# Comparison between Hadoop and Spark

**Houssam BENBRAHIM, Hanaâ HACHIMI and Aouatif AMINE**
GS Laboratory "LGS", BOSS-Team, National School of Applied Sciences, Ibn Tofail University, Kenitra, Morocco.
houssam.benbrahim@uit.ac.ma, hanaa.hachimi@univ-ibntofail.ac.ma and amine_aouatif@univ-ibntofail.ac.ma

## Abstract

Big Data is a technology that aims to capture, store, manage, and analyze large numbers of data with different types: structured, semi-structured and unstructured. These data are guided by the rule of the 5 Vs: Volume, Velocity, Variety, Veracity and Value. To analyze a large amount of information coming from several sources, the technological world of Big Data is based on clearly identified tools, including the Hadoop Framework and the Apache Spark. Hadoop allows massive data storage with the Hadoop Distributed File System (HDFS) model, as well as the analysis with the MapReduce model, on a cluster that has one or more machines. Apache Spark analysis the distributed data, but it doesn't contain a system storage. This article presents a comparative between the Big Data analysis methods offered by Hadoop and Spark, their architectures, operating modes and performances. Indeed, We have first begun by a general overview of Big Data technology. We discuss about the Hadoop Framework and their components: HDFS and MapReduce the first element of our comparison, we offer a study of their methodology, their mode of data processing and their architecture, at the end of this section we present the Hadoop Ecosystem. We then continue with Apache Spark Framework the second element of our study, we expos their features, their Ecosystem and their mode of analysis of large data. At the end, we compare these last two elements Hadoop and Spark based on a detailed study.

## Keywords
Big Data, Hadoop, Spark, HDFS, MapReduce, Cluster, Analysis methods, Comparison.

## 1. Introduction

The Big Data term origins was raised by the Meta Group research firm (now Gartner) in 2001 (Laney, 2001). This concept has profoundly transformed our society and it has became a new technology focus both in science and in industry. It is obvious that we are living at the era of the data deluge, evidenced by the huge volume of data collected from different sources and by the growth rate of the data generated (Corp, 2014). The question is: Where does this data come from?

The movement began in the 2000s with the birth of Web 2.0, the emergence of social networks, the development of smartphones and continued with the orientation towards connected objects (Institut Montaigne, 2015). All messages, all documents, all images and videos shared, for example on Facebook, Twitter or Google represent a data. The IDC (International Data Corporation) report (Gantz & Reinsel, 2011) predicts that, from 2005 to 2020, the overall volume of data can grow from one hundred and thirty exabytes to forty thousand exabytes, representing double growth every two years.

The Big Data approach has a strong impact on many sectors: finance, health, public sector and more (Corp, 2016). In fact, the data are presented as a new gold, which should be discovered and exploited, for instance, the Mckinsey report (McKinsey & Company, 2011) states that the potential value of global personal location data is estimated to be $100 billion in revenue to service providers over the next ten years and be as much as $700 billion in value to consumer, business and users.

Today, the biggest problem is the analysis of mass data with all the constraints and difficulties surrounding it. This challenge offers new opportunities in terms of information management, and understanding of structured data and unstructured allows improved the results directly.

The area of data analysis is based on a powerful technology, at the heart of them, we find an framework that known as the Hadoop MapReduce. This module works in a parallel environment that performs advanced processing functions in a short time. Introduced by Google in 2004 (Dean & Ghemawat, 2004), MapReduce help programmers to perform a data analysis, delegated and managed this data under an architecture, which can include thousands of computers running at the same time, this is called a «cluster» (LeMagIT/TechTarget, 2014). Also, we have Spark a Big Data processing framework, it is an open source project of the Apache Foundation. This module built to perform sophisticated analysis and designed for speed and easy of use (Apache Spark). Apache Spark offers a complete and unified ecosystem to meet the needs of Big Data processing for various data sets, various in nature (text, graph, ...) as well as various in source (batch, streaming or real-time) (What is Apache Spark?). In this paper, we start with the Big Data definition and their 5Vs properties. We continue with the Hadoop technology storage and analysis. Next, we discuss about the Apache Spark and their mode of treatment. Following that, we present a comparison between the previous two technologies Hadoop and Spark. Finally, we outline a brief conclusion and suggestions for further research.

## 2. Big Data: definition and characteristics

### 2.1 Definition

Firstly, it is sometimes difficult to agree a single definition of Big Data. In 2011, the McKinsey and Company firm defines Big Data as follows: “datasets whose size is beyond the ability of typical database software tools to capture, store, manage, and analyze.” (McKinsey & Company, 2011). Likewise, IDC defines Big Data with: “ Big data technologies describe a new generation of technologies and architectures, designed to economically extract value from very large volumes of a wide variety of data, by enabling high-velocity capture, discovery, and/or analysis.” (Gantz & Reinsel, 2011). Also, IBM defines Big Data by: “ Big data is being generated by everything around us at all times. Every digital process and social media exchange produces it. Systems, sensors and mobile devices transmit it. Big data is arriving from multiple sources at an alarming velocity, volume and variety. To extract meaningful value from big data, you need optimal processing power, analytics capabilities and skills.”(IBM). However, there are several definitions of Big Data, each of definitions focuses on a specific aspect of this phenomenon.

