Scientific Journal of Impact Factor(SJIF): 3.134
e-ISSN(O): 2348-4470
p-ISSN(P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 2,Issue 5, May -2015

Fast Analysis of Sensor Data over MapReduce using Spark

Mansi Shah[1], Vatika Tayal[2]

[1]M. Tech. Scholar, Computer Science and Engineering Department, N.S.I.T, Jetalpur, Gujarat, mnshah201@gmail.com
[2]Assistant Professor, Computer Science and Engineering Department, N.S.I.T, Jetalpur, Gujarat, vatika.sharma15@gmail.com

Abstract — *Big data analysis is emerging rapidly due to the tremendous volume of data, velocity at which the data is flowing in the organizations and the variety of data. In recent years due to the spurt in Internet of Things (IoT), data generated by the sensors is growing exponentially thus transforming into big data. Thus data collection, processing and extracting useful information from such increasing high velocity and high volume of sensor data poses a challenge for the researchers. Apache Spark is an open source, a general purpose engine for rapid large-scale data processing. To overcome the data replication and disk I/O overhead of sharing data between parallel operations in Hadoop, Spark uses the primitive called Resilient Distributed Datasets (RDD's) which provides the programmers a fault tolerant and in-memory data storage across cluster nodes without replication that increases the processing speed of the applications to several magnitudes. We propose a method to analyze the sensor data using the Spark.*

Keywords- *big data, Resilient Distributed Datasets, Spark, MapReduce, Hadoop*

I. INTRODUCTION

In this electronic era due to the spurt in the volume of digital and social media as well as Internet-of-Things (IoT) the amount of data generated by sensor networks has increased enormously. Extracting knowledge from sensor data can be used for proactive maintenance, improving reliability, reducing heating and cooling expenses, product monitoring, reduce unplanned service work and so on. Big data is the collection of data sets so large and complex that it becomes difficult to process those using conventional tools or applications. The three attributes that characterize big data are velocity, volume, and variety. The prime challenge with Big Data includes extract, search, storage, retrieval, analysis and visualization of data. Big Data analysis focusses to process data that is massive, unstructured, speedy and extracting useful and hidden information from it[1]

Google's MapReduce became the most popular framework to process rapidly enormous amounts of data in parallel on large clusters of machines. However, Hadoop/MapReduce is designed for batch processing of huge volumes of data but is not optimized for iterative computations, interactive mining and stream processing due to redundant and wasteful processing[2]

In this paper, we propose a method to analyze sensor data over distributed data processing engine Spark that is faster than Hadoop MapReduce.

II. EXISTING METHOD AND LIMITATIONS

Hadoop is an open source framework for writing and running distributed applications which are capable of batch processing large sets of data. Hadoop framework is mainly known for MapReduce and its distributed file system (HDFS)[12] MapReduce is a programming model that enables large-scale and distributed processing of big data on a set of commodity machines. MapReduce defines the computation as two functions: map and reduce. The input is a set of key/value pairs, and the output is a list of key/value pairs. The reduce function takes an intermediate key and a list of intermediate values associated with that key as its input, and results set of final key/value pairs as the output.

However, Hadoop/MapReduce is designed for batch processing of huge volumes of data but is not optimized for iterative computations, interactive mining and stream processing due to redundant and wasteful processing [2].The main problem is that both iterative and interactive applications need data sharing across multiple MapReduce steps. In MapReduce the data sharing between parallel operations is done by writing the intermediate results to distributed file system HDFS which adds overhead due to disk I/O, redundant and wasteful processing and data replication [3]. It also suffers from various Configuration and Automation issues which require numerous configuration parameters to set when deploying a Hadoop MapReduce cluster. These Programming Model issues makes it unsuitable for machine learning algorithms and graph processing which often require iterations or incremental computations [4].

III. SYSTEM OVERVIEW

Apache Spark is an open source in-memory cluster computing framework for fast and flexible large-scale data analysis initially developed in the AMP Lab at UC Berkeley. Spark is built on the top of Hadoop Distributed File System, but instead of using Hadoop MapReduce, it depends on its own parallel data processing framework which starts by placing data in Resilient Distributed Datasets (RDDs), a distributed memory abstraction that executes calculations on huge clusters in a fault tolerant way. Spark also adds flexibility to its speed by providing APIs that permits developers to write queries in Java, Python or Scala[5]

The Spark working architecture is depicted in Figure 1. The Spark cluster consists of a driver program where the execution of application logic is started with several workers that process data in parallel. Even though this is not mandated, data are typically collocated with the worker and partitioned across the same set of commodity machines within the cluster. The driver programs will pass the code to the worker machine during execution where the processing of the corresponding partition of data will be conducted. To elude data shuffling across machines, the data will go through different steps of transformation while remaining in the same partition as much as possible. Actions will be executed at the workers and the final result is then returned back to the driver program[6]

