

Big Data analysis: Apache Spark Perspective

Pankaj Lathar

Department of Information Technology, C.B.P. Government Engineering College, Jaffarpur, New Delhi 110073,
drpankajlathar@gmail.com

Abstract-*The development of Data volumes in industry and research postures colossal openings, and in addition gigantic computational difficulties. As information sizes have outpaced the capacities of single machines, clients have required new frameworks to scale out calculations to different hubs. Accordingly, there has been a blast of new bunch programming models focusing on assorted registering workloads. At to start with, these models were moderately specific, with new models produced for new workloads; for instance, MapReduce bolstered clump handling, and however Google additionally created Dremel for intuitive SQL inquiries and Prege for iterative diagram calculations. In the open source Apache Hadoop stack, frameworks like Storm1 and Impala are likewise particular. Indeed, even in the social database world, the pattern has been to move far from "one-estimate fits-all" frameworks. Lamentably, most huge information applications need to join a wide range of preparing composes. The very idea of "enormous information" is that it is assorted and chaotic; a run of the mill pipeline will require MapReduce-like code for information stacking, SQL-like inquiries, and iterative machine learning. Particular motors would thus be able to make both many-sided quality and wastefulness; clients must join together dissimilar frameworks, and a few applications just can't be communicated proficiently in any motor. Start has a programming model like MapReduce yet broadens it with an information sharing reflection called "Strong Distributed Datasets," or RDDs.*

Keywords: *Apache, Big Data, Map reduce, Hadoop*

I. INTRODUCTION

Spark's generality has several important benefits. Initially, applications are less demanding to create in light of the fact that they utilize a bound together API. Second, it is more effective to consolidate handling undertakings; though earlier frameworks required written work the information to capacity to pass it to another motor, spark can run different capacities over similar information, regularly in memory (Zaharia et al. 2016). Apache start: a brought together motor for huge information preparing. Interchanges of the ACM. At last, Spark empowers new applications, (for example, intelligent questions on a chart and gushing machine realizing) that were impractical with past frameworks. One capable similarity for the estimation of unification is to contrast cell phones with the different versatile gadgets that existed before them, (for example, cameras, cell phones, and devices). In bringing together the elements of these gadgets, cell phones empowered new applications that consolidate their capacities, (for example, video informing and Waze) that would not have been conceivable on any one gadget (Alsheikh et al. 2016). Versatile enormous information examination utilizing profound learning and apache start. IEEE organize. These days, a ton of information is being produced from different sources known as Big information (Manyika et al. 2011).

Big data: The next frontier for innovation, competition, and productivity. These data sets can be either in a structured or unstructured form. Big data are mostly used by the companies for making decisions, analysis purpose etc. The drawback with the traditional methods for handling such data sets was: storage, processing etc. was not possible.

Most of the industries nowadays are using the Hadoop framework for processing, analyzing, storing etc. their data sets. The MapReduce model which is present in the Hadoop framework makes the computation of results faster, reliable etc. The main problem with this method was that it took a lot of time in processing such huge data sets. In terms of time needed to wait for the queries to be sent and the time needed for running these programs (Dittrich et al. 2012). Efficient big data processing in Hadoop MapReduce. Proceedings of the VLDB Endowment, 2014-2015.

Spark can capture a wide range of processing workloads that previously needed separate engines, including SQL, streaming, machine learning, and graph processing. These implementations use the same optimizations as specialized engines (such as column-oriented processing and incremental updates) and achieve similar performance but run as libraries over a common engine, making them easy and efficient to compose. Rather than being specific. (Armbrust et. al. 2015).

Some of the drawbacks with Hadoop were:

Could not be used for Machine Learning Algorithms (Iterative Algorithms).

Application Programming Interfaces (API) options were limited.

Based only on MapReduce model.

Output obtained from one task could not be easily sent as an input to the next task.

The problems of Hadoop were overcome by the introduction of Spark.

II. APACHE SPARK

Apache Spark is used to speed up the computation process in Hadoop. Spark has got its own cluster management computation. Thus does not depend on Hadoop. It uses Hadoop for two purposes: storage as well as processing. Mostly used for storage activities only (Shoro et al. 2015). An important feature of Apache Spark is its “in-memory cluster computation” which basically helps in the increase of speed during computation process.

