International Journal for Modern Trends in Science and Technology, 6(8S): 76-81, 2020

ISSN: 2455-3778 online
DOI: https://doi.org/10.46501/IJMTSTCIET15
Available online at: http://www.ijmtst.com/ncracse2020.html

Machine Learning for Query Processing System and Query Response Time using Hadoop

M.Srikanth[1] | R.N.V.Jagan Mohan[2]

[1] Research Scholar, Department of CSE, GIET University, Gunupur, Andhra Pradesh, India
[2]Associate Professor, Department of CSE, SRKR Engineering College, Bhimavaram, Andhra Pradesh, India

To Cite this Article
M.Srikanth and R.N.V.Jagan Mohan, "Machine Learning for Query Processing System and Query Response Time using Hadoop*", International Journal for Modern Trends in Science and Technology*, 6(8S): 76-81, 2020.

Article Info
Received on 16-July-2020, Revised on 15-August-2020, Accepted on 25-August-2020, Published on 28-August-2020.

ABSTRACT

Now a day's 1000 of companies using hadoop. Hadoop is used for storing and processing big data. It is actually storing huge amount of data and processing system. Here you can store any type of data like (Structured data, Unstructured data, Video and Audio data) here data storing process is not a problem. But data retrieving is low query processing then response time is also low. In previously we study some papers then we find some issues of retrieving query processing and response time also. We propose to machine learning for query processing system and response time using hadoop. In this paper are simple changes to retrieving query process in hadoop using some techniques and the queries performance will make a huge difference in the response times.

KEYWORDS: *Hadoop, Hive Query Performance, Query response time, Map reduce, Machine Learning, Supervised learning.*

I. INTRODUCTION

Hadoop is one of the most popular Big Data Analytics tools in the heavy information industry. Hadoop is an open source framework for processing large amounts of data and storing data costly service hardware. Currently Hadoop faces less query processing and response time to retrieve data in the Hadoop database. To improve the query processing system and response time in this paper we will use some elements like (Mapreduce, hive query processing system, response time and supervised practice). Map Reduce Performance Tuning provides you with policies to improve the overall performance of your Hadoop cluster and get a first-class end result from your programming in Hadoop. This includes memory tuning in Hadoop, map disk spill in Hadoop, tuning mapper responsibilities, speculative execution in Big Data Hadoop, and plenty of different accessory ideas for Hadoop Map that reduce performance tuning. Now when we use these methods we use some hive bee quest processing system methods, which will definitely improve the query processing system and the response time in Hadoop. Here we also use the machine leaning model because this algorithm has the ability to learn from past experience. There are actually three types of models, but we will approach the supervised
learning model. Supervised practice We can do two types of work, namely classification and regression. In taxonomy, we try to find out if the test has the best of certainty if it is entered. In regression, we try to get the actual values output to examine the input, if the format system finds a

dataset that has a numeric output corresponding to each input. Knowing the method to be monitored involves learning a sample using academic statistics and checking a sample of invisible test information to assess reform accuracy. This model helps to classify the data and evaluate the monitored practice to improve the question processing and question response time using output.

II. IMPROVE RETRIEVING QUERY PERFORMANCE OF HADOOP WITH QUERY PROCESS

Question processing using a variety of widely spoken questions becomes a good-sized application area for Hadoop. Despite great efforts to improve the performance of those languages, performance dependence on basic configuration parameters can no longer be considered. Basic configuration parameters include the performance of standard types of queries. We select three queries from the Lehigh University benchmark, which have the maximum standard challenges and we consider the dependence on the parameters along with them: dataset length, range of nodes, reducer range and overhead load. The results suggest a strong dependence on the size of the reducers and the IO performance of the cluster, which proves the general opinion that Mapreduce is the IO determinant. These results help to assess the performance behavior of different languages and serve as the basis for knowledge of the impact of configuration parameters on the final overall performance.

A.QUERY PROCESSING ON HADOOP

External questions send some of the questions to the engine in Hadoop. Standard data dictionary / meta statistics in the Hadoop cluster (e.g., hive). Layout of standard records in HDFS files (e.g., ORC). The combination of external and internal Hadoop engines can limit data types on the combination of Hadoop and Query engine capabilities. Combines records and analytics in both systems and the compatibility of requirements is usually high and the query engine is generally mature. Data in Hadoop "Transportable" and can be read by other machines / engines. Here we apply hive query processing methods to help with question response time.

