

Handling Non-Relational Databases on Cloud using Scheduling Approach with Performance Analysis

*¹Bansri Kotecha, ²Hetal Joshiyara

¹PG Scholar, ²Assistant Professor

^{1,2}Computer science Department,

^{1,2}LD college of Engineering, Ahmedabad, India

Abstract: Non-relational database computing in Clouds is a new model for upcoming generation analytics development, enabling unstructured data organization, sharing, and exploration of large volumes rapidly growing variety forms of data using Cloud computing technologies as a back end large scale service oriented computational infrastructure facility. Advances in information technology and its extensive growth in numerous areas of business, engineering, medical and scientific studies are resulting in information and data explosion. There are many techniques to handle non-relational database on cloud. This thesis focus on handling non-relational database using scheduling approach. Hadoop is an open source framework that is used to process large amounts of data in an inexpensive and efficient way, and job scheduling is a key factor for achieving high performance in big data processing. This paper uses Google cloud services like big query. Proposed scheduling algorithm was applied on interactive and batch query using cached data and without using cached data. Proposed algorithm reduces waiting time and thus improves the query execution time.

IndexTerms: Bigquery, scheduling, cloud, performance, cached data

1. INTRODUCTION

A non-relational database is any database that does not follow the relational model provided by traditional relational database management systems. This category of databases, also referred to as NoSQL databases, has seen steady adoption growth in recent years with the rise of Big Data applications. [1]

Non-relational databases have grown in popularity because they were designed to overcome the limitations of relational databases in dealing with Big Data demands. Big Data refers to data that is growing and moving too fast, and is too diverse in structure for conventional technologies to handle While these NoSQL technologies vary greatly, these databases are typically more scalable and flexible than their relational counterparts [2].

Non-relational databases have evolved from relational technology in these ways:

- **Data models:** Unlike relational models which require predefined schema, NoSQL databases offer flexible schema design that make it much easier to update the database to handle changing application requirements.
- **Data structure:** Non-relational databases are designed to handle unstructured data that doesn't fit neatly into rows and columns. This matters as most of the data generated today is unstructured.
- **Scaling:** You can scale your system horizontally by taking advantage of cheap, commodity servers.
- **Development model:** NoSQL databases are typically open source which means you don't have to pay any software licensing fees upfront.

1.1 INTRODUCTION TO BIGQUERY AND GOOGLE CLOUD

BigQuery is Cloud Platform's fully managed data warehouse that lets you economically query massive volumes of data at speeds one would expect from Google. Pay as you go, taking advantage of our pricing benefits and the scalability and security of Google's world-class infrastructure to power your business insights. Following are the key features of Big Data at google scale

Fully Managed, Server less Insight

Google Cloud Platform leads the industry in the ability to let you analyze data at the scale of the entire web, with the familiarity of SQL and in a fully managed, server less architecture where backend infrastructure is fully handled on your behalf. Our big data analytics products are able to scale automatically while you focus only on the business insight you want to uncover.

Fast Queries on Petabyte-scale Datasets

BigQuery is Cloud Platform’s fully managed data warehouse that lets you economically query massive volumes of data at speeds one would expect from Google. Pay as you go, taking advantage of our pricing benefits and the scalability and security of Google’s world-class infrastructure to power your business insights.

Unified Batch and Stream Processing

Cloud Dataflow is an innovative, fully managed service for developing and executing a wide range of data processing patterns including ETL, batch computation, and stream analytics. Express your computation with no switching cost as you use a single tool and programming model for both batch and continuous stream processing flows.

Spark and Hadoop in the Cloud

Companies standardized on great open source tools including Spark, Hadoop/MapReduce, Hive, and Pig, will find a natural transition in Cloud Dataproc. Never worry about your data pipelines outgrowing clusters as Dataproc lets you create and resize clusters quickly at any time. Per-second billing on the underlying Compute Engine resources ensures you pay only for the resources that you use, and you can spin down to zero when your analysis completes.

