2018 1st Annual International Conference on Information and Sciences (AICIS)

Recent trends in distributed online stream processing platform for big data: Survey

Ahmed Hussein Ali
Ph.D. candidate, ICCI, Informatics Institute for Postgraduate Studies
Baghdad, Iraq
msc.ahmed.h.ali@gmail.com

Mahmood Zaki Abdullah
Assistant Professor Doctor of Computer Engineering, Al-Mustansiriyah
Baghdad, Iraq
drmzaali@uomustansiriyah.edu.iq

***Abstract*— There is no doubt that big data has become an important source of information and knowledge, especially for large profitability companies such as Facebook and Amazon. But, dealing with this kind of data comes with great difficulties; thus, several techniques have been used to analyze them. Many techniques handle big data and give decisions based on off-line batch analysis. Today, we need to make a constructive decision based on online streaming data analysis. Stream computing is gaining interest because it provides the opportunity for real-time data analytics. The objective of this paper is to give an in-depth analysis of efficient big data streaming analysis platforms, as well as to provide solutions for some on-line big data processing problems. Additionally, some recent works in big data streaming were highlighted.**

Keywords- Component; Big data; Streaming; MOA; Big data frameworks; Big data platforms.

I. INTRODUCTION

The amount of data in our world has been rapidly growing in recent time. contrary to the conventional data, Big Data refers to different formats of large and rapidly increasing datasets. Big data comes in different forms, such as unstructured, semi-structured, and structured [1]. Big data analytics is the process of realizing the value of a dataset; it helps in gaining new insights from datasets, resulting in better and faster decisions. Typically, Big data analytics employs techniques that range from simple statistics to sophisticated analytics which involves machine learning, such as text analytics, predictive analytics, and data mining. Big data mining provides a range of attractive possibilities. However, big data experts are faced with various challenges when examining Big datasets. They also experience problems when extracting information and value from such information mining. Distributed stream computing has become an inevitable processing method because it makes it possible to continuously get results from large volumes of fresh data in real time [2]. Internet of Things (IoT) is considered one of the central stores of Big data applications. The IoT and Big data are intimately connected; Big data contributes in the IoT by providing several analytics platforms for the analysis of huge data gathered by IoT. There are several instances that typify the working together of Big data and IoT to provide insight and analysis, and a typical example of such corporation is found in the shipping organizations where sensor data and Big data analytics are utilized to enhance efficiency, save cost, and reduce the environmental influence of their activities. The IoT indeed facilitated stream computing development through its rapid expansion and deployment of sensors at various geographic locations [1, 2]. The rest of this paper was structured thus: section 2 presented a discussion of the works related to this study while section 3 explained the streaming frameworks. Section 4 presented a comparison for Big data streaming frameworks while the problems and solutions are discussed in section 5. The latest works in Big data streaming and the research trends were presented in Section 6 while section 7 presented the conclusions drawn from this study.

II. RELATED WORK

Several Big data surveys have been proposed by the research community and most of the survey articles related to Big data streaming did not provide a deep analysis of the problems and solutions; they did not provide appropriate comparisons as well. In this survey, we strived to provide a review of data stream processing techniques in the Big data area with the and the aim of providing a fast introduction and survey of the technical solutions for Big data stream processing based on the relevant comparisons in term of learning type, supported languages, supported machine learning tools, in-memory processing, low latency, and fault tolerance. Sara et al [3] presented a survey of the machine learning techniques and open source tools for batch Big data processing, while Dilpreet & Reddy [4] presented a comprehensive survey of Big data analytics platforms and their characteristics. Furthermore, Annie et al [5] presented a quick review of streaming data analytics in relation to anomaly detection with a set of cases in the given area.

