International Conference on Information Engineering, Management and Security
2015 [ICIEMS 2015]

ISBN	978-81-929742-7-9	**VOL**	01
Website	www.iciems.in	**eMail**	iciems@asdf.res.in
Received	10 - July - 2015	**Accepted**	31- July - 2015
Article ID	ICIEMS021	**eAID**	ICIEMS.2015.021

BIGDATA ANALYTICS WITH SPARK

SUBHASH KUMAR
IT Department, St Xavier's College, Mahapalika Marg, New Marine Lines,
Mumbai, PIN-400001, Maharashtra, India

Abstract: *Current generation is witnessing data explosion most of it is unstructured and is called Big Data. This data has characteristics of high volume, velocity, variety and veracity. HDFS, GFS, Ceph, Lustre, PVFS etc are used as file system for storing Big Data. MapReduce processes program in parallel across clusters and generates output. Spark framework improves performance by 10x when datasets are stored in hard disk and performance improves by 100x when data is stored in memory. This paper proposes optimization of Big Data processing using Spark framework.*

Keywords: *Volume, velocity, veracity, Big Data, HDFS, Spark.*

I. INTRODUCTION

Huge amount of data is being generated every second. It puts a new challenge for managing this data. Social media like Facebook generates about 600 TB of data every day, whereas Twitter generates about 120 TB each day and Google generates about 20 PB of data each day. It is evident that data is collected in an exponential rate and we have already reached Terabyte and PetaByte stage (see Figure 1). 1PB=1024TB. 1EB=1024PB.

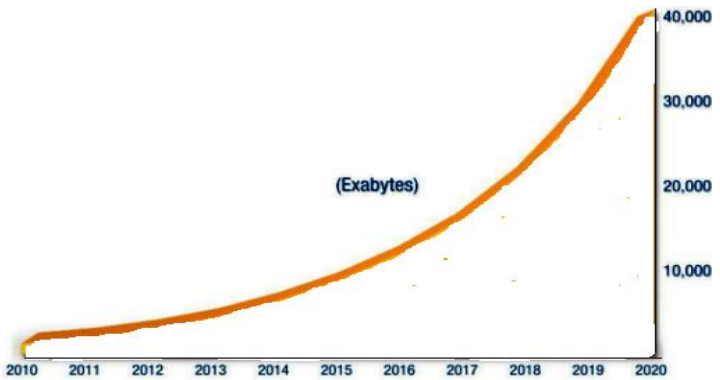

Figure 1: Digital Universe expansion

As per IDC [6], digital universe in 2010 was 1227 ExaBytes and by end of 2020 data would reach to 40ZB.

This paper is prepared exclusively for **International Conference on Information Engineering, Management and Security** 2015 [ICIEMS] which is published by ASDF International, Registered in London, United Kingdom. Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honoured. For all other uses, contact the owner/author(s). Copyright Holder can be reached at copy@asdf.international for distribution.
2015 © Reserved by ASDF.international

Cite this article as: SUBHASH KUMAR. "BIGDATA ANALYTICS WITH SPARK." *International Conference on Information Engineering, Management and Security (2015)*: 129-133. Print.

This Big Data needs to be stored in multiple machines in commodity hardware as clusters since single machine cannot store it. This is managed with Hadoop Framework. HDFS (Hadoop Distributed File System), GFS (Google File System), PVFS, Ceph etc is used to store Big Data. Default data size in HDFS is 64MB. These chunks (64MB) are stored in commodity hardware. In hadoop MapReduce, the tuples generated in Map and Reduce task are stored in disk (see Figure 2). This takes time. This is improved by using apache spark framework.

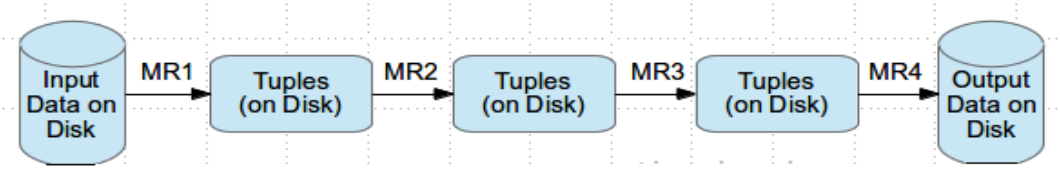

Figure 2: MapReduce iteration in Hadoop

Apache Spark is an open-source cluster computing framework developed in 2009 at AMPLab at University of California. Spark can read from any data source (relational, NoSQL, file systems, etc) and offers unified API for batch analytics, SQL queries, machine learning and graph processing, real-time analysis.

