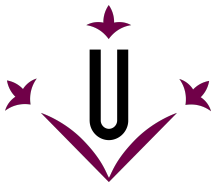

University of Lleida

Degree on Computer Engineering

Hadoop + Spark: Architecture and applications

Distributed Computing subject

Jordi Farrera Palou
Joel Ampurdanés Bonjoch

October 30, 2022

Index of Contents

1 Introduction **3**

2 Spark **3**
- 2.1 What is Spark . . . 3
- 2.2 Apache Spark features . . . 3
- 2.3 Spark architecture . . . 4
  - 2.3.1 Worker Nodes . . . 5
- 2.4 Two Main Abstractions of Apache Spark . . . 5
  - 2.4.1 Resilient Distributed Datasets (RDD) . . . 5
  - 2.4.2 Directed Acyclic Graph (DAG) . . . 6
- 2.5 Applications of Spark . . . 6
  - 2.5.1 Finance Industry . . . 6
  - 2.5.2 Healthcare . . . 6
  - 2.5.3 Media and Entertainment Industry . . . 6
  - 2.5.4 Travel Industry . . . 6
  - 2.5.5 Gaming Industry . . . 7

3 Hadoop **7**
- 3.1 What is Hadoop . . . 7
- 3.2 Features of Hadoop . . . 7
- 3.3 Hadoop Distributed File System (HDFS) . . . 8
  - 3.3.1 The goals of HDFS . . . 8
  - 3.3.2 Example of HDFS . . . 9
- 3.4 MapReduce . . . 9
  - 3.4.1 Example MapReduce . . . 9
- 3.5 Yet Another Resource Negotiator (YARN) . . . 10
  - 3.5.1 Hadoop YARN features and functions . . . 10
  - 3.5.2 Key components of Hadoop YARN . . . 10
  - 3.5.3 YARN advantages . . . 11
- 3.6 Applications of Hadoop . . . 12
  - 3.6.1 Website Tracking . . . 12
  - 3.6.2 Geographical Data . . . 12
  - 3.6.3 Retail Industry . . . 13
  - 3.6.4 Financial Industry . . . 13
  - 3.6.5 Healthcare Industry . . . 13
  - 3.6.6 Digital Marketing . . . 13

4 Differences between Hadoop and Spark **13**

5 Combine Hadoop and Spark **15**

6 Conclusions **15**

7 Bibliography **15**

Index of Figures

1 Spark architecture 4
2 Worker Nodes 5
3 Hadoop Features 7
4 YARN's architecture 11
5 Applications of Hadoop 12
6 Spark-and-Hadoop architecture 15

Index of Tables

1 Differences between Hadoop and Spark 14

1 Introduction

In this practice we will explain what Hadoop and Spark are all about, the applications in which they are used, the differences between them and the possible advantages of combining both.

2 Spark

2.1 What is Spark

Apache Spark is an open-source, distributed processing system used for big data workloads. It uses in-memory caching and optimized query execution for fast queries against data of any size. Simply put, Spark is a fast and general engine for large-scale data processing.

The fast part means that it's faster than previous approaches to work with Big Data like classical MapReduce. The secret for being faster is that Spark runs on memory (RAM), and that makes the processing much faster than on disk services.

The general part means that it can be used for multiple things like running distributed SQL, creating data pipelines, ingesting data into a database, running Machine Learning algorithms, working with graphs or data streams, and much more.

2.2 Apache Spark features

- **Speed**: Spark performs up to 100 times faster than MapReduce for processing large amounts of data. It is also able to divide the data into chunks in a controlled way.
- **Powerful Caching**: Powerful caching and disk persistence capabilities are offered by a simple programming layer.
- **Deployment**: Mesos, Hadoop via YARN, or Spark's own cluster manager can all be used to deploy it.
- **Real-Time**: Because of its in-memory processing, it offers real-time computation and low latency.
- **Polyglot**: In addition to Java, Scala, Python, and R, Spark also supports all four of these languages. You can write Spark code in any one of these languages. Spark also provides a command-line interface in Scala and Python.

