Comparative Analysis of Big Data Frameworks: A Review

[1]Prof. Madhuri Chopade, [2]Dr. Seema Mahajan
[1]Assistant Professor, [2]Head of the Department
[1]Information Technology Department, Gandhinagar Institute of Technology, Gandhinagar, Gujrat, India.
[2]Department of Computer Engineering, Indus University, Gujarat, India.

Abstract: Big data is the buzz word in today’s data world. Due to the social network websites, research works, google map data, and many more example can be given to describe the need of big data. There were techniques like data warehouse and data mining. These techniques are only limited to structured data. They are not appropriate option for storing and managing unstructured data. To manage the unstructured data, there was the born of big data techniques. There are many big data frameworks which are available. This paper focus on Apache Hadoop, Apache Storm, Apache Samza, Apache Spark, Apache Flink and Apache Kafka framework. This paper introduces those framework and comparative analysis among these big data frameworks

***Index Terms* - Big Data Framework, Hadoop, Samza, Flink, Spark, Storm, Kafka**

I. INTRODUCTION

Big data is data sets of huge or complex data which cannot be processed using any traditional data processing application software. Big data functionalities includes privacy, data storage, capturing data, data analysis, searching, sharing, visualization, querying, updating, transfer, and information security. There are many big data techniques that can be used to store the data, to perform task faster, to make the system parallel, to increase the speed of processing and to analyze the data [2]. Many distributed computation systems are there which can process Big Data in real time or near-real time. Following are the best Apache frameworks:

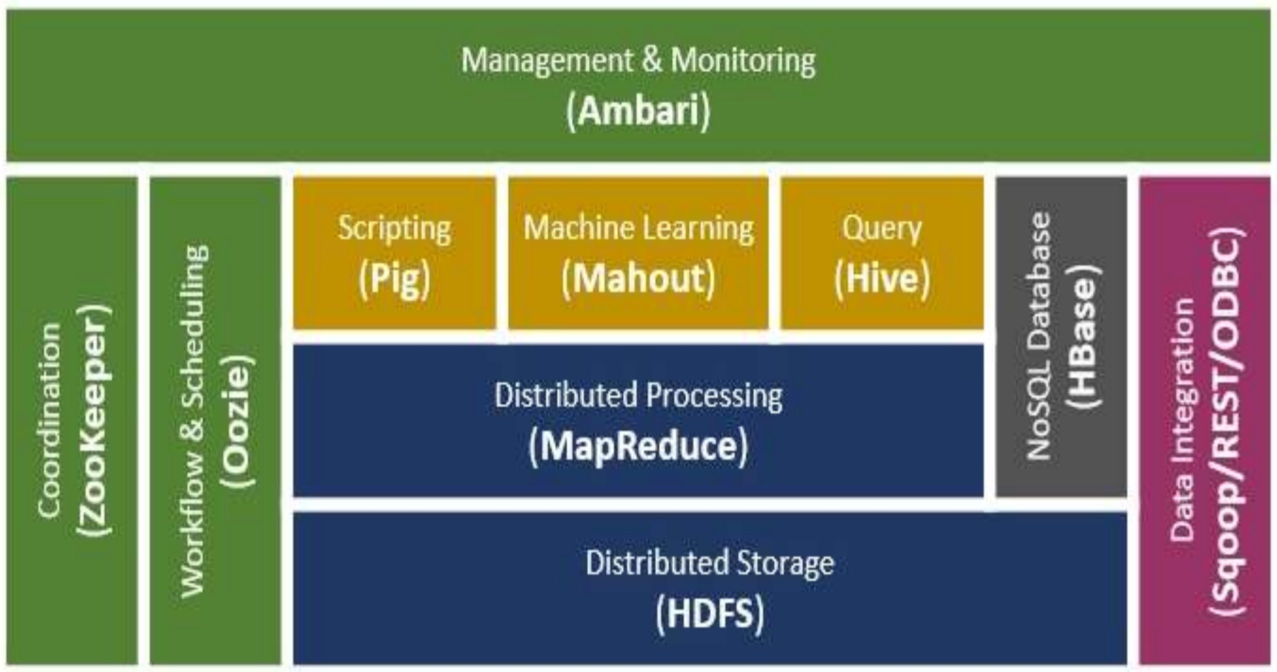
- Apache Hadoop
- Apache Storm
- Apache Samza
- Apache Spark
- Apache Flink
- Apache Kafka