Spark is not only designed to run many more workloads, but it can do so much faster than older systems. Spark is 10 times faster than Hadoop MapReduce when data is read from disk and 100 times faster when data is read from memory. Spark is highly scalable. The largest Spark cluster has about 8,000 nodes.

Spark improves efficiency through in memory computation, general computation graph. It has rich API in java, python, scala and interactive shell where programmers write less line of codes.

II. LITERATURE REVIEW

Yanfeng Zhang et al. [4] paper discusses PrIter, which is the prioritized execution of iterative computations. PrIter stores intermediate data in memory for fast convergence or stores intermediate data in files for scaling to larger data sets. PrIter was evaluated on a local cluster of machines as well as on Amazon EC2 Cloud. The results show that PrIter achieves up to 50 × speedup over Hadoop for iterative algorithms type problems. In addition, PrIter is shown better performance for iterative computations than other distributed frameworks such as Spark and Piccolo

Yanfeng Zhang et al. [2] paper discusses iMapReduce is a distributive framework that significantly improves the performance of iterative computations by (1) reducing the creation new MapReduce jobs again and again, (2) shuffling of static data gets eliminated, and (3) asynchronous execution of map tasks is allowed, iMapReduce prototype shows that it can achieve up to 5 times speedup in implementing iterative algorithms.

Xu, X et al. [3] pointed that if TaskTracker could adjust to change of load as per its computing ability, results can be obtained faster.

Weizhong Zhao et al. [4] said that Parallel K-Means clustering based on MapReduce can process datasets efficiently using commodity hardware.

Matei Zaharia et al. [5] pointed out that the resilient distributed dataset (RDD), which represents a read-only collection of objects can be partitioned across multiple set of machines in cluster and can be rebuilt even if a partition is lost. If a partition of an RDD is lost, the RDD has enough information and uses other RDDs to rebuild just that partition by using lineage information. Lineage is the sequence of transformations used to build the current RDD. .
Matei Zaharia et al. [6] paper showed following results of spark: spark outperforms Hadoop by up to 20x in iterative machine learning and graph applications. The speedup comes from avoiding I/O and deserialization costs by storing data in memory as Java objects. Applications written in spark perform and scale well. In particular, spark speeds up an analytics report that was running on Hadoop by 40x. When nodes fail, Spark shows recovery strategy by rebuilding only the lost RDD partitions. Spark can to query a 1 TB dataset interactively with latencies of 5–7 seconds.

As per apache spark [7], spark runs much faster than hadoop which is evident from the figure below (see Figure 3) for logic regression.

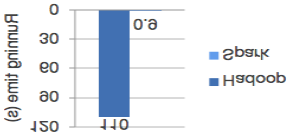

Figure 3: Logic Regression in Hadoop and Spark

Cite this article as: SUBHASH KUMAR. "BIGDATA ANALYTICS WITH SPARK." *International Conference on Information Engineering, Management and Security (2015)*: 129-133. Print.

III. SPARK FRAMEWORK

A. RDD (Resilent Distributed Dataset)

It is an abstraction layer which is read only and represents collection of objects that can be stored in memory or disk in cluster. It can be rebuilt on failure. It has parallel functional transformation. RDDs support operations known as transformations, which create new dataset from an existing one, and actions, which after running a computation on the dataset return a value to the driver program. Some of the actions in spark are filter, count, union, join, sort, groupBy, groupByKey, pipe, cross, mapWith etc.

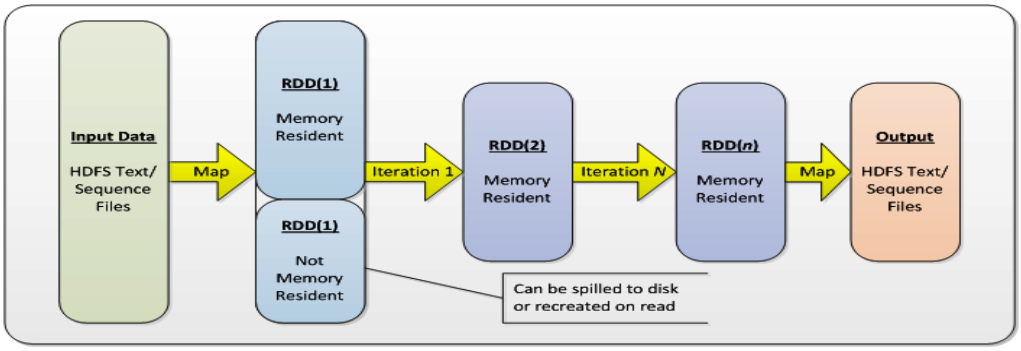

Figure 4: Iteration in RDD using SPARK

From Figure 4, the first map operation into RDD (1) is shown where, not all data could fit in the memory space so some data is passed to the hard disk. Data is first searched in the memory for the reading and also writing occurs in memory. This method makes system to be 100X faster than other methods that rely purely on disk storage.