2.3 Spark architecture

The Apache Spark base architecture diagram is:

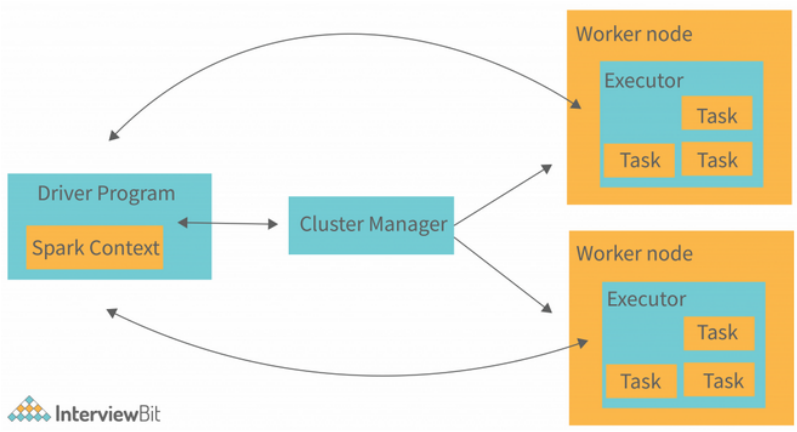

Figure 1: Spark architecture

When the Driver Program in the Apache Spark architecture executes, it calls the real program of an application and creates a Spark Context. Spark Context contains all of the basic functions.

The Spark Driver includes several other components, including a DAG Scheduler, Task Scheduler, Backend Scheduler, and Block Manager, all of which are responsible for translating user-written code into jobs that are actually executed on the cluster.

The Cluster Manager manages the execution of various jobs in the cluster. Spark Driver works in conjunction with the Cluster Manager to control the execution of various other jobs. The Cluster Manager does the task of allocating resources for the job. Once the job has been broken down into smaller jobs, which are then distributed to worker nodes, Spark Driver will control the execution.

Many worker nodes can be used to process an RDD created in the Spark Context, and the results can also be cached.

The Spark Context receives task information from the Cluster Manager and enqueues it on worker nodes.

The Executor is in charge of carrying out these duties. The lifespan of executors is the same as that of the Spark Application. We can increase the number of workers if we want to imporve he performance of the system. In this way, we can divide jobs into more coherent parts.

2.3.1 Worker Nodes

The slave nodes function as executors, processing tasks, and returning the results back to the spark context.

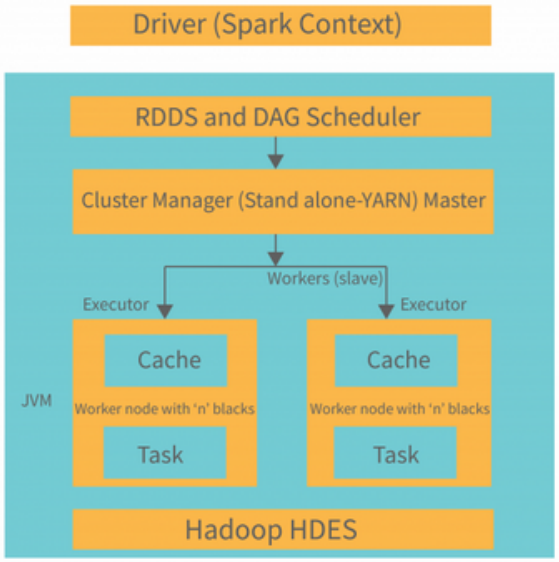

Figure 2: Worker Nodes

1. There are multiple executor processes for each application, which run tasks on multiple threads over the course of the whole application.
2. Even on a cluster manager that also supports other applications, Spark can be run if it can acquire executor processes and these communicate with each other. It's relatively easy for Spark to operate even on a cluster manager if this can be done even with other applications (YARN).
3. The driver program must listen for and accept incoming connections from its executors throughout its lifetime. Workers must be able to connect to the driver program via the network.
4. The driver is responsible for scheduling tasks on the cluster. It should be run on the same local network as the worker nodes, preferably on the same machine.

2.4 Two Main Abstractions of Apache Spark

The Apache Spark architecture consists of two main abstraction layers:

2.4.1 Resilient Distributed Datasets (RDD)

It is a key tool for data computation. It enables you to recheck data in the event of a failure, and it acts as an interface for immutable data.

