Apache Spark Software Technology Evaluation Project

Jose Juan Pena Gomez and Enrique Vilchez Campillejo

November 29, 2019

Abstract

The software technology which will be discussed in this report is **Apache Spark**, an unified tool for analysing and processing data. In addition, **PySpark** is going to be used to write the code in **iPython Notebook**.

The purpose of this report is to check the different ways of processing big data using **Distributed Machine Learning** techniques, working with clusters, to see which way of **clustering** is more efficient.

The dataset taken for this project contains taxi travels with information about source location, target location, amount of passengers and taxi fare. This **large dataset** is a proficient approach to work with Spark in big data computing purposes.

There will be three Distributed Machine Learning models presented, in two approaches, one way for **Native Clustering**, which means one core acting as a cluster and the other approach is **Local Clustering** which means as many clusters as cores the processor has.

The models used are **Linear Regression**, **Decision Tree** and **Random Forest** generating several results. Basically, the results acknowledge the efficiency to work with Local Clustering rather than working with Native Clustering. Another improvement would be to work with **Remote Clustering** in a network with several machines, but it wasn't possible to implement it in the end. Remote clustering is more efficient because of the quantity of cores used to parallelize the tasks.

1 Introduction

The main purpose of the project is to use a unified analytics engine (Apache Spark) to compare how clustering works with machine learning and how the technology works itself. As we worked before in projects of Machine Learning, we think it would be interesting to work with distributed machine-learning framework.

In 2008, approximately,the basis for creating this technology appeared, Hive and HBase, which are two tools of the Hadoop ecosystem. Spark was founded at UC Berkeley in 2009, it was born practically from a Google paper and from there it evolved, going through the mapreduce processes.

Comparing this technology with others, for example with sklearn, the most well-known library for machine learning porpoises, we are satisfied with our results because the sklearn library is even not able to load big datasets. In addition we are satisfied with the clustering management, it is 60 % faster than without clustering management.

This rest of this report is organised as follows:

- Section 2 gives an overview of the key concepts and the architecture.

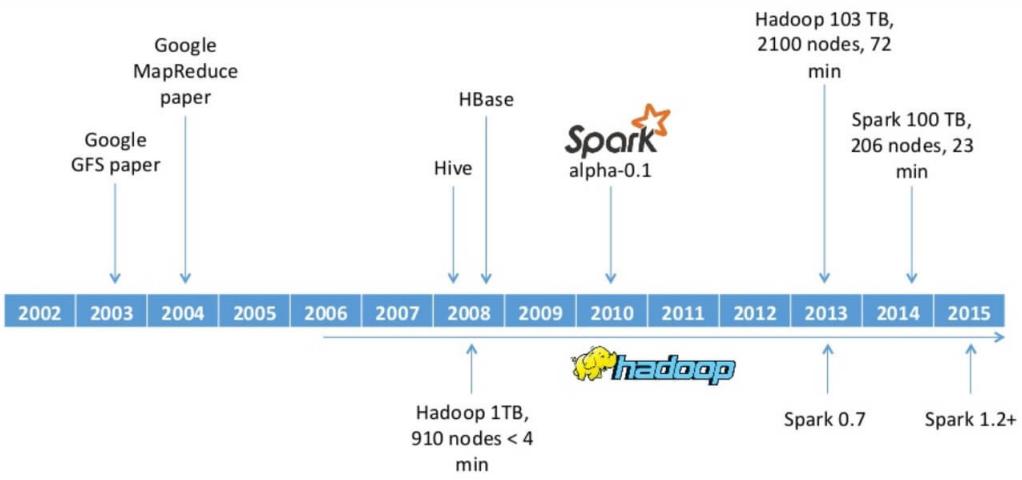

Figure 1: Apache Spark Development Timeline

- Section 3 gives a view of the prototype and its implementation
- Section 4 gives an explanation of the test-bed environment used and the results of the experiments
- Section 5 gives a conclusion about the main concepts that we appreciated from this technology.

2 The Software Technology

Apache Spark is an open-source distributed general-purpose cluster-computing framework. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance. Originally developed at the University of California, Berkeley's AMPLab, the Spark codebase was later donated to the Apache Software Foundation, which has maintained it since. It is based on a data computing and analytics system based on Hadoop Map Reduce.