Apache Spark supports batch applications, interactive queries, iterative algorithms etc.

Evolution

- Developed in 2009 by Materi Zaharia in UC Berkeley's Lab.
- In 2010, Spark became open sourced under BSD license.
- Apache Foundation adopted Spark in 2013 .
- Spark has now become the top level project in Apache Foundation from February 2014 onwards.

Apache Spark Features

- Different kinds of operations are supported by Spark when compared to the MapReduce functions.
- Speed - In Hadoop cluster, an application could run 100 times faster in memory and 10 times faster on the disk. The intermediate results are stored in the memory rather than in the disk. Half of the data will be stored in the memory and the rest in the disk.
- Multiple programming languages such as Java, Python etc API's are supported by Apache Spark.
- Very well integrated with SQL, HDFS, NoSQL etc.
- Help in the decision making of real-time data.
- Supports Streaming data, Iterative Algorithms like Machine Learning, SQL queries etc Zaharia et al. (2010).

Different Ways of Implementing Spark on Hadoop

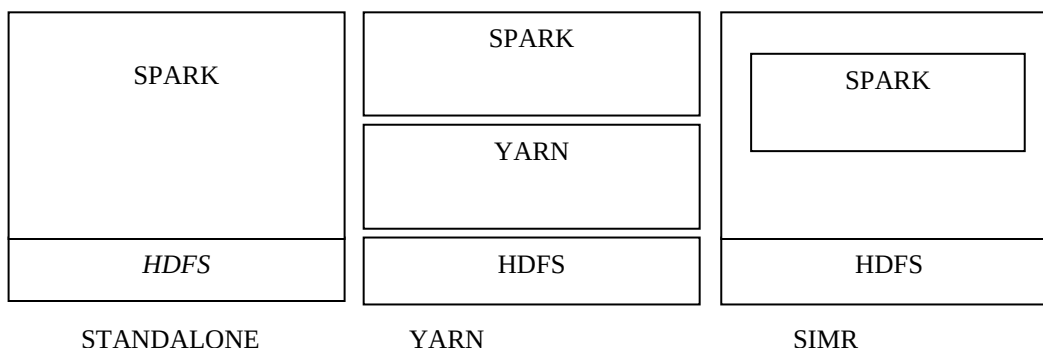


Figure 1: SPARK implemented on different platforms

- In Standalone deployment, Spark is placed on top of the HDFS. Space is given explicitly for the HDFS. Spark and MapReduce run in parallel, thus making all the spark jobs in the cluster completed.
- In Hadoop YARN deployment, Spark runs on top of YARN. It does not require any pre-installation to be done. With this deployment, it helps Spark to be easily integrated with Hadoop stack.
- In Hadoop MapReduce deployment users can easily start with Spark and also use its shell without the requirement of administrative access (Reyes-Ortiz et al. 2015).

Spark Ecosystem

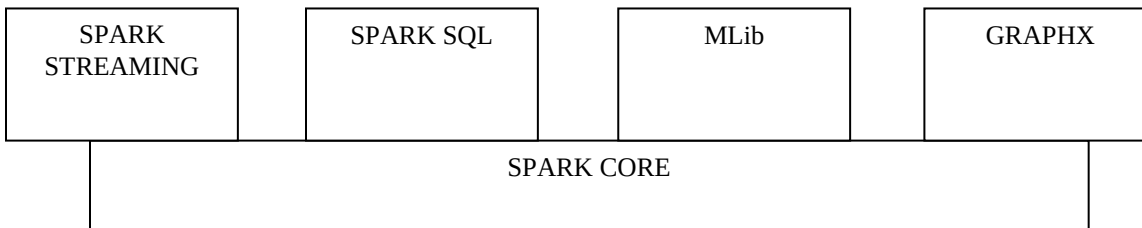


Figure 2: Components of Apache Spark

Apache Spark comprises of the following components

- *Spark Core* consists of the basic I/O functionalities. It includes components for memory management, scheduling tasks, fault recovery systems etc. Supports the API that defines the RDD (Resilient Distributed Data Sets). Supports in-memory computing.
- *Spark SQL* supports the user to work with structured data and is included in the Spark Package. The user can write queries using SQL or HQL (Hive Query Language). Also supports many data sources such as the Parquet, JSON etc.