2.4.2 Directed Acyclic Graph (DAG)

The driver converts the program into a DAG for each job. The Apache Spark Eco-system includes various components such as the API core, Spark SQL, Streaming and real-time processing, MLIB, and Graph X. A sequence of connection between nodes is referred to as a driver. As a result, you can read volumes of data using the Spark shell. You can also use the Spark context -cancel run a job, task (work), and job (computation) to stop a job.

2.5 Applications of Spark

Here are some industry specific spark use cases that demonstrate its ability to build and run fast big data applications.

2.5.1 Finance Industry

Banks are using Spark to access and analyse the social media profiles, call recordings, complaint logs, emails, forum discussions, etc. to gain insights that can help them make the right business decisions for credit risk assessment, targeted advertising and customer segmentation.

With the use of Apache Spark on Hadoop, financial institutions can detect fraudulent transactions in real-time, based on previous fraud footprints. All the incoming transactions are validated against a database, if there a match then a trigger is sent to the call centre. The call centre personnel immediately checks with the credit card owner to validate the transaction before any fraud can happen.

2.5.2 Healthcare

Many healthcare providers are using Apache Spark to analyse patient records along with past clinical data to identify which patients are likely to face health issues after being discharged from the clinic. This helps hospitals prevent hospital re-admittance as they can deploy home healthcare services to the identified patient, saving on costs for both the hospitals and patients. Also, Apache Spark is used in genomic sequencing to reduce the time needed to process genome data.

2.5.3 Media and Entertainment Industry

Apache Spark is used in the gaming industry to identify patterns from the real-time in-game events and respond to them to harvest lucrative business opportunities like targeted advertising, auto adjustment of gaming levels based on complexity, player retention and many more.

Few of the video sharing websites use Apache Spark along with MongoDB to show relevant advertisements to its users based on the videos they view, share and browse.

2.5.4 Travel Industry

A Travel industry like TripAdvisor, uses Apache Spark to provide advice to millions of travellers by comparing hundreds of websites to find the best hotel prices for its customers.

The time taken to read and process the reviews of the hotels in a readable format is done with the help of Apache Spark.

2.5.5 Gaming Industry

Apache Spark is used in the gaming industry to identify patterns from real-time in-game events. It helps companies to harvest lucrative business opportunities like targeted advertising, auto adjustment of gaming levels based on complexity. It also provides in-game monitoring, player retention, detailed insights, and many more.

3 Hadoop

3.1 What is Hadoop

Apache Hadoop is an open source framework that is used to efficiently store and process large datasets ranging in size from gigabytes to petabytes of data. Hadoop allows clustering clustering multiple computers to analyze massive datasets in parallel more quickly.

Hadoop consists of four main modules:

- Hadoop Distributed File System (HDFS): A distributed file system that runs on standard or low-end hardware.
- MapReduce: A framework that helps programs do the parallel computation on data.
- Yet Another Resource Negotiator (YARN): Manages and monitors cluster nodes and resource usage. It schedules jobs and tasks.
- Hadoop Common: Provides common Java libraries that can be used across all modules.

3.2 Features of Hadoop

The features of Hadoop are detailed below:

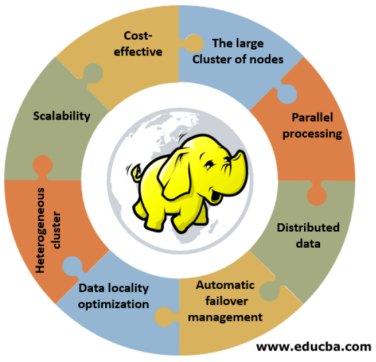

Figure 3: Hadoop Features

1. **Cost-effective**: Hadoop does not require specialized or effective hardware to implement it. It can be implemented on simple hardware, which is community hardware.
2. **Large cluster of nodes**: A cluster can consist of hundreds or thousands of nodes. The benefit of having a large cluster is that it offers more computing power and a huge storage system to the clients.
3. **Parallel processing**: Data can be processed simultaneously across all the clusters, saving a lot of time. The traditional system was not able to do this task.
4. **Distributed data**: Hadoop framework takes care of splitting and distributing the data across all the nodes within a cluster. It replicates data over all the clusters. The replication factor is 3.
5. **Automatic failover management**: If any of the cluster nodes fails, the Hadoop framework will replace the failure machine with a new one.
6. **Data locality optimization**: Suppose the programmer needs node data from a database located at a different location. The programmer will send a byte of code to the database. It will save bandwidth and time.
7. **Heterogeneous cluster**: It has a different node supporting different machines with different versions. For example, an IBM machine supports Red hat Linux.
8. **Scalability**: Adding or removing nodes and adding or removing hardware components to or from the cluster. We can perform this task without disturbing cluster operations. For example, RAM or Hard Drive can be added or removed from the cluster.

