Apache Spark: A Big Data Processing Engine

Conference Paper · November 2019

DOI: 10.1109/MENACOMM46666.2019.8988541

4 authors, including:

Iman Ahmed Mohiuddin
Prince Mohammad University
3 PUBLICATIONS **121** CITATIONS
SEE PROFILE

Irum Nahvi
Prince Mohammad University
3 PUBLICATIONS **75** CITATIONS
SEE PROFILE

Apache Spark: A Big Data Processing Engine

Eman Shaikh
College of Computer Engineering and Science,
Prince Mohammad Bin Fahd University,
Khobar, Saudi Arabia.
emanshaikh26@gmail.com

Iman Mohiuddin
College of Computer Engineering and Science,
Prince Mohammad Bin Fahd University,
Khobar, Saudi Arabia.
iman28198@gmail.com

Yasmeen Alufaisan
College of Computer Engineering and Science,
Prince Mohammad Bin Fahd University,
Khobar, Saudi Arabia.
yalufaisan@pmu.edu.sa

Irum Nahvi
College of Computer Engineering and Science,
Prince Mohammad Bin Fahd University,
Khobar, Saudi Arabia.
inahvi@pmu.edu.sa

***Abstract*—Big data analysis has influenced the industry market. It has a significant impact on large and varied datasets to exhibit the hidden patterns and other revelations. Apache Hadoop, Apache Flink and Apache Storm are some commonly used frameworks for big data analysis. Apache Spark is a consolidated big data analytics engine and provides absolute data parallelism. This paper scrutinizes a technical review on big data analytics using Apache Spark and how it uses in-memory computation that makes it remarkably faster as compared to other corresponding frameworks. Moreover, Spark also provides exceptional batch processing and stream processing capabilities. Furthermore, it also discuses over the multithreading and concurrency capabilities of Apache Spark. The point of convergence is architecture, hardware requirements, ecosystem, use cases, features of Apache Spark and the use of Spark in emerging technologies.**

***Index Terms*—Big Data Analytics , Machine Learning, Stream Processing, Resilient Distributed Datasets.**

I. INTRODUCTION

Big data analytics research has vastly exerted influence in the market of industries. It is a strategy of fetching large volumes of data from an extensive variety of sources, organizing the same data completely and then analyzing those big sets of data to locate meaningful facts and figures from the data collected. As a result of heterogeneous data aggregation, many organizations that provides web services like Amazon and Microsoft use cluster of commodity servers.

The term big data is used to refer to a particularly huge amount of complex datasets that are evolved from new data sources and are examined for traditional data processing application software. Big data is used for non-conventional strategies and technologies that require gathering insights from large datasets as well as organizing and processing these data. In a nutshell, big data analysis is a process of finding knowledge from bulk of data. Therefore, in order to analyze such huge data, it is necessary to use some kind of analysis tool or processing framework.

Processing framework is observed to be one of the most important constituent of big data systems. Processing frameworks determine data in the system, either by consuming the data into the system as it is or by reading the data from a non-volatile storage. Processing frameworks are categorized by the type and condition of the data they are designed to operate on. Where some systems deals with data in batches, other systems undertake data in an uninterrupted stream as it moves into the system. There are some systems that can handle data in both ways as well [1]. Mainly, big data processing frameworks can be divided into the following three categories: batch-only framework, stream-only framework and hybrid framework.

A batch processing system collects all data into a group which is stored and processed later in the future. On the other hand, stream processing systems process the data as soon as they arrive, i.e. real-time processing. In the hybrid processing systems, both batch and stream workloads can be handled. This results in a simpler, more general data processing since we can apply the same features or APIs for both batch and stream data.

Apache Spark is a substantial consolidated analytics engine for a comprehensive distributed data processing and machine learning workloads. It has established a broad domain to solve data science and engineering problems using programming languages like Python. Apache Spark reinforces techniques such as in-memory processing, stream and batch processing of big data workloads. These techniques will be discussed further in Section III. Apache Spark has rapidly been embraced by an infinite range of industries. It is not only active projects in Apache Software Foundation but widely accepted open source project. The act of assembling, processing and storing large volume of data is big data.