Managed Databases, Object Storage and Archival

Specific business questions you might ask in the future are difficult to predict. Never discard events and valuable metadata in your business environment store them economically to mine insight later. Choose from a variety of globally available storage products for your data, from managed SQL to NoSQL options, including our category-defining archival product near line.

The Next Stage of Machine Intelligence

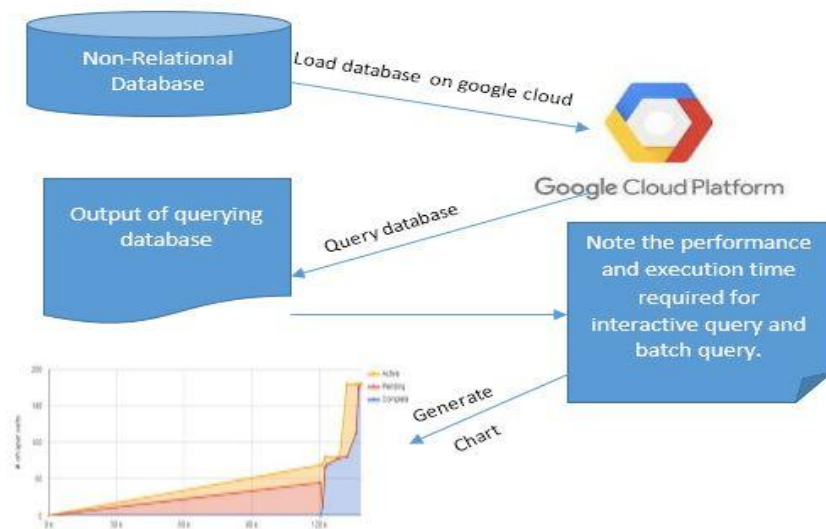
The long-term opportunity for companies lies in applying Google’s heritage of machine learning and analytics at web-scale to real-world data relevant to your business. Cloud Platform enables modest-sized teams to aggregate and run machine learning workloads on massive data to do predictive analytics. To disseminate the use of machine learning, Google has recently opened-sourced its library for machine intelligence TensorFlow and launched Cloud Machine Learning products, including several pre-trained models usable out-of-the-box such as Cloud Vision API, Cloud Speech API, and Google Cloud Translation API.

Tap In to Innovation

Google has led the industry with innovations in data processing technologies such as MapReduce, Bigtable, and Dremel. Now, Google is making the latest generation of its data processing tools available to everyone, including industry leading programming tools and programming models.

2. PROPOSED WORK

2.1 Proposed architecture



[Figure 2.1. Proposed architecture]

2.2 Description of all steps

Step 1: Load your non-relational Database on google cloud

You can load data in any format like

CSV

JSON (newline delimited only)


Avro

Parquet (Beta)

When you load data from Cloud Storage into a BigQuery table, the dataset that contains the table must be in the same regional or multi-regional location as the Cloud Storage bucket.

Go to the BigQuery web UI.

<https://bigquery.cloud.google.com/>

In the navigation panel, hover on a dataset, click the down arrow icon , and click Create new table. The process for loading data is the same as the process for creating an empty table.

On the Create Table page, in the Source Data section:

For Location, select Google Cloud Storage and in the source field, enter the Cloud Storage URI. Note that you cannot include multiple URIs in the BigQuery web UI, but wildcards are supported. The Cloud Storage bucket must be in the same location as the dataset that contains the table you're creating.

For File format, select Comma-separated values (CSV).

On the Create Table page, in the Destination Table section:

For Table name, choose the appropriate dataset, and in the table name field, enter the name of the table you're creating in BigQuery.

Verify that Table type is set to Native table.

In the Schema section, enter the schema definition.

For CSV files, you can check the Auto-detect option to enable schema auto-detection.