3.3 Hadoop Distributed File System (HDFS)

HDFS handles large data sets running on commodity hardware. It is used to scale a single Apache Hadoop cluster to hundreds (and even thousands) of nodes.

3.3.1 The goals of HDFS

Fast recovery from hardware failures

Because one HDFS instance may consist of thousands of servers, failure of at least one server is inevitable. HDFS has been built to detect faults and automatically recover quickly.

Acces to streaming data

HDFS is intended more for batch processing versus interactive use, so the emphasis in the design is for high data throughput rates, which accommodate streaming access to data sets.

Accomodation of large data sets

HDFS accommodates applications that have data sets typically gigabytes to terabytes in size. HDFS provides high aggregate data bandwidth and can scale to hundreds of nodes in a single cluster.

Portability

To facilitate adoption, HDFS is designed to be portable across multiple hardware platforms and to be compatible with a variety of underlying operating systems.

3.3.2 Example of HDFS

Consider a file that includes the phone numbers for everyone in Catalonia, the numbers for people whom last name starting with A might be stored on server 1, B on server 2, and so on.

With Hadoop, pieces of this phonebook would be stored across the cluster, and to reconstruct the entire phonebook, your program would need the blocks from every server in the cluster.

To ensure availability if and when a server fails, HDFS replicates these smaller pieces onto two additional servers by default. This redundancy offers multiple benefits, the most obvious being higher availability.

The redundancy also allows the Hadoop cluster to break up work into smaller chunks and run those jobs on all the servers in the cluster for better scalability. Finally, you gain the benefit of data locality, which is critical when working with large data sets.

3.4 MapReduce

MapReduce is a programming paradigm that enables massive scalability across hundreds or thousands of servers in a Hadoop cluster. The term "MapReduce" refers to two separate and distinct tasks that Hadoop programs perform. The first is the map job, which takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs).

MapReduce programming offers several benefits to help you gain valuable insights from your big data:

- **Scalability**: Businesses can process petabytes of data stored in the Hadoop Distributed File System (HDFS).
- **Flexibility**: Hadoop enables easier access to multiple sources of data and multiple types of data.
- **Speed**: With parallel processing and minimal data movement, Hadoop offers fast processing of massive amounts of data.
- **Simple**: Developers can write code in a choice of languages, including Java, C++ and Python.

3.4.1 Example MapReduce

This is a very simple example of MapReduce.

Assume you have five files, and each file contains two columns (a key and a value in Hadoop terms) that represent a city and the corresponding temperature recorded in that

city for the various measurement days. The city is the key, and the temperature is the value. For example: (Lleida, 20). Out of all the data we have collected, you want to find the maximum temperature for each city across the data files (note that each file might have the same city represented multiple times).

Using the MapReduce framework, you can break this down into five map tasks, where each mapper works on one of the five files. The mapper task goes through the data and returns the maximum temperature for each city.

For example, the results produced from one mapper task for the data above would look like this: (Lleida, 20) (Barcelona, 25) (New York, 22) (Rome, 33)

Assume the other four mapper tasks (working on the other four files not shown here) produced the following intermediate results:

(Lleida, 18) (Barcelona, 27) (New York, 32) (Rome, 37) (Lleida, 32) (Barcelona, 20) (New York, 33) (Rome, 38) (Lleida, 22) (Barcelona, 19) (New York, 20) (Rome, 31) (Lleida, 31) (Barcelona, 22) (New York, 19) (Rome, 30)

All five of these output streams would be fed into the reduce tasks, which combine the input results and output a single value for each city, producing a final result set as follows: (Lleida, 32) (Barcelona, 27) (New York, 33) (Rome, 38).

3.5 Yet Another Resource Negotiator (YARN)

YARN is responsible for allocating system resources to the various applications running in a Hadoop cluster and scheduling tasks to be executed on different cluster nodes.