B. IMPROVE HIVE QUERY PERFORMANCE WITH HADOOP

The Apache Hive is a query and analysis engine built on top of the Apache Hapdoop and utilizes the Mapreduce programming model. It provides the use of the Java API as an abstract layer to query large-facts using SQL syntax by imposing traditional SQL queries. Important components of hive are: metastore, driver, compiler, optimizer, executor and client. Hadoop / hive can process almost any size of information, but optimizations lead to large savings in proportion to the amount of information in terms of processing time and price. It has a lot of optimizations that apply inside the hive. Let us look at the optimization methods we will cover:

a) Use Tez Engine:

Apache Tez is a nutrient-side library that acts as an execution engine under Hive and Pig, allowing for the traditional MapReduce engine to set up DAG to process jobs quickly. To explore how Tez facilitates character optimization, we will first examine the stereotyped processing sequence of the MapReduce job: The System Mapper feature reads information from the file system and processes it into key-value pairs, in addition to storing it temporarily on a neighbor's disk. These key-price pairs grouped on key values are sent to reducers on the community. On the nodes where Red Developer is to be run, the information is retrieved and stored on the local disk and waits for the records to reach all mappers. Then, the total values for a secret are studied as a single reducer, processed and similarly written output, which is then replicated mainly based on the configuration. Know As you know it makes sense in a non-meaningful read / write overhead unmarried mapreduce process. Multiple MapReduce are run to handle the same hive query and all outputs of MapReduce are first written inside the DFS, then transferred to the nodes and the cycle is repeated to see that there is no coordination between the two MapReduce jobs.

Apache Tez optimizes bee-question without breaking it in more than one Mapreduce. Tez is a nutrition-face library for orchestrating the processing of map-reducing jobs. Tez optimizes jobs using steps as follows:

- Skip the DFS script by the reducer and pipe the reducer's output into the next mapper as soon as it is entered.
- Cascading the chain of reducers without interfering with mapper stages.
- Reusing the boxes for the next steps of processing.
- Pre Optimal Resource Usage Use of pre-warmed containers.
- Cost-based optimizations.
- Ector vectorized quarry processing.

We can set the execution engine using the following query, or by putting it within the hive-website.Xml.

```
set hive.execution.engine=tez;
```

b) Use Vectorization:

Vectorization improves performance by obtaining an unmarried row using 1,024 rows in an unmarried operation. It improves the overall performance for filter out, one component, aggregation etc. activities.

Vectorization in the environment is enabled by executing the following commands.

```
set hive.vectorized.execution.enabled=true;
set
hive.vectorized.execution.reduce.enabled=t
rue;
```

A. c) Use ORCFile:

The optimized row column format provides green ways to store hive records by reducing records storage format by 75%. The ORCFile format is more than just the layout of hive documents on the subject of reading, writing and processing records. It uses strategies such as predict push-down, compression and more to improve the overall performance of the query.

Consider two tables: Worker and Employee_Details, tables saved in the text content file. Tell us what we can use to join to get information from each of the tables.

```
Select a.EmployeeID, a.EmployeeName,
b.Address,b.Designation from Employee a
Join Employee_Details b
On a.EmployeeID=b.EmployeeID;
```

Above query will take a long time, as the table is stored as text. Converting this table into ORC File format will significantly reduce the query execution time.

```
Create Table Employee_ORC (EmployeeID
int, EmployeeName varchar(100),Age int)
STORED AS ORC
tblproperties("compress.mode"="SNAPPY");
Select * from Employee Insert into
Employee_ORC;
Create Table Employee_Details_ORC
(EmployeeID int, Address varchar(100)
                    ,Designation
Varchar(100),Salary int)
STORED AS ORC
tblproperties("compress.mode"="SNAPPY");
Select * from Employee_Details Insert into
Employee_Details_ORC;
Select a.EmployeeID, a.EmployeeName,
b.Address,b.Designation from
Employee_ORC a
Join Employee_Details_ORC b
On a.EmployeeID=b.EmployeeID;
```

ORC supports compressed (ZLIB and Snappy), as well as uncompressed storage.

d) Use Partitioning:

Partitioning is the process of dividing a table into parts based on the values of the exact columns. The table contains more than one partition columns to find a specific partition. Asking questions on pieces of information is straightforward by using the partition. Partition columns information is not stored in documents. When checking the file structure, you will notice that the partition column creates folders on the idea of values. This ensures that it examines the best relevant statistics to implement a particular process, minimizing the I / O time required by the query. Thus, the question enhances the overall performance. When we query information on a partitioned desk, it simply's going to scan the walls to be applied and skip the irrelevant partitions. Now, even on the partition, suppose the records in the partition have become too heavy; additionally divide it into bucketing 'usable extra parts.