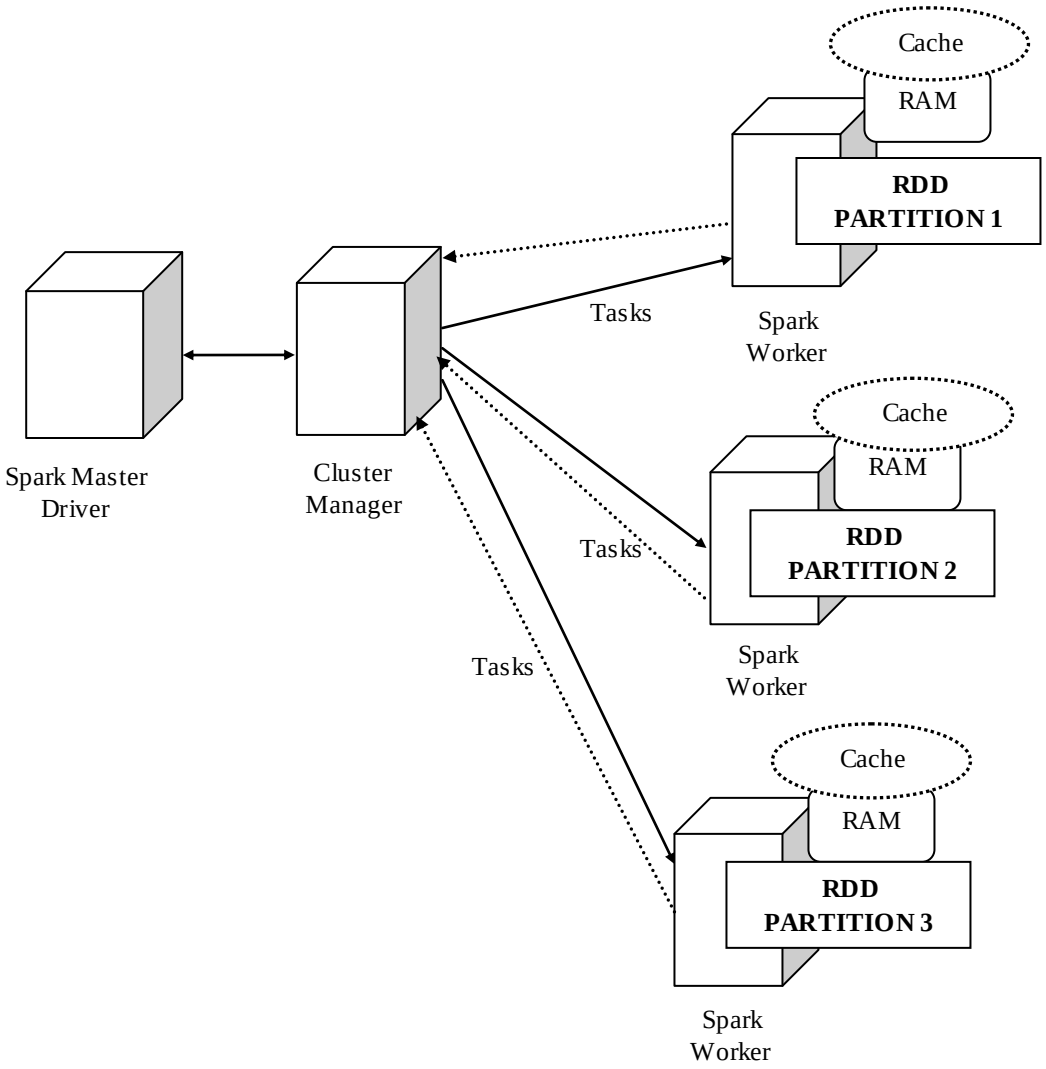

Figure 1. Spark Working Architecture

3.1. Spark Programming Interface

The key concept in Spark is Resilient Distributed Datasets (RDDs), which are immutable, fault-tolerant collections of objects distributed across cluster nodes which can be operated on in parallel. RDDs are created by users by implementing operations known as transformations, such as map, filter and group in a durable storage system. Controlling how various RDD's are co-partitioned (with the same keys) across commodity machines can decrease inter-machine shuffling of data within a cluster [7] [8].

Spark provides a partition operator which produces a new RDD by redistributing the data in the original RDD across machines within the cluster. RDD can optionally be cached in RAM and therefore providing faster access. Presently, the granularity of caching is implemented at the RDD level, either the complete or none of the RDD is cached. Spark will try to cache the RDD if and only if sufficient memory is available in the cluster, depending on the LRU (Least Recently Used) eviction algorithm.RDD presents an abstract data structure from which the application logic can be depicted as a series of transformation processing, without being concerned about the intrinsically distributed nature of the data. The application logic is typically depicted in terms of a series of TRANSFORMATION and ACTION [6].