3.5.1 Hadoop YARN features and functions

In a cluster architecture, Apache Hadoop YARN sits between HDFS and the processing engines being used to run applications. It combines a central resource manager with containers, application coordinators and node-level agents that monitor processing operations in individual cluster nodes. YARN can dynamically allocate resources to applications as needed, a capability designed to improve resource utilization and application performance compared with MapReduce's more static allocation approach.

YARN supports multiple scheduling methods, all based on a queue format for submitting processing jobs.

Another pluggable tool, called Capacity Scheduler, enables Hadoop clusters to be run as multi-tenant systems shared by different units in one organization or by multiple companies, with each getting guaranteed processing capacity based on individual service-level agreements.

Hadoop YARN also includes a Reservation System feature that lets users reserve cluster resources in advance for important processing jobs to ensure they run smoothly.

3.5.2 Key components of Hadoop YARN

Apache Hadoop YARN decentralizes execution and monitoring of processing jobs by separating the various responsibilities into these components:

- A global ResourceManager that accepts job submissions from users, schedules the jobs and allocates resources to them.
- A NodeManager slave that's installed at each node and functions as a monitoring and reporting agent of the ResourceManager.
- An ApplicationMaster that's created for each application to negotiate for resources and work with the NodeManager to execute and monitor tasks.
- Resource containers that are controlled by NodeManagers and assigned the system resources allocated to individual applications.

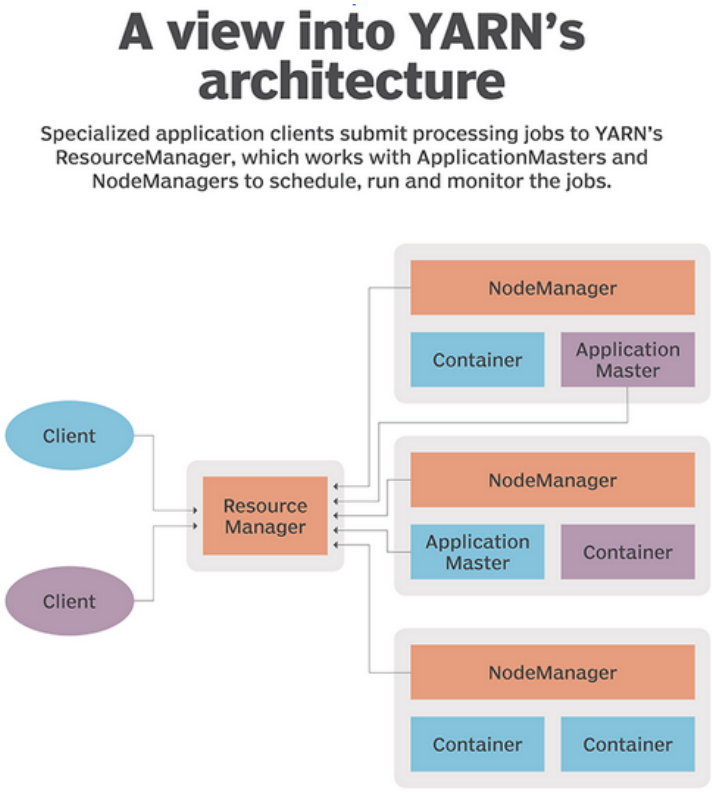

Figure 4: YARN's architecture

3.5.3 YARN advantages

Using Apache Hadoop YARN to separate HDFS from MapReduce made the Hadoop environment more suitable for real-time processing uses and other applications that can't wait for batch jobs to finish. Now, MapReduce is just one of many processing engines that

can run Hadoop applications. It doesn't even have a lock on batch processing in Hadoop anymore: In a lot of cases, users are replacing it with Spark to get faster performance on batch applications, such as extract, transform and load jobs.

Spark can also run stream processing applications in Hadoop clusters thanks to YARN, as can technologies including Apache Flink and Apache Storm. YARN has also opened up new uses for Apache HBase, a companion database to HDFS, and for Apache Hive, Apache Drill, Apache Impala, Presto and other SQL-on-Hadoop query engines. In addition to more application and technology choices, YARN offers scalability, resource utilization, high availability and performance improvements over MapReduce.