III. STREAMING PLATFORMS/FRAMEWORKS

Apache Spark

Apache Spark is an open-source parallel in-memory processing platform for Big data analytics paradigm and machine learning (ML) algorithms [6]. The Spark provides many advantages compared to the other platforms for Big data analytics such as Storm and Hadoop which uses MapReduce platform. Compared to MapReduce, Spark is low latency and provides a high speed because of its ability to reduce disk input and output operation. Data process in Spark is faster than in MapReduce because Spark has in-memory computation feature. In Spark, intermediate data is stored in Resilient Distributed Datasets (RDD). The RDD is a cached in-memory; thus, data processing is faster compared to data processing in Hadoop [7]. Spark can execute complex analysis on huge data sets, and as such, Big data analysis in Spark is more than 100x faster compared to Apache Hadoop and Hive which uses MapReduce platform. The Spark project contains a set of levels or components for process scheduling, memory management (Spark scheduler, YARN, Mesos), error recovery, and for interacting with the storage systems (Local, HDFS, NoSQL, S3, RDBMS). The components or levels in Spark are illustrated in Figure1 and listed as follows:

- **Spark streaming**: This is another Spark component which provides scalability, automatic parallelization, and fault-tolerant streaming. This component makes it possible to stream tasks by writing batch-like processes in Scala and Java, and to integrate interactive queries and batch jobs. With this component, each streaming process is executed as a series of short batch tasks on in-memory data in RDDs.

- **MLlib**: This is a distributed ML platform which was built on top of Spark. In terms of performance, MLlib provides various ML frameworks for the optimization of clustering, classification, and regression tasks [8]. MLlib, just like Mahout, is important for ML categories. They provide frequent pattern mining and topic modeling algorithms. They also support regression frameworks. However, such models are not supported by Mahout although MLlib is quite a new platform compared to Mahout.

- **Spark SQL**: This important feature in Spark unified RDD and relational tables, enabling programmers to mix SQL commands with ease when querying external datasets with complicated analytics. With SQL, programmers can easily query both externally-sourced datasets (such as Hive Tables and Parquet files) and data saved in the existing RDDs. Furthermore, Spark SQL makes it possible to write out RDDs to Parquet files or Hive tables. It also enables fast parallel data query processing over large distributed datasets. For this reason, Spark SQL uses HiveQL as a query language. For the development of a fast application, Catalyst framework which enables users to rapidly incorporate new optimizations via Spark SQL was developed.

- **GraphX**: This is composed of a library for graph manipulation and graph-parallel computation execution. GraphX extends Spark RDD API features just like Spark SQL and Spark Streaming, thereby making it possible for users to create a directed graph with imaginary features attached to each edge and vertex. Several operators for graph manipulation such as mapVertices and subgraph are provided by GraphX. It also provides several graph frameworks such as triangle counting and PageRank.