In this paper, we focus specifically on Apache Spark big data processing framework. We discuss Spark batch and stream processing abilities. We further describe the different features that makes Spark a unique framework. We also discuss some of the main use cases of Apache Spark. After that, we review Spark ecosystem, architecture and hardware requirements. Finally, we address Spark multi-threading and concurrency capabilities and the use of Spark in emerging technologies.

The remaining paper is organized as follows: Section II

978-1-7281-3687-5/19/$31.00 ©2019 IEEE

introduces the literature work related to this paper, Section III discusses Apache Spark, and Section IV concludes the paper.

II. RELATED WORK

In [2], the authors initiated the Apache Spark project that provides an integrated analytic engine for a wide range of disseminated data processing. Spark allows programming whole cluster in parallel. It expands its model to an elementary data structure called as Resilient Distributed Datasets (RDDs), even though having a programming model same as MapReduce. Spark is the foremost information processing system for comprehensive SQL, stream processing, graph processing and machine learning. Therefore, Apache Spark model can effectively aid present workloads and provide ample benefits to the users.

Salloum et al. [3] focused on the key components and distinctive characteristics of Apache Spark in big data analytics. Some heterogeneous functionalities for design and implementation are produced by Apache Spark, which comprises of machine learning pipelines API. Apache Spark is a wide spread cluster computing framework which is popular not only in academic community but in the market of industry. Consequently, this paper directed a great deal of attention towards the research and growth correlated to Apache Spark in big data analytics.

In [4], the authors proposed solutions to overcome the significant confrontations faced during big data analysis. In their work, they use Apache Storm framework with a sample of Twitter data. Apache Storm was able to successfully overcome those challenges in turn proving that it can process real-time streams with very low latency.

In [5], the authors developed a new pipeline for functional magnetic resonance imaging (fMRI) using the PySpark on a single node. PySpark is a language for data analysis and pipelines, which exposes Spark programming model to Python. In this pipeline, the brain networks are extricated from fMRI data by template matching and the sum of squared differences (SSD) method. In terms of processing time, this pipeline is four times faster than the one developed in Python. It has upgraded in-memory data processing in parallel, converted the data to Resilient Distributed Datasets and stored results in other formats like data frames.

Gopalani et al. [6] mainly presented a comparison between Apache Hadoops Map Reduce and Apache Spark framework as both options are used in big data analysis. Furthermore, the paper compares the two frameworks on various parameters as well as provides a performance analysis on them using the K-Means algorithm and it was concluded that the Apache Spark framework will bring a major change in the big data world due to its ability of in-memory processing.

In [7], the authors performed a comparative study of Apache Spark and Apache Flink. In particular, the paper focused on comparing machine learning libraries in these frameworks for batch data processing. The machine learning algorithms used in the study are Support Vector Machines and Linear Regression. The paper showed with empirical results that Spark outperforms Flink in terms of performance.

From the usability and ease of use perspective, distributed data flow-oriented platforms- Apache Hadoop MapReduce, Apache Spark and Apache Flink were compared in [8]. MapReduce pursuits the challenges like scalability and built-in redundancy, while as latter two focus on the need of efficient data flow, data caching and declarative data processing operators. The main intention is to provide a route to select a suitable platform and provide better understanding for the functionality of big data processing systems.

Our paper is different from these previous research as it reviews Apache Spark from various aspects. We focus on Sparks key features, batch processing and stream processing abilities, use cases, ecosystem, architecture, multi-threading and concurrency capabilities. Lastly we also mention how Spark has become an absolute vital tool used in current emerging technologies.

III. APACHE SPARK

Apache Spark is a powerful big data processing platform which adapts the hybrid framework. A hybrid framework offers support for both batch and stream processing capabilities. Even though Spark uses many similar principles to Hadoops MapReduce engine, Spark outperform the latter in terms of performance. For instance, given the same batch processing workload, Spark can be faster than MapReduce due to the ”full in-memory computation” feature used by Spark compared to the traditional read from and write to the disk used by MapReduce. Spark can run in standalone mode or it can be combined with Hadoop to replace MapReduce engine.

