International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 04 Issue 05
April 2017

Putting Avro into Hive

S. Sreekanth[1]. A Sai Ram Pramodhini[2]. Ch S Likita[3]. Chiluka Manisha[4]
Associate Professor, Department of CSE, Guru Nanak Institutions, Ibrahimpatnam, Hyderabad, India[1]
B.Tech Student, Department of Computer Science & Engineering, Guru Nanak Institutions, Hyderabad, India[2]
B.Tech Student, Department of Computer Science & Engineering, Guru Nanak Institutions, Hyderabad, India[3]
B.Tech Student, Department of Computer Science & Engineering, Guru Nanak Institutions, Hyderabad, India[4]

ABSTRACT

Avro is an Apache™ open source project that provides data serialization and data exchange services for Hadoop®. These services can be used together or independently. Using Avro, big data can be exchanged between programs written in any language. Using the serialization service, programs can efficiently serialize data into files or into messages. The data storage is compact and efficient. Avro stores both the data definition and the data together in one message or file making it easy for programs to dynamically understand the information stored in an Avro file or message. Avro stores the data definition in JSON format making it easy to read and interpret, the data itself is stored in binary format making it compact and efficient. Avro files include markers that can be used to splitting large data sets into subsets suitable for MapReduce processing. Some data exchange services use a code generator to interpret the data definition and produce code to access the data. Avro doesn't require this step, making it ideal for scripting languages. Overview – Working with Avro from Hive The AvroSerde allows users to read or write Avro data as Hive tables. The AvroSerde's bullet points: Infers the schema of the Hive table from the Avro schema. Starting in Hive 0.14, the Avro schema can be inferred from the Hive table schema. Reads all Avro files within a table against a specified schema, taking advantage of Avro's backwards compatibility abilities Supports arbitrarily nested schemas. Translates all Avro data types into equivalent Hive types. Most types map exactly, but some Avro types don't exist in

Hive and are automatically converted by the AvroSerde. Understands compressed Avro files. Transparently converts the Avro idiom of handling nullable types as Union[T, null] into just T and returns null when appropriate. Writes any Hive table to Avro files. Has worked reliably against our most convoluted Avro schemas in our ETL process. Starting in Hive 0.14, columns can be added to an Avro backed Hive table using the Alter Table statement.

Keywords: Data exchange, Hive tables, Avroserde, Mapreduce, markers

INTRODUCTION

BIG DATA

-"Big data refers to data sets whose size is beyond the ability of typical database software tools to capture, store, manage and analyze." - The McKinsey Global Institute, 2012

The term big data refers to the data that is generating around us everyday life. It is generally exceeds the capacity of normal conventional traditional databases. Big data is recycled ubiquitously at the present in disseminated archetype on web. It is the group of collections of massive volume of data. That's the reason why big data came into picture in the real time business analysis of processing data. Some well-known internet companies like Google, Amazon, LinkedIn, Yahoo! etc. have generated a huge amount of structured and unstructured data every day. This exponential growth of data leads to some challenges like processing of large data sets, extraction of useful information from online generated data sets etc. This company have generated a huge amount of structured and unstructured data every day.

EXISITING SYSTEM

In Relational Database System in order to extract large amount of data it takes large amount of time and complexity arises. In non-partitioned tables, Hive would have to read all the files in a table's data directory and subsequently apply filters on it. This is slow and expensive –especially in cases of large tables.

DISADVANTAGES

- Complexity is high
- Runtime for query processing is high

PROPOSED SYSTEM

The concept of partitioning is not new for folks who are familiar with relational databases. Partitions are essentially

horizontal slices of data which allow larger sets of data to be separated into more manageable chunks. In Hive, partitioning is supported for both managed and external tables in the table. Multi-column partition is supported just we need to include the partition columns in the table definition and can still use them in query projections. The partition statement lets Hive alter the way it manages the underlying structures of the table's data directory.

ADVANTAGES

- Hive can improve developer productivity when working with challenging data formats or complex analytical tasks.
- Less time is required for processing a query

PROJECT DESCRIPTION

In this we will be discussing the concept of AVRO in Hive, which gives a fine structure to Hive tables while performing queries on large datasets.

As we all know, Partition helps in increasing the efficiency when performing a query on a table. Instead of scanning the whole table, it will only scan for the partitioned set and does not scan or operate on the unpartitioned sets, which helps us to provide results in lesser time and the details will be displayed very quickly because of Hive Partition.

