ISSN: 0976-3031

Available Online at http://www.recentscientific.com

CODEN: IJRSFP (USA)

International Journal of Recent Scientific Research
Vol. 8, Issue, 8, pp. 19477-19481, August, 2017

International Journal of Recent Scientific Research

DOI: 10.24327/IJRSR

Research Article

ANALYSIS AND PROCESSING OF STREAMING DATA USING SPARK

Amratansh Sharma and Poovammal E

Department of Computer Science and Engineering, SRM University, Chennai, India

DOI: http://dx.doi.org/10.24327/ijrsr.2017.0808.0701

ARTICLE INFO

Article History:

Received 06th April, 2017
Received in revised form 14th June, 2017
Accepted 23rd July, 2017
Published online 28th August, 2017

Key Words:

Spark, Streaming Data, Big Data Processing Engines, Pipelines, real time processing, Big Data Analytics.

ABSTRACT

The data flow is consistently increasing with the volume of data. The batch processing which requires different programs and computations for input, processing and output will definitely lead to lower efficiency of the big data management systems in terms of cost and speed. Hence it is important to handle the streaming data which is the real time processing. The rate at which it ingest in, the variety of with which occurs gives rise to the need of having big data processing engines and software which brings computation to the data and analyze the data in an efficient manner. One such tool which is described in details is spark. Besides the elucidation of various concepts and tools for the data processing, some analysis is done on the streaming data set which is the weather data set containing different attributes and values at different time stamps. The analysis is done taking into account different window size, the time interval which it takes to extract the data and different sliding size.

INTRODUCTION

The infinite flow of data which comes from the data source is known as streaming data [1]. Some of the real life examples which we see are sensor data from instruments, stock price data, etc. The rate at which data arrives vary, sometimes it can be too fast while on the other side it can be too slow as well. It is often processed in memory. Moreover because of the high velocity of data, it needs to be processed immediately. The streaming data can be used for event detection and prediction.

Big Data Management Systems are designed for parallel and distributed processing. It may not guarantee the consistency for every update, however it more likely assure the eventual consistency. Traditional database lacks storage space for very large data and cannot cope up with the speed of data coming in. Therefore buffers or queues are used to process data in chunks. As the demand increases, the processing nodes also go high. There are several processing nodes, as if one of the node goes down, the system need not restart the processing which, it will relieve data loss. Batch processing [2] is slow and costly. As the scalability increases the complexity decreases. Spark helps in data parallel scalability as shown in Figure 1. Some of the requirements of Big Data Systems are: they support big data operations, handle fault tolerance, enable adding more racks, optimized and extensible for many data types, enable both streaming and batch processing.

Pipelining in simple words can be understood as one after the other execution. Data flows along the pipeline having going through various transformations. For instance, in the case of Map-Reduce [3] we have the data transformations as shown in Figure 2 in which the data first gets partitioned, some user defined functions are applied and at the end results are combined, merged or reduced. Big Data pipelines elucidate the scenario when the output of one running program is given as the input to another program.

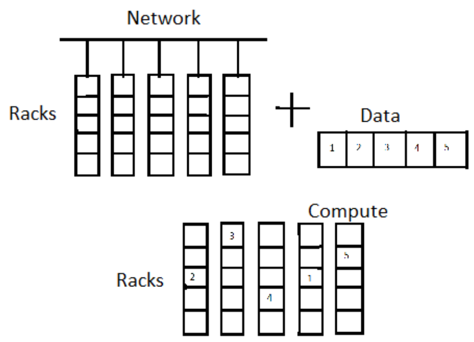

Figure 1 Data Parallel Scalability

Data goes through various transformations which are indeed the tools for giving a shape to data. Data parallelism can be defined as running the same function simultaneously on

Corresponding author: **Amratansh Sharma**
Department of Computer Science and Engineering, SRM University, Chennai, India

different pipelines/ partition of data sets on multiple cores. After the big data integration platform, the streaming data processing platform comes which give either the real time view or batch view. Some of the operations on big data pipelines are manifested in the following sections.