1) Spark Batch Processing Model: The strongest advantage of Spark over MapReduce is the in-memory computation. Spark interacts with the disk only for two tasks: loading the data initially into the memory and storing the final results back to the memory. All other results in-between is processed in memory. This in-memory processing makes Spark significantly faster than its competitive batch processing framework Hadoop. Furthermore, holistic optimization used by Spark contributes further to its high speed where a complete set of tasks can be analyzed ahead of time. This is accomplished by generating Directed Acyclic Graphs (DAGs) that are used to represent all the operations, data and the relationship between them [1]. To support the in-memory computation feature, Spark uses Resilient Distributed Datasets (RDD). RDD is a read-only data structure maintained in memory to make Spark a fault tolerance framework without having to write to the disk after every operation.

2) Spark Stream Processing Model : In addition to batch processing, Spark provides stream processing abilities with the use of micro-batches. In micro-batching data streams are treated as a group of very small batches which are in turn handled as a regular task by Spark batch engine [1]. Even though this micro-batching procedure works well, it could still lead to some differences in terms of performance as opposed to a true stream processing frameworks.

A. FEATURES

Apache Spark has many distinguishable features. Following is a description of some of these features:

- *Speed:* Apache Spark is a tool that can be used for running Spark applications in Apache Hadoop cluster. Apache Spark is hundred times faster than Apache Hadoop and ten times faster than accessing data from the disk. Spark utilizes the idea of a Resilient Distributed Dataset (RDD), and enables it to distinctly store data inside the memory [9].
- *Usability:* Spark enables users to swiftly write applications in various programming languages such as Java, Scala, R and Python. This helps programmers to develop and run their applications on languages familiar to them which makes it easy to develop parallel applications
- *Advanced analytics:* Besides the simple map and reduce operations, Sparks favors SQL queries, data streaming, and other complicated analytics such as machine learning, and graph algorithms
- *Runs everywhere:* Apache Spark can be run on various platforms such as Apache Hadoop YARN, Mesos, EC2, Kubernetes or in the cloud using the Apache Spark standalone cluster mode. It can retrieve several data information that are HDFS, Cassandra, HBase etc.
- *In-memory computing:* In-memory cluster computation allows Spark to run iterative machine learning algorithms and aids bilateral querying and data streaming analysis at super-fast speeds. Spark keeps data in the RAM of the servers so that the stored data can be accessed quickly
- *Real-time stream processing:* Spark streaming grasps real-time stream processing along with other configurations concluding that spark streaming is simple, fault tolerant and unsegregated [9].

B. USE CASES OF APACHE SPARK

- *Healthcare:* Spark is used in the healthcare sector as it provides a thorough analysis of patient records along with previous medical data. This helps to identify which patients are prone to face health complications in the near future and therefore avoids hospital re-admittance which thereby reduces cost for the hospitals and the patients as it is now feasible to deploy home services for the identified patient [10]. Furthermore, Spark is also used in genomic sequencing as it can reduce the processing time required to process genome data which earlier would take several weeks to organize all the chemical compounds with genes. MyFitnessPal is company that utilizes Spark [11].
- *Finance:* Apache Spark provides insights that help to make correct choices over various issues such as customer segmentation, credit risk assessment and targeted advertising [10]. Financial institutions often use big data to figure out the exact time and location of when the fraud had occurred, so that it can be stopped. Various models are already present which are used to detect fake transactions and a majority of them are deployed in batch environment. With the help of Apache Spark on Hadoop, financial institutions can detect fake transactions in real-time, based on previous fraud footprints [11].
- *E-commerce:* Spark is used in the e-commerce industry to find information concerning real-time transactions that are passed to a streaming clustering algorithms such K-means clustering algorithm or alternating least squares. It also improves the recommendations to customers based on latest trends. Alibaba and eBay are examples of companies that use Spark in e-commerce.
- *Entertainment:* In the gaming industry, Apache Spark helps in recognizing patterns from real-time in-game events and then respond to them to yield fruitful business opportunities such as selective advertising, player retention or the automatic changing of gaming levels based on its difficulty. Furthermore, Spark combined with MongoDB is also used in video sharing websites such as Pinterest, Netflix, and Yahoo. These websites show related advertisements to its users based on the videos viewed, shared, and browsed by the users [10].