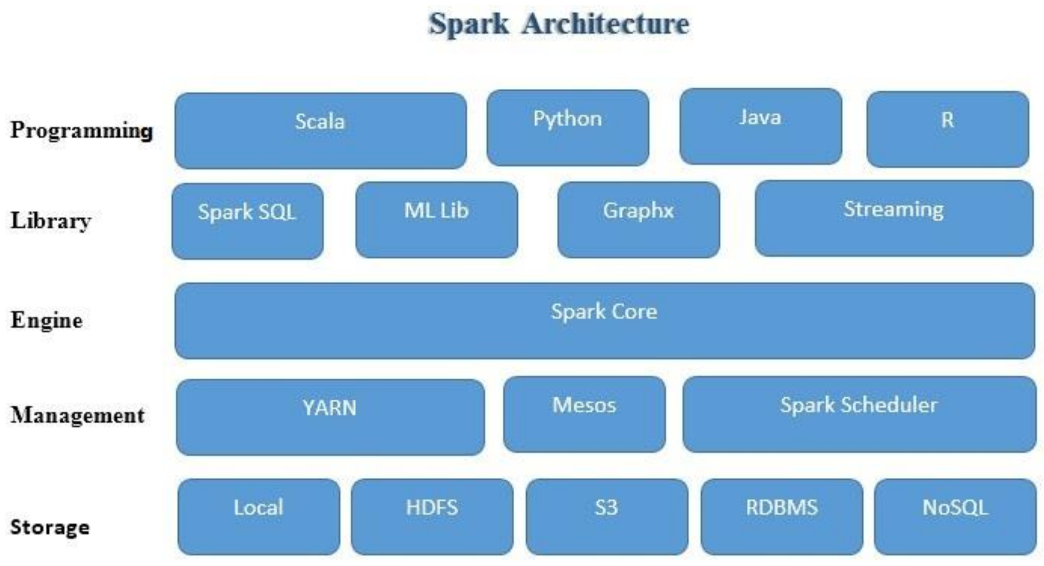

Fig.1: Spark architecture

The major issue is the way to manage data cluster and push it for spark processing. First, the input dataset has to be uploaded to the Hadoop directory for onward usage by MapReduce nodes. Next, the input dataset will be divided by the HDFS into data splits for onward storage in a cluster, taking note of the replication factor for fault tolerance. Then, a Task tracker will parallelly process all the data splits for the Map and Reduce tasks. Either an external or embedded cluster manager is used in the distributed computing frameworks. A cluster manager is deployed for the management of a cluster of computers. Specifically, cluster manages are used to manage computer resources such as CPU, ports, memory, storage, and other available resources on a node cluster. It pools the available resources on each node for onward sharing to different applications. There are two components of a standalone cluster manager: master and worker (Figure 2). The computer resources are managed by the worker at the nodal level while the master processes the computer resources pooled from the workers and allocate them accordingly. The master process can either run simultaneously with one worker node or can be run on a separate server.

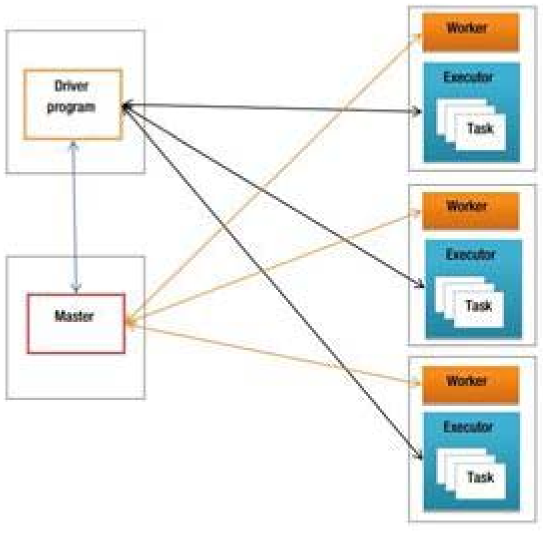

Fig.2: Data distributing over clusters

Massive Online Analysis (**MOA)**

The MOA [9] framework is an analytics platform with several ML algorithms for online data streams learning. Hence, MOA is appropriate for stream ML but not support for distributed processing. It can only run on one machine, and when required, cannot be scaled into more machines [10]. The MOA supports both ML and data mining algorithm in getting fast-optimized solutions to nature and evolution of data streams. It is easy to implement and with special Graphical User Interface (GUI). Furthermore, it can be simplified using Java API, thereby, making it suitable for programmers with varying levels of experience. A commonly recognized capability of MOA is its modularity. Except for the core program, any programmer can opt to expand only the modules of interest, thereby saving time and effort in scouting non-relevant features. Tasks such as classification, clustering, regression, outlier detection, change detection, and recommender systems are supported in the latest version of MOA. The framework is also compatible with algorithms that run on single node machines, as well as those that study algorithmic capabilities for either scale up (vertical scaling) or scale out (horizontal scaling) processes. One of the most important analysis languages is the R programming language because it offers a simple way developing functions that accommodate loops, conditionals, output and input facilities which as earlier mentioned, can be merged with functions which are written in other programming languages but can execute computationally intensive tasks. The MOA works as follows: first, the data feed method is chosen; then, an algorithm is used to train the data (multiple algorithms that executes the same task are used to train the data for comparisons sake); finally, the desired settings are analyzed using a selected evaluation method.