3.6 Applications of Hadoop

The Applications of Hadoop are given below:

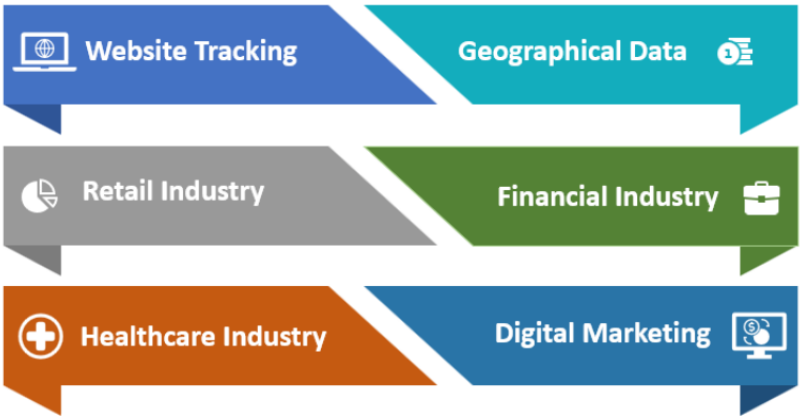

Figure 5: Applications of Hadoop

3.6.1 Website Tracking

Suppose you have created a website and want to know about visitors' details. Hadoop will capture a massive amount of data about this. It will give information about the visitor's location, which page visitors visited first and most, how much time spent on the website and on which page, how many times a visitor has visited the page, and what visitors like most. This will provide a predictive analysis of visitors' interests. Website performance will predict what would be users' interests. Hadoop accepts data in multiple formats from multiple sources. Apache HIVE will be used to process millions of data.

3.6.2 Geographical Data

When we buy products from an e-commerce website, the website will track the user's location and predict customer purchases using smartphones and tablets. Hadoop cluster will help to figure out business in geo-location. This will help the industries to show the business graph in each area (positive or negative).

3.6.3 Retail Industry

Retailers will use the data of customers, which is present in the structured and unstructured format, to understand and analyze the data. This will help a user to understand customer requirements and serve them with better benefits and improved services.

3.6.4 Financial Industry

Financial Industry and Financial companies will assess the financial risk and market value and build the model which will give customers and the industry better results in terms of investment like the stock market, FD, etc. Understand the trading algorithm. Hadoop will run the build model.

3.6.5 Healthcare Industry

Hadoop can store large amounts of data. Medical data is present in an unstructured format. This will help the doctor for a better diagnosis. Hadoop will store a patient medical history for more than one year and will analyze symptoms of the disease.

3.6.6 Digital Marketing

We are in the era of the 20s, and every single person is connected digitally. Information is reached to the user over mobile phones or laptops, and people get aware of every single detail about news, products, etc. Hadoop will store massively online generated data, store, analyze and provide the result to digital marketing companies.

4 Differences between Hadoop and Spark

To find the differences between Hadoop and Spark, we need to compare Spark with the MapReduce modul, because Spark is a Hadoop enhancement to MapReduce.

The below table shows the difference between Hadoop and Spark, listed on the basis of some parameters like performance, cost, etc.

Basis	Hadoop	Spark
Processing Speed	Hadoop's MapReduce model reads and writes from a disk, thus slowing down the processing speed.	Spark reduces the number of read/write cycles to disk and stores intermediate data in memory, hence faster-processing speed.
Usage	Hadoop is designed to handle batch processing efficiently.	Spark is designed to handle real-time data efficiently.
Latency	Hadoop is a high latency computing framework, which does not have an interactive mode.	Spark is a low latency computing and can process data interactively.

Data	With Hadoop MapReduce, a developer can only process data in batch mode only.	Spark can process real-time data, from real-time events like Twitter, and Facebook.
Cost	Hadoop is a cheaper option available while comparing it in terms of cost	Spark requires a lot of RAM to run in-memory, thus increasing the cluster and hence cost.
Algorithm Used	The PageRank algorithm is used in Hadoop.	Graph computation library called GraphX is used by Spark.
Fault Tolerance	Hadoop is a highly fault-tolerant system where data is replicated across the nodes and used the data in case of any issue.	Spark uses a DAG to rebuild the data across the nodes.
Security	Hadoop supports LDAP, ACLs, SLAs, etc and hence it is extremely secure.	Spark is not secure, it relies on the integration with Hadoop to achieve the necessary security level.
Machine Learning	Data fragments in Hadoop can be too large and can create bottlenecks. Thus, it is slower than Spark.	Spark is much faster as it uses MLib for computations and has in-memory processing.
Performance	Hadoop has a slower performance as it uses disk for storage and depends upon disk read and write operations.	It has fast performance with reduced disk reading and writing operations.
Scalability	Hadoop is easily scalable by adding nodes and disk for storage. It supports tens of thousands of nodes.	It is quite difficult to scale as it relies on RAM for computations. It supports thousands of nodes in a cluster.
Language support	It uses Java or Python for MapReduce apps.	It uses Java, R, Scala, Python, or Spark SQL for the APIs.
User-friendliness	It is more difficult to use.	It is more user-friendly.
Resource Management	YARN is the most common option for resource management.	It has built-in tools for resource management.