CREATE TABLE table_name (column1 data_type, column2 data_type, ...) PARTITIONED BY (partition1 data_type, partition2 data_type,....);

Partition Columns are not defined in the Column List of the table. In insert queries, partitions are mentioned in the start and their column values are also given along with the values of the other columns but at the end.

Static Partitioning

This is practiced when we are aware of the data partitions we are about to load. This should be a priority when loading data into the table from large files. It is performed in strict mode:

set hive.mapred.mode = strict;

Dynamic Partitioning

It is used when we have no knowledge about the partitions of data. It takes longer to load the data in the table. Typically, we load data into a table using another table that contains undivided data.

To enable dynamic partitioning in the hive:

SET hive.exec.dynamic.partition = true;

There are two modes of **dynamic partitioning:**

Strict: This needs at least one column to be static while loading the data.

Non-strict: This allows us to have dynamic values of all the partition columns.

SET hive.exec.dynamic.partition.mode = nonstrict;

Some other things are to be configured when using dynamic partitioning, like

Hive.exec.max.dynamic.partitions.pernode: Ma ximum number of partitions to be created in each mapper/reducer node

Hive.exec.max.dynamic.partitions: Maximum number of dynamic partitions allowed to be created in total

Hive.exec.max.created.files: Maximum number of HDFS files created by all mappers/reducers in a MapReduce job

Hive.error.on.empty.partition: Whether to throw an exception if the dynamic partition insert generates empty results

e) Use Bucketing:

Bucket provides the flexibility to further divide records into additional sections called buckets or clusters. Depending on the bucket hash feature, it depends on the type of bucket column. Records that can be bucketed through a single column fee are constantly stored in equal buckets. Clustered by provision to divide the table into buckets. It works well for columns with high cardinality.

CREATE TABLE table_name (column1 data_type, column2 data_type, …) PARTITIONED BY (partition1 data_type, partition2 data_type,….) CLUSTERED BY (clus_col1) SORTED BY (sort_col2) INTO n BUCKETS;

In Hive Partition, each **partition** will be created as a **directory**. But in Hive Buckets, each **bucket** will be created as a **file**.

set hive.enforce.bucketing = true;

Using the bucket we can also type statistics using one or more columns. Since datasheets are single-dimensional parts, map-face joining in bucket tables is faster. The bucket additionally has its own advantage when used with ORC files and when used as a joining column. We will also discuss these benefits.

f) Cost-Based Query Optimization:

Hive bees optimize the logical and physical implementation plan of each question before submitting it for final execution. However, this does not always depend on the cost in question in the early version of the hive. During subsequent versions of the hive, the question was optimized step by step with a fee (what kind should be undertaken, a way to join orders, a diploma of parallelism, etc.).

To use fee-based optimization, set it below the parameters at the beginning of the query.

```
set hive.cbo.enable=true;
set hive.compute.query.using.stats=true;
set hive.stats.fetch.column.stats=true;
set hive.stats.fetch.partition.stats=true;
```

III. MAP REDUCE IN HADOOP

MapReduce is a programming model suitable for processing large data. Hadoop can run Mapreduce programs written in different languages: Java, Ruby, Python and C ++. MapReduce programs are parallel in nature and therefore very useful for large-scale data analysis using multiple machines in a cluster.

MapReduce programs work in two phases:

1. Map phase
2. Reduce phase.

Input key-value pairs for each step. Additionally, each programmer must specify two functions: map function and minimize function.

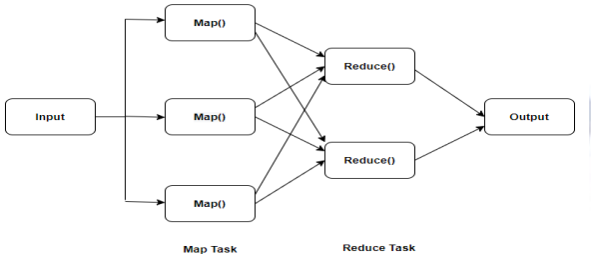


Figure. MapReduce

IV. QUERY RESPONSE TIME

Response time measures the overall performance of an individual transaction or query. Response time is usually treated as the time elapsed from the moment the user enters the command or turns on the feature until the software indicates that the command or feature is complete. Improve query processing methods when you use some hive, and then improve the query response time and the following formula is to calculate the response time better.

V. SUPERVISED LEARNING

In supervised practice, you train the machine using nicely "labeled" facts. It consulted some facts and was already tagged with an appropriate answer. This can be compared to the practice of taking acreage in the presence of a supervisor or

teacher. Learning supervised study rules from categorized academic information can help you anticipate effects for unexpectedly unpredictable statistics. Successfully building, scaling and implementing a properly monitored machine to learn fashions takes time and technology especially from a team of professional information scientists. In addition, the data scientist must reconstruct the models so that the insights given up to its statistical changes are correct. Supervised learning process: A two-step training and testing approach. One training: Learn the model using training data. Another test: Test the model using invisible test data to assess model accuracy.