### 2.2 Characteristics

In the early 2000s, and more precisely in 2001, when the analyst Doug Laney noted in his research by Meta Group (Laney, 2001) that data growth challenges and opportunities are three Vs (Russom, 2011)(Sagiroglu & Sinanc, 2013)(McAfee & Brynjolfsson, 2012)(Besse & Loubes, 2014):

- Volume: that we are talking about a large volume of data. Data is growing exponentially, causing forecasters to mention 40 zettabytes will be generated in 2020 (EMC Corporation). This data are collected from a variety of sources like computers, smartphones connected to the internet, the access to social networks, the interconnected objects, etc. With this large volume generated, the challenge is to store and analyze this Big Data in real time (Corp, 2015).
- Velocity: this means processing of huge data in a short time, sometimes in real time, with very advanced technologies like Hadoop and Spark. The large volume of data collected have added value that quickly reduces with time. They have no value if they are not fast used.
- Variety: all types and forms of data (structured or unstructured). The data comes in various formats: text document, email, picture, audio, video, location data, log files and more. The data collected are very diverse in nature.

These three Vs: Volume, Velocity and Variety, are the original properties of Big Data. The last two new "V" are (Demchenko, Ngo, & Membrey, 2013)(Teboul & Berthier, 2015)(SAS Company):

- Value: the exploitation of data to draw information. What we want is turning the data into value, this fourth property is the most important V in the era of Big Data. This means money, and the recovery of benefits for businesses.
- Veracity: refers to the reliability of data, with this high volume, velocity and variety of data, the quality and accuracy are less controllable. The uncertain and unpredictable nature of data like abbreviations, typos and colloquial speech that are shared via the internet and social networks are not necessarily correct. Big data

technologies work with this types of data to bring order to this and combine analytic that correspond to user needs.

These five elements are the main keys to understanding the word of Big Data. The goal is to process in a short time a large volume of data, with innovative technologies in order to develop the untapped information. Big data can be described by 5 characteristics (Demchenko, Grosso, De Laat, & Membrey, 2013) represented by the Figure 1.

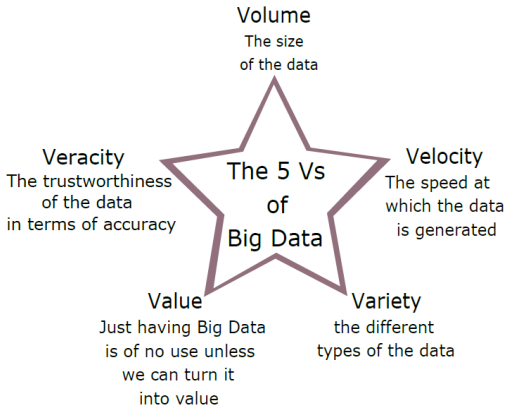


Figure 1. The 5 Vs of Big Data

## 3. Hadoop Framework

### 3.1 Hadoop Core

Apache Hadoop is an open-source software written in Java that supports massive data storage and processing on a cluster which contains several machines (Apache™ Hadoop). In 2004, Google has published a research article on the algorithm MapReduce, designed to achieve of the analytical operations to large scale on a multiple servers, as well on the file system in the cluster, Google File System (GFS) (Dean & Ghemawat, 2004). Doug Cutting, who worked on the development of the search engine free Apache Lucene has inspired the concepts described in Google article and decided to replicate in open source the tools developed by Google for its needs (LeMagIT/TechTarget, 2014). Hadoop Core offers a distributed file system that gives high-speed access to application data, from another side, it provides a system for parallel analysis of large data sets (Intel IT Center, 2013).

#### 3.1.1 Hadoop Distributed File System (HDFS)

HDFS is a Java project used to store very large volume of data, structured or not, on a cluster. Like other technologies related to Hadoop, HDFS has become a key tool for managing Big Data pools and supporting analytic applications. HDFS is designed to ensure data security by replicating the data written to the cluster multiple times, it means that the data is pipelined between various DataNodes (LeMagIT/TechTarget, 2014).

HDFS has a master/slave architecture with one NameNode as a master and several DataNodes as a slaves. The NameNode stores the informations of the file and their characteristics centralized way. It also supports the DataNodes that stores the content of the files, fragmented into blocks (64MB by default) (Borthakur, 2008).

HDFS facilitates the work with the parallel mode, that is when running a query, for example, each node will run it with its data available. The NameNode is the arbitrator and repository for all HDFS metadata. HDFS Client read and write requests are in the responsibility of the DataNodes, in addition to these, they perform block creation, deletion and replication instructed by the NameNode (Borthakur, 2008). The Figure 2 shows the architecture of HDFS in a cluster based on Hadoop.