1.1. Apache Hadoop

Apache Hadoop framework which is written in Java is an Open source, Scalable, and Fault tolerant. It is a processing framework that exclusively provides batch processing. It efficiently processes large volumes of data on a cluster of commodity hardware [1]. Hadoop is not only a storage system but is a platform for large data storage as well as processing.The Hadoop ecosystem consists of HDFS and MapReduce. This is accompanied by a series of other projects like Pig, Hive, Oozie, Zookeeper, Sqoop, Flume etc. There are various categories of Hadoop that exist like Cloudera, Hortonworks and IBM Big Insights. Hadoop comes with its flexibility, scalability, fault tolerance and cost effectiveness which makes it more useful for big enterprises.

When software developers discuss software, they generally qualify a software tool for its specific usage. For example, software developer may say that Apache Tomcat is a web server and MySQL is a database but when it comes to Hadoop, however, things become a little bit more complicated as it includes multiple tools.Hadoop comes with multiple tools that are designed and implemented to work together. As a result, Hadoop can be used for many things, people often define it on the basis of the way they are using it. For some people, Hadoop is a data management system bringing together huge amounts of structured and unstructured data that touch nearly every layer of the traditional enterprise data stack [5]. It is able to occupy a central place within a data center. It is a massively parallel execution framework which brings the power of supercomputing to the masses, positioned to fuel execution of organization applications.

Some people view Hadoop as an open source community creating tools and software for solving Big Data problems. The reason behind that is Hadoop provides such a wide variety of capabilities that can be adapted to solve many problems [7]. It is considered as basic framework.



[Fig: 1.1 Apache Hadoop Architecture]

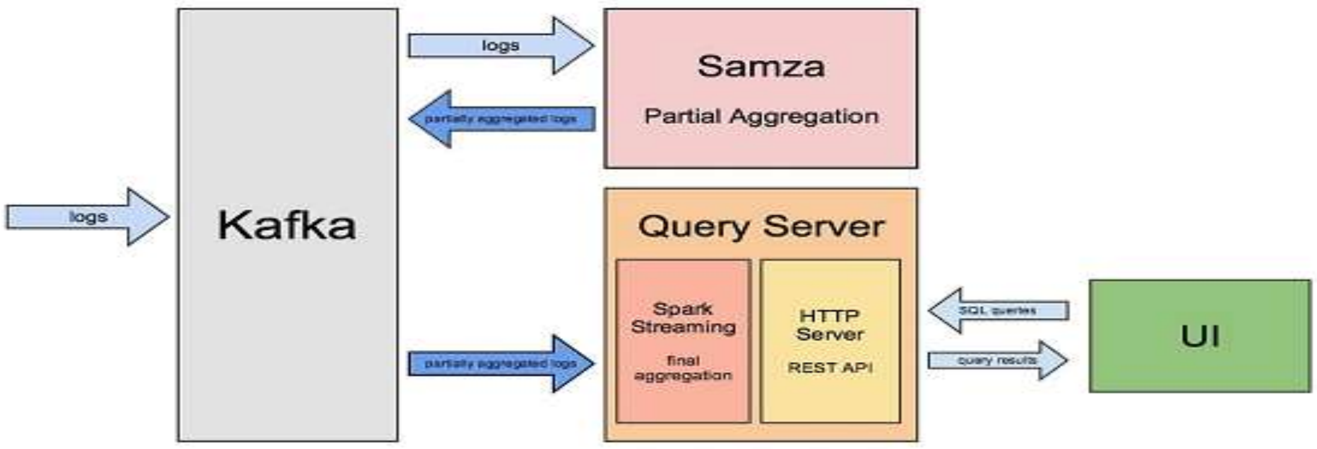
1.2. Apache Storm

A stream processing framework that emphases on extremely low latency and is perhaps the best option for workloads that require near real-time processing is nothing but Apache Storm [6]. It can handle huge amount of data with and deliver results with less latency than other solutions. Storm is simple, can be used with any programming language. Storm has many application as mentioned below:

- real time analytics
- online machine learning
- continuous computation
- distributed RPC
- ETL, and more

Storm is fast: a benchmark clocked it at over a million tuples processed per second per node. It is scalable, fault-tolerant, guarantees your data will be processed, and is easy to set up and operate [13].

