ANALYSIS OF BILL OF MATERIAL DATA USING KAFKA AND SPARK

Ashwitha Jain[*], Dr. Venkatramana Bhat P [**]

[*] Student, Department of Computer Science & Engineering, Mangalore Institute of Technology & Engineering
[**] Professor, Department of Computer Science & Engineering, Mangalore Institute of Technology & Engineering

Abstract- With the growing technology and exploding data the need to manage data and process it in real-time is also increasing. Hence all the incoming data needs to be handled in a fraction of seconds to get the value out of it. This type of data can be addressed as Big Data with respect to its Volume, Velocity and Variety. Managing such huge data is expensive and time consuming. The data set considered in this paper is the Engineering Bill of Material data. Since the BOM data is inevitable, it needs to be managed to ensure its completeness, consistency and correctness. In-order to outdo these problems in this paper Hadoop is proposed to store data which is cost effective. To address the data processing issues Spark and Kafka are used. The data is streamed via Kafka which is a message streaming tool into Spark which is a data processing tool and then exported into local database via Sqoop. Both Kafka and Spark can deal with incoming data in real-time and hence it addresses the issue of time consumption. The purpose of this paper is to compare the incoming and existing Engineering BOM data in Hive or Hadoop Distributed File System. Finally the resultant changes are exported to different production plants.

Index Terms - Big data, Hadoop, HDFS, Bill of Material (BOM), Kafka, Spark, Hive, Sqoop.

I. INTRODUCTION

With frequent innovations and evolutions in the emerging technology the data being produced is increasing every day. It has been tuned that data is being generated in-terms of terabytes to petabytes every day. Big data is most often produced due to the sensors, log files, audio messages, video messages, social media websites, network packets and web. Data is being produced in enormous volumes with greater velocity and in all the formats including structured data, unstructured data and semi-structured data for multiple sources. It is necessary to focus on how to store these data, process it and then further analyze it efficiently so that it can be productive. Many of the organizations are trying to capture every bit of this data and analyze it because it has several hidden information in it. This hidden information can be exposed only when the entire data is captured and analyzed. Unfiltered data can add unpredicted value to the business in the decision making processes. This kind of processing involves very complicated workloads beyond the boundaries of traditional data management and data warehousing techniques. The actual techniques and technologies used depend on how complex the processing workload is, what is the volume of data and what are its varieties. In addition to the techniques and technologies used, it is necessary to understand the benefits of timely decision making. This is possible only when the incoming data can be stored efficiently, handled and processed in real-time.

Hadoop can be the best possible choice to fulfill these requirements since it can store bulk data with greater operational efficiency and cost reduction. In addition to this it also ensures other beneficial features like reliability, scalability and flexibility [1]. Hadoop has the ability to store enormous data in Hadoop Distributed File System (HDFS) and it allows this data to be analyzed using its MapReduce programming model [2]. Hadoop allows processing of structured and unstructured data ranging from terabytes to petabytes. During the Map phase input data is broken down into key-value pairs for processing purpose. During the reduce phase the processed data is accumulated back together based on the key.

HDFS has two components- namenode and the datanode. Namenode contains the metadata and datanode contains the actual data stored in it. Hadoop aims at processing the data with least possible latency, to achieve this mapper is executed on the same node where the data resides this is known as data locality. In addition to this, Hadoop exhibits *Scale out* capability and it can process data in parallel. By executing the data in parallel it reduces the data execution time massively and scale out architecture allows the system to add new hardware and configure it as the need arises. This characteristic of Hadoop is advantageous over the traditional database server which makes use of the *Scale up* architecture where additional resources like processors and shared storage needs to be added in same server and hence processing cannot be done in parallel. Fig 1 shows the Scale out architecture where the system expands horizontally i.e, new data storage devices and hardware is added and configured when the need arises. Fig 2 shows the scale up architecture where the system expands vertically i.e, existing resources in the system are upgraded rather than adding new resources.