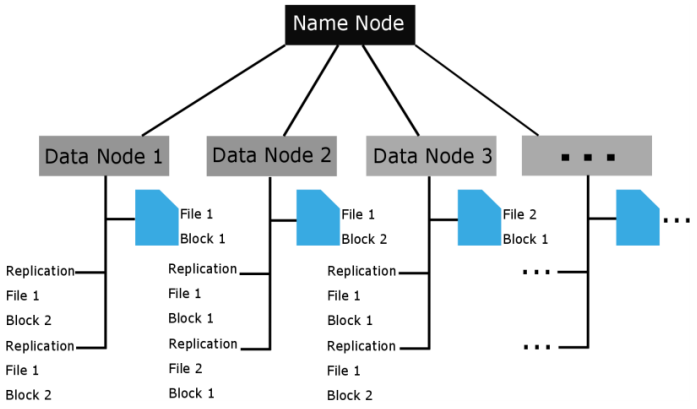


Figure 2. HDFS architecture

### 3.1.2 MapReduce

The second important component of Hadoop is MapReduce, which was introduced by Google. It's a programming model and an associated implementation for processing and generating large data sets in a cluster. The MapReduce algorithm is based on two functions which are: Map() function and Reduce() function (Schneider, 2015).
The calculation takes a group of input key/value pairs, and produces a group of output key/value pairs. The Map() function transform the input data into a series of pairs Key/value (White, 2012), and it will group the data by associating them with keys, which are chosen from the key/value pairs have a meaning withe the work to be solved. Moreover, this operation must be parallelizable: it must be possible to splitting the input data into multiple fragments, and executing the operation Map() to each cluster machine on a separate fragment. The Reduce() function apply a processing to all the values produced by the Map() operation, and at the end of the Reduce() operation, we will have a result for each of the distinct keys. Here, each cluster machine will be assigned one of the unique key produced by Map(), and it lists the values associated with the key, also will then perform the Reduce() operation for that key. Finally the desired result is produced in an output file (White, 2012). The Figure 3 represents the execution of the MapReduce Job.

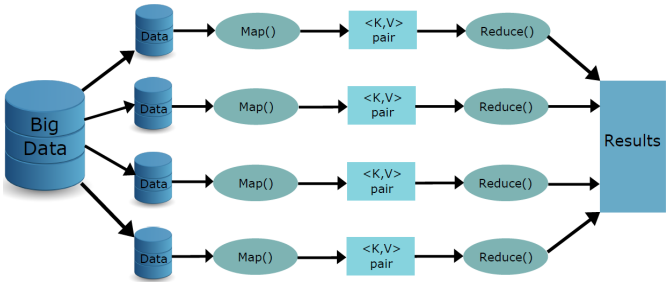


Figure 3. The MapReduce Job

We try to enumerate all the distinct words of a textual source, with for each of them the number of times they are present within the source. The Map() function sends each word with a number of occurrences found, after that the Reduce() function sums together all counts emitted for a particular word. We can detail this work with the Figure 4.

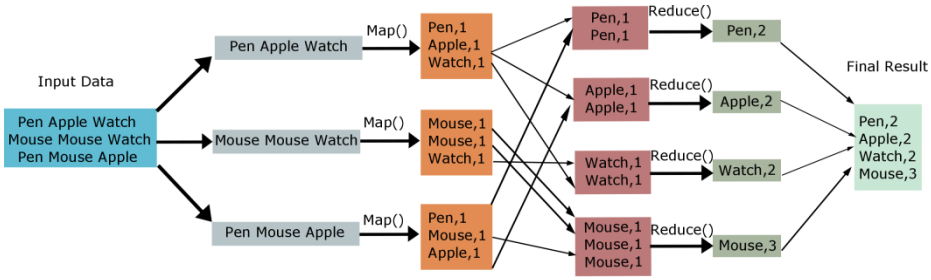


Figure 4. The overall MapReduce word count process

When a MapReduce program is run with Hadoop, the job is sent to the master node which is the JobTracker. It is one on the cluster, the JobTracker receives Map/Reduce tasks and organizes their execution on the cluster. The TaskTracker, multiple on the cluster, runs the Map/Reduce Job itself (in the form of the Map() and Reduce() functions associated with the input data), each one of the TaskTrackers is a computing unit in the cluster. The JobTracker is in communication with HDFS, knows where are stored the input data to run the Map/Reduce programs, and it knows where are stored the output data. So to run a Map/Reduce program, we should write input data into the HDFS, submit the program to the JobTrackers, and retrieve output data from HDFS (LeMagIT/TechTarget, 2014)(Schneider, 2015). The Figure 5 describe the MapReduce architecture in a cluster based on Hadoop.

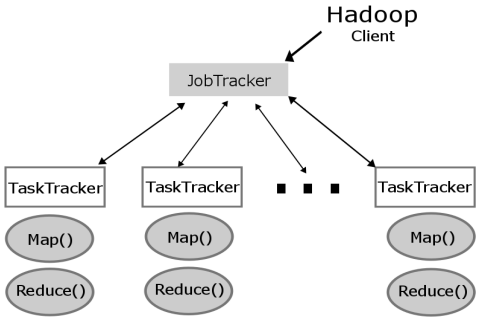


Figure 5. The MapReduce architecture

## 3.2 Hadoop Ecosystem

Hadoop cluster globally can be considered as that the Hadoop Core and the Hadoop Ecosystem, as we have already seen, the Hadoop Core consists on HDFS and MapReduce. Since the beginning of the project, a lot of others softwares have grown up around it, which are broadly used to make Hadoop easy to develop and more usable, this is called the Hadoop Ecosystem (Saldhi, 2014). The following section provides additional informations on the Hadoop components:

- HBase: is an open source, a distributed, and a non-relational database system written in Java. It runs on the top of HDFS, and it uses a not only SQL approach. HBase can serve as the input and the output data for the MapReduce (Apache HBase).
- Pig: is a high-level platform for the MapReduce programs used with Hadoop. It is a data processing system, where the data sets are analyzed and produced with a language called PigLatin (Apache Pig).
- Hive: is a Data Warehouse application that supports the SQL interface and relational model. The Hive architecture is built on the top of Hadoop in order to provide synthesis, querys and analysis via a language called HiveQL (Apache Hive).
- Sqoop: is a platform dedicated to transfer bulk data between relational data-bases and the Apache Hadoop (Apache Sqoop).
- Oozie: is a java based web-application, it streamlines the workflow and it coordinats among the tasks dedicated to the Hadoop. Oozie manages the Hadoop Jobs (Apache Oozie).

- Flume: is a high-level application which used for data streaming which come from several sources (Apache Flume).
- Zookeeper: is a centralized service that provides to maintain configuration information, as well as naming and providing distributed synchronization, likewise adduceing group services (Apache ZooKeeper).

This is not an exhaustive list, but there are many other projects that belongs to the Hadoop Ecosystem.

### 3.3 The Evolution of the Hadoop Ecosystem

Important evolution: the two functionalities of MapReduce are decoupled. The resource management of the cluster is ensured by the new YARN layer (Yet Another Resource Negotiator) and MapReduce is summarized for batch work processing. An improvement that no longer requires to pass by the MapReduce freamwork to access at the data in parallel mode (LeMagIT/TechTarget, 2014). With YARN, MapReduce's resource management and job scheduling/monitoring tasks are separated into two stand-alone daemons. The idea behind the Apache Foundation is to have a global Resource Manager (distributing tasks according to memory, CPU and network criteria) and an application manager, which manages jobs (in the MapReduce sense) (Apache Hadoop YARN).

## 4. Apache Spark

### 4.1 Presentation and history

Apache Spark is an open source Big Data processing Framework. It was originally developed by AMPLab, UC Berkeley University in 2009 and passed as an open source project from Apache Software Foundation in 2010 (Bichutskiy, 2015)(Carlos Hinojosa, 2016). The main feature of Spark is the massive data analysis distributed in-memory that increases the processing speed of an application.
Spark is designed to cover a wide range of workloads such as batch applications, iterative algorithms, interactive queries and streaming. It is a quick and general engine for large-scale traitement (Apache Spark)(What is Apache Spark?)(Abdul Ghaffar Shoro, 2015).

### 4.2 Features of Apache Spark

Apache Spark has a following features (Apache Spark)(What is Apache Spark)(Tutorialspoint, 2015)(Apache Spark | MapR):

- Speed: Apache spark is very fast in memory, it stores input data, output and intermediate data in-memory as Resilient Distributed Datasets (RDDs). Spark is also faster when data is stored on disk, it holds the world record for large-scale on-disk sorting.
- Supports multiple languages: Apache Spark writes applications quickly in Java, Scala, Python or R. Therefore, it can build applications in different languages. The Spark comes up with 80 high-level operators for interactive querying.
- Advanced analytics: Spark comes packaged with higher-level libraries, including support for SQL queries, machine learning (MLlib), stream processing (Spark Streaming), and graph processing (GraphX). These libraries are tightly integrated into the Apache Spark Ecosystem and they can be seamlessly combined to create complex workflows.
- Integration everywhere: Spark runs everywhere. It is designed to run on different platforms and multiple distributed systems. Apache spark runs on Hadoop, Apache Mesos, Standalone, or in the Cloud. It can access to diverse data sources, including HDFS, Cassandra, HBase, and S3. Programmers can start Spark and use their data sources for better use.

### 4.3 Apache Spark Ecosystem

In addition to Spark main APIs, the Ecosystem contains additional libraries that makes it very strong, in order to operate and analyze the Big Data. We can cite the components of the Spark Ecosystem as follows (Bichutskiy, 2015)(Frampton, 2015)(Parziale, Bostian, Kumar, Seelbach, & Ye, 2016):

- Apache Spark Core: is the general execution engine for the Spark platform, that all other features built on it. Spark Core is responsible for basic I/O functionalities, scheduling and monitoring the jobs on spark clusters, task dispatching, networking with different storage systems, fault recovery and efficient memory management. It is provided APIs for a variety of commonly-used languages: R, SQL, Python, Scala, and Java.
- Spark SQL: is a module for structured data processing. It provides a programming abstraction called Data Frames and can also act as distributed SQL query engine. Spark developers can run SQL like queries on Spark data, that is present in RDDs and other external sources.
- Spark Streaming: is enables scalable, high-throughput, fault-tolerant stream processing, while inheriting Spark's ease of use. To process data in real-time Spark Streaming makes use of a continuous stream of input data (Discretized Stream or Stream a series of RDD's).
- MLlib (Machine Learning Library): running on top of Spark, MLlib contains common learning algorithms and utilities, including classification, regression, clustering and more.
- GraphX: is a new component in a Spark for graphs and graph-parallel computation. GraphX comes complete a library of common algorithms.