1.3. Apache Samza



[Fig: 1.3 Apache Samza Architecture]

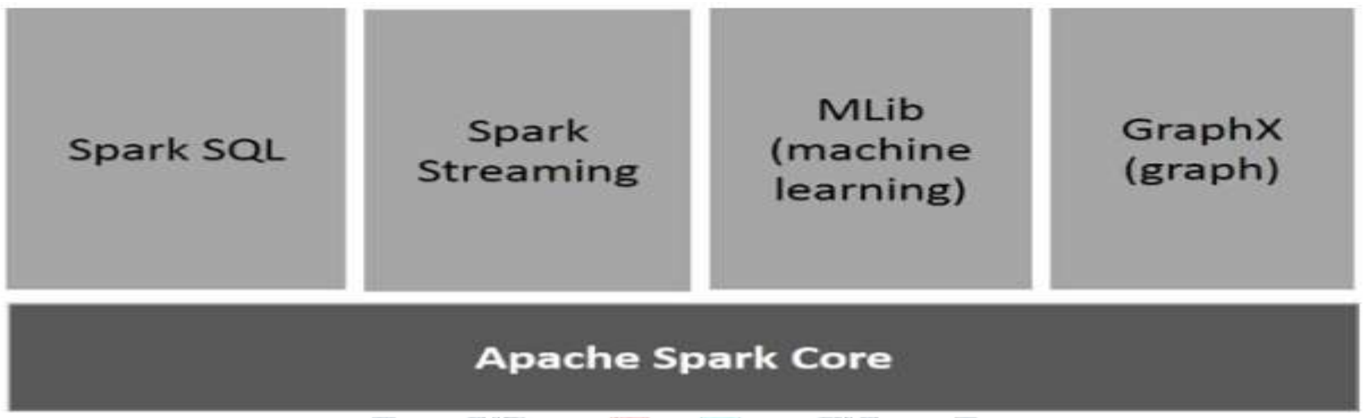
Apache Samza is a stream processing framework that is tightly tied to the Apache Kafka messaging system. While Kafka can be used by many stream processing systems, Samza is designed specifically to take advantage of Kafka's unique architecture and guarantees. It uses Kafka to provide fault tolerance, buffering, and state storage.

Samza uses YARN for resource negotiation. This means that by default, a Hadoop cluster is required (at least HDFS and YARN), but it also means that Samza can rely on the rich features built into YARN [14].

1.4. Apache Spark

Apache Spark is a cluster computing platform designed to be fast and general-purpose. To increase the processing speed of big data, Spark extends the popular MapReduce model to efficiently support more types of computations, including interactive queries and stream processing. To process the large datasets speed is very important, as it means the difference between exploring data interactively and waiting minutes or hours. One of the main features Spark offers for speed is the ability to run computations faster in memory, but the system is also more efficient than MapReduce for complex applications running on disk [7.8.9].

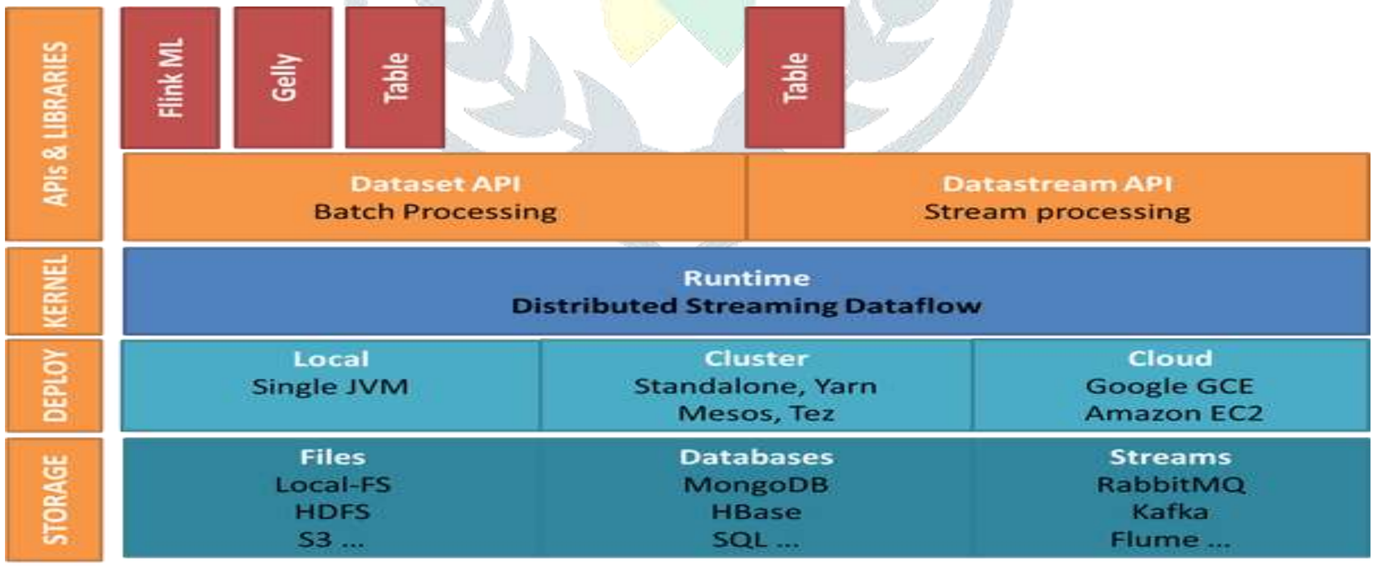
Spark has numerous benefits compared to other big data and MapReduce techniques like Hadoop and Storm. The resilient distributed dataset (RDD) is an application programming interface centered on a data structure provided by Apache Spark. It is a read-only multi set of data items distributed over a cluster of machines that is maintained in a fault-tolerant way. It was developed to overcome the limitations in the Map Reduce cluster computing paradigm, which forces a particular linear dataflow structure on distributed programs [11]. MapReduce programs works in 4 stages. The first is to read input data from disk. Second is to map a function across the data. Third is to reduce the results of the map and fourth is to store reduction results on disk.