Intermixing of SQL queries along with program data manipulations is supported by RDD in Java, Python, and Scala etc. SparkSQL was included in Spark V1.0 older version of SQL in Spark was SHARK Shyam et al. (2015).

Spark Streaming supports streams of live data to process in a fast, reliable manner.

Such data is usually obtained from log files, query messages send to the web server to know the status and updates of the server etc. Also supports API's for the streaming of data which are similar to the API of the RDD.

Thus making the manipulation of the data stream in application easier in the memory, disks etc. The developers can reuse the same code for batch processing and streaming data thus making the integration of streaming data with historical data possible and provides fault-tolerance, scalability, a throughput of the same degree of Spark Core.

Some of the Applications using Spark Streaming are:

- Online Advertisements & Campaigns
- Finance
- Intrusion Detection Systems.

Spark provides a low-level machine learning library with the help of MLlib. Easy to use, scalable, compatible with the different programming language and very well integrated with various tools available.

Algorithms such as K-means clustering, linear regression, a classification such as Naive Bayes, collaborative filtering etc. can be implemented using MLlib. This library is nine times faster than the Apache Mahout (disk based). According to a Survey conducted by Data Bricks in 2015, it is seen that use of Spark Mlib component has increased from 11% to 15%.

III. USE OF MLIB BY DATA SCIENTISTS

- A simple API is provided by MLlib to the data scientists who use the data programming languages such as Python and R.
- Also provides a standard component “Spark Framework” which helps the data scientists to develop their own applications using R, Scala, and Python etc.
- The data problems can be iterated 100x faster by the data scientists when compared to using the MapReduce model.

The code developed for machine learning can be run on a big cluster as well as on the PC without any breakdown.

Some of the Applications that uses MLlib

- Advertising Optimization - Used to calculate the number of clicks obtained on an advertisement by the user.
- Fraud Detection – Keep track of the behavior of the users.
- Supply Chain maintenance and optimization.
- GRAPHX is an API that is present on top of Spark. Manipulation of graphs and computations can be done using GraphX library. Resilient Distributed Graph (RDG) is introduced by Apache Spark which provides abstractions of the Spark RDD's. In RDG, the records are represented as the edges and vertices of the graph.

Using the computational primitives (expressive), data scientists can do various operations on the graph with the help of RDG. With the help of these primitives, developers can develop Page Rank abstraction within 20 lines of the program code.

Some of the Applications that uses GraphX

- Analysis of Social Networks
- Fraud Detection

Resilient Distributed Datasets (RDD)

Resilient Distributed Datasets is a collection of data. Each of these datasets is logically partitioned and can be computed by the different nodes present in the cluster consists of JAVA, Scala objects as well as user-defined classes. These datasets are fault tolerant and they can be only read.

These datasets can be created by two ways:

1. Parallelizing the collections which are present in the driver program.
2. Referencing dataset which may be present in the external storage such as Hbase, HDFS etc.

IV. COMPARISON BETWEEN MAPREDUCE AND SPARK

Iterative Operations using MapReduce

In this framework, the intermediate results obtained during iterations are passed across different stages for processing across different applications. Overhead occurs because of disk replication, disk I/O, serialization. This decreases the performance of the system and makes it slow.

