ISSN: 2321-7782 (Online)
Volume 3, Issue 4, April 2015
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Data processing in Big Data by using Hive Interface

N. Pushpalatha[1]
Assoc.Professor in CSE
Marri Laxman Reddy Institute of Technology & Management
India

P. Sudheer[2]
Asst.Professor in CSE
Marri Laxman Reddy Institute of Technology & Management
India

Abstract: Telecom service providers are dealing the huge amounts of data cards usage records every day. There is a great challenge not only to store and manage such a large amount of data, but also to analyze and extract meaningful information from it and getting the benefit out of that analysis. There are several approaches to collecting, storing, processing, and analysing big data .Present these analysis activities are happening using data warehousing technologies. But it is more expensive and time consuming. To help better in this area, we are using the Hadoop and Hadoop Eco-systems.

The main objective of this paper is finding the business insights of current user records data. And get the benefits for business growth. The parameters to be considered for analysis are Daily user count and bytes transmitted on a particular time slot. Area wise business (usage) share in the total business. Since every network owner will be depending on partners to get the service where they does not have the service tower.

Keywords: Hive, Big Data, Hadoop, HDFS

I. Introduction

There is a lot of buzz around "big data" and rightly so. Organizations that are capturing and analysing large amounts of data in real time or near–real time are creating significant competitive advantages for themselves, their customers and business partners. Communications service providers (CSPs) are no exception. CSPs that can ingest and analyse network, location and customer data in real time or near–real time have much to gain. They will be able to quickly introduce new capabilities such as location-based services, intelligent marketing campaigns, next best actions for sales and service, social media insights, network intelligence and fraud detection to significantly increase revenues and reduce costs.

We are living in an information age and there is enormous amount of data that is flowing between systems, internet, telephones, and other media. The data is being collected and stored at unprecedented rates. There is a great challenge not only to store and manage the large volume of data, but also to analyze and extract meaningful information from it. There are several approaches to collecting, storing, processing, and analyzing big data. The main focus of the paper is to draw an analogy for data management between the traditional relational database systems and the Big Data technologies. Aim of this project is finding the business insights of current user records data (i.e., data cards usage records). And get the benefits for business growth. The parameters to be considered for analysis are:

Daily user count and bytes transmitted on a particular time slot.

Area wise business(usage) share in the total business

Data processing with Hive

Hive is a Data Warehouse software that facilitates querying and managing huge data residing in distributed storage .Instead of writing huge raw map reduce programs in some programming language, Hive provides a SQL-like interface to data stored in Hadoop File System. And there is another popular Hadoop eco-system i.e Pig which is a scripting language with a focus on data flows. Hive provides a database query interface to Apache Hadoop. People often ask why do Pig and Hive exist when they seem to do much of the same thing. Hive because of its SQL like query language is often used as the interface to an Apache Hadoop based data warehouse. Hive is considered friendlier and more familiar to users who are used to using SQL for querying data. Pig fits in through its data flow strengths where it takes on the tasks of bringing data into Apache Hadoop and working with it to get it into the form for querying. A good overview of how this works is in Alan Gates posting on the Yahoo Developer blog titled Pig and Hive at Yahoo! From a technical point of view both Pig and Hive are feature complete so you can do tasks in either tool. However you will find one tool or the other will be preferred by the different groups that have to use Apache Hadoop. The good part is they have a choice and both tools work together. Since every network owner will be depending on partners to get the service where they does not have the service tower.

II. Proposed Architecture

After analyzing all the requirements I have designed and going to implement the fallowing architecture. As we see in the fallowing figure first we are going to load the user's data card usage records data (.csv files) from MySql RDBMS into Hadoop HDFS and then that data we are going to process with some other bigdata technology called Hive and after that we will be exporting the results back to our MySql RDBMS and generating the reports on that.

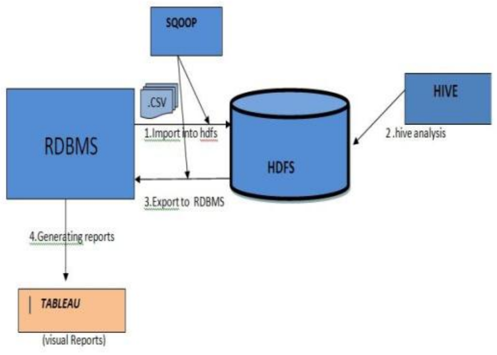


Figure :Proposed Architecture

HDFS Architecture Design

The figure below gives a run-time view of the architecture showing three types of address spaces: the application, the NameNode and the DataNode. An essential portion of HDFS is that there are multiple instances of DataNode.