Now, let's assume a condition that there is a huge dataset. At times, even after partitioning on a particular field or fields, the partitioned file size doesn't match with the actual expectation and remains huge and we want to manage the partition results into different parts. To overcome this problem of partitioning, Hive provides Bucketing concept, which allows user to divide table data sets into more manageable parts.

BIG DATA TECHNOLOGIES

Big data technologies are important in providing more accurate analysis, which may lead to more concrete decision-making resulting in greater operational efficiencies, cost reductions, and reduced risks for the business. To harness the power of big data, you would require an infrastructure that can manage and process huge volumes of structured and unstructured data in real-time and can protect data privacy and security. There are various technologies in the market from different vendors including Amazon, IBM, Microsoft, etc., to handle big data. While looking into the technologies that

handle big data, we examine the following two classes of technology:

(i) **Operational Big Data**

(ii) **Analytical Big Data**

AVRO

Avro is a remote procedure call and data serialization framework developed within Apache's Hadoop project. It uses JSON for defining data types and protocols, and serializes data in a compact binary format. Its primary use is in Apache Hadoop, where it can provide both a serialization format for persistent data, and a wire format for communication between Hadoop nodes, and from client programs to the Hadoop services.

It is similar to Thrift and Protocol Buffers, but does not require running a code-generation program when a schema changes (unless desired for statically-typed languages). Apache Avro™ is a data serialization system.

Avro provides:

- Rich data structures.
- A compact, fast, binary data format.
- A container file, to store persistent data.
- Remote procedure call (RPC).
- Simple integration with dynamic languages. Code generation is not required to read or write data files nor to use or implement RPC protocols. Code generation as an optional optimization, only worth implementing for statically typed languages.

Schemas

Avro relies on schemas. When Avro data is read, the schema used when writing it is always present. This permits each datum to be written with no per-value overheads, making serialization both fast and small. This also facilitates use with dynamic, scripting languages, since data, together with its schema, is fully self-describing.

When Avro data is stored in a file, its schema is stored with it, so that files may be processed later by any program. If the program reading the data expects a different schema this can be easily resolved, since both schemas are present.

When Avro is used in RPC, the client and server exchange schemas in the connection handshake. (This can be optimized so that, for most calls, no schemas are actually

transmitted.) Since both client and server both have the other's full schema, correspondence between same named fields, missing fields, extra fields, etc. can all be easily resolved.

Avro schemas are defined with JSON . This facilitates implementation in languages that already have JSON libraries.

DEVELOPMENT TOOLS

This chapter is about the software language and the tools used in the development of the project. The platform used here is JAVA.

FEATURES OF JAVA

Java is a programming language originally developed by James Gosling at Sun Microsystems and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to byte code that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is general-purpose, concurrent, class-based, and object-oriented, and is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere".

Java is considered by many as one of the most influential programming languages of the 20th century, and is widely used from application software to web applications The java framework is a new platform independent that simplifies application development internet.Java technology's versatility, efficiency, platform portability, and security make it the ideal technology for network computing. From laptops to datacenters, game consoles to scientific supercomputers, cell phones to the Internet, Java is everywhere!

HADOOP (HDFS)

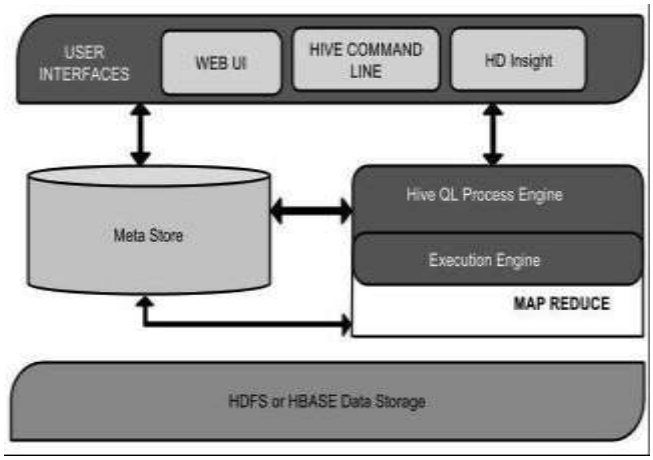
The **Hadoop distributed file system** (**HDFS**) is a distributed, scalable, and portable file-system written in Java for the Hadoop framework. A Hadoop cluster has nominally a single name node plus a cluster of data nodes, although redundancy options are available for the name node due to its criticality. Each data node serves up blocks of data over the network using a block protocol specific to HDFS. The file system uses TCP/IP sockets for communication. Clients use remote procedure call (RPC) to