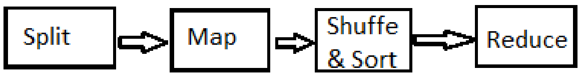

Figure 2 Map Reduce Transformations

Map

Map is the basic data block of big data pipeline which applies same operations to the number of collections. For instance, colour each member of a set, discount each product price by 10%.

Reduce

Reduce operation collectively apply the same process to objects of similar nature. For instance collection of object which have same keys as shown in Figure 3.

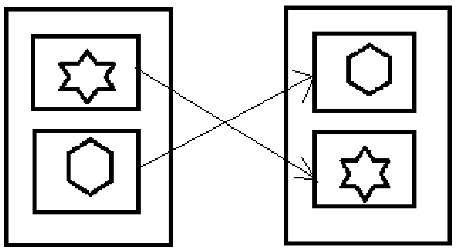

Figure 3 Reduce Illustration

Cross/Cartesian

Cross operation takes place when two data sets are identified by the same key. Some process has to be done on each pair of the two sets, thereby each data set gets paired with all other data partition regardless of the keys.

Match/Join

Match is also known as selective multiplication. Some process has to be done to each pair of the data sets which have the same keys. In the end only the keys of both data gets joined and become the part of final output of data transformation.

Co- Group

The Co-Group operation groups the common items initially. It collects the similar things first. Further some process is to be applied to each collection.

Filter

Filter selects the elements that match a certain criteria. It lists the data sets with all keys even if they don't exist in the database. Analytical operations are basically performed for extracting the meaningful patterns and trends. The flow of analytics is shown in Figure 4.

Patterns
Insights
Decision

Figure 4 Analytics Flow

Analytical operations are also performed in order to gain insights into the problem and to make the data driven decisions. Some of the analytical operations are: Classification, Clustering, Path Analysis and Connectivity Analysis. Path Analysis can be finding the shortest path from home to school for an instance or to find the roots from one location to another. Connectivity Analysis can be analyzing tweets, extracting conversation trends, interacting groups, etc.

Depending on the resourced and applications, there are several considerations which need to be evaluated and plays a crux role for choosing the software for big data. Some of the important factors are: Execution Model, latency, scalability, programming language and fault tolerance. Few of the big data processing engines which are supported by apache are described in the following sections.

Map Reduce

The Map Reduce implementation of hadoop supports the batch processing execution model which lets the data gets loaded before it gets processed. It does have a high latency and low scalability resulting from the lack of in memory processing. Programming languages which can be used are java, python. The fault tolerance is enabled because of the data replication which also affects the execution speed and scalability.

Spark

Spark [4] is built to support iterative and interactive processing using in memory structure called resilient distributed database (RDD). It also provides support to join and filter operations other than map and reduce. The RDD extraction is done to handle fault tolerance. Moreover, spark can read data from many other storage platforms and supports the processing of streaming data as well. The technique is called micro-batching. The latency depends upon the batch size which can be low for small micro batch size. Programming languages which can be used are scala, python, java and R.

Flink

Flink provides direct support for the streaming data, making it lower latency framework. It also provides connection interface with data engines such as kafka and flume. The programming languages which it supports are java and scala. It has its own execution engine called Nephele. In addition to map reduce, it also provides the other data operations like join and group by.

Beam

Batch is a new big data engine which supports batch processing as well as stream processing. It is developed by google and was initially used for the cloud data flow. Moreover flink is a low latency environment with high fault tolerance. The programming languages which it supports are java, scala and python software development kits. It provides a very strong streaming framework.

Storm

Storm [5] is been designed for the stream processing and do possess a very low latency. The input stream abstraction is defined as spouts and the computational abstractions as bots. They can be pipelined together by a data flow approach. There is a master node which keeps track of the running jobs and ensures that all data is processed. The lambda architecture was developed using storm for stream processing and hadoop map reduce for the batch processing.