### 4.4 Resilient Distributed Dataset

One of the main principles of Spark is the use of Resilient Distributed Datasets (RDDs). Resilient means that they can be rebuilt on failure based on the information stored. The RDDs are data distributed over a cluster of machines and stored in memory (RAM), in order to prevent system to read data each time from disk. Whenever data is needed for a calculation, and if the requested partition is too large, the system will fetch some data from disk. It is also possible to create new RDDs by updating others (Zaharia, Chowdhury, Das, & Dave, 2012).

### 4.5 Iterative and Interactive Operations on Spark RDDs

Resilient Distributed Datasets store the data in-memory, so that the user can re-query the subset of data. The Figure 6 describes how RDDs store the intermediate outputs in distributed memory instead of stable storage (Disk), making the system faster (Tutorialspoint, 2015)(Zaharia et al., 2012).

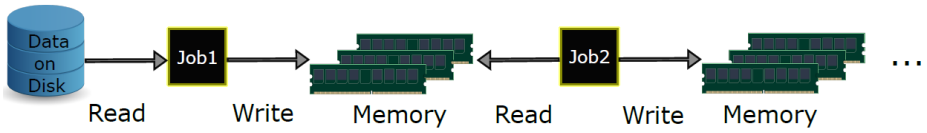


Figure 6. Iterative Processing in Spark

In interactive data applications, users can run different queries on the same subset of data continuously by keeping it in-memory, in order to attain for a minimum execution time (Tutorialspoint, 2015)(Zaharia et al., 2012). The Figure 7 shows how interactive operations are performed on Spark in-memory.

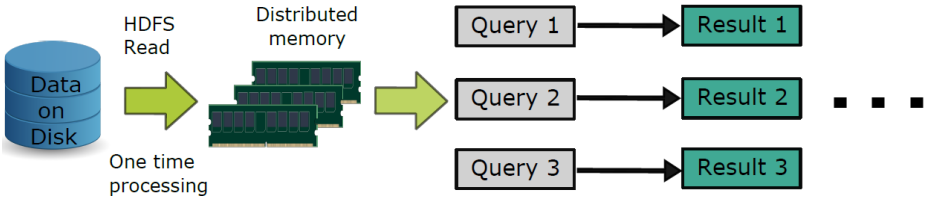


Figure 7. Interactive Operations on Spark RDDs

### 4.6 Spark in Cluster Mode Overview

Apache Spark doesn't comes up with resource management module like YARN, it manages the resources in standalone mode in single node cluster setup. But for distributed cluster mode, Spark can be integrated with resource management modules like YARN or Mesos (Parziale et al., 2016).
The following is a description of the Apache Spark components in Cluster Mode in more details (Bichutskiy, 2015)(Tutorialspoint, 2015)(Parziale et al., 2016):

- Driver Program: the declaration of transformations and the action on the RDDs are made by the Spark Driver, as well as the submission of the requests to the master. Spark Driver runs the Main() function into the application that prepares the SparkContext. In the main program, the SparkContext object orchestrates all of the activities within a Spark cluster.
- Cluster Manager: is an associate external service for getting resources on the cluster. There is a connection between the SparkContext object and the Cluster Manager to allocate resources between applications. On the Spark Cluster, the Spark applications run as subprograms and that each of them independently of the others. Driver Programs manages executor processes via external Resource Manager (like YARN or Mesos).
- Worker Node and Executor: a Worker Node means a node that can run application code in the cluster. An Executor runs tasks and store data across a Worker Node.

The Figure 8 describes how Apache Spark works in the distributed cluster mode.

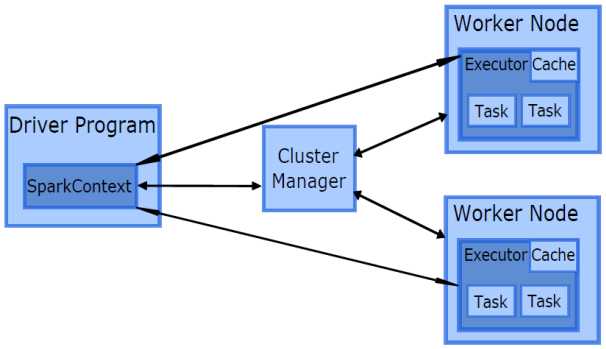


Figure 8. Apache Spark Cluster Overview

## 5. Comparing Hadoop and Spark

### 5.1 Compare Projects

Hadoop and Spark are popular Apache projects in the Big Data Ecosystem. A direct comparison between Hadoop and Spark is difficult because they do the same thing but are also non-overlapping in some areas, for example, Spark has no file management and therefor must rely on HDFS or some other solution.
MapReduce is a startup system for several companies that began their journey in Big Data when Apache Hadoop appeared. This Framework has been the first fruit of the projects which is interested in the analysis of the distributed data in the sense of Big Data. One of the significant challenges for systems developed with this Framework is its high-latency, batch-mode response (Comparing Hadoop, MapReduce, Spark, Flink, and Storm - MetiStream).
Some developers would complain that it is difficult to maintain MapReduce because of inherent inefficiencies in its design and in its code, even if MapReduce has been evolved over the last thirteen years (MapReduce and Spark - Cloudera VISION). On the other hand, Spark is more nimble and better suited for Big Data analytics. It can run on a variety of file systems and databases, Spark processes work 100 times faster than the same process on MapReduce. The Table 1 shows a matrix comparison between Hadoop MapReduce and Spark (Comparison between Apache Spark and Hadoop MapReduce - DataFlair)(Stockinger, 2015).