C. ECOSYSTEM

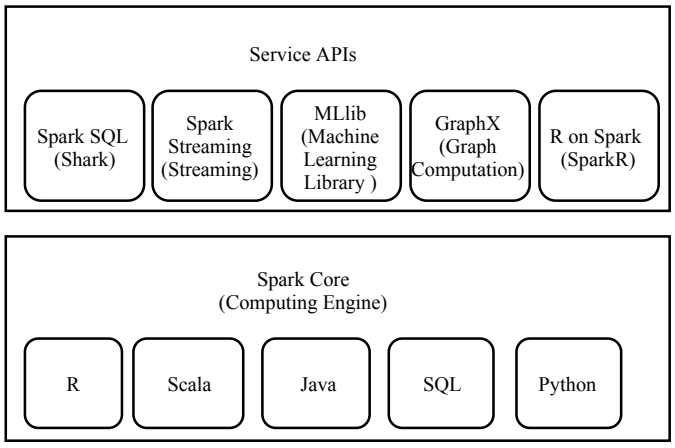

Fig. 1. Apache Spark Ecosystem

The Apache Spark ecosystem consists of the following main components:

- *Spark SQL:* Formerly known as Shark. Spark SQL is a distributed framework that works with structured and semi-structured data. It facilitates analytical and interactive application for both streaming and historical data which can be accessed from various sources such as JSON, Parquet and Hive table [12].
- *Spark Streaming:* It enables users to process streaming of data in real time. In order to perform streaming analysis, Spark streaming enhances the fast scheduling capability of Apache Spark by inserting data into mini batches. An operation known as transformation is then applied to those mini batches that can be easily obtained from live streams and data sources such as Twitter, Apache Kafka, IoT sensors, and Amazon Kinesis.

- *MLlib:* It delivers high-quality algorithms with high speed and makes machine learning easy to use and scale. Several machine learning algorithms such as regression, classification, clustering, and linear algebra are present. It also provides a library for lower level machine learning primitives like the generic gradient descent optimization algorithm. It also provides other functions such as model evaluation and data import. It can be used in Java, Scala, and Python.
- *GraphX:* It is a graph computation engine that enables the building, manipulation, transformation and execution of graph-structured data at a large scale. It consists of various Spark RDD API that facilitates the creation of directed graphs.
- *Spark Core:* Various functionalities of Apache Spark are built on top of the Spark core. It provides a vast range of APIs as well as applications for programming languages such as Scala, Java, and Python APIs to facilitate the ease of development. In-memory computation is implemented in Spark core in order to deliver speed and to solve the issue of MapReduce.
- *SparkR:* It is a package for R that enables data scientists to leverage the power of Spark from R shell. Since DataFrame is the basic data structure for data processing in R, similarly SparkR DataFrame is the fundamental unit of SparkR. It can perform various functions on large datasets such as selection, filtering and aggregation [12].

D. ARCHITECTURE

As illustrated in Fig. 2, the architecture of Apache Spark consists of a master node which has a driver program that is responsible for calling the main program of an application. The driver program is either the code written by the user or if an interactive shell is used, it is a shell. This driver program is responsible for creating Spark context. Spark context behaves like a gateway to all of the functionalities of Apache Spark. It works with the cluster manager that is responsible to manage different jobs [13]. Both Spark context and the driver program collectively handle the execution of the job within the cluster.