There were certain shortcomings and drawbacks of hadoop [5] map reduce such as it can only be used for map reduce based computation, relies on reading data from hadoop based file system (HDFS), native support of java only , no interactive shell support and no support for streaming which gave rise for an expressive programming model called Spark. It provides in memory processing which makes it efficient for iterative applications. Moreover it supports diverse workloads such as batch and streaming at once. It also provides API's for python, java, scala and SQL programming through an interactive shell. The spark layer diagram called as the spark stack [6] as shown in Figure 5,

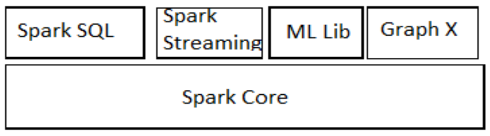

Figure 5 Spark Stack

consists of components that are built on top of spark computational engines. Through all these layers spark provides diverse, scalable interactive management and analyses of big data. The engine distributes and monitors tasks across the nodes of commodity cluster. The interactive shell allows the scientists to explore, build and scale the data. Spark does in memory processing using RDD abstraction. In hadoop map reduce as shown in Figure 6, the pipeline reads from the disk to memory, performs the computation and then write back the output to the disk. Writing data to disk is costly operation and it increases as the volume of data increases.

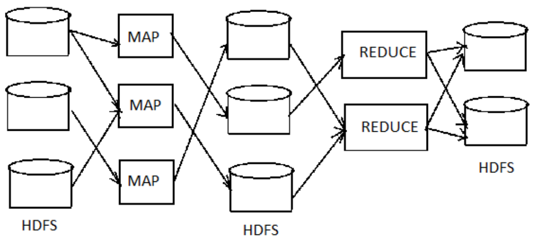

Figure 6 Map Reduce

On the other hand memory operations can be of very high speed. Spark thus does the in memory processing and allows for immediate results of transformation in different stages of pipeline in memory as shown in Figure 7. Here the outputs of MAP operations are shared to the reduce operations without been written in the disk. The storage where the data gets stored in memory are called Resilient Distributed Datasets (RDD). Usually data sets come from batch data storage like HDFS, when spark reads the data it generates RDD which are immutable and cannot be changed. By distributed data we mean the partition data structure computations which are divided in partitions and atomic chunks of data. Resilient manifests that the recovery from the error is possible, for instance if any node goes down or gets failed, it can get recovered. In case if any portion of data is missed, track history of each portion is available.

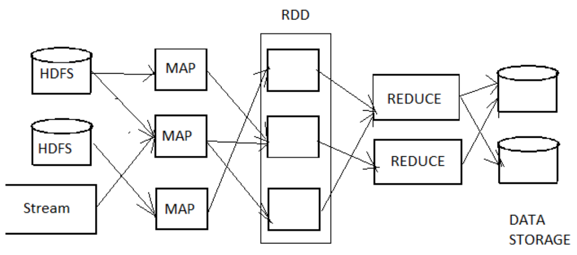

Figure 7 Spark

The Spark mainly consists of two major components which are the driver program and the worker node as shown in Figure 8. The application starts in the driver program which distributes the RDD in the computation clusters and performs the transformations on them [7].

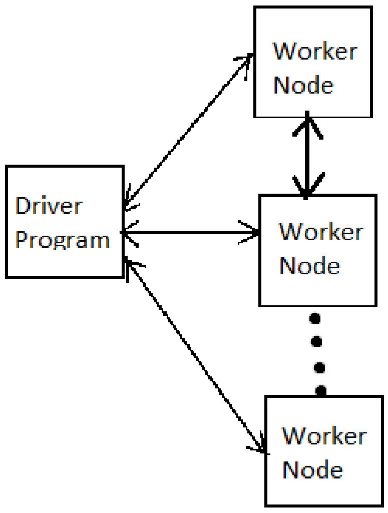

Figure 8 Spark Architecture

The driver program creates a connection with the spark with the spark clusters through the spark context object. It also maintains the coordination with the myriad nodes called the worker nodes. The execution takes place in the worker node. The Java Virtual Machine (JVM) which the spark keeps running is called the executor. For example if we have HDFS as the storage system, then on each worker node some of the data will be available. The main job of this system is to being computation to the data. Hence the spark will send computational jobs to be executed that are available in machine. Data will be read from HDFS and the in memory processing will be done, further data will get stored as one RDD. In the real time scenario, there are many worker nodes which execute the specific tasks.