[Figure 2.2 Auto-detect schema]

You can also enter schema information manually by:

Clicking Edit as text and entering the table schema as a JSON array:

```
Schema ?
[
  {
    "description": "quarter",
    "mode": "REQUIRED",
    "name": "qtr",
    "type": "STRING"
  },
  {
    "description": "sales representative",
    "mode": "NULLABLE",
    "name": "rep",
    "type": "STRING"
  },
  {
    "description": "total sales",
    "mode": "NULLABLE",
    "name": "sales",
    "type": "INTEGER"
  }
]
```

[Figure 2.3 schema as JSON]

Using Add Field to manually input the schema:



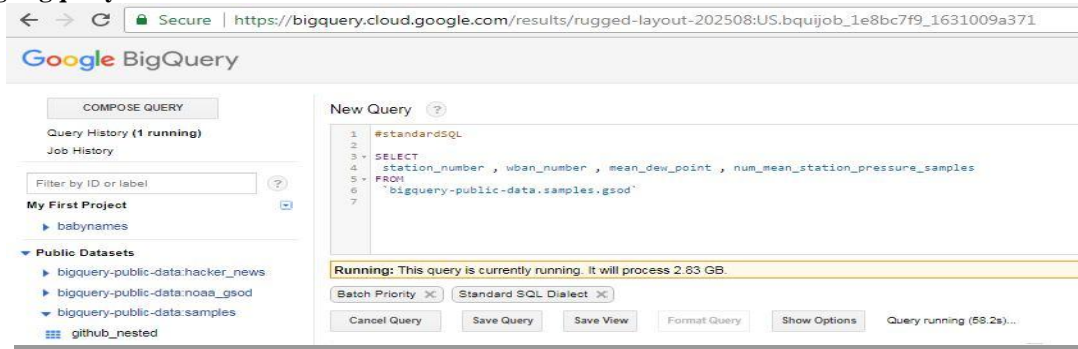
[Figure 2.4 Using Add Field manually]

Select applicable items in the Options section and then click Create Table.

Step 2: Query the database

You can query the database in two manners.

Using Bigquery WEB UI



[Figure: 2.5 querying the database]

You can see on the left side Compose Query button, Click on it and write the query in the New Query section and click on Run Query.

Using the bq Command-Line Tool

It is a python-based, command-line tool for BigQuery. This section contains general information on using the bq command-line tool.

Download and Install Google SDK cloud shell

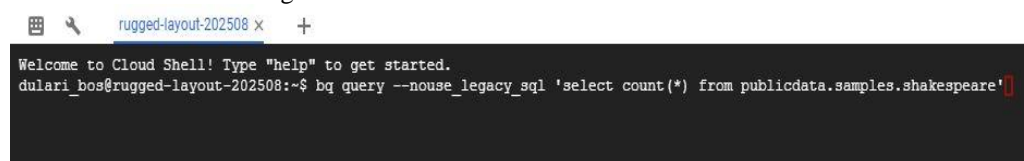
Sign in to your google account

In top right corner you will see following highlighted icon. Click on that and yourshell will open



[Figure 2.6 Google SDK Shell]

You will see the following window



[Figure 2.7 Google sdk shell]

Write the query and take the output

```

Welcome to Cloud Shell! Type "help" to get started.
dulari_bos@rugged-layout-202508:~$ bq query --nouse_legacy_sql 'select count(*) from publicdata.samples.shakespeare'
Waiting on bqjob_r3f4352f071d92531_00000163195b7a8c_1 ... (0s) Current status: DONE
+-----+
| f0    |
+-----+
| 164656 |
+-----+
dulari_bos@rugged-layout-202508:~$
    
```

[Figure 2.8 Querying in Google sdk shell]

Step 3: Note the performance and execution time required for interactive query and batch query

As shown in Fig. 4. 9, you can set the query execution parameters like Interactive or batch query and you can also set that whether you want to use cached data or you don't want to use cached data. Select the parameters run the sample data.

Step 4: Generate chart

Generate the chart regarding which public data sample took how much seconds to run the query. Also plot the chart for interactive and batch query both. You can also distinguish it by using cached result and using non cached result.