[Fig. 1.4 Apache Spark Components]

1.5. Apache Spark

Apache Flink is an open source platform. It is a streaming data flow engine to provide communication, fault-tolerance, and data-distribution for distributed computations over data streams. Flink is a scalable data analytics framework. Apache Flink is fully compatible to Hadoop framework. It can execute both stream processing and batch processing easily [10].

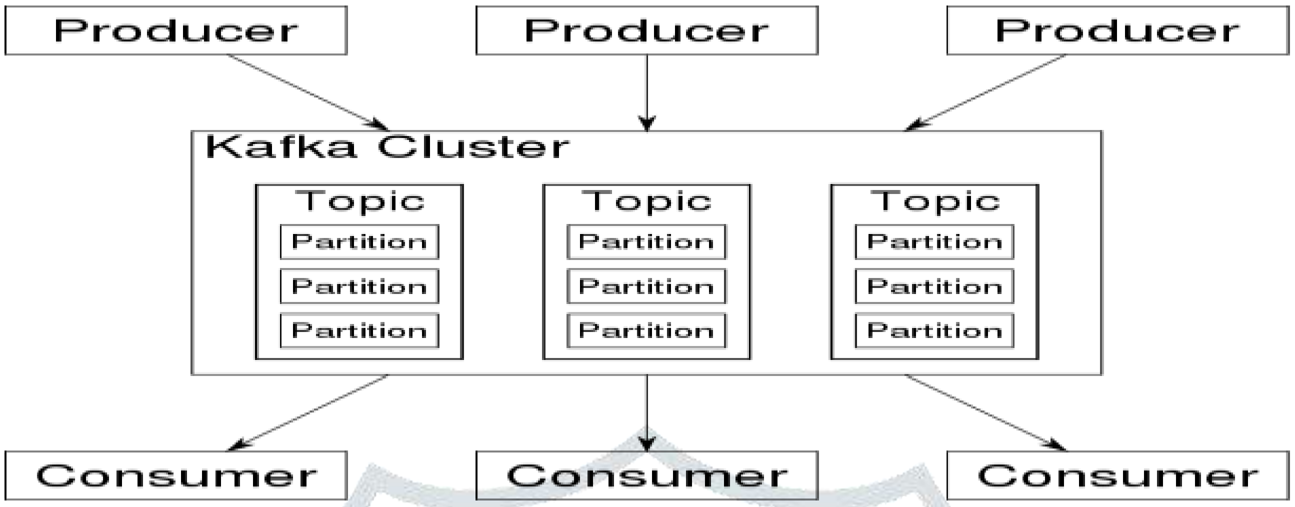


[Fig: 1.5 Apache Flink]

1.6 Apache Kafka

Apache Kafka is a distributed streaming platform, written in Scala and Java. Apache Kafka is a fast, scalable, durable, and fault-tolerant publish-subscribe messaging system. Kafka is often used in place of traditional message brokers like JMS and AMQP because of its higher throughput, reliability and replication.