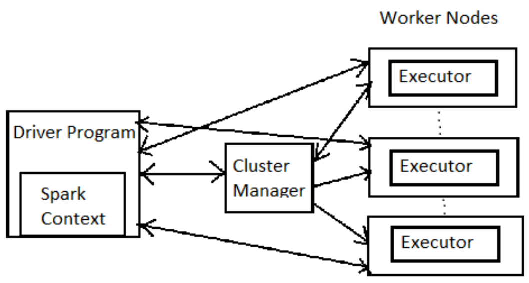

Figure 9 The Cluster manager coordination

So there should be a system which takes the responsibility of coordination and managing the nodes. This system is called cluster manager as shown in Figure 9. The driver program coordinates with the cluster manager for monitoring the resources and it also communicates with the worker nodes for executing the tasks.

Spark uses a lazy evaluation for transformation, which means that it will wait for an action to be performed and will not be executed immediately. Spark offers scalable processing for real time data and run on top of the spark core. The continuous data streams are converted into discrete resilient distributed databases (RDD) [8]. Further the RDD's are processed in parallel. Spar can read the data from different data sources such as kafka, flume, hdfs. Also it can read the data from real time data sources such as twitter, machineries, etc. The streaming data which is read by spark is converted into micro batches which are called Dstreams or discretized streams. For instance as shown in Figure 10, the 6 second stream is converted into three RDDs of batch length of 2 second each. All the operations which are applicable for RDDs such as map, filter, reduce also works for the Dstreams. Further the window size consists of 2 seconds. The Sliding interval depends on the amount of batch size it moves forward.

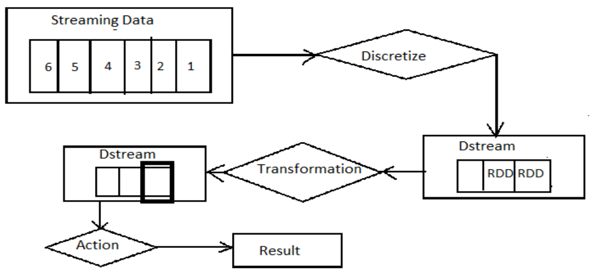

Figure 10 Spark Streaming

EXPERIMENT AND RESULTS

The experiment is performed in the Jupyter Python Notebook in the cloudera [9] virtual machine. The data set is of weather station [10] in which the streaming data is analysed. Every specific line of data contains the time stamp and abbreviation of the attribute, such as Dn for wind direction minimum speed, Sm for average wind speed. The function is called which parses every line and returns the particular desired attribute. The spark streaming context and spark context are to be imported and a batch interval of 1 second is given. A connection is needed to be made with the streaming data and henceforth the Dstreams are created. Further, we create a sliding window which contains two arguments which are indeed the time intervals, one is the window size which collects the data according to specified time period and the other attribute is of window sliding size which specifies by what time interval the window will move forward. We print the maximum and minimum values of the streaming data in different windows. To have a better and pervasive analysis we consider two attributes Sm and Dn. Moreover we change the window arguments, which are the window size and window sliding size to get different results for the analysis. In Figure 11 and Figure 14 the slide size of window remains the same. In Figure 11 the analysis is done for wind degree minimum whose unit are in degrees and in Figure 14 the analysis is done for the Average wind speed. In Figure 12 and Figure 15 the slide size of window changes while the window size remains the same. In Figure 12 the analysis is done for wind degree minimum and in Figure 15 the analysis is done for the Average wind speed. In Figure 13 and Figure 16 the slide size of window and the window size both remains same, the slide size is 5 and the window size is 10. In Figure 13 the analysis is done for wind degree minimum for the two periods which are in the gap of 6 hrs and in Figure 16 the analysis is done for the Average wind speed for two periods which are in the gap of 6 hours as well.