communicate between each other.HDFS stores large files (typically in the range of gigabytes to terabytes) across multiple machines. It achieves reliability by replicating the data across multiple hosts, and hence theoretically does not require RAID storage on hosts (but to increase I/O performance some RAID configurations are still useful). With the default replication value, 3, data is stored on three nodes: two on the same rack, and one on a different rack. Data nodes can talk to each other to rebalance data, to move copies around, and to keep the replication of data high. HDFS is not fully POSIX-compliant, because the requirements for a POSIX file-system differ from the target goals for a Hadoop application. The tradeoff of not having a fully POSIX-compliant file-system is increased performance for data throughput and support for non-POSIX operations such as Append.

HDFS added the high-availability capabilities, as announced for release 2.0 in May 2012, letting the main metadata server (the NameNode) fail over manually to a backup. The project has also started developing automatic fail-over.

ARCHITECTURE OF HIVE

The following component diagram depicts the architecture of Hive:



Hive Architecture

IMPLEMENTATION

All Hadoop sub-projects such as Hive, Pig, and HBase support Linux operating system. Therefore, you need to install any Linux flavored OS. The following simple steps are executed for Hive installation

VERIFYING JAVA INSTALLATION

Java must be installed on your system before installing Hive. Let us verify java installation using the following command:

$ java –version

If Java is already installed on your system, you get to see the following response:

java version "1.7.0_71"

Java(TM) SE Runtime Environment (build 1.7.0_71-b13)

Java HotSpot(TM) Client VM (build 25.0-b02, mixed mode)

If java is not installed in your system, then follow the steps given below for installing java.

CONCLUSION

This improvement was especially evident in the case of tables that were holding large historical data — prior to partitioning, a full table scan of these tables was done in order to collect the stats. Partitioning also enabled us to selectively expire portions of data without having to rebuild the table.

By separating the schema and tables , the data can be edited easily .the tables structure is stored in hive and the schema is stored in hdfs. The data can be added and removed easily without effecting the entire table directly but changing the schema which in turn will make necessary changes in the table. By this even in large amount of data, the data can be added or removed from the tables. Avro is an Apache provides data serialization and data exchange services for Hadoop. Changes are done only to the schema in hadoop but not to the tables which are stored in hive.

REFERENCES

[1] Bittencourt, L.F. and Madeira, E.R.M. "A Performance-Oriented Adaptive Scheduler for Dependent Tasks on Grids," Concurrency and Computation: Practice and Experience.

[2] Caron, E. Chis, A. Desprez, F. And Su, A. "Design of Plug-in Schedulers for a GRIDRPC Environment," Future Generation Computer Systems, vol. 24, no. 1, pp. 46-57.

[3] Dinda, P.A. And O'Hallaron, D.R. "Host Load Prediction Using Linear Models," Cluster Computing, vol. 3, no. 4, pp. 265-280.

[4] Dinda, P.A. "Design, Implementation, and Performance of an Extensible Toolkit for Resource Prediction in Distributed Systems," IEEE Trans. Parallel and Distributed Systems, vol. 17, no. 2, b pp. 160-173.

[5] Eddy Caron, Andreea Chis, Frederic Desprez, Alan Su (November 2011) "Plug-

in Scheduler Design for a Distributed Grid Environment".

[6] Liang Hu, Xi-Long Che, (2012)"Online System for Grid Resource Monitoring and Machine Learning-Based Prediction" IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 23.

[7] Massie, M.L. Chun, B.N. And Culler, D.E. "The Ganglia Distributed Monitoring System: Design, Implementation, and Experience," Parallel Computing, vol. 30, no. 7, pp. 817-840.

[8] Peter Dinda, A. and David R. O'Halloran (July 2012) "AN Extensible Toolkit for Resource Prediction in Distributed Systems" School of Computer Science Carnegie Mellon University Pittsburgh, PA, 15213.

[9]Sam Verboven, Peter Hellinckx, Frans Arickx and Jan Broeckhove (2011) "Runtime Prediction based Grid Scheduling of Parameter Sweep Jobs" University of Antwerp Antwerp, Belgium.