Kafka works in combination with Apache Storm, Apache HBase and Apache Spark for real-time analysis and rendering of streaming data. Kafka can message geospatial data from a fleet of long-haul trucks or sensor data from heating and cooling equipment in office buildings. Whatever the industry or use case, Kafka brokers massive message streams for low-latency analysis



in Enterprise Apache Hadoop[15].

[Fig: 1.6 Apache kafka Cluster Architecture]

II. Comparison of Big Data Frameworks

Comparative study of big data frameworks [5][16] is provided in below table.

Table 2. 1. Comparison of Big Data Frameworks

Sr. No.	Features	Hadoop	Spark	Flink	Storm	Samza	Kafka
1	**Data Processing**	It is a batch processing System	It is a batch processing as well as stream processing System.	It is a batch processing as well as stream processing System.	It supports stream processing.	It supports stream processing .	It supports distributed stream processing.
2	**Streaming Engine**	Map-reduce is batch-oriented processing tool. Large data set is given as input , processes it and produces the result	Data streams are processed in micro-batches in Apache Spark Streaming processes .	Apache Flink is the true streaming engine. Streams are used for workloads: streaming, SQL, micro-batch, and batch in Apache Flink streaming engine.	Storm processes events one by one	Samza processes events one by one	Kafka Streaming processes data streams in small-batches.

3	**Data Flow**	MapReduce computation data flow does not have any loops. It is a sequence of stages .	Spark represents it as (DAG) direct acyclic graph	It supports controlled cyclic dependency graph in run time .	Storm's topology is designed as a directed acyclic graph (DAG) with spouts, bolts, and streams used to process data.	Samza relies on Kafka's semantics to define the way that streams are handled .	In Kafka a stream processor is anything that takes continual streams of data from input topics, performs some processing on this input, and produces continual streams of data to output topics.
4	**Fault tolerance**	MapReduce is highly fault-tolerant. There is no need to restart the application from scratch in case of any failure in Hadoop.	Apache Spark Streaming recovers lost work and with no extra code or configuration, it delivers exactly-once semantics out of the box .	The fault tolerance is based Lightweight Distributed Snapshots	When workers die, Storm will automatically restart them. If a node dies, the worker will be restarted on another node .	Whenever a machine in the cluster fails, Samza works with YARN to transparently migrate your tasks to another machine	The Kafka cluster can handle failures with the masters and databases.
5	**Scalability**	MapReduce has huge scalability potential. It has been used in production on huge no. of Nodes .	It is highly scalable, we can keep adding n number of nodes in the cluster. A large known as Spark cluster is of 8000 nodes .	"Apache Flink is also highly scalable, we can keep adding n number of nodes in the cluster A large known Flink cluster is of thousands of nodes."	Scalable - with parallel calculations that run across a cluster of machines	Samza is partitioned and distributed at every level. Kafka provides ordered, partitioned, fault-tolerant streams. YARN provides a distributed environment for Samza .	Apache Kafka can handle scalability in all the four dimensions, i.e. event producers, event processors, event consumers and event connectors. In other words, Kafka scales easily without downtime.
6	**Language Support**	Java, c, c++, ruby, groovy, Perl, Python are supported by it.	It supports Java, Scala, Python and R.	"It Supports Java, Scala, Python and R".	Storm is written in Java and Closure but has good support for non-JVM languages.	Samza is written in Java and Scala Also supports JVM languages.	Kafka is written in Java and Scala.
7	**Latency**	Hadoop has higher latency than both Spark and Flink .	It is faster than Hadoop hence offers lower latancy than Hadoop.	With small attempts in configuration, Apache Flink's data streaming runtime achieves low latency and high throughput .	Focuses on extremely low latency.	Offers low latency performance	Provides Low latency