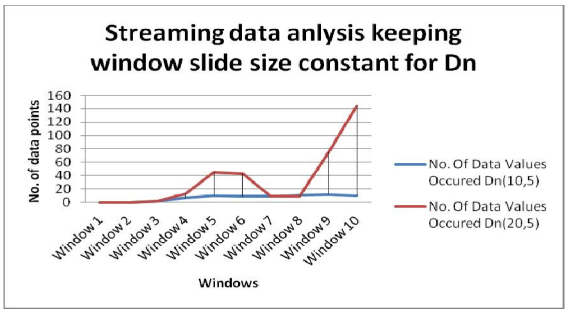

Figure 11 Window sliding size constant for Dn

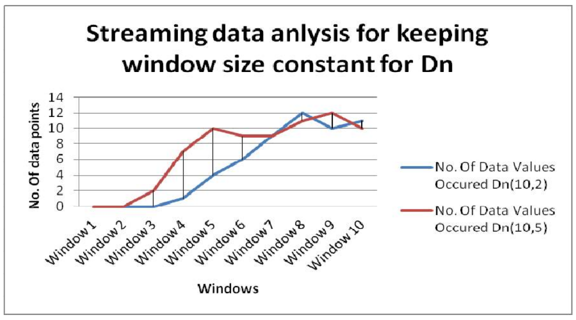

Figure 12 Window size constant for Dn

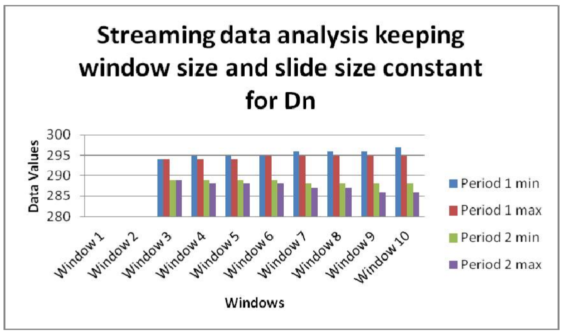

Figure 13 Window sliding size and window size constant for Dn

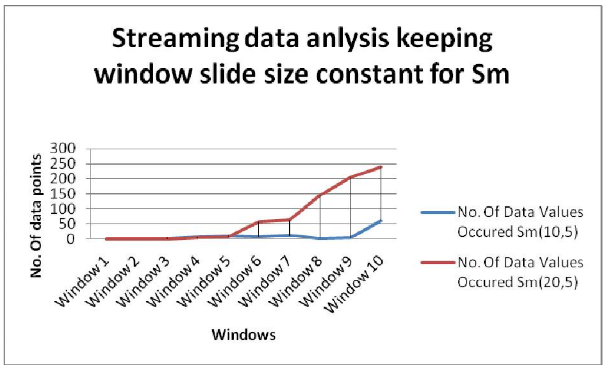

Figure 14 Window sliding size constant for Sm

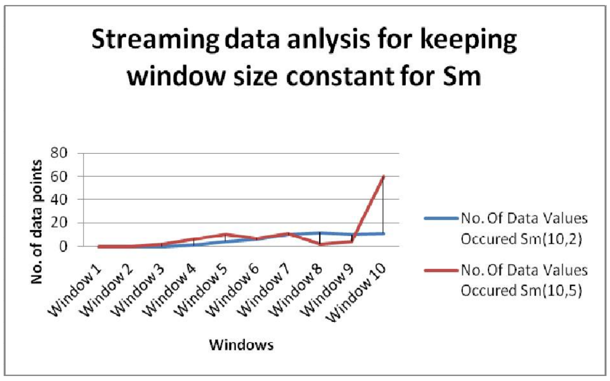

Figure 15 Window size constant for Sm

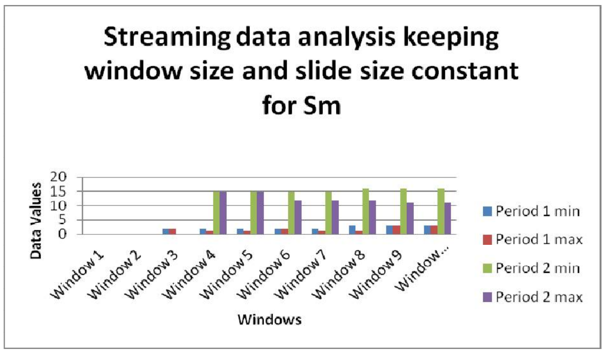

Figure 16 Window sliding size and window size constant for Sm

CONCLUSION

In the paper we have discussed various big data processing machines and had some deep insights of the spark processing. The data streaming is analysed in spark considering two different attributes for the weather station data sets, one is wind direction degree (Dm) and other is average wind speed (Sm). We have varied the arguments of window which gave different observations and results regarding the data values in the window, occurrence of values in the window and the minimum and maximum values of data. When the window sliding size is increased from 2 to 5 keeping the window size constant, we can observe the same pattern for both the sliding size, although there are certain rise and fall but it happens for both the sliding size. For both Sm and Dn it starts increasing gradually, starting from zero. When the window sliding size is constant and the window size is increased from 10 to 20, we can see a steep increase in the number of data points eventually after few windows. Once it gets start increasing when the window size is 20, it goes on increasing and never seems to come down. When the window sliding size and the window size remains constant and the minimum-maximum values of data inside the window are calculated for two different periods, we can observe that for the Dn case, the minimum and maximum values are larger for period 1 as compared to period 2 which shows the difference in the wind direction for 2 periods which are having 6 hours gap in afternoon. Moreover when the window sliding size and the window size remains constant and the minimum maximum values of data inside the window are calculated for Sm case, the minimum and maximum values are smaller for period 1 as compared to period 2 which shows that the wind speed rises gradually as the day time increases. For both the Dn and Sm cases we