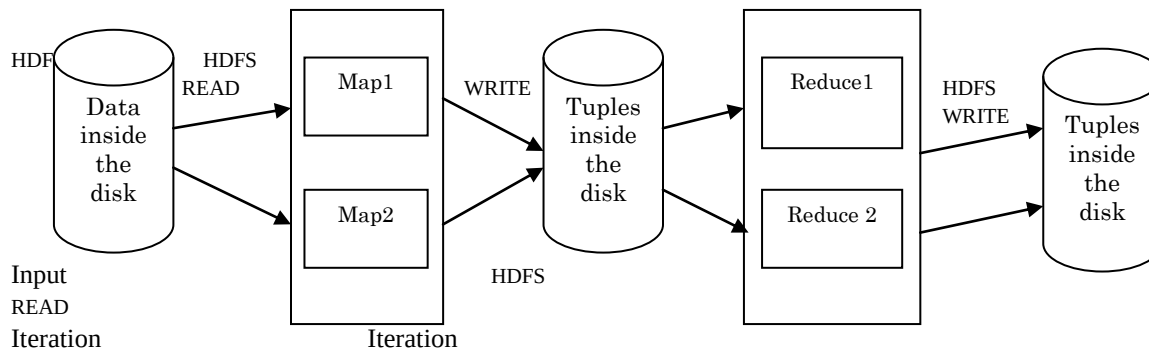


Figure 3: Using MapReduce the data is processed

Iterative Operations Using Spark RDD

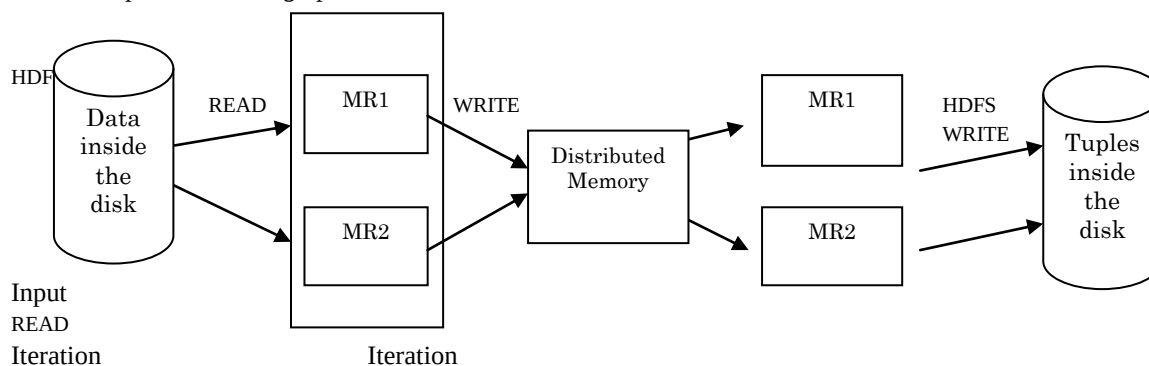


Figure 4: Using Spark RDD the data is processed.

Instead of storing the intermediate results in the disk, the results are now stored in the distributed memory. This makes the system to operate faster. In case, if the data is not stored fully in the memory (RAM) then it is sent to the disk.

Interactive Operations Using MapReduce

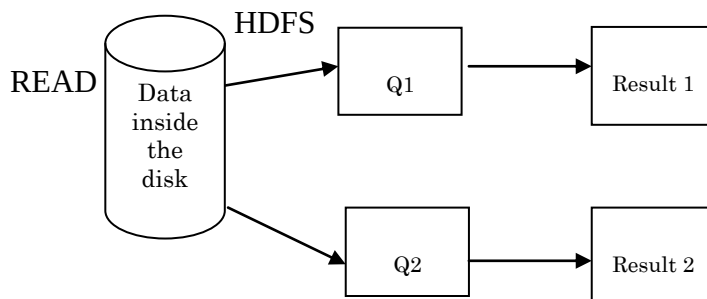


Figure 5: Using MapReduce for Interactive Operations

Users write or run queries on the same set of data. Therefore each of these queries does a disk I/O operation on the storage disks. This lead to more execution time of the application.

Interactive operations using Spark

If the user wants to run different queries on the same set of data, then it is better to store the data in the memory. This reduces the overhead required for the disk I/O operation and hence makes the execution of the application to be done at a faster rate.