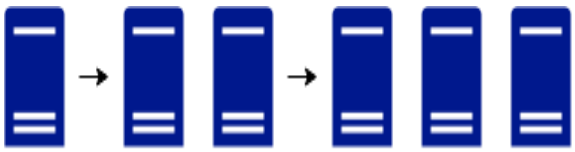


Fig 1: Scale out Architecture

Fig 2: Scale up Architecture

Processing data in Hadoop is efficient using its MapReduce programming model but, in certain scenarios when it has to deal with iterative algorithms MapReduce is not the right choice [3]. After every Map phase, the resultant output is taken from memory and it is exported into disk. Hence in an iterative algorithm every time the data needs to be imported and exported to and from disk. This is a very time consuming process. Solution to this problem is to make use of Spark [4], which performs all the computations in memory and persist the data in memory whenever necessary.

II. Bill of Material

Bill of material is a document which includes all the information about what are the parts included in manufacturing a product [5]. This BOM data contains all the necessary information starting from a smallest component to the most complex ones. There are different varieties of Bill of material [6]:

A. Engineering Bill of Material

Engineering BOM can be defined as an alternative for the manufacturing BOM. After you release the engineering bill to manufacturing and take customer orders for specific configurations, Bill of Material creates the new configuration item and automatically assigns values to each catalog descriptive element.

B. Standard Bill of Material

A standard bill of material is the most common type of bill and lists the mandatory components, the required quantity of each component, and information to control work in process, material planning.

C. Option Class Bill of Material

An option class is an item that groups optional components on a bill. An option class is an item that becomes a level in your model bill of material. Option classes can also have mandatory components that apply for all of its options.

D. Planning Bill of Material

A planning bill of material is a bill of material structure that includes a percentage distribution for its components.

In this paper the Engineering Bill of material is considered. So here the Engineering BOM contains all the information required to manufacture a product.

III. DATA INGESTION & PROCESSING

In this research the first step is to gather the Bill of material data and then ingest it into Kafka, it is the data ingestion phase. This is followed by processing phase where the ingested Bill of material data is compared with the existing Bill of material data already stored in HDFS or Hive. In this paper Kafka and Spark are the two tools used for data ingestion and data processing in real-time. In data ingestion phase data is extracted from the data source and ingested via Kafka. In data processing phase, Spark can then be configured to pull data from Kafka as and when it is available.

A. KAFKA

Kafka is a distributed message streaming system that has the ability to stream data in real-time. It is based on publish-subscribe model. Since Kafka is distributed in nature, it can be scaled-out easily [7]. Kafka can persist messages on disk and hence can be used to stream data in real-time applications. In this paper Kafka is used to stream messages in real-time.

Kafka includes several components which plays an important role in streaming messages, namely: Kafka producer, Kafka consumer, Kafka topic, Kafka broker and zookeeper. Kafka topic [8] is a collection of messages, all the data that is of interest to the application are collected here. Here the Bill of material data is published with the help of Kafka producer and all the consumers subscribe to this topic in order to consume the messages stored in Kafka topic.

(i)Kafka Topic- Topic can be defined as stream to which the messages are published. Hence a Kafka Topic can be seen as a collection of messages. Kafka topics can be created and deleted as and when required by the user. It is possible to set the replication factor for the topics.

(ii)Kafka Producer- Producer is the one who publishes the messages into Kafka topic. Producer first sends messages to Kafka Broker where the leader is located which is then published into the topic.

(iii)Kafka Consumer- Consumer is the one who subscribes to the messages in the Topic. Consumer pulls the messages from the topic when it is ready to consume the messages. It also has the ability to pull messages corresponding to its current capacity.

(iv)Kafka Broker- Kafka broker is stateless it does not have to maintain the number of messages consumed by the Kafka consumer. It is done by the Kafka consumer itself by maintaining offsets. Hence Kafka handles this by using time based retention policy where the broker keeps the message for certain amount of time after which it is not accessible

(v)Zookeeper- It is used to maintain the coordination between the Kafka brokers. Zookeeper is used to notify the producers and the consumers about the availability or failure of the Kafka brokers.