Step 5 Apply proposed scheduling approach

To apply proposed scheduling algorithm create a scheduling application with the API. To do so, follow the following steps. This are just example steps, original code steps are explained in next chapter.

Follow the following steps to implement your scheduling algorithm

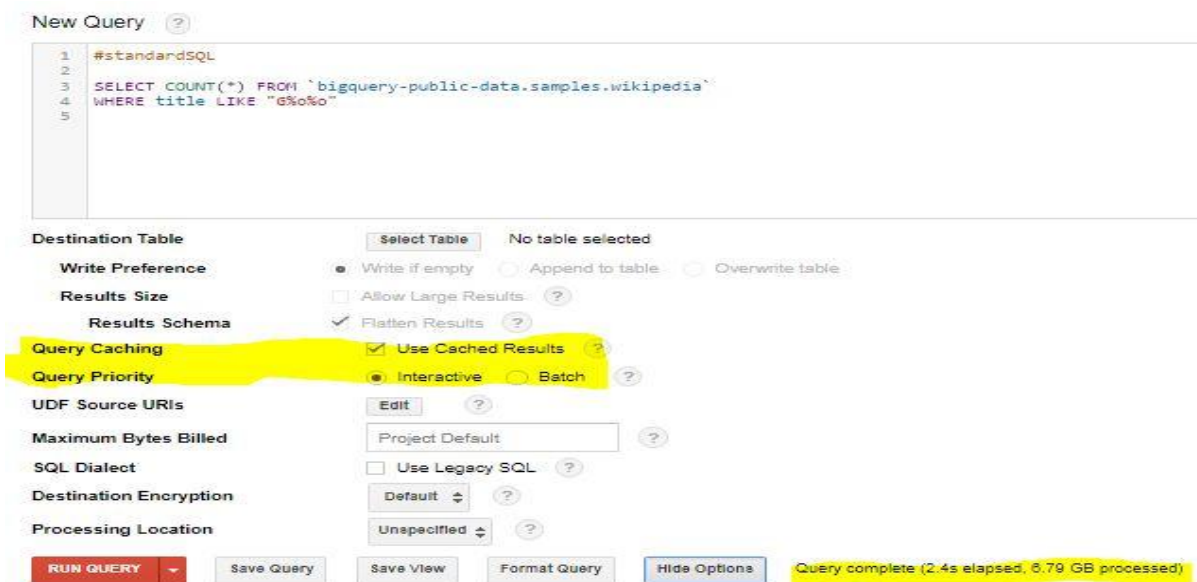
Understand BigQuery concepts and terminology.

Create a project with the BigQuery API enabled.

Set a local development environment for your preferred language from any of the following.



[Figure 2.9 language options]



[Figure: 2.10 Options to run Query]

Step 6: Performance analysis

Run the query for all sample programs and note the results and compare it with previous result. Draw the conclusion and plot the chart.

3. RESULTS AND DISCUSSION

Table ID (bigquery-public-data: samples.)	Table size	Long term storage size	Number of rows
github_nested	1.58 GB	1.58 GB	2,541,639
github_timeline	3.54 GB	3.54 GB	6,219,749
gsod	16.1 GB	16.1 GB	114,420,316
nativity	21.9 GB	21.9 GB	137,826,763
shakespeare	6.13 MB	6.13 MB	164,656
trigrams	258 GB	258 GB	68,051,509
wikipedia	35.7 GB	35.7 GB	313,797,035

[Table 3.1 Benchmark database]

Table ID (bigquery-public-data: samples.)	Table size	Query	Processed data
github_nested	1.58 GB	SELECT repository.url , repository.forks , repository.size , repository.language FROM `bigquery-public- data.samples.github_nested`	139 MB
github_timeline	3.54 GB	SELECT repository_url , repository_forks , repository_has_wiki , repository_name FROM `bigquery-public- data.samples.github_timeline`	350 MB
gsod	16.1 GB	SELECT station_number , wban_number , mean_dew_point , num_mean_station_pressure_sa mples FROM `bigquery-public- data.samples.gsod`	2.83 GB
nativity	21.9 GB	SELECT source_year , state , weight_pounds , mother_married FROM `bigquery-public data.samples.nativity`	2.62 GB
shakespeare	6.13 MB	SELECT word , word_count , corpus , corpus_date FROM `bigquery-public- data.samples.shakespeare`	6.13 MB.