Samza

Apache Samza is a distributed online stream processing platform. Apache samza depends on both Apache Kafka and Apache YARN; therefore, it performs messaging by Kafka and provides better resource management, processor isolation, security, and fault tolerance by YARN [11]. Samza is used to process data streams which must be made up of a similar type of fixed messages. A Samza process comprises of a code which logically transforms a set of input data stream and appends the output of the transformation process to set of output data streams [12, 13]. To scale the stream processor's throughput in Samza, job streams are collected into smaller parallel units (partitions and tasks).

Apache Storm

This is a free and openly available system for real-time distributed stream processing of Big data. Both Hadoop and Storm facilitates Big data processing; Storm for high-velocity data stream and Hadoop for large volume data patch processing. Spout and Bolt are two components of Storm; Spout is a stream source while Bolt is for stream data processing. With these components, streams can be easily transformed from one form to another. Both components have an interface that can allow a user to achieve specific application logic. Storm can execute tasks such as distributed topologies in clusters [14]. For task execution, Storm offers 2 parallel mechanisms [15, 16] to achieve the desired processing capability. One of the mechanisms is horizontal parallelism which implies adding Bolts horizontally in order to enhance the processing capability. The other mechanism is vertical parallelism which implies achieving parallelism in each node. Storm creates several Java Virtual Machines (JVMs) at each node and executes multiple threads in each of the nodes.

There are many features of Storm, such as [17]:

- Speed of processing: About 1 million tuples can be handled by storm per second
- Horizontal scaling: Apache storm is more scalable or linearly scalable. Processing capacity can be increased by adding more nodes to the storm cluster.
- Ensuring processing of data: Storm provides the ability to replay when there is lost tuple and failure. At least, once the message can be processed, storm provides a high level of guarantee. Little maintenance is required to deploy the cluster.
- Clear and suitable to use: Apache storm is suitable and easy in management and deployment.
- Fault allowance: In apache storm, the main unit of labor is carried out by the worker. If any worker at any time stop working (fail), then, the storm will switch that worker to another node in the cluster.

Kafka

Until now, the most common framework data streams ingestion into the processing platforms is Apache Kafka [2, 18]. It is an open source processing framework which was put together by Apache Software Foundation. It is commonly used because of its scalable storage layer and its capability to handle large data feeds in real-time. Kafka is comprised of broker, topics, producers, and consumers. In Kafka, a broker is a cluster made up of one or more servers [21], while topic is a category for storing and publishing messages. Kafka also has partition log in which messages are identified by a specific ID called offset. Actually, there are more than one partition logs inside the topic. The messages replicated to many servers are stored in portion logs. Each partition has many servers; one server serves as the leader while the others act as followers. It is the duty of the leader to accept and perform all the reading and writing tasks in the server. The followers are exactly the same as the leader; in case a leader server fails, a follower is chosen by default to be the new leader. Producers are processes that separate the data into the topics of interest. Only selected messages from selected partitions in a topic can be produced by the producer. The producer executes this task in a manner that is most appropriate to the condition at hand. Now comes the consumers; the consumer consumes the published messages under the topics. It is always labeled with the name of a consumer group. The messages are delivered to the to the subscribed consumer groups based on their instances.

IV. Comparison of Big Data Streaming Frameworks

The comparison of the frameworks of Big data analytics is shown in Table 1. The frameworks are compared in term of supported languages, machine learning tools, latency, and fault tolerance.

V. Problems and Solutions

In this section, a brief explanation of the recent Big data streaming frameworks was provided. They were compared based on the available literature. The challenges that can be encountered in Big data analytics due to changes in environments, input data, and systems were also identified. The computation load can be increased by the surge in Big data as it can fill the "input" system. One way to avoid such challenges is to incorporate more computation resources, while another way is to deploy different computation nodes for the analysis task. There are also some problems while working with Hadoop and Spark because of their complexity. So, the solution is to use Hadoop framework as an example. Radoop is purely a visual workflow designer for Hadoop and Spark. Radoop eliminates data preparation complexities and ML on Hadoop and Spark. Similarly, a combination of the frameworks may be ideal for one algorithm; it can also make such algorithm highly scalable and efficient in executing real-time analysis.