Transformation takes an input RDD and creates a new RDD. When transformation function is called, nothing gets calculated. **Action** computes and returns a new value to the program or writes the output to the external stable storage system.When the action operation is executed by the user; the scheduler will build a DAG of stages to execute by examining the RDD's lineage graph [6].

IV. SENSOR DATA ANALYSIS FLOW

The flowchart of sensor data analysis is shown in Figure 2. First sensor data is loaded from a file into the Spark framework, mapped to RDD's. The data is filtered and temperature parameter is extracted from the available data using RDD transformation. We then apply RDD Map transformation to generate key-value pairs based on if-else rules. Finally RDD reduce transformation is applied on the key-value pairs to produce an aggregated final result which is saved to the local file system using RDD action. The result gives the count of the readings with and without Normal temperature for each area in the lab as well as sensor readings not satisfying the rules. We also calculate the accuracy of the rules.

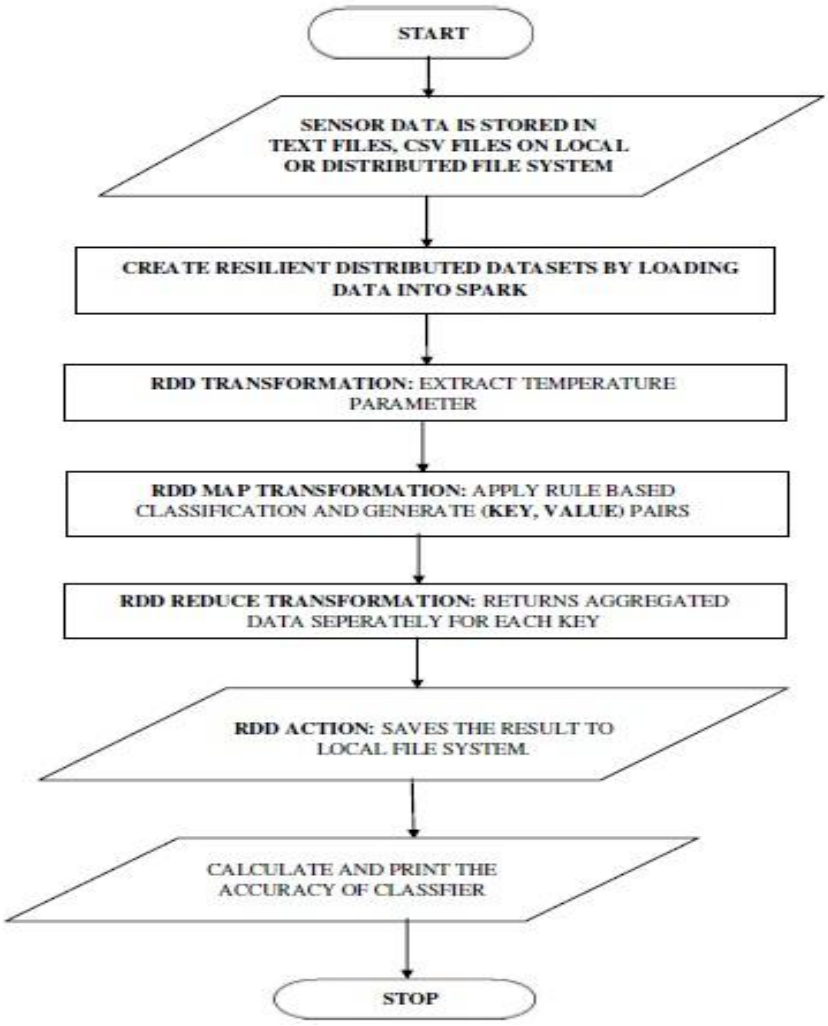

Figure 2. Data Analysis flow

V. EXPERIMENTS AND RESULTS

The sensor dataset used in this experiment is generated by 54 sensors deployed in the Intel Berkeley Research Lab. The frameworks used for analysis are Spark (0.9.0-incubating) and Hadoop (0.20.2). Below is the table of the different number of the data and the file size of the data that we analyzed using the Spark. The results we obtained are mentioned in the Table 1. And the performance comparison of Spark and Hadoop is shown in Figure 3. The performance of the developed system depends strongly on the machine it is installed on. CPU, Memory Usage and Disk I/O.

Table 1. Experimental results of Hadoop and Spark

Data Size	File Size	Execution Time	
		Hadoop	Spark
10,000	634KB	48 Sec.	20 Sec.
100,000	6.25MB	55 Sec.	23 Sec.
1,000,000	64.3MB	106 Sec.	39 Sec.