trigrams	258 GB	SELECT first , second , third , fourth FROM `bigquery-public- data.samples.trigrams`	1.25 GB
wikipedia	35.7 GB	SELECT title , id , language , is_redirect FROM `bigquery-public- data.samples.wikipedia`	9.53GB

[Table 3.2: Results before applying scheduling algorithm]

Table ID (bigquery-public-data: samples.)	Table size	Number of rows	Interactive Query completion time (in sec)	Batch Query completion time (in sec)
github_nested	1.58 GB	2,541,639	16.8s	111.5s
github_timeline	3.54 GB	6,219,749	30.0s	116.5s
gsod	16.1 GB	114,420,316	20.9s	135.7s
nativity	21.9 GB	137,826,763	23.1s	125.2s
shakespeare	6.13 MB	164,656	13.4s	120.0s
trigrams	258 GB	68,051,509	48.0s	152.5s
wikipedia	35.7 GB	313,797,035	54.5s	154.0s

[Table 3.3 execution query time without using cached data]

Table ID (bigquery-public-data: samples.)	Table size	Number of rows	Interactive Query completion time (in sec)	Batch Query completion time (in sec)
github_nested	1.58 GB	2,541,639	2.8s	106.5s
github_timeline	3.54 GB	6,219,749	1.8s	123.6s
gsod	16.1 GB	114,420,316	1.9s	91.8s
nativity	21.9 GB	137,826,763	1.9s	117.0s
shakespeare	6.13 MB	164,656	1.7s	109.0s
trigrams	258 GB	68,051,509	2.8s	119.5s
wikipedia	35.7 GB	313,797,035	2.0s	106.4s

[Table 3.4 execution query time using cached data]

Results after applying scheduling algorithm

Table ID (bigquery-public-data: samples.)	Table size	Number of rows	Interactive Query completion time (in sec)	Batch Query completion time (in sec)
github_nested	1.58 GB	2,541,639	10.8s	90.8s

github_timeline	3.54 GB	6,219,749	25.0s	96.5s
gsod	16.1 GB	114,420,316	16.9s	95.6s
natality	21.9 GB	137,826,763	18.1s	105.2s
shakespeare	6.13 MB	164,656	13.0s	96.0s
trigrams	258 GB	68,051,509	32.8s	91.5s
wikipedia	35.7 GB	313,797,035	44.5s	118.2s

[Table 3.5 execution query time without using cached data]

Table ID (bigquery-public-data: samples.)	Table size	Number of rows	Interactive Query completion time (in sec)	Batch Query completion time (in sec)
github_nested	1.58 GB	2,541,639	1.8s	80.8s
github_timeline	3.54 GB	6,219,749	0.9s	76.5s
gsod	16.1 GB	114,420,316	1.0s	85.6s
natality	21.9 GB	137,826,763	0.9s	85.2s
shakespeare	6.13 MB	164,656	0.7s	76.0s
trigrams	258 GB	68,051,509	1.8s	71.5s
wikipedia	35.7 GB	313,797,035	1.1s	88.2s

[Table 3.6 execution query time using cached data]

3.1 Performance Analysis

Table ID (bigquery-public- data: samples.)	Interactive Query completion time Before Scheduling (in sec)	Interactive Query completion time after Scheduling (in sec)
github_nested	16.8s	10.8s
github_timeline	30.0s	25.0s
gsod	20.9s	16.9s
natality	23.1s	18.1s
shakespeare	13.4s	13.0s
trigrams	48.0s	32.8s
wikipedia	54.5s	44.5s

[Table 3.1 analysis of batch query without using cached data]