The cluster manager first takes care of the resource allocating work. Then the job is split into numerous tasks that is further allocated to the worker or slave nodes. The moment an RDD is created in Spark context, it can be allocated across the different slave nodes and can be cached there too. The slave nodes play a role in executing the tasks that were assigned to them by the cluster manager. They then return these tasks back to Spark context. The executor carries out the execution of the tasks. The lifetime of the executors is the same as Spark. In order to increase the system performance, the number of worker nodes must be increased so that the jobs can be divided further into more number of logical portions [14].

E. HARDWARE REQUIREMENTS

- *Storage Systems:* It is necessary to keep storage systems very close to Spark systems, because most of Spark jobs read input data from an external storage system such as HDFS (Hadoop Distributed File System) or HBase [15].
- *Local Disks:* Even though Spark performs a lot of its computation in memory, it still uses local disks, which does not fit in RAM, to store data and to preserve intermediate output between stages. It is better to have 4-8 disks per node, which are configured without RAID.
- *Memory:* Spark runs well on memory ranging from 8 Gigabyte to hundreds of Gigabyte per machine. It is better to allocate at most 75% of the memory for Spark in all cases, and the rest should be assigned to the operating system and buffer cache.
- *Network:* Numerous Spark applications are network-bound when the data is found inside the memory. These applications can be made faster by using a 10 Gigabit or higher network. This is mainly true for distributed reduce applications such as SQL joins.
- *CPU Cores:* Spark performs minimal sharing between threads and therefore it scales well to tens of CPU cores per machine. At least 8-16 cores per machine must be provisioned. Provisioning the cores depend on the cost of the CPU workload [15].

F. MULTITHREADING AND CONCURRENCY

1) Multithreading: Apache Spark has APIs for many languages such as Scala, Python Java, and R. The most popular use for Spark is with Scala and Python. Choosing which language to use with Spark depends on the features we are interested in utilizing [16]. Regarding multithreading, Python is at a disadvantage compared to Scala since Python does not support multithreading. Scala on the other hand supports multithreading. Having a multithreaded program means we can run more than task at the same time concurrently. A thread is a lightweight process that consumes less memory than the heavyweight process. Creating a thread in Scala can be done in two ways. It can be done by either extending the Thread class or the Runnable interface. The run() method can be used after that [17]. A Scala thread can go through five different states during its lifetime described as follows:

1) New: The initial state of the thread.
2) Runnable: The thread is ready to run but it has not been picked by the scheduler yet.
3) Running: The thread is being executed.
4) Blocked: The thread is waiting for some event such as inputs or resources.
5) Terminated: The thread finished executing.

Furthermore, Scala thread flow can be controlled using different Scala thread methods [17]. For example, we can use sleep() method to put a thread to sleep for a specific period of time and join() method to let the thread wait for another thread to terminate.

2) Concurrency: In concurrency, a task is said to have completed when all the working threads and sub threads are done processing. In Spark, all the tasks run inside the executor JVM. The number of tasks that can run at the same time is controlled by the number of cores which is handled by the