8	**Cost**	MapReduce can typically run on less expensive hardware than some alternatives since it does not attempt to store everything in memory.	As spark requires a lot of RAM to run in-memory, increasing it in the cluster, gradually increases its cost.	Apache Flink also requires a lot of RAM to run in-memory, so it will increase its cost gradually .	Apache Storm also requires a lot of RAM to run in-memory, so it will increase its cost gradually.	Apache Samza also requires a lot of RAM to run in-memory, so it will increase its cost gradually.	Kafka is cost effective even though retention of cost is 10X-100Xmore expensive than using cloud storage.
9	**Security**	Kerberos authentication is supported by it. Some vendors have enabled organizations to leverage Active Directory Kerberos and LDAP for authentication	Apache Spark's security is only supported by password authentication. Spark can also run on YARN to use Kerberos authentication .	Flink is supported by user authentication via the Hadoop / Kerberos infrastructure. To run Flink on YARN, itrequired Kerberos tokens of the user that submits programs, and authenticate itself at YARN, HDFS,and HBase with that.	Storm cluster that authenticates users via Kerberos	Samza uses apache Hadoop YARN to provide security	Kafka supports cluster encryption and authentication, including a mix of authenticated and unauthenticated, and encrypted and non-encrypted clients
10	**Scheduler**	Scheduler in Hadoop becomes the pluggable component. There are two schedulers: Fair Scheduler and Capacity Scheduler. To schedule complex flows, MapReduce needs an external job scheduler like Oozie .	Due to in-memory computation, spark acts its own flow scheduler .	Flink can use YARN Scheduler but Flink also has its own Scheduler .	Storm now has 4 kinds of built-in schedulers: DefaultScheduler, IsolationScheduler, MultitenantScheduler, ResourceAwareScheduler .	Samza use YARN Scheduler.	Kafka uses its own Kafka Scheduler.

III. Conclusion

After performing comparative analysis of different big data frameworks, it can be concluded that any one framework cannot be a good option for performing every task. As per the application need, the best framework can be decided. For performing stream processing hadoop is not good option but other frameworks can be the best option. Comparison table explains the need and advantage of each framework.

References

[1] Oussous, A., et al. Big Data technologies: A survey. Journal of King Saud University – Computer and Information Sciences (2017)
[2] V Srinivas Jonnalagadda , P Srikanth , Krishnamachari Thumati , Sri Hari Nallamala - A Review Study of Apache Spark in Big Data Processing, International Journal of Computer Science Trends and Technology (IJCST) – Volume 4 Issue 3, May - Jun 2016
[3] https://www.kdnuggets.com/2016/03/top-big-data-processing-frameworks.html
[4] Rabl, T., Gómez-Villamor, S., Sadoghi, M., Muntés-Mulero, V., Jacobsen, H.A., Mankovskii, S.: Solving big data challenges for enterprise application performance management. Proc. VLDB Endow. 5, 1724–1735 (2012)
[5] https://www.digitalocean.com/community/tutorials/hadoop-storm-samza-spark-and-flink-big-data-frameworks-compared,Copyright©2018 DigitalOcean™ Inc.

[6] Mrigank Mridul, Akashdeep Khajuria, Snehasish Dutta, Kumar N " Analysis of Bidgata using Apache Hadoop and Map Reduce" Volume 4, Issue 5, May 2014" 27

[7] Jonathan Stuart Ward and Adam Barker "Undefined By Data: A Survey of Big Data Definitions" Stamford, CT: Gartner, 2012.

[8] Bernice Purcell "The emergence of big data technology and analytics" Journal of Technology Research 2013.

[9] Harshawardhan S. Bhosale1, Prof. Devendra P. Gadekar "A Review Paper on Big Data and Hadoop" International Journal of Scientific and Research Publications, Volume 4, Issue 10, October 2014

[10] BIG DATA: Challenges and opportunities, Infosys Lab Briefings,Vol 11 No 1, 2013.

[11] John A. Miller, Casey Bowman, Vishnu Gowda Harish and Shannon Quinn "Open Source Big Data Analytics Frameworks Written in Scala" 2016 IEEE International Congress on Big Data

[12] Wissem Inoublia, Sabeur Aridhib, Haithem Meznic, Alexander Jung "An Experimental Survey on Big Data Frameworks" Available from: https://www.researchgate.net/publication/309572509_Big_Data_Frameworks_A_Comparative_Study [accessed Jul 14 2018].

[13] S. Landset, T. M. Khoshgoftaar, A. N. Richter, and T. Hasanin. "A survey of open source tools for machine learning with Big data in the Hadoop ecosystem.Journal of Big Data", 2(1):1, 2015.

[14] S. Aridhi and E. M. Nguifo. Big graph mining: Frameworks and techniques.Big Data Research, 6:1–10, 2016.

[15] https://hortonworks.com/apache/kafka

[16] https://opensourceforu.com/2018/03/a-quick-comparison-of-the-five-best-big-data-frameworks

JETIR