Spark follows lazy loading that is it doesn't perform transformation on RDD immediately. Instead, it piles this transformation and forms batch which is then processed.

B. SPARK STACK

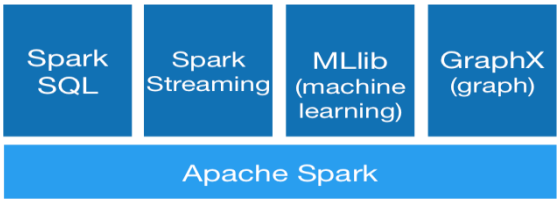

Figure 5: Spark Stack

Spark Stack consists of four major components Spark SQL, Spark Streaming, MLib, GraphX (see Figure 5)

C. SPARK SQL

The two useful components of Spark SQL are DataFrame and SQLContext. DataFrame provides an abstraction which can act as distributed SQL query engine. A DataFrame is a distributed collection of data which is organized into named columns. DataFrames can be converted to RDDs and vice versa. DataFrames can be created from different data sources such as: Hive tables, Existing RDDs, JSON datasets, structured data files, External databases. Spark SQL lets you query structured data as a distributed dataset (RDD) in Spark, with integrated APIs in Python, Scala and Java. This tight integration makes it easy to run SQL queries alongside complex analytic algorithms.

D. SPARK STREAMING

Spark Streaming allows one to process large data streams in real time. This helps to find fraud detection. Spark Streaming allows live streaming as well as post processing in batch. There is no other framework which can do both.

The live stream is divided into small batch of x second which is then passed to spark framework and it treats this batch as RDD and processes it in batch (see Figure 6).

Cite this article as: SUBHASH KUMAR. "BIGDATA ANALYTICS WITH SPARK." *International Conference on Information Engineering, Management and Security (2015)*: 129-133. Print.

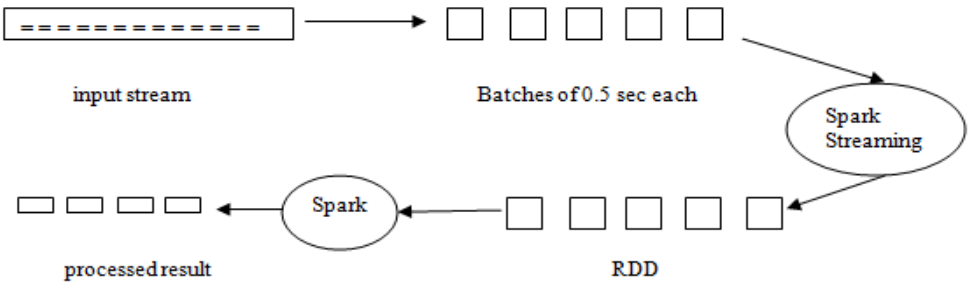

Figure 6: Illustration of Spark Streaming

E. MLLIB

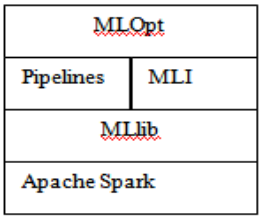

Figure 7: Building blocks of MLlib

MLOpt is declarative layer which automates hyperparameter tuning. Pipelines and MLI are API for simplifying development of machine learning such as distributed table, distributed matrices. MLlib is machine learning core library (see Figure 7). It consists of Gradient descent algorithm for optimization, K-Means algorithm for clustering, Logistic Regression for prediction, feature transformation etc.

F. GRAPHX

GraphX is the new Spark API for graph-parallel computation. GraphX extends the Spark RDD with graph concept where properties are attached to each vertex and edge. GraphX exposes fundamental operators such as subgraph, joinVertices etc. GraphX has collection of graph algorithms which simplifes graph analytics. With one can view the same data as both graphs and collections, join and transform graphs with RDD efficiently.