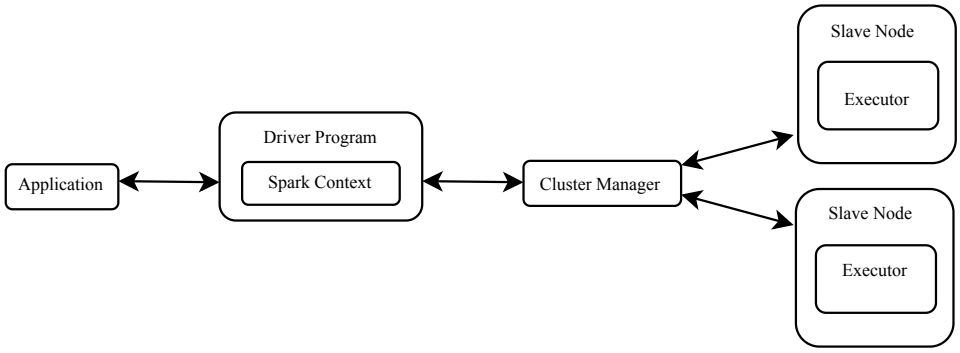

Fig. 2. Apache Spark Architecture

executor. The setting is defined during job submission and is constant in general but can vary if the task uses dynamic allocation. In general, a task is a single thread that runs the serialized code which is written for that particular task. The code within the task is single-threaded and synchronous except when the code directly indicates that it is not synchronous [18]. As Scala runs on JVM, it has full access to all its multi-threading capabilities. However, unlike Java, Scala is not by default just limited to the concept of threads for achieving concurrency. It provides other advance options that can achieve concurrency such as Futures and Actors. On the contrary, both Python and R languages do not support true concurrency and multi-threading. Multi-threading can only run in parallel for some I\O tasks, but can only run one at a time for CPU-bound multiple core tasks. Therefore, more overhead is produced in managing memory and data [19] .

G. APACHE SPARK IN EMERGING TECHNOLOGIES

1) Fog Computing: Fog computing requires extremely low latency, parallel processing of machine learning and complex graph analytical algorithms that is provided by Apache Spark. Spark streaming along with MLlib and Apache Kafka forms the base of a fake financial transaction detection. Credit card transactions of an individual can be obtained to classify the individuals spending patterns. Models can be further formed and trained to forecast any abnormality in the card transaction and along with the Kafka and Spark streaming in real time. Spark can also be used in interactive analysis since it is extremely fast as compared to MapReduce that provide tools like Pig and Hive for interactive analysis [20].

2) Machine Learning: Apache spark has a highly powerful API for machine learning applications known as MLlib that consists of several machine learning algorithms. For instance, we can use Support Vector Machine (SVM) in Spark. SVM is a machine learning algorithm used for classification and regression analysis. The only optimizer available for SVM in Spark is the SGD optimizer [21]. Furthermore, Spark also supports another machine learning algorithm called XGBoost or eXtreme Gradient Boosting. This algorithm enables the users to build a unified pipeline by embedding XGBoost into the data processing system which is based on Apache Spark [22].

Another example of using machine learning in Spark is Deep Learning (DL). Since DL is computationally very heavy, distributing its processes is a good solution and Apache Spark is one of the easiest way to implement it. DL can be implemented in Apache Spark in many ways, some examples are as follows [23]:

- Elephas uses Distributed DL with Keras and PySpark
- Yahoo Inc. uses TensorFlowOnSpark
- CERN uses Distributed Keras
- Qubole uses tutorial Keras and Spark

Furthermore, Deep Learning Pipelines is an open source library which provides high-level APIs that can perform scalable deep learning in Python with Apache Spark. Some of the advantages of this library are [23]:

- With the help of Spark's ML libraries, we can have easy-to-use APIs that produce deep learning within few lines of code.
- It does not compromise the performance while focusing on ease of use and integration.
- As it is built by the creators of Apache Spark it has a greater chance to merge as an official API.
- As Python is the language used it makes it easier to integrate with all of its famous libraries. Moreover, it uses the power of the two main libraries for Deep Learning which are TensorFlow and Keras.

IV. CONCLUSION

Big data is a term that refers to an excessively large amount of datasets that are used to computationally reveal patterns and trends. In order to analyze and find knowledge from this bulk of data, a processing framework is required. There are various types of commonly used big data frameworks such as Apache Hadoop, Apache Storm, Apache Spark, Apache Flink etc. In this paper we talk about Apache Spark's batch processing and stream processing abilities, use cases, ecosystem, architecture, multi-threading and concurrency capabilities and lastly the use of Spark in emerging technologies.

REFERENCES

[1] J. Ellingwood, "Hadoop, storm, samza, spark, and flink: Big data frameworks compared," Website, 10 2016. [Online]. Available: https://www.digitalocean.com/community/tutorials/hadoop-storm-samza-spark-and-flink-big-data-frameworks-compared

[2] M. Zaharia, R. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, and I. Stoica, "Apache spark: a unified engine for big data processing," *Commun. ACM*, vol. 59, pp. 56–65, 2016.