Table 1. A general comparison between Hadoop MapReduce and Apache Spark

| | Hadoop MapReduce | Apache Spark |
|---|---|---|
| Developed at | Google | UC BerKeley |
| Designed for | Batch processing | Real time and streaming processing. |
| Written in | Java | Scala |
| Intermediate results | Stored in hard disk | Stored in memory |
| Programming Language support | Primarily Java, other languages like C, C++, Ruby, Groovy, Perl, and Python. | Scala, Java, Python, R, and SQL. |
| Storage | Disk only | In-memory or on disk |
| OS support | Supports cross platform | Supports cross platform |
| Speed | Reads and writes from a disk which slows down the processing speed. | Reads and writes from memory which makes processing speed very fast. |
| Real time | MapReduce cannot succeed, because it has been dedicated to performing batch analysis on large amounts of data. | Spark can process real-time data, that means, data from real-time event streams at millions of events per second, sharing / viewing from social networks for example. It is dedicated to the treatment in real-time. |
| Streaming | Can only process data in batch mode. | Spark makes it possible to process data which are fixed at a time T. Thanks for the Spark Streaming module, it is possible to process data streams which arrive continuously, and thus to process these data as they arrive. |
| Difficulty | The developers In MapReduce are needed to hand code each and every operation, which makes MapReduce very difficult to program. | With Resilient Distributed Datasets (RDDs) Spark is easy to program. |

Apache Spark saw tremendous growth compared to Hadoop, Spark's growth comes not only from a huge increase in the number of contributors, devellopers and users but also from increases in usage across a variety of organizations and functional roles. Apache spark is one of the most active Big Data projects (Databricks, 2015), but it is wiser to compare Hadoop MapReduce to Spark, because they're more comparable. The following section provides a comparison between them, we will cover a general differences.

## 5.1 Case studies "Hadoop vs Spark"

Spark can be used to analyze data that is too large. In addition, like Hadoop, Spark can work on unstructured, semi-structured or structured data. In Daytona Graysort Challenge (Daytona 100TB) on 2014, Spark largely outpaced MapReduce, with a team from Databricks including Spark committers, Reynold Xin, Xiangrui Meng, and Matei Zaharia. Spark won a tie with the Themis team from UCSD, and set a new world record in sorting. With 206 EC2 i2.8xlarge machines, Spark reaches a record of 23 minutes to sort 100TB of data. On the other hand, the record established by Hadoop MapReduce cluster was 72 minutes, with 2100 nodes to sort the same quantity (100TB). This means that Apache Spark is 3 times faster than Hadoop MapReduce, and it uses 10 times less machines. All the sorting took place on disk (HDFS), without using Spark's in-memory cache (Spark wins Daytona Gray Sort 100TB

Benchmark | Apache Spark) (Databricks)(Sort Benchmark). The Figures 9 and 10 represent the results of this challenge.

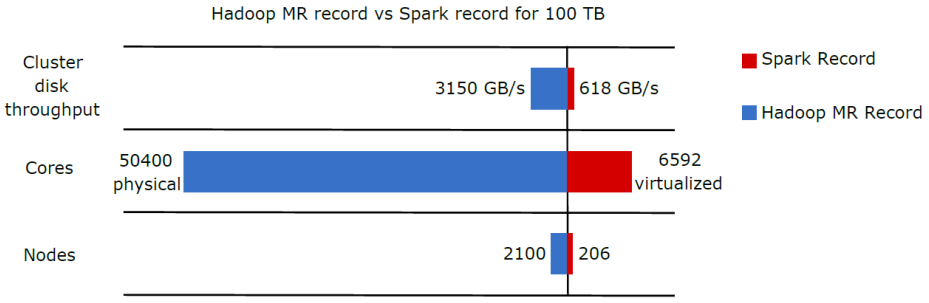


Figure 9. The results correspond to Daytona Graysort Challenge for Cluster disk throughput, Cores and Nodes

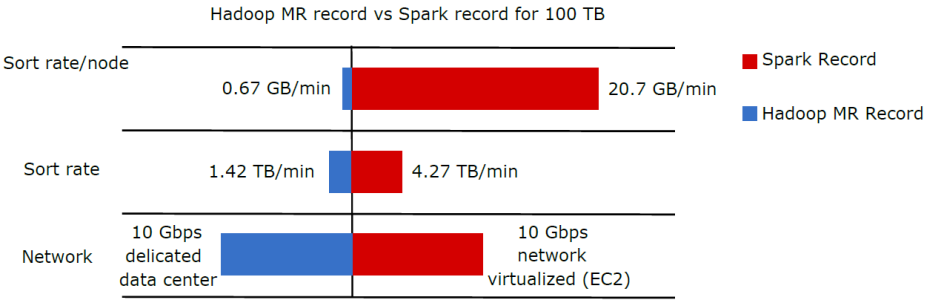


Figure 10. The results correspond to Daytona Graysort Challenge for Sort rate/node, Sort rate and Network

Apache Spark really wins the challenge, it achieves superior performance than Hadoop MapReduce clusters on sorting. Which means a great work of the team that is behind Spark, and also demonstrates that Spark is fulfilling its promise to serve as a faster and more scalable engine for data processing of all sizes GBs, TBs and PBs.