The official web defines Apache Spark as an unified analytics engine for large-scale data processing.[2]

Apache Spark has as its architectural foundation the resilient distributed dataset (RDD), a read-only multiset of data items distributed over a cluster of machines, that is maintained in a fault-tolerant way.[6] The Dataframe API was released as an abstraction on top of the RDD, followed by the Dataset API. In Spark 1.x, the RDD was the primary application programming interface (API), but as of Spark 2.x use of the Dataset API is encouraged [1] even though the RDD API is not deprecated The RDD technology still underlies the Dataset API.

It works in memory, which results in a much faster processing speed, it allows you to work on disk with large amount of data. In this way if for example we have a very large file or a quantity of information that does not fit in memory, the tool allows to store part in disk, which makes lose speed. This means that we have to try to find the balance between what is stored in memory and what is stored in disk, to have a good speed and so that the cost is not too high, as the memory is always much

more expensive than the disk.

Spark and its RDDs were developed in 2012 in response to limitations in the MapReduce cluster computing paradigm, which forces a particular linear dataflow structure on distributed programs: MapReduce programs read input data from disk, map a function across the data, reduce the results of the map, and store reduction results on disk. Spark's RDDs function as a working set for distributed programs that offers a (deliberately) restricted form of distributed shared memory.[5]

Spark facilitates the implementation of both iterative algorithms, which visit their data set multiple times in a loop, and interactive/exploratory data analysis, i.e., the repeated database-style querying of data. The latency of such applications may be reduced by several orders of magnitude compared to Apache Hadoop MapReduce implementation Among the class of iterative algorithms are the training algorithms for machine learning systems, which formed the initial impetus for developing Apache Spark.

Apache Spark requires a cluster manager and a distributed storage system. For cluster management, Spark supports standalone (native Spark cluster, where you can launch a cluster either manually or use the launch scripts provided by the install package. It is also possible to run these daemons on a single machine for testing), Hadoop YARN, Apache Mesos or Kubernetes. For distributed storage, Spark can interface with a wide variety, including Alluxio, Hadoop Distributed File System (HDFS), MapR File System (MapR-FS), Cassandra, OpenStack Swift, Amazon S3, Kudu, Lustre file system, or a custom solution can be implemented. Spark also supports a pseudo-distributed local mode, usually used only for development or testing purposes, where distributed storage is not required and the local file system can be used instead; in such a scenario, Spark is run on a single machine with one executor per CPU core.[4]

The friendly APIs ,which Apache Spark provides, natively supports Java, Scala, R, and Python, giving you a variety of languages for building your applications. These APIs make it easy for developers, because they hide the complexity of distributed processing behind simple, high-level operators that dramatically lowers the amount of code required.

It allows real time processing, with a module called Spark Streaming, which combined with Spark SQL will allow us to process the data in real time. As we are injecting the data we can transform them and turn them to a final result. Resilient Distributed Dataset (RDD), it uses the lazy evaluation, which means that all the transformations that we are carrying out on the RDD, are not resolved, but are stored in a directed acyclic graph (DAG), and when we execute an action, that is to say, when the tool does not have more option than to execute all the transformations, it will be when they are executed. This is a double-edged sword, as it has an advantage and a disadvantage. The advantage is that you gain speed by not making transformations continuously, but only when necessary. The disadvantage is that if some transformation raises some kind of exception, it will not be detected until the action is executed, so it is more difficult to debug or program.

2.1 Spark Architecture

The main components that make up the framework are the following ones:

Spark Core : It is the base or set of libraries where the rest of modules are supported, is the core

of the framework.It is also shelter to API that contains the backbone of Spark that is RDDs (resilient distributed datasets). The basic functionality of Spark is present in Spark Core like:

1. Memory management
2. Fault recovery
3. Interaction with the storage system.
4. It is in charge of essential I/O functionalities like:
   - Programming and observing the role of Spark cluster
   - Task dispatching
   - Fault recovery
   - It overcomes the snag of MapReduce by using in-memory computation.