VI. The Latest Works in Big Data Streaming and Research Trends

Ramirez-Gallego et al [22] proposed a novel parallel classifier based on the lazy KNN algorithm. This algorithm is executed in Apache Spark for high-speed Big data streams. Baojun et al [23] proposed a parallel monitoring system for huge data traffic based on Apache Spark. They used JPCAP as a collector to capture packets, Kafka as a messaging system, and Spark streaming as a stream processor. Babak et al [24] developed a new Big data processing framework for online traffic data stream. This framework used Apache Kafka as a data ingestion component and Spark streaming as data processing component to provide more sophisticated and high-level data stream analytics. Guipeng et al [25] proposed a novel partition method called SP-Partitioner for live data stream processing. The new partitioner solved the problem of imbalance intermediate data stream in Spark Streaming. Glaudia et al [26] proposed a novel approach called Pipeline Composition (PiCo) for real-time data processing. PiCo was proposed to enhance the process of Big data stream analysis compared to Kafka and Spark Streaming. To usher in Big data researchers into this new age, we present the highest impact research trends in the field of Big data streaming as follow:

- There is no doubt that reducing computational time using parallel computing is one of the remarkable future trends.
- How to make machine learning algorithms operate in the parallel method is a challenge since these algorithms were not designed for parallel processing environments.
- Because of the importance of social media, the gathering and processing of social media data will be the highest impact research trend in term of Big data streaming.

Table 1 Big data streaming comparison

Frameworks	Learning type	Supported languages	Supported machine learning tools	In memory processing	Low latency	Fault tolerance	Representative references
Spark	Stream learning, batch learning	scala, Python, R, and Java Virtual Machine (JVM) languages	H2O, machine learning MLlib, and Mahout	Yes	Yes	Yes	[4]
MOA	Streaming	R	Weka	Yes (single machine)	Yes	No	[19]
Storm	Streaming	Any	SAMOA	Yes	Yes	Yes	[16]
Kafka	Streaming	Java, Scala	Kafka stream	Yes	Yes	Yes	[20]
Samza	Streaming	Java Virtual Machine (JVM) languages	Kafka stream	Yes	No	Yes	[7]

VII. Conclusion

This paper focused on open source Big data streaming analysis frameworks/platforms such as Apache Spark, MOA, Samza, Storm, and Kafka. These frameworks/platforms have their own features and application. They are for unstructured data, online stream processing, fast in-memory calculations, fast OLAP, and real-time computing, respectively. The adoption of these platforms depends on different scenarios. This article provided a comprehensive comparison of the most important Big data streaming frameworks and the problems of using such platforms, with suitable solutions based on literature and some experiments. We have reviewed some recent studies in Big data streaming and the areas several works can be done in the future. Many issues relating to Big data streaming and machine learning algorithms still remain challenging areas of research in Big data.

References

[1] A. Oussous, F.-Z. Benjelloun, A. A. Lahcen, and S. Belfkih, "Big Data technologies: A survey," Journal of King Saud University-Computer and Information Sciences, 2017.

[2] P. Le Noac'H, A. Costan, and L. Bougé, "A Performance Evaluation of Apache Kafka in Support of Big Data Streaming Applications," in IEEE Big Data 2017, 2017.

[3] 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, vol. 2, p. 24, 2015.

[4] D. Singh and C. K. Reddy, "A survey on platforms for big data analytics," Journal of Big Data, vol. 2, p. 8, 2015.

[5] A. I. Rana, G. Estrada, M. Solé, and V. Muntés, "Anomaly Detection Guidelines for Data Streams in Big Data," in Soft Computing & Machine Intelligence (ISCMI), 2016 3rd International Conference on, 2016, pp. 94-98.

[6] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," HotCloud, vol. 10, p. 95, 2010.