## 6. Conclusion

To conclude, firstly Hadoop has a powerful distributed storage file (HDFS), with performances and features that are available for Big Data processing systems such as Hadoop MapReduce and Apache Spark. Hadoop MapReduce is a platform built for batch processing. It was very famous and very usable by multiple companies in several areas, because Hadoop MapReduce was the first technology that analyze mass distributed data.

On the other hand, Apache Spark is the new brightest platform on the Big Data technology, it has better performances. The Spark is very profitable thanks to the data processing in memory, it is compatible with all of the data sources and file formats of Hadoop, also it has several APIs. Apache Spark even includes graph processing and machine-learning capabilities.

Although some companies feel compelled to choose between Hadoop and Spark, the fact that isn't a direct fight. They are complementary technologies that work in tandem in some cases, or separately in other cases, depending on the data or the desired results. The truth is that Spark and Hadoop have a dependent relationship with each other. Hadoop provides options that Spark doesn't possess like HDFS, and Spark provides real-time, in-memory processing. The perfect scenario of Big Data, is that Hadoop and Spark work together on the same team.

Our future work would focus on the Big Data analytics approach, more specifically in the area of health in Morocco. What we want precisely is to design a new model of treatment that is interested to health data, with powerful techniques and technologies of Big Data, which will benefit from the advantages and powers of the two Frameworks Hadoop and Spark. This model will take in two methods of treatment: the batch-processing paradigm and the real-time processing paradigm.

## Acknowledgements

The authors wish to thank the IEOM Society for organizing the International Conference on Industrial Engineering and Operations Management Bangkok, Thailand. We would also like to show our gratitude to the "anonymous" reviewers for evaluation of the manuscript.

## References

Abdul Ghaffar Shoro, T. R. S. (2015). Big Data Analysis: Apache Spark Perspective. *Global Journal of Computer Science and Technology*, *15*(1), 7–14. Retrieved from http://www.computerresearch.org/index.php/computer/article/viewFile/1137/1124

Apache Flume. Retrieved July 17, 2017, from https://flume.apache.org/

Apache Hadoop YARN. Retrieved July 26, 2017, from https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

Apache HBase. Retrieved July 8, 2017, from https://hbase.apache.org/

Apache Hive. Retrieved July 7, 2017, from https://hive.apache.org/

Apache Oozie. Retrieved July 2, 2017, from http://oozie.apache.org/

Apache Pig. Retrieved July 11, 2017, from https://pig.apache.org/

Apache Spark. Retrieved June 5, 2017, from http://spark.apache.org/

Apache Spark | MapR. Retrieved July 29, 2017, from https://mapr.com/products/apache-spark/

Apache Sqoop. Retrieved July 13, 2017, from http://sqoop.apache.org/

Apache ZooKeeper. Retrieved July 21, 2017, from https://zookeeper.apache.org/

Apache™ Hadoop. Retrieved July 1, 2017, from http://hadoop.apache.org/

Besse, P., & Loubes, J. (2014). Big Data Analytics - Retour vers le Futur 3 ; De a Data Scientist à Data Scientist.

Bichutskiy, V. (2015). Spark overview.

Borthakur, D. (2008). HDFS architecture guide. *Hadoop Apache Project*, 1–13. Retrieved from http://archive.cloudera.com/cdh/3/hadoop-0.20.2-cdh3u6/hdfs_design.pdf%5Cnpapers3://publication/uuid/BE03DF70-D0C1-441E-A65F-1888C84992D6

Carlos Hinojosa. (2016). *Probabilistic Topic Modeling with Latent Dirichlet Allocation on Apache Spark*. Faculty of Mathematics. University of Barcelona.

Comparing Hadoop, MapReduce, Spark, Flink, and Storm - MetiStream. Retrieved September 1, 2017, from http://www.metistream.com/comparing-hadoop-mapreduce-spark-flink-storm/

Comparison between Apache Spark and Hadoop MapReduce - DataFlair. Retrieved September 7, 2017, from http://data-flair.training/blogs/comparison-between-apache-spark-vs-hadoop-mapreduce/

Corp. (2014). *Guide Du Big Data 2013/2014*. *Big Data Paris Congress*.

Corp. (2015). *Guide Du Big Data 2014/2015*. *Big Data Paris Congress*.

Corp. (2016). *Guide Du Big Data 2015/2016*. *Big Data Paris Congress*.

Databricks. Apache Spark the fastest open source engine for sorting a petabyte. Retrieved September 12, 2017, from https://databricks.com/blog/2014/10/10/spark-petabyte-sort.html

Databricks. (2015). Apache Spark Survey Results 2015.

Dean, J., & Ghemawat, S. (2004). Simplified data processing on large clusters. In *OSDI'04 Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation* (Vol. 51, pp. 107–113). San Francisco, CA, USA. http://doi.org/10.1145/1327452.1327492

Demchenko, Y., Grosso, P., De Laat, C., & Membrey, P. (2013). Addressing big data issues in Scientific Data Infrastructure. *IEEE International Conference on Collaboration Technologies and Systems, CTS 2013*, 48–55.