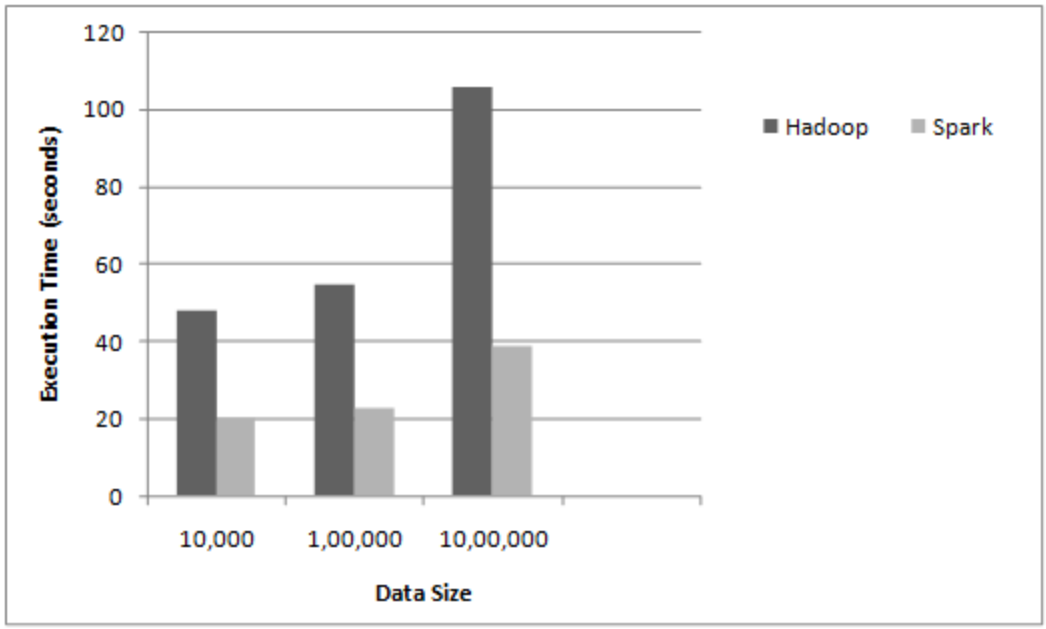

Figure 3. Performance comparision of Hadoop and Spark

VI. CONCLUSION

Big Data is an emerging field because today the data is generated in huge volume, at a fast velocity with variant types. Owing to machine-to-machine communication and Internet of Things based wireless sensor networks the amount of data generated by sensors has increased enormously. Initially, the knowledge extraction from the sensor data was easy as the number of wireless sensor networks were limited. But as the sensor networks and applications grow capturing, managing and analyzing the increasingly high volumes and high-velocity sensor data become a challenge by the existing systems. So there is a need to introduce new approaches which will process and analyse such large volume of sensor data at a faster speed. In this paper we have proposed a method of implementing rule based classification on distributed systems using Spark. Also, by using Spark framework the system memory is fully utilized, which leads to better performance than Hadoop.

ACKNOWLEDGEMENT

We would like to thank the Intel Berkeley Research Lab [9] for providing us the sensor dataset.

REFERENCES

[1] Rabi Prasad Padhy, "Big Data Processing with Hadoop-MapReduce in Cloud Systems", International Journal of Cloud Computing and Services Science (IJCLOSER), February 2013, vol.2, No.1, 16-27.

[2] Saeed Shahrivari, "Beyond Batch Processing: Towards Real-Time and Streaming Big Data", Computer 2014.

[3] Matei Zaharia; Mosharaf Chowdhury; Tathagata Das; Ankur Dave; Justinma; Murphy Mccauley; Michael J; Scott Shenker; Ion Stoica , "Fast and Interactive Analytics over Hadoop Data with Spark" , Usenix, pp. 45-51 Aug 2012.

[4] Vasiliki Kalavri, Vladimir Vlassov," MapReduce: Limitations, Optimizations and Open Issues", 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communication, 2013, pp. 1031-1038, doi: 10.1109/TrustCom.2013.126.

[5] Spark as a Service. [Online]. Available: http://www.qubole.com/resources/articles/hadoop-spark/

[6] Spark: Low latency, massively parallel processing framework. [Online]. Available: http://horicky.blogspot.in/2013/12/spark-low-latency-massively-parallel.html

[7] Jie Deng, Zhiguo Qu, Yongxu Zhu, Muntean, Gabriel-Miro, Xiaojun Wang, "Towards Efficient And Scalable Data Mining Using Spark", Information and Communications Technologies (ICT 2014), 2014 International Conference, pp. 1-6.

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

[9] Intel Lab Data. [Online]. Available: http://db.csail.mit.edu/labdata/labdata.html