Spark SQL : It is the module for the processing of structured and semi-structured data. With this module we will be able to transform and perform operations on RDD or dataframes. It is designed exclusively for data processing.

Spark Streaming : It is the one that allows the ingestion of data in real time. If we have a source, for example Kafka or Twitter, with this module we can ingest data from that source and dump them to a destination. Between the ingestion of data and its subsequent dump, we can have a series of transformations.

Spark MLLib : It is a very complete library that contains numerous Machine Learning algorithms, both clustering, classification, regression, and so on. It allows us, in a friendly way, to use Machine Learning algorithms.

Spark Graph : Allows the processing of graphs (DAG). It does not allow to paint graphs, but it allows to create operations with graphs, with their nodes and edges, and to go making operations.

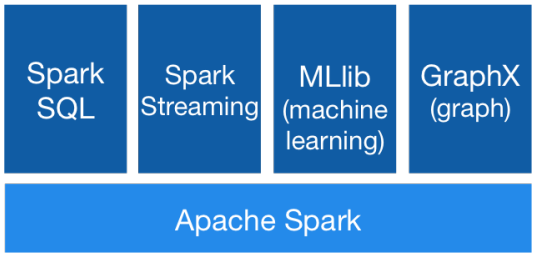

Figure 2: Components of Apache Spark

2.2 MLlib

MLIB is part of Spark APIs and is interoperable with Python NumPy as well as the R libraries. Implemented with Spark it is possible to use any type of data from any source or from the Hadoop platform

such as HDFS, HBase, related database data sources or local data sources such as text documents.

Spark excels in iterative calculation processes allowing the processes written in the MLlib libraries to be executed quickly allowing their use and above all at an industrial level.

MLlib provides many types of algorithms, as well as numerous useful functions. ML includes classification algorithms, regression, decision trees, recommendation algorithms, grouping. Among the most used utilities can include the characteristics of transformations, standardization and normalization, statistical functions and linear algebra.

2.3 Clustering

Apache Spark can be configured to run on distributed mode on the cluster as a master node or slave node. Its master/slave architecture is composed by many main daemons and a cluster manager. T schedules and divides resource in the host machine which forms the cluster.

- Master Daemon (Master/Driver Process)
- Worker Daemon (Slave Process)
- Cluster Manager. Schedules and divides resources in the host machine which forms the cluster

As we can see in the figure 3 .

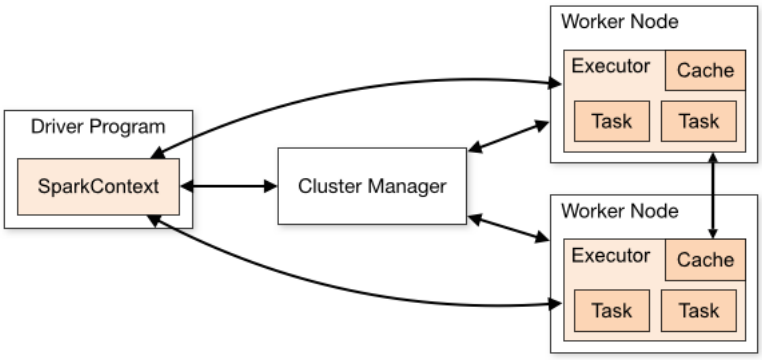

Figure 3: Cluster management of Apache Spark

3 Demonstrator Prototype

About 5 pages that gives:

1. High-level view of the demonstrator and its purpose.
2. Details of how the demonstrator has been implemented.
3. May involve presentation of code snippets.

The example below shows how you may include code. There are similar styles for many other langages - in case you do not use Java in your project. You can wrap the listing into a figure in case you need to refer to it. How to create a figure was shown in Section 2.

```
1  public class BoksVolum {
2
3     public static void main(String[] args) {
4
5        int b, h, d;
6        String btext, htext, dtext;
7
8        [ ... ]
9
10       int volum = b * h * d;
11
12       String respons =
13             "Volum [" + htext + "," + btext + "," + dtext + "] = " + volum;
14
15    }
16 }
```

Borrar encima

3.1 High-Level View

As any other machine learning project, this prototype learns how to predict the results of new data through the results of a given dataset with the results already set. As a consequence we design a prototype with a big input dataset of 6 Gb to test the technology because implicitly the library prepared for machine learning, the one that we are using, is splitting, parallelizing and clustering when we use the data structure of this library call RDD.