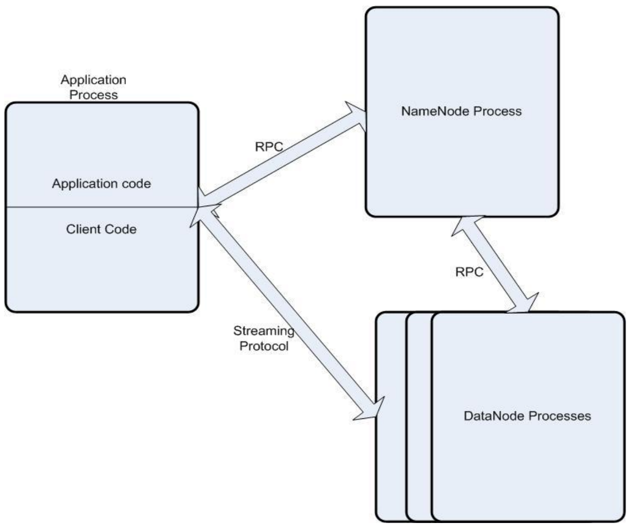


Figure :HDFS Design Architecture

The application incorporates the HDFS client library into its address space. The client library manages all communication from the application to the NameNode and the DataNode. An HDFS cluster consists of a single NameNode—a master server that manages the file system namespace and regulates access to files by clients. In addition, there are a number of DataNodes, usually one per computer node in the cluster, which manage storage attached to the nodes that they run on.

The NameNode and DataNode are pieces of software designed to run on commodity machines. These machines typically run a GNU/Linux operating system (OS). HDFS is built using the Java language; any machine that supports Java can run the NameNode or the DataNode software. Usage of the Java language means that HDFS can be deployed on a wide range of machines. A typical deployment has a dedicated machine that runs only the NameNode software. Each of the other machines in the cluster runs one instance of the DataNode software. The architecture does not preclude running multiple DataNodes on the same machine but in a real deployment that is rarely the case. **The figure below shows how blocks are replicated on different DataNodes.**

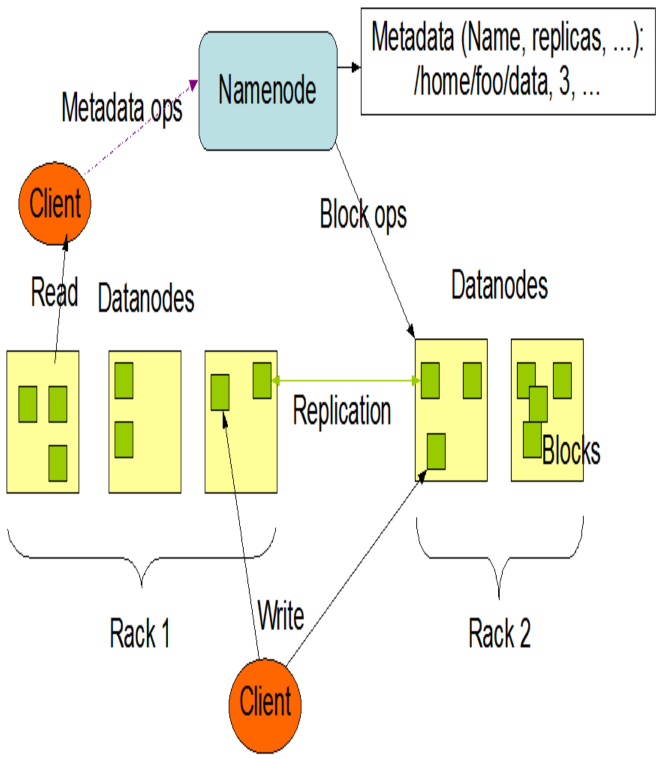


Figure :Blocks Replication in DataNodes

Blocks are linked to the file through INode. Each block is given a timestamp that is used to determine whether a replica is current.

Apache Hive

Hive is a data warehouse that uses MapReduce to analyze data stored on HDFS. In particular,it provides a query language called HiveQL that closely resembles the common StructuredQuery Language (SQL) standard.