[7] H. E. Ciritoglu, L. Batista de Almeida, E. Cunha de Almeida, T. S. Buda, J. Murphy, and C. Thorpe, "Investigation of Replication Factor for Performance Enhancement in the Hadoop Distributed File System," in Companion of the 2018 ACM/SPEC International Conference on Performance Engineering, 2018, pp. 135-140.

[8] Saba A. Salman, Al-Hakam A. Salih, Ahmed H. Ali, Mohammad. K. Khaleel, and Mostafa A. Mohammed. A. Mohammed, "A New Model for Iris Classification Based on Naïve Bayes Grid Parameters Optimization."

[9] W. Romsaiyud, "Automatic extraction of topics on big data streams through scalable advanced analysis," in Computer Science and Engineering Conference (ICSEC), 2014 International, 2014, pp. 255-260.

[10] A. Valsamis, K. Tserpes, D. Zissis, D. Anagnostopoulos, and T. Varvarigou, "Employing traditional machine learning algorithms for big data streams analysis: The case of object trajectory prediction," Journal of Systems and Software, vol. 127, pp. 249-257, 2017.

[11] http://samza.apache.org/.

[12] N. Kazanskiy, V. Protsenko, and P. Serafimovich, "Performance analysis of real-time face detection system based on stream data mining frameworks," Procedia Engineering, vol. 201, pp. 806-816, 2017.

[13] F. Su, Z. Wang, S. Yang, K. Li, X. Lu, Y. Wu, et al., "A Survey on Big Data Analytics Technologies," in International Conference on 5G for Future Wireless Networks, 2017, pp. 359-370.

[14] S. Tang, B.-S. Lee, and B. He, "Fair resource allocation for data-intensive computing in the cloud," IEEE Transactions on Services Computing, 2016.

[15] S. Liu, Y. Ji, D. Zhang, Y. Yuan, J. Gong, and R. Wang, "An Online Prediction Algorithm of Traffic in Big Data Based on the Storm," in

Advanced Cloud and Big Data (CBD), 2017 Fifth International Conference on, 2017, pp. 129-134.

[16] T. Li, J. Tang, and J. Xu, "Performance modeling and predictive scheduling for distributed stream data processing," IEEE Transactions on Big Data, vol. 2, pp. 353-364, 2016.

[17] A. Jain and A. Nalya, Learning storm: Packt Publishing, 2014.

[18] https://kafka.apache.org/.

[19] D. Marrón, J. Read, A. Bifet, and N. Navarro, "Data stream classification using random feature functions and novel method combinations," Journal of Systems and Software, vol. 127, pp. 195-204, 2017.

[20] P. Le Noac'H, A. Costan, and L. Bougé, "A performance evaluation of Apache Kafka in support of big data streaming applications," in Big Data (Big Data), 2017 IEEE International Conference on, 2017, pp. 4803-4806.

[21] N. Garg, Learning Apache Kafka: Packt Publishing Ltd, 2015.

[22] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, J. M. Benítez, and F. Herrera, "Nearest Neighbor Classification for High-Speed Big Data Streams Using Spark," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 47, pp. 2727-2739, 2017.

[23] B. Zhou, J. Li, X. Wang, Y. Gu, L. Xu, Y. Hu, et al., "Online Internet traffic monitoring system using spark streaming," Big Data Mining and Analytics, vol. 1, pp. 47-56, 2018.

[24] B. Yadranjiaghdam, S. Yasrobi, and N. Tabrizi, "Developing a Real-time Data Analytics Framework For Twitter Streaming Data," in Big Data (BigData Congress), 2017 IEEE International Congress on, 2017, pp. 329-336.

[25] G. Liu, X. Zhu, J. Wang, D. Guo, W. Bao, and H. Guo, "SP-Partitioner: A novel partition method to handle intermediate data skew in spark streaming," Future Generation Computer Systems, 2017.

[26] C. Misale, M. Drocco, G. Tremblay, and M. Aldinucci, "Pico: a novel approach to stream data analytics," in European Conference on Parallel Processing, 2017, pp. 118-128.