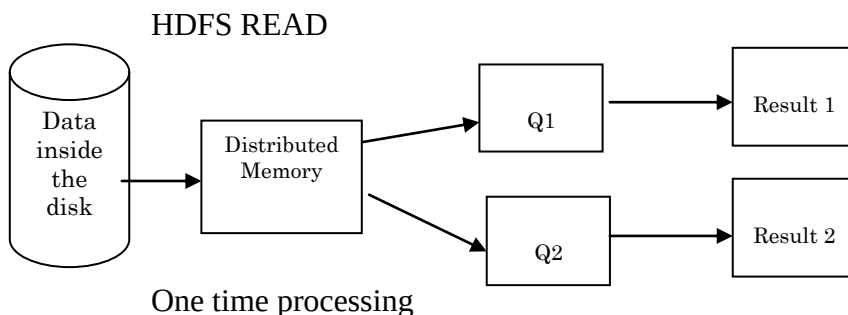


Figure 6: Using SPARK for Interactive Operations

V. PROPERTIES OF RDD IN APACHE SPARK

- *Immutable* – The data records can be shared across different process. Therefore caching, replication, sharing etc. can be done in a faster speed. Data can be created and retrieved anytime and from anywhere. Helps to maintain consistency during computation.
- *Partitioned* – Each set of data records are logically partitioned and are the basic units of parallelism in RDD. Users can develop their own criteria's for partitioning the data.
- *Fault -Tolerant* - As the name says “Resilient” it means the data can be entirely recovered even if a failure occurs.
- *Lazily Evaluated* - Whenever the action is performed for the first time, Spark computes the RDD lazily.
- *Persistence* - The data can be stored either in the memory or disks depending on the operations performed on it.

Operations of RDD in Apache Spark

1. In Transformation operation, a new RDD can be created from an existing RDD using the transformation operation. When a transformation is applied to an existing RDD, a DAG graph is created first. The operations are not applied immediately. The graph keeps growing until any action is specified on it. Therefore, the transformations in spark are mostly said to be lazy.

Some of the transformations and their meanings

- **Map** - By passing elements from the existing source in the function, a new dataset is returned.
- **Filter** -A new dataset is returned, when the function returns a true value for a set of elements from the source dataset.
- **Union (another dataset)** - A new RDD is returned with the elements which are obtained by the union of the source dataset and another dataset passed as an argument.
- **Intersection (another dataset)** – A new RDD is returned with the elements which are obtained by the intersection of the source dataset and another dataset passed as an argument.

2. In Action operation, a new RDD is obtained when an action is performed on the existing RDD (DAG Graph) with the transformations lined up. .

Some of the Actions and their meanings are

- Reduce (func) – When this action is performed, the elements present in the dataset are aggregated i.e. input – consists of two arguments, output- returns a single value. For correct parallel computation, functions like commutative or associative should be used.
- Collect - Elements that are present in the dataset are returned to the driver program as an array. This function is mostly used after the filter function. Dataset size returned is relatively small.
- First - The value of the first element present in the dataset is returned.
- Take(n) – An array containing the first 'n' elements from the source dataset is returned.

In this paper we have implemented a real-time anomaly detection framework using Apache Spark. It has the ability to manage streaming data and guarantees data delivery. It is also fault tolerance and is flexible to increasing parallelism.

Key-Value Pairs

Spark operations usually deal with RDD's containing different object types. There are special operations that are available only for RDD's containing key-value pairs. In scala, the key-value pair operations are already present in the RDD's having the Tuple2 objects. Tuple2 represent (p,q) form. This operation is present in the PairRDD Function class.

Code using the reduce By Key operation of the key-value pairs to count the number of times a line occurs in a text.

```
Val lines1 = sc.textfile("text.txt")

val pair = lines.map( x=> (x,1))

val count = pair.reduceByKey((p,q)=> p+q)
```

SHUFFLE

SHUFFLE Supports re-distributing of data across the partitions which does the grouping of data differently and involves copying of data across different machines, thus making the shuffle operation costly and more complex.

Let us consider the Reduce By Key Operation. When this operation is applied on an existing RDD, new RDD is formed. It contains a tuple where all the values of a single key is combined.

Challenge is that values for the single key are sometimes obtained from different partitions. The solution is to co-locate all of them to obtain the result.

In Spark, usually the data is not available across all the partitions. Usually a single task does its computation on one partition. Therefore it is necessary to obtain the values for all the keys, merge them together to obtain a single result for each of the keys. This process is known as Shuffle.

In the newly shuffled data, the order of the partitions and the order of the elements will not be same. Therefore they can be ordered by using:

Map Partitions – used for sorting the partitions.

Sort By – get a sorted order of RDD's.

VI. PERFORMANCE ANALYSIS

The shuffle operation includes disk I/O and data serialization. For this operation, data must be organized. This is done with the help of 2 tasks:

1. Map task -Organizes data
2. Reduce task- Aggregation of data

The results obtained from these individual tasks are placed in the memory. Then they are sorted based on the partition specified and is send to a single file as an output. In the reduce side. The tasks read the sorted blocks .Large amount of heap memory is consumed as the processes built a in-memory data structure before transferring them. When the data structures are overloaded with data, they are split across the disks.