Why Hive

Actually we are Developing Map Reduce Programs, we have Hadoop Streaming and explained that one large benefit of Streaming is how it allows faster turn-around in the development of Map Reduce jobs. Hive takes this a step further. Instead of providing away of more quickly developing map and reduce tasks, it offers a query language based on the industry standard SQL. Hive takes these HiveQL statements and immediately and automatically translates the queries into one or more Map Reduce jobs. It then executes the overall Map Reduce program and returns the results to the user. Whereas HadoopStreaming reduces the required code/compile/submit cycle, Hive removes it entirely and instead only requires the composition of HiveQL statements. This interface to Hadoop not only accelerates the time required to produce results from data analysis, it significantly broadens who can use Hadoop and Map Reduce. Instead of requiring software development skills, anyone with a familiarity with SQL can use Hive.The combination of these attributes is that Hive is often used as a tool for business and dataanalysts to perform ad hoc queries on the data stored on HDFS. Direct use of MapReducerequires map and reduce tasks to be written before the job can be executed which meansa necessary delay from the idea of a possible query to its execution. With Hive, the dataanalyst can work on refining HiveQL queries without the ongoing involvement of a software developer. There are of course operational and practical limitations (a badly written querywill be inefficient regardless of technology) but the broad principle is compelling.

Hive Internal Working

Hive internal design includes the fallowing

UI - The user interface for users to submit queries and other operations to the system. Currently the system has a command line interface and a web based GUI is being developed.

Driver - The component which receives the queries. This component implements the notion of session handles and provides execute and fetch APIs modeled on JDBC/ODBC interfaces.

Compiler - The component that parses the query, does semantic analysis on the different qurey blocks and query expressions and eventually generates an execution plan with the help of the table and partition metadata looked up from the metastore.

Metastore - The component that stores all the structure information of the various tables and partitions in the warehouse including column and column type information, the serializers and de-serializers necessary to read and write data and the corresponding hdfs files where the data is stored.

Execution Engine - The component which executes the execution plan created by the compiler. The plan is a DAG of stages. The execution engine manages the dependencies between these different stages of the plan and executes these stages on the appropriate system components.

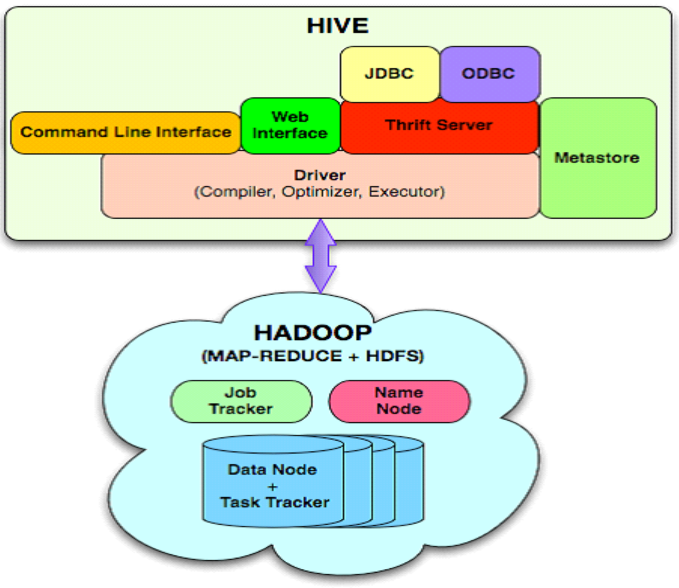


Figure :Hive Architecture

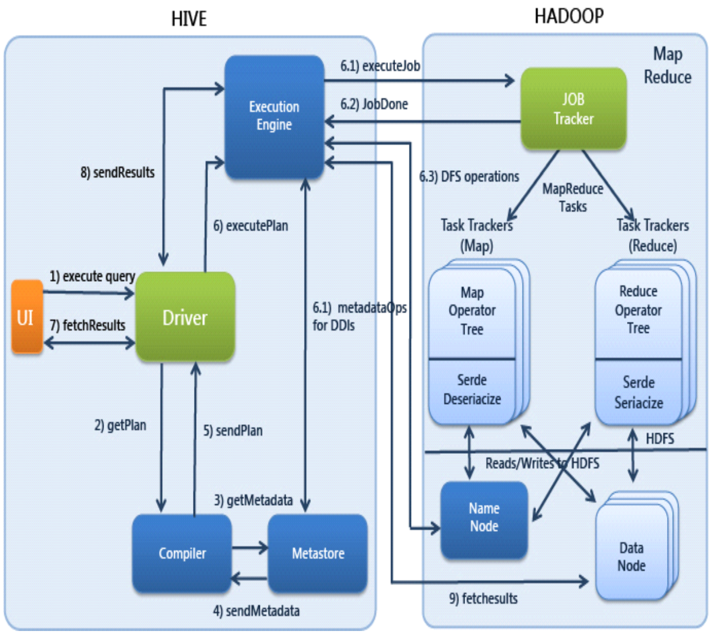


Figure:Hive Job execution flow

Step 1

The UI calls the execute interface to the Driver.