IV. SPARK RUNTIME

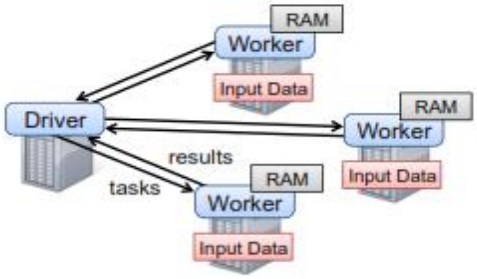

Figure 8: Spark Runtime

See Figure 8, where Driver program launches multiple worker threads that read data blocks from a distributed file system and persists computed RDD partitions in memory. Developers write a driver program which can connect to a cluster of workers, as shown in Figure 2. The driver defines one or more RDDs and invokes actions on them. The workers can store RDD partitions in RAM. A driver performs two types of operations on a dataset: action and transformation. action performs computation on dataset and returns value to the driver; transformation creates new dataset from an existing dataset.

V. PROGRAMMING ILLUSTRATION

Here illustration is shown in scala language which is functional programming language.
This program collects all errors having DB2 written in ERROR Log.

```
//RDD is created using hdfs
val txt = spark.textFile("hdfs://scrapper/user/alltweets.txt"")
//New RDD created by transformation which searches for ERROR
val errors = txt.filter(line => line.contains("ERROR"))
```

Cite this article as: SUBHASH KUMAR. "BIGDATA ANALYTICS WITH SPARK." *International Conference on Information Engineering, Management and Security (2015)*: 129-133. Print.

```
// Count all the errors. Action is performed
errors.count()
// Count errors mentioning DB2
errors.filter(line => line.contains("DB2")).count()
// Fetch the DB2errors as an array of strings
errors.filter(line => line.contains("DB2")).collect()
```

Another example for storing data in RAM using cache() method . It is illustrated below

step1
Go to the bin dir of spark installation
$ /home/spark-1.2.1//bin
step2
$ run the spark-shell which takes to scala prompt
./spark-shell

step3
Create RDD of TEST.txt
scala>val tf=sc.textFile("TEST.txt")

step4
Use transformation on RDD and creat new RDD
scala>val ramtxt=tf.filter(line=>line.contains("Data")

step5
Store this RDD in RAM for faster access
scala>ramtxt.cache()

CONCLUSION

Spark framework supports big data processing. This framework can give faster result than existing hadoop system. It has capability of doing in-memory computation using languages scala, java, python. One can work with cluster writing less lines of code in scala as it is functional programming language. Spark has rich set of libraries for data streaming, machine learning and spark sql. Spark framework improves performance by 10x when datasets are stored in hard disk and performance improves by 100x when data is on memory.

ACKNOWLEDGMENT

I would like to thank Knowledge Centre of St Xavier's College for providing the infrastructure for setting cluster. Also I would like to thank our Principal (Dr.) Agnelo Menezes and Dr Siby Abraham for supporting research activities.

REFERENCES

[1] Vernon Turner, John F. Gantz, David Reinsel and Stephen Minton (2014). White paper The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things. URL:http://idcdocserv.com/1678

[2] Yanfeng Zhang; Qinxin Gao; Lixin Gao; Cuirong Wang, "iMapReduce: A Distributed Computing Framework for Iterative Computation," Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on, vol., no., pp.1112,1121,16-20May2011
doi: 10.1109/IPDPS.2011.260

[3] Xu, X.; Cao, L.; Wang, X. (2014), Adaptive Task Scheduling Strategy Based on Dynamic Workload Adjustment for Heterogeneous Hadoop Clusters Systems Journal, IEEE , vol.PP, no.99, pp.1,12. doi: 10.1109/JSYST.2014.2323112.
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6832443&isnumber=4357939

[4] Yanfeng Zhang, Qixin Gao, Lixin Gao, Cuirong Wang (Sept.2013), PrIter: A Distributed Framework for Prioritizing Iterative Computations, Parallel and Distributed Systems, IEEE Transactions on, vol.24, no.9, pp.1884,1893.

[5] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica University of California, Berkeley: Spark: Cluster Computing with Working Sets.

[6] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, Ion Stoica University of California, Berkeley: Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing.

Cite this article as: SUBHASH KUMAR. "BIGDATA ANALYTICS WITH SPARK." *International Conference on Information Engineering, Management and Security (2015)*: 129-133. Print.