Write a program in Scala to implement Word Count.

Create a text file named Input.txt people application

Write a Scala program containing the main function of the transformations and actions to be performed. Save it as PgWordSpark.scala

Flat Map() - tokenizes the lines to words. Map () - Used for calculating word frequency.

Reduce By Key() - Used for counting number times a word is repeated. In the terminal execute all these steps in the application directory of spark.

Spark JAR file should be downloaded. Can be done from the official download page. Next step is the program should be compiled.JAR file of the application should be created say for example Sparkspword.jar .Spark Application needs to be submitted using the commands in spark.

Outfile displays the output present in the application directory of spark. Output will be of the form (application,1)(people, 2)

Limitations of using Apache Spark

- 1) For processing, more memory is required.
- 2) Issues occur due to the over usage of memory is not dealt in a friendly manner.
- 3) Plenty of resources is needed for processing purpose.

Real Time use Cases of Apache Spark

- In Retails, Spark can be used to analyze usage of coupons.
- In Health , using Spark Core, SQL ,Spark Streaming a HealthCare System can be built,
- In Websites, Spark's Machine learning capability can be used to detect fake profiles and also provide matches of product to the customers interest.
- In Airlines ,models can be built to predict the travel delay of airlines

VII. CONCLUSION

Scalable data handling will be important for the succeeding generation of computer applications but typically involves a complex sequence of processing steps with different computing systems. To streamline this errand, the Spark venture presented a bound together programming model and motor for huge information applications. Our experience shows such a model can effectively bolster the present workloads and conveys considerable advantages to clients. We trust Apache Spark features the significance of compos capacity in programming libraries for huge information and supports advancement of all the more effortlessly interoperable libraries. In this paper we have executed an ongoing abnormality recognition system utilizing Apache Spark. It can oversee gushing information and ensures information conveyance. It is additionally adaptation to non-critical failure and is adaptable to expanding parallelism.

REFERENCES

1. Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., ... & Ghodsi, A. (2016). Apache spark: a unified engine for big data processing. *Communications of the ACM*, 59(11), 56-65.
2. Kuhn, K. A., & Giuse, D. A. (2001). From hospital information systems to health information systems. *Methods of information in medicine*, 40(04), 275-287.
3. Alsheikh, M. A., Niyato, D., Lin, S., Tan, H. P., & Han, Z. (2016). Mobile big data analytics using deep learning and apache spark. *IEEE network*, 30(3), 22-29.
4. Alsheikh, M. A., Niyato, D., Lin, S., Tan, H. P., & Han, Z. (2016). Mobile big data analytics using deep learning and apache spark. *IEEE network*, 30(3), 22-29.
5. Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Byers, A. H. (2011). Big data: The next frontier for innovation, competition, and productivity.
6. Dittrich, J., & Quiané-Ruiz, J. A. (2012). Efficient big data processing in Hadoop MapReduce. *Proceedings of the VLDB Endowment*, 5(12), 2014-2015.
7. Armbrust, M., Xin, R. S., Lian, C., Huai, Y., Liu, D., Bradley, J. K. & Zaharia, M. (2015, May). Spark sql: Relational data processing in spark. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*(pp. 1383-1394). ACM.
8. Shoro, A. G., & Soomro, T. R. (2015). Big data analysis: Apache spark perspective. *Global Journal of Computer Science and Technology*.
9. Talia, D., Trunfio, P., & Marozzo, F. (2015). *Data analysis in the cloud: models, techniques and applications*. Elsevier.
10. Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., Mccauley, M., ... & Stoica, I. (2012). Fast and interactive analytics over Hadoop data with Spark. *Usenix Login*, 37(4), 45-51.
11. Gopalani, S., & Arora, R. (2015). Comparing apache spark and map reduce with performance analysis using k-means. *International journal of computer applications*, 113(1).
12. Gopalani, S., & Arora, R. (2015). Comparing apache spark and map reduce with performance analysis using k-means. *International journal of computer applications*, 113(1).
13. Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., ... & Ghodsi, A. (2016). Apache spark: a unified engine for big data processing. *Communications of the ACM*, 59(11), 56-65.