Accuracy $= \frac{Number\ \ of\ \ correct\ \ classifications}{Total\ \ number\ \ of\ \ test\ \ cases}$.

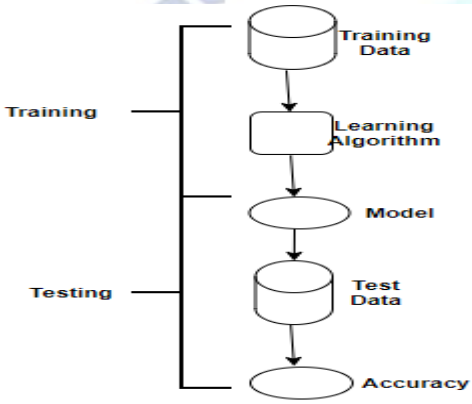


Figure. Supervised learning process.

VI. WORKING PROCESS

If you take any kind of input query, we will connect the monitored learning algorithm and the data (observations, measurements, etc.) in this model will be labeled with pre-defined classes. Test data were also categorized into these classes. Data processes a supervised slope process such as (training and testing). The next step is to translate the query processing and high-diploma questions into low-level expression. This is a step-by-step clever way that can be used in the actual implementation of the query to get the physical diploma, query optimization and result of the file gadget. Here we also apply some hive bee query processing methods, after which the response time measures the overall performance of the individual transaction or query. Response time is usually treated as the time elapsed from the moment the user enters the command or turns on the feature until the software indicates that the command or feature is complete.

Response time : $\frac{Average\ \ first\ \ response\ \ time}{sum\ \ of\ \ all\ \ first\ \ response\ \ time}$.

Now that all of its processes have been completed, retrieval query processing in Hadoop improves the response time as well as the query processing performance system.

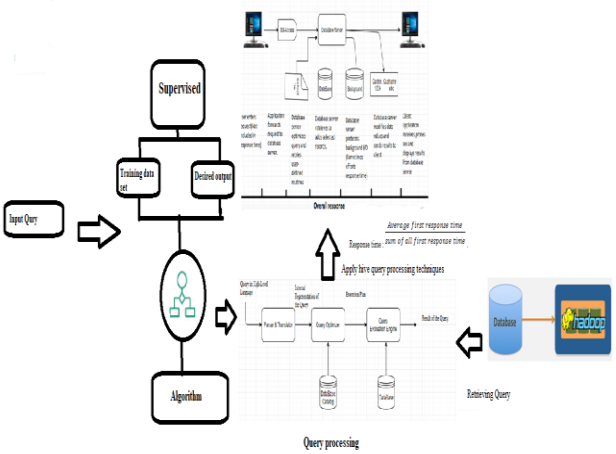


Figure. Architecture

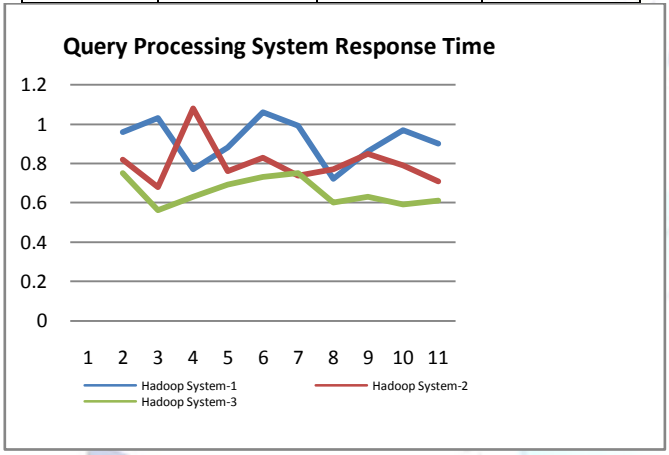
VII. EXPERIMENTAL RESULTS

Statistical method for implementing the regression model for the provided dataset. Suppose we are given the number of statistical data units. It is formula is as follows: $Q= e_0 +a_0x_0 + a_1x_1+ a_2x_2 + a_3x_3$. Therefore, Q is the target variable, the response variable is xi, the descriptive variables are, and e0 is the sum of the word squared error, in which sound is as careful. To get an additional correct estimate, we need to minimize this error term to the potential moment with the help of a regression called function. So knowing the method to be monitored means learning a model using academic statistics and checking a sample of invisible test information to assess reform accuracy. Accuracy = $\frac{\text{Number of correct classifications}}{\text{Total number of test cases}}$