Table ID (bigquery-public- data: samples.)	Batch Query completion time Before Scheduling (in sec)	Batch Query completion time after Scheduling (in sec)
github_nested	111.5s	106.5s
github_timeline	116.5s	97.6s
gsod	135.7s	99.8s
nativity	125.2s	110.0s
shakespeare	120.0s	115.0s
trigrams	152.5s	119.5s
wikipedia	154.0s	106.4s

[Figure 3.8 Performance analysis of Batch Query without using cached data]

[Table 3.8 analysis of Interactive query with using cached data]

Table ID (bigquery-public- data: samples.)	Interactive Query completion time Before Scheduling (in sec)	Interactive Query completion time after Scheduling (in sec)
github_nested	2.8s	1.8s
github_timeline	1.8s	0.9s
gsod	1.9s	1.0s
nativity	1.9s	0.9s
shakespeare	1.7s	1.2s
trigrams	2.8s	1.8s
wikipedia	2.0s	1.1s

[Figure 3.9 Performance analysis of Batch Query with using cached data]

References

- [1]. NishthaJatana, SahilPuri, Mehak Ahuja, IshitaKathuria, DishantGosain, A Survey and Comparison of Relational and Non-Relational Database International Journal of Engineering Research & Technology (IJERT) Vol. 1 Issue 6, ISSN: 2278-0181 August - 2012
- [2]. Ibrahim AbakerTargio Hashem , IbrarYaqoob , Nor BadrulAnuar , SalimahMokhtar , Abdullah Gani , SameeUllah Khan, The rise of “big data” on cloud computing: Review and open research issues in Elsevier Information Systems 47 (2015) 98–115
- [3]. Mehdi Sookhaka , Abdullah Gani , Muhammad KhurramKhanb , RajkumarBuyyac, Dynamic remote data auditing for securing big data storage in cloud computing in Elsevier Information Sciences 000 (2015) 1–16
- [4]. M. Ali, S.U. Khan, A.V. Vasilakos, Security in cloud computing: opportunities and challenges, Inf. Sci. 305 (2015) 357–383.
- [5]. X.Fei, L.Fangming, J.Hai, A.V.Vasilakos, Managing performance overhead of virtual machines in cloud computing : a survey , state of the art , and future directions , Proc. IEEE102 (2014)11–31.
- [6]. Nagina, Dr.SunitaDhingra Scheduling Algorithms in Big Data: A Survey International Journal Of Engineering And Computer Science ISSN:2319-7242 Volume 5 Issue 8 August 2016 Page No. 17737-17743

- [7]. Raj ED, L.D DB —A Two Pass Scheduling Policy based Resource allocation for MapReduce International Conference on Information and Communication Technologies (ICICT 2014) Procedia Computer Science, Vol-46 ,pp. 627 – 634,2015, ISSN:1877 -0509.
- [8]. Hassan MM, Song B, Hossain MS, Alamri A —Efficient Resource Scheduling for Big Data Processing in Cloud Platform IDCS, LNCS 8729, pp. 51–63, 2014, ISSN: 0302 9743.
- [9]. Polo J, Castillo C, Carrera D, Becerra Y, Whalley I, Steinder M, Torres J, Ayguade E —Resource-Aware Adaptive Scheduling for MapReduce Clusters Middleware, LNCS 7049, pp. 187–207, 2011, ISSN: 0302- 9743.
- [10]. Nikhil B, Riddhikesh B, Balu P, Mukesh T —A Survey On Scheduling In Hadoop For Bigdata Processing Multidisciplinary Journal of Research in Engineering and Technology, Volume 2, Issue 3, pp. 497-501,2015, ISSN: 2348 – 6953.

Link references:

- [1]. <https://www.mongodb.com/scale/what-is-a-non-relational-database>
- [2]. <https://www.mongodb.com/scale/what-is-a-non-relational-database>
- [3]. <https://www.qubole.com/resources/big-data-cloud-database-and-computing/>
- [4]. <https://hadoop.apache.org/>
- [5]. https://hadoop.apache.org/docs/r1.2.1/fair_scheduler.html