[3] S. Salloum, R. Dautov, X. Chen, P. X. Peng, and J. Z. Huang, "Big data analytics on apache spark," *International Journal of Data Science and Analytics*, vol. 1, no. 3, pp. 145–164, Nov 2016. [Online]. Available: https://doi.org/10.1007/s41060-016-0027-9

[4] M. Iqbal and T. Soomro, "Big data analysis: Apache storm perspective," *International Journal of Computer Trends and Technology*, vol. 19, pp. 9–14, 01 2015.

[5] S. Sarraf and M. Ostadhashem, "Big data application in functional magnetic resonance imaging using apache spark," in *2016 Future Technologies Conference (FTC)*, Dec 2016, pp. 281–284.

[6] S. Gopalani and R. Arora, "Comparing apache spark and map reduce with performance analysis using k-means," *International Journal of Computer Applications*, vol. 113, pp. 8–11, 03 2015.

[7] D. García-Gil, S. Ramírez-Gallego, S. García, and F. Herrera, "A comparison on scalability for batch big data processing on apache spark and apache flink," *Big Data Analytics*, vol. 2, no. 1, p. 1, Mar 2017. [Online]. Available: https://doi.org/10.1186/s41044-016-0020-2

[8] B. Akil, Y. Zhou, and U. Rhm, "On the usability of hadoop mapreduce, apache spark apache flink for data science," in *2017 IEEE International Conference on Big Data (Big Data)*, Dec 2017, pp. 303–310.

[9] V. S. Jonnalagadda, P. Srikanth, K. Thumati, and S. H. Nallamala, "A review study of apache spark in big data processing," *International Journal of Computer Science Trends and Technology (IJCST)*, vol. 4, no. 3, pp. 93–98, Jun 2016.

[10] "Apache spark use cases in real time," Website, 11 2018. [Online]. Available: https://data-flair.training/blogs/spark-use-cases/

[11] "Top 5 apache spark use cases," Website, 6 2016. [Online]. Available: https://www.dezyre.com/article/top-5-apache-spark-use-cases/271

[12] "Spark notes for beginners & experienced," Website, 6 2016. [Online]. Available: https://data-flair.training/blogs/spark-notes/

[13] Naveen, "Apache spark architecture," Website, 2 2017. [Online]. Available: https://intellipaat.com/blog/tutorial/spark-tutorial/spark-architecture/

[14] N. Vaidya, "Apache spark architecture spark cluster architecture explained," Website, 5 2019. [Online]. Available: https://www.edureka.co/blog/spark-architecture/

[15] "Hardware provisioning," Website. [Online]. Available: https://spark.apache.org/docs/2.1.0/hardware-provisioning.html

[16] P. Gandhi, "Apache spark : Python vs. scala," Website, 2018. [Online]. Available: https://www.kdnuggets.com/2018/05/apache-spark-python-scala.html

[17] "What is scala thread & multithreading: File handling in scala," Website, 9 2018. [Online]. Available: https://data-flair.training/blogs/scala-thread/

[18] R. Spitzer, "Concurrency in spark," Website, 2 2017. [Online]. Available: http://www.russellspitzer.com/2017/02/27/Concurrency-In-Spark/

[19] S. Vaid, "Choosing the right programming language for machine learning algorithms with apache spark," Website, 6 2018. [Online]. Available: https://blogs.opentext.com/choosing-the-right-programming-language-for-machine-learning-algorithms-with-apache-spark/

[20] N. Kumar, "Apache spark use cases & applications," Website, 6 2019. [Online]. Available: https://www.knowledgehut.com/blog/big-data/spark-use-cases-applications

[21] S. Yasrobi, J. Alston, B. Yadranjiaghdam, and N. Tabrizi, "Performance analysis of sparks machine learning library," *Transactions on Machine Learning and Data Mining*, vol. 2, pp. 67–77, 2017.

[22] J. Brownlee, "A gentle introduction to xgboost for applied machine learning," Website, 8 2016. [Online]. Available: https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

[23] F. Vzquez, "Deep learning with apache spark part 1," Website, 4 2018. [Online]. Available: https://towardsdatascience.com/deep-learning-with-apache-spark-part-1-6d397c16abd