Compare three different interactive hadoop systems with respect to their response times to an amendment request. Due to fluctuations in other query transactions in the process, it was decided to take ten sets of samples at randomly selected times for each hadoop system and record the average response time as follows:

Query Processing System Response time (sec)			
Query session	**Hadoop System-1**	**Hadoop System-2**	**Hadoop System-3**
1	0.96	0.82	0.75
2	1.03	0.68	0.56
3	0.77	1.08	0.63
4	0.88	0.76	0.69

5	1.06	0.83	0.73
6	0.99	0.74	0.75
7	0.72	0.77	0.60
8	0.86	0.85	0.63
9	0.97	0.79	0.59
10	0.90	0.71	0.61



The ANOVAs table for this problem can be formulated as follows:

Source of variation	**Sum of Squares**	**Degrees of Freedom**	**Mean Square**	**F**
Between treatments	0.34041	2	0.1702033	17.4276
Error	0.26369	27	0.0097663	
Total	**0.60410**	**29**		

F 2,27,0.01 5.49, and the observed value 17.4276, we rejected the null hypothesis at the 1 percent significance level. In other words there is a significant difference in the perception of the three systems.

VIII. CONCLUSION

In this work, we propose to monitor practice for response time using the query processing system and Hadoop. Fix processing and response time to recover the Hadoop query in this paper here. In this work we will apply some hive bee question processing performance methods to question processing in query retrieval in Hadoop. These simple changes in question performance can make a huge difference in response times. Improve further graphical viewing performance in the future and reduce the thirteen major limitations of the Hadoop query process.

REFERENCE

[1] Query Support for Data Processing and Analysis on Ethereum Blockchain Fariz Azmi Pratama ; Kusprasapta Mutijarsa International Symposium on Electronics and Smart Devices (ISESD), (Kusprasapta Mutijarsa et al..2018).

[2] Optimization of Multiple Queries for Big Data with Apache Hadoop/Hive Varun Garg 2015 International Conference on Computational Intelligence and Communication Networks (CICN)

[3] The research on the query optimization on the distributed heterogeneous database based on the response time Zhang Zhenyou ; Luo Bin ; Cao Zhi Proceedings of 2011 International Conference on Computer Science and Network Technology

[4] Scheduling for response time in Hadoop MapReduce Xiangming Dai ; Brahim Bensaou 2016 IEEE International Conference on Communications (ICC) Year: 2016 | Conference Paper | Publisher: IEEE.

[5] Priyank Jain,Manasi Gyanchandani & Nilay Khare"Enhanced Secured Map Reduce layer for Big Data privacy and security"in Springer,2019.

[6] Foto N. Afrati and Jeffrey D. Ullman "Optimizing Multiway Joins in a Map-Reduce Environment" in IEEE,2011.

[7] M. Srikanth ; R. N. V. Jagan Mohan"Block-level based Query Data Access Service Availability for Query Process System" in IEEE in 2020.

[8] Fariz Azmi Pratama and Kusprasapta Mutijarsa "Query Support for Data Processing and Analysis on Ethereum Blockchain" International Symposium on Electronics and Smart Devices (ISESD),

[9] M.Srikanth presented a paper titled "Query Response Time in Blockchain Using Big Query Optimization" in International conference ICRTIB-2019 – Apple Academic Press (Apple Academic Press is partnered with CRC Press, a member of the Taylor & Francis Group, for marketing and distribution worldwide) on 19th – 20th Oct 2019 at GIET University, Gunupur, Odisha.

[10] D. Vujii, D. Jagodi and S. Rani, "Blockchain technology, bitcoin, and Ethereum: A brief overview," in Proceedings of 17th International Symposium INFOTEH-JAHORINA (INFOTEH), East Sarajevo, 2018, pp. 1-6.

[11] L. Yang, Z. Kai, Y. Ying, L. Qi and Z. Xiaofang, "EtherQL: A Query Layer for Blockchain System," in Proceedings of 22nd International Conference on Database Systems for Advanced Applications (DASFAA), Suzhou, China, 2017, pp. 556-567.

[12] Ellervee, R. Matulevicius and N. Mayer, "A Comprehensive Reference Model for Blockchain-based Distributed Ledger Technology" in Proceedings of the ER Forum and the ER Demo Track, Valencia, Spain, 2017, pp 320-333.

[13] Jongbeen Han,Heemin Kim,Hyeonsang Eom,Jonathan Coignard,Kesheng Wu and Yongseok Son "Enabling SQL-Query Processing for Ethereum-based Blockchain Systems" the 9th International Conference on June 2019.