can observe that for individual periods the difference between the minimum and maximum values are almost the same and do not vary much, though they at different periods difference in the wind direction for 2 periods which are having 6 hours gap. It can also be observed that it is not necessary that if the sliding size of window is 5 then, the last five values of data for previous frame would be the first five values of consecutive frame. It works for Dn having window size 10 and sliding size 5 but not exactly for the other cases. Finally, it is observed in all cases that after the start execution is given, the empty windows arrive initially, the data starts coming at least after 2 or three windows, sometimes it may be more delayed as well. It can sometime occur because the server is down or we need to change the port which will enable the data ingestion. The patterns can definitely help us to predict the future outcomes of different attributes of weather station. More patterns and analysis could be extracted from the streaming data which will definitely lead to better decision making and a betterment of analytics.

References

1. Shazia Nousheen M, Dr. Prasad G R, A Survey Paper on Data Stream Mining, IJERT, August 2016, Vol. 05, Issue 08.
2. Swati M. Gavali, Supriya Sarkar, Survey Paper on Big Data Processing and Hadoop Components, IJSR, July 2016, Volume 5, Issue 7.
3. Kyong-Ha Lee Yoon-Joon Lee, Hyunsik Choi Yon Dohn Chung, Bongki Moon, Parallel Data Processing with MapReduce: A Survey, SIGMOD Record, December 2011, Vol. 40, No. 4.
4. Spark Streaming. https://spark.apache.org/streaming/. Downloaded spark in September 2017.
5. Telmo da Silva Morais, Survey on Frameworks for Distributed Computing: Hadoop, Spark and Storm, DSIE, 1 st Edition, 2015 - ISBN: 978-972-752-173-9.
6. Coursera, UC San Diego, Big Data Integration and Processing, https://www.coursera.org/learn/big-data-integrationprocessing/home/week/4
7. Getting Started with Apache Spark, https://mapr.com/ebooks/spark/03apache-spark-architecture-overview.html
8. Zeba Khanam and Shafali Agarwal, Map-Reduce Implementations: Survey and Performance Comparison, IJCSIT, August 2015, Vol. 7, No. 4.
9. Cloudera. http://www.cloudera.com/. Downloaded Cloudera Quickstart virtual machines in September 2017.
10. https://github.com/words-sdsc/coursera/blob/master/big-data-/, downloaded in September 2017.

****