Table 1: Differences between Hadoop and Spark

5 Combine Hadoop and Spark

Previously, on **section 4** we have explained the difference between Hadoop MapReduce and Spark, but you can also combine Hadoop with Spark, and have the best things of them. Below you can see a simplified version of Spark-and-Hadoop architecture:

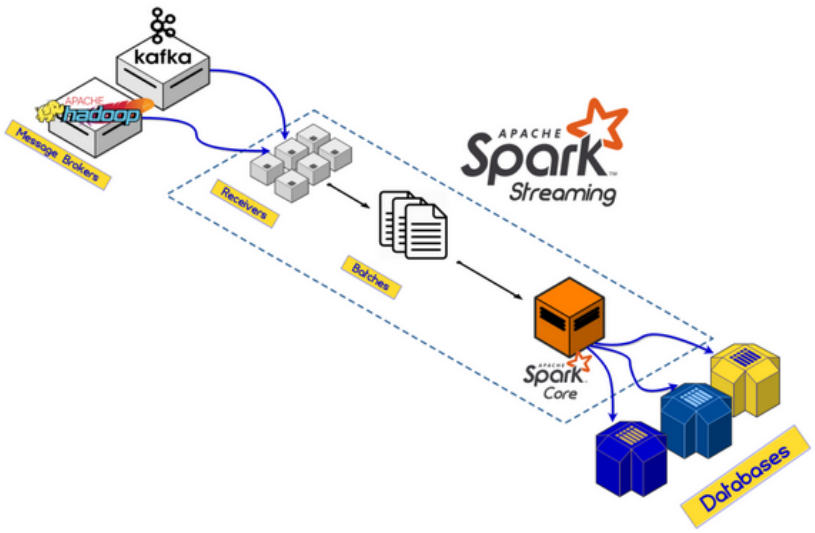

Figure 6: Spark-and-Hadoop architecture

Organizations that need batch analysis and stream analysis for different services can see the benefit of using both tools. Hadoop can—at a lower price—deal with heavier operations while Spark processes the more numerous smaller jobs that need instantaneous turnaround.

YARN also makes archiving and analysis of archived data possible, whereas it isn't with Apache Spark. Thus, Hadoop and YARN in particular becomes a critical thread for tying together the real-time processing, machine learning and reiterated graph processing.

6 Conclusions

Doing this activity we have learnt an interesting topic that has a lot of advantages on the Big Data world. Also, as we have explained in the report, we conclude that the best use of Hadoop and Spark is the union between them.

7 Bibliography

https://www.juanbarrios.com/que-son-hadoop-y-spark/

https://www.interviewbit.com/blog/apache-spark-architecture/

https://www.youtube.com/watch?v=Z-iNJypyZTQ

https://www.youtube.com/watch?v=T6YYDDgVtvw

https://www.geeksforgeeks.org/difference-between-hadoop-and-spark/

https://logz.io/blog/hadoop-vs-spark/

https://www.ibm.com/topics/mapreduce

https://www.ibm.com/topics/hdfs

https://aprenderbigdata.com/hadoop-mapreduce/

https://aprenderbigdata.com/hadoop-yarn/

https://www.upgrad.com/blog/features-applications-of-hadoop/

https://data-flair.training/blogs/hadoop-applications/

https://www.educba.com/what-is-hadoop/

https://www.integrate.io/blog/storing-apache-hadoop-data-cloud-hdfs-vs-s3/

https://www.upgrad.com/blog/apache-spark-applications-use-cases/