be billed for. So, this can be considered as an optional step.

Conclusion

Google BigTable provides the efficient storage and retrieval of huge volumes of semi-structured or unstructured data. Google Cloud Big Table with its high performance and low latency provides unparalleled levels of scalability, speed for the continuously evolving enterprise database business problems.
Every Technology has its own drawbacks, Google Big Table may not be ideal solution for all the cases because it lacks the ACID properties. Choosing the right technology for the business needs is vital.

[5] Cloud BigTable is not a relational database; it does not support SQL queries or joins, nor does it support multi-row transactions. Also, it is not a good solution for small amounts of data (< 1 TB). It is a NoOps, NoSQL, Big Data analysis tool, meant to be used at massive scale in conjunction with other Big Data tools. [6] Cloud BigTable is designed for larger companies and enterprises where extensive data processing is required, and where workloads are more complex. For example, if an organization needs to stream data into, run analytics on and serve data out of a single database at scale – Cloud BigTable is the right system.

[4] Unlike other clouds, GCP compute and storage are separate. You need to consider the following three parts when calculating the cost.

1. The type of Cloud instance, and the number of nodes in the instance.

2. The total amount of storage your tables use.

3. The amount of network bandwidth used. Please note: some part of network traffic is free.

Citations

[1] Overview of Cloud BigTable | Cloud BigTable Documentation | Google Cloud Platform."

Google Cloud Platform. N.p., n.d. Web

[2] "Differences between the HBase and Cloud BigTable APIs | Cloud BigTable Documentation |

Google Cloud Platform." Google Cloud Platform. N.p., n.d. Web

[3] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike

Burrows (Dec 10, 2016) "BigTable: A Distributed Storage System for Structured Data."

[4] My Big Data World | Google Cloud SQL vs Cloud Datastore vs BigTable vs Big Query vs Spanner

Posted on June 10, 2017 By Weidong Zhou. N.p., n.d. Web

[5] When to Pick Google BigTable vs Other Cloud Platform Databases |May 8, 2015 by Terrence Ryan. N.p., n.d. Web

[6] Google Launches Cloud BigTable, A Highly Scalable and Performant NoSQL Database

Posted May 6, 2015 by Frederic Lardinois N.p., 2015. Web

[7] "Differences between the HBase and Cloud BigTable APIs | Cloud BigTable Documentation |

Google Cloud Platform." Google Cloud Platform. N.p., n.d. Web

[8] Hall, K. B., Gilpin, S., & Mann, G. (2010, December). MapReduce/BigTable for distributed

optimization. In NIPS LCCC Workshop. N.p., 2010. Web

[9] Zdenko Hrcek, Disadvantages of Cloud BigTable | Quora" Available - https://www.quora.com/What-are-the-disadvantages-of-using-BigTable-over-MySQL-PostgreSQL-or-MongoDB

[10] Querying Cloud BigTable Data | Cloud BigTable Documentation | Google Cloud Platform."

Google Cloud Platform. N.p., n.d. Web