B. SPARK

Spark is a distributed messaging processing tool that can process the data in-memory and persist it memory. Spark makes use of RDDs (Resilient Distributed Datasets) to process data [9]. RDDs are simply collection of datasets. These RDDs can be created in two ways either by parallelizing an RDD from already existing collection or by making a reference to RDDs stored in external storage. There are two operations that can be performed in an RDD namely:

(i)Transformations: Transformations crate new datasets by applying certain operations onto the existing dataset. These operations may be applying a map function to a dataset or applying any other filter operation.

(ii)Actions: Actions give the result of the transformations. They return corresponding value to the sparks driver program on completing the execution.

The transformation operation performed in Spark is called lazy operation since it does not give the result immediately to the driver program rather it memorizes the transformation operations applied on datasets. Spark has the ability to persist dataset in memory for any time frame specified by the user. When persist operation is applied in Spark all the nodes that have its partitions will persist them. This feature of Spark allows the computations to be faster.

C. SQOOP

Sqoop is a data transfer tool that is extensively used to import data from local database into Hadoop or it can also be used to export data from Hadoop into local database [10]. Sqoop performs data transfer in parallel hence the performance is faster and efficient system utilization. Sqoop can import and export data between local database and other Hadoop supported tools like Hive, HBase.

D. HIVE

Hive provides the data warehousing solution for Hadoop. Hive enables storing data in Hadoop and querying it using HiveQL which is a query language like SQL. In this paper the data is stored in Hive after comparison and then it is exported into local databases using Sqoop. HiveSerDe is another feature of Hive used in this paper which is used to read the semi-structured xml data.

IV. INTEGRATION OF KAFKA & SPARK

In this paper Engineering BOM data is streamed using Kafka and it is further processed using Spark. Both the data ingestion and data processing operations can be performed in real-time. Extracting the BOM data as and when it is available in the data source (data source can be a data server) and then immediately streaming it via Kafka makes the BOM data available for real-time processing.

In the data processing phase, the resultant value after comparing the BOM data streamed into Spark with the BOM data already stored in Hive or HDFS is exported using Sqoop. To carry out this process, it is essential to check the correctness of the BOM data initially, because after the ingestion phase the BOM data is directly compared with other existing BOM data stored in HDFS. Hence any defects in BOM will lead to improper productions. Fig 3 shows the flow chart for data ingestion, processing and then exporting the resultant delta value.

Here the engineering BOM data is extracted from the data source and it is ingested via Kafka into Spark. In Kafka when the message is published to the Kafka topic zookeeper will be updated. This message will be consumed by the consumer based on its requests. The consumer sends the request along with an offset. This offset specifies the position of the message from where it wants to read. These offsets are maintained by the Zookeeper and every time there is communication between the Kafka producer, Kafka consumer and the Zookeeper. This communication ensures that the message is ingested into Spark only once and hence it avoids redundancy. Since zookeeper keeps Kafka topic, producer and consumer coordinated with each other, Kafka consumer can be accurate about which messages are successfully ingested into Spark. This is done by keeping track of the message offsets. Hence it ensures that no message is delivered to Spark multiple times.