Step 2

The Driver creates a session handle for the query and sends the query to the compiler to generate an execution plan.

Step 3

The compiler gets the necessary metadata from the metastore. This metadata is used to typecheck the expressions in the query tree as well as to prune partitions based on query predicates .

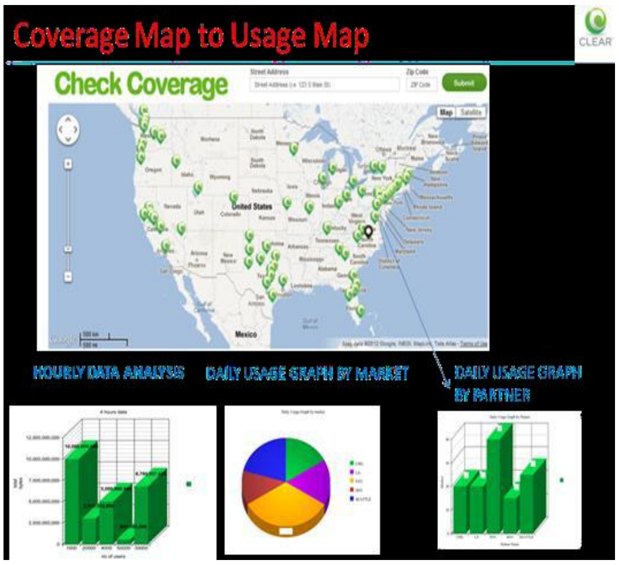
Step 4

The plan generated by the compiler is a DAG(Direct acyclic graph) of stages with each stage being either a map/reduce job, a metadata operation or an operations on hdfs. For map/reduce stages, the plan contains map operator trees(operator trees that are executed on the mappers) and a reduce operator tree(for operations that need reducers).

Step 5

The execution engines submit these stages to appropriate components. In each task(mapper/reducer) the de-serializer associated with the table or intermediate outputs is used to read the rows from hdfs files and these are passed through the associated operator tree. Once the output is generated, it is written to a temporary hdfs file though the serializer(this happens in the mapper in case the operation does not need a reduce). The temporaryfiles are used to provide data to subsequent map/reduce stages of the plan. For DML operations the final temporary file is moved to the tables location. This scheme is used to ensure that dirty data is not read(file rename being an atomic operation in hdfs). For queries, the contents of the temporary file are read by the execution engine directly from hdfs as part of the fetch call from the Driver.

Expecting Results from the reporting tool


Coverage Map to Usage Map
CLEAR
Check Coverage
Submit
HOURLY DATA ANALYSIS
DAILY USAGE GRAPH BY MARKET
DAILY USAGE GRAPH BY PARTNER

Expecting Output Reports

III. CONCLUSION

We found the business insights of current user records data (i.e., data cards usage records). And get the benefits for business growth. The parameters to be considered for analysis and gave them the results like Daily user count and bytes transmitted on a particular time slot, Area wise business(usage) share in the total business and Since every network owner will be depending on partners to get the service where they does not have the service tower. We solved the Problem Statement present in existing system in this project.

References

1. Big Data and Cloud Computing: Current State and Future Opportunities 2013.
2. D. Agrawal, S. Das, and A. E. Abbadi. Big data and cloud computing: New wine or just new bottles? PVLDB, 3(2):1647–1648,2010.
3. D. Agrawal, A. El Abbadi, S. Antony, and S. Das. Data Management Challenges in Cloud Computing Infrastructures. In DNIS, pages1–10, 2010.
4. P. Agrawal, A. Silberstein, B. F. Cooper, U. Srivastava, and R. Ramakrishnan. Asynchronous view maintenance for vlsd databases. In SIGMOD Conference, pages 179–192, 2009.
5. S. Aulbach, D. Jacobs, A. Kemper, and M. Seibold. A comparison of flexible schemas for software as a service. In SIGMOD, pages881–888, 2009.
6. "Understanding Hadoop Clusters and the Network." Available at http://bradhedlund.com. Accessed on June 1, 2013.
7. Sammer, E. 2012. Hadoop Operations. Sebastopol, CA: O'Reilly Media.