Demchenko, Y., Ngo, C., & Membrey, P. (2013). Architecture framework and components for the big data ecosystem. *Journal of System and Network Engineering*, 1–31.

EMC Corporation. ANNUAL REPORT Big Data. Retrieved June 19, 2017, from https://www.emc.com/corporate/annual-report/big-data.htm

Frampton, M. (2015). *Mastering apache Spark*. *Packt Publishing Ltd*.

Gantz, J., & Reinsel, D. (2011). *Extracting Value from Chaos*. *IDC iView*. Retrieved from http://idcdocserv.com/1142

IBM. What is Big Data? Retrieved June 1, 2017, from http://www.ibm.com/big-data/

Insititut Montaigne. (2015). *Big data et objets connectés Faire de la France un champion de la révolution numérique*.

Intel IT Center. (2013). Guide de planification Démarrer avec le Big Data.
Laney, D. (2001). 3-D Data Management: Controlling Data Volume, Velocity and Variety. *META Group*.
LeMagIT/TechTarget. (2014). *Tout savoir sur Hadoop : Vulgarisation de la technologie et les stratégies de certains acteurs*.
MapReduce and Spark - Cloudera VISION. Retrieved September 4, 2017, from https://vision.cloudera.com/mapreduce-spark/
McAfee, A., & Brynjolfsson, E. (2012). Big Data. The management revolution. *Harvard Buiness Review*, *90*(10), 61–68. http://doi.org/10.1007/s12599-013-0249-5
McKinsey & Company. (2011). *Big data: The next frontier for innovation, competition, and productivity*. *McKinsey Global Institute*.
Parziale, L., Bostian, J., Kumar, R., Seelbach, U., & Ye, Z. Y. (2016). *Apache Spark Implementation on IBM z / OS*. IBM Corporation.
Russom, P. (2011). *BIG DATA ANALYTICS*.
Sagiroglu, S., & Sinanc, D. (2013). Big data: A review. In *2013 International Conference on Collaboration Technologies and Systems (CTS)* (pp. 42–47). http://doi.org/10.1109/CTS.2013.6567202
Saldhi, A. (2014). Big Data Analysis Using Hadoop Cluster. In *2014 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)* (pp. 0–3).
SAS Company. Big Data Analytics What it is and why it matters. Retrieved June 23, 2017, from https://www.sas.com/en_us/insights/analytics/big-data-analytics.html
Schneider, R. D. (2015). *Hadoop Buyer's Guide*. *Ubuntu*.
Sort Benchmark. Retrieved September 17, 2017, from http://sortbenchmark.org/
Spark wins Daytona Gray Sort 100TB Benchmark | Apache Spark. Retrieved September 11, 2017, from http://spark.apache.org/news/spark-wins-daytona-gray-sort-100tb-benchmark.html
Stockinger, K. (2015). Brave New World : Hadoop vs . Spark.
Teboul, B., & Berthier, T. (2015). Valeur et Véracité de la donnée : Enjeux pour l'entreprise et défis pour le Data Scientist. In *La donnée n'est pas donnée. Military School. Paris. France*.
Tutorialspoint. (2015). Apache Spark - Core Programming, 2.
What is Apache Spark?. Retrieved June 13, 2017, from https://databricks.com/spark/about
What is the difference between Hadoop and Spark? - Quora. Retrieved September 20, 2017, from https://www.quora.com/What-is-the-difference-between-Hadoop-and-Spark
White, T. (2012). *Hadoop: The Definitive Guide. The 2nd edition*. O'Reilly Media, Inc.
Zaharia, M., Chowdhury, M., Das, T., & Dave, A. (2012). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation. USENIX Association* (pp. 2–2). Retrieved from https://www.usenix.org/system/files/conference/nsdi12/nsdi12-final138.pdf

## Bibliography

**Houssam BENBRAHIM** is a Ph.D. student in Computer Engineering, affiliated to the Systems Engineering Laboratory in the National School of Applied Sciences (ENSA), Ibn Tofail University, Kenitra, Morocco. He received his Master's degree in Computer Engineering: Networks and Systems from the Faculty of Sciences and Techniques of Tangier, Abdelmalek Essaadi University. He is currently working in the field of Big Data in the healthcare with the BOSS team: Big Data, Optimization, Service, and Security.

**Prof. Hanaâ Hachimi**, Ph.D. in Applied Mathematics and Computer Science and a Ph.D. in Mechanics and Systems Reliability. She is a professor and researcher in the ENSA of Kenitra, Ibn Tofail University. She is affiliated with the Systems Engineering Laboratory since October 2013, precisely with the BOSS team. She is responsible and coordinator of the courses: Operational Research, Graph Theory, Statistics, and Probability.

**Prof. Aouatif AMINE** has been an associate professor in the Ibn Tofail University - Kenitra, Morocco, since 2014. In July 2010, she joined the ENSA of Kenitra. Ex Vice President of IEEE SPS Morocco Chapter, Secretary General of the Moroccan Association for the Development of Electronics, Electrical Engineering, Computer Science and Automation (AMADEIA). Her areas of research include: data classification, object tracking, road safety for intelligent vehicles and pattern recognition.