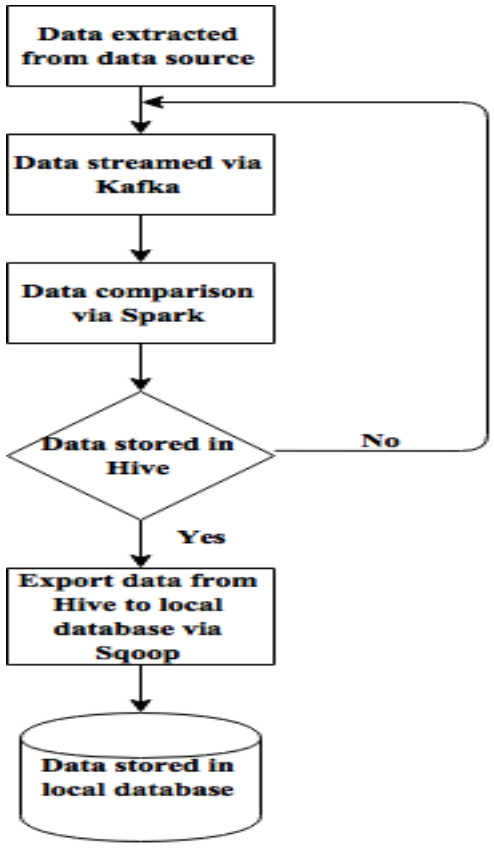


Fig 3: Flow chart for ingestion, processing and exporting Bill of material data

BOM data ingested via Kafka is in xml format. Spark puts this xml data into HDFS or Hive so it can be furthered compared. During the comparison, if the BOM data is ingested for the first time, it is directly stored without any comparison. Next time when the BOM data is ingested, it is compared with the already existing BOM data. After the comparison the old BOM data is deleted and it is then replaced with new BOM data. Finally, the change between these two BOM data is stored in HDFS. This computed value can then be exported into local database via Sqoop so that it can be queried and accessed from local database.

V. RESULT & ANALYSIS

The map reduce model used by Hadoop dumps the data into disk after every map and reduce task. Fetching the data from disk after every map task and reduce task is very much time consuming. This makes it disadvantageous to be used in iterative algorithms. Figure 4 shows the performance measurements for Spark and Hadoop. Figure 5 shows the time taken by Spark and Hadoop to execute an iterative algorithm.

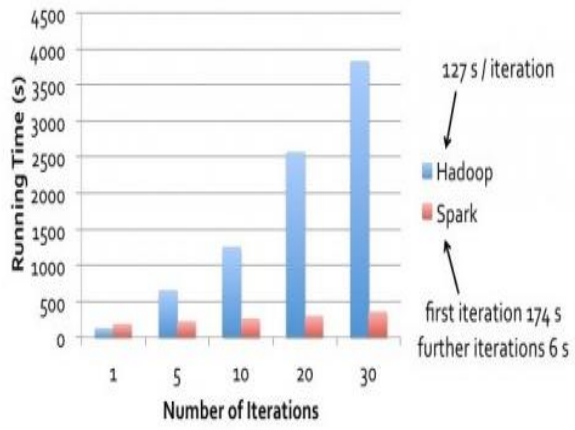
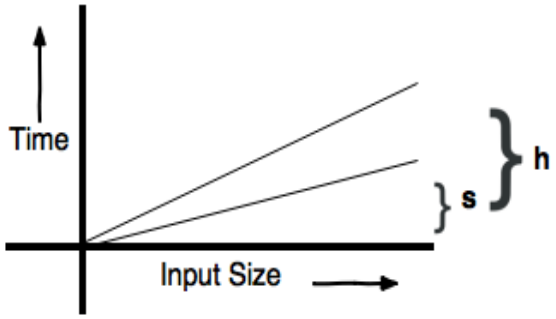


Fig 4: Performance measurement for Spark and Hadoop



s = Time taken by Spark.

h = Time taken by Hadoop.

Fig 5: Time taken to execute an iterative algorithm in Spark and Hadoop

From Figure 4 and 5, it is seen that as the input data size increases in an iterative algorithm the time taken for execution also increases in Hadoop as well as Spark. But the time taken by Spark to complete the execution is approximately 10 times less than that of Hadoop. Thus in Hadoop time taken to execute an iterative algorithm increases exponentially with respect to the size of data. This can be represented using asymptotic notation as follows:

$$\mathbf{T(n) = O(n^2)}$$

But, in Spark the time taken to execute an iterative algorithm varies linearly with respect to the size of dataset. This can be represented using asymptotic notation as follows:

$$\mathbf{T(n) = O(n)}$$

To overcome this, Spark is used for data processing, it can process data in real-time. Hadoop takes 100 times more time than Spark to process iterative algorithm. Spark set a new world record in 100TB sorting, beating the previous record held by Hadoop MapReduce by three times, using only one-tenth of the resources; it received a new SQL query engine with a state-of-the-art optimizer; and many of its built-in algorithms became five times faster [11]. Spark claims to process data 100x faster than MapReduce, while 10x faster with the disks [12]. The data that can be processed in hours using Hadoop alone can be processed in seconds using Spark. When Kafka and Spark are integrated together, it has the ability to perform ingestion and processing tasks in real-time. Real-time data analysis means processing data generated by the real-time event streams coming in at the rate of millions of events per second.

VI. CONCLUSION

Hadoop is an ideal platform used to manage and process data efficiently. The ability of Hadoop to process data in parallel, batch processing mode makes it ideal and efficient. But when it comes to iterative algorithms it has several drawbacks. After every map and reduce phase, the data needs to be fetched from the disk. But, Spark can persist data in memory.

The strength of Spark lies in its abilities to support streaming of data along with distributed processing. This is a useful combination that delivers near real-time processing of data. MapReduce is handicapped of such an advantage as it was designed to perform batch and distributed processing on large amounts of data. Real-time data can still be processed on MapReduce but its speed is nowhere close to that of Spark.

REFERENCES

[1] Gu, Lei, and Huan Li. "Memory or time: Performance evaluation for iterative operation on hadoop and spark." High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference on. IEEE, (November - 2013, Harvard).

[2] Wikipedia – MapReduce: Overview, Dataflow.

https://en.wikipedia.org/wiki/MapReduce

[3] Grolinger, Katarina, Michael Hayes, Wilson A. Higashino, Alexandra L'Heureux, David S. Allison, and Miriam AM Capretz. "Challenges for mapreduce in big data." In 2014 IEEE World Congress on Services, pp. 182-189. IEEE, (June-27, 2014, Chicago).

[4] Zaharia, Matei, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. "Spark: cluster computing with working sets." HotCloud 10 (2010, June-22): 10-10.

[5] Arena solutions - Creating a Bill of Materials, Filed under : BOM Management.

http://www.arenasolutions.com/resources/articles/ creating-bill-of-materials/

[6] Oracle Application : Bill of Material Types

https://docs.oracle.com/cd/A60725_05/html/comnls/us/bom/bomov01.htm

[7] Kreps, Jay, Neha Narkhede, and Jun Rao. "Kafka: A distributed messaging system for log processing." In Proceedings of the NetDB, pp. 1-7, (June 2011).

[8] Thein, Khin Me Me. "Apache Kafka: Next Generation Distributed Messaging System." International Journal of Scientific Engineering and Technology Research .Vol-3, Issue-47 (December - 2014): 9478-9483.

[9] Zaharia M, Chowdhury M, Das T, Dave A, Ma J, McCauley M, Franklin MJ, Shenker S, Stoica I. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation (2012, April-25), pp:2-2. USENIX Association.

[10] Hortonworks – Apache Sqoop : Sqoop Overview, What Sqoop Does, How Sqoop Works.

http://hortonworks.com/apache/sqoop/

[11] Databricks - Recent performance improvements in Apache Spark: SQL, Python, DataFrames, and more.

https://databricks.com/blog/2015/04/24/recent-performance-improvements-in-apache-spark-sql-python-dataframes-and-more.html

[12] Edureka - Apache Spark vs Hadoop MapReduce

http://www.edureka.co/blog/apache-spark-vs-hadoop-mapreduce

AUTHORS

First Author - Ashwitha Jain, Student – M.Tech (Department of Computer Science & Engineering), Mangalore Institute of Technology & Engineering, ashu71192@gmail.com

Second Author – Dr. Venkatramana Bhat P, Professor (Department of Computer Science & Engineering), Mangalore Institute of Technology & Engineering, venkatramana@mite.ac.in