So during that process we are able to test the performance of Apache Spark and its component for machine learning, called MLLib. We check the differences in performance using the time spending in every training of a model because it has to compute a big amount of data, as consequence we can test this technology in terms about clustering and parallelizing huge amount of data and compare the results between native clustering and local clustering, by native we mean that we use the component by default with only one core for clustering and by local we mean to use the component with as many core as the machine has (there is a third one call remote clustering that we didnt achieve to implement so doesnt make so much sense to talk about that in the report but we did it in the presentation).

3.2 Implementation

4 Test-bed Environment and Experimental Results

About 3 pages that:

Describes the software used to establish the test-bed and for implementing the demonstrator prototype.

Config	Property	States	Edges	Peak	E-Time	C-Time	T-Time
22-2	A	7,944	22,419	6.6 %	7 ms	42.9%	485.7%
22-2	A	7,944	22,419	6.6 %	7 ms	42.9%	471.4%
30-2	B	14,672	41,611	4.9 %	14 ms	42.9%	464.3%
30-2	C	14,672	41,611	4.9 %	15 ms	40.0%	420.0%
10-3	D	24,052	98,671	19.8 %	35 ms	31.4%	285.7%
10-3	E	24,052	98,671	19.8 %	35 ms	34.3%	308.6%

Table 1: Selected experimental results on the communication protocol example.

Explains what experiments have been done and the results.

For some reports you may have to include a table with experimental results are other kinds of tables that for instance compares technologies. Table 2 gives an example of how to create a table.

4.1 Jupyter Notebooks

The software we used to experiment with the data was Jupyter Notebook to write python code and PySpark to run that python code into spark. Jupyter Notebook (formerly IPython Notebook) is an open-source web application that lets you create and share documents containing live code, equations, visualizations, and narrative text.

Also, it is a widely used application in the field of Data Science to create and share documents including data cleansing and transformation, numerical simulation, statistical modeling, data visualization, automatic learning and much more.
It allows you to edit and run notebook documents through any web browser of your choice, and it can run on a local desktop that does not require Internet access, or it can be installed on a remote server and accessed through the Internet. We can also run Jupyter Notebook without any installation.

4.2 PySpark

PySpark is a python API for spark released by the Apache Spark community to support python with Spark. Using PySpark, one can easily integrate and work with RDD in python programming language too.

Numerous features make PySpark such an amazing framework when it comes to working with huge datasets. Whether it is to perform computations on large data sets or to just analyze them, Data engineers are turning to this tool. Following are some of the said features.

The following features are the key features of PySpark:

Real-time computations : Because of the in-memory processing in PySpark framework, it shows low latency.

Polyglot : PySpark framework is compatible with various languages like Scala, Java, Python, and R, which makes it one of the most preferable frameworks for processing huge datasets.

Caching and disk persistence : PySpark framework provides powerful caching and very good disk persistence.

Fast processing : PySpark framework is way faster than other traditional frameworks for big data processing.

Works well with RDD : Python programming language is dynamically typed which helps when working with RDD.

4.3 About the dataset Taxi Problem

[3]

The goal of this dataset is predicting the fare amount (inclusive of tolls) for a taxi ride in New York City given the pickup and dropoff locations. While you can get a basic estimate based on just the distance between the two points, this will result in an RMSE of $5-8$, depending on the model used. Our challenge is to do it using distributed Machine Learning techniques.

We also choose this dataset because contains 6 Gb of data, so we think that is a proper amount of data for test our results.

4.4 Experiments

We have done six experiments (Each experiment its a model testing), to try Linear regression, Decision tree regression and Random forest regression in Native Clustering and the rest to try Linear regression, Decision tree regression and Random forest regression in Local Cluster mode.

Mode	Machine Learning Model	Time-1	RMSE-1	Time-2	RMSE-2	Time-3	RMSE-3
Native Cluster	Linear Regression	4:04 min	20.7	11:47 min	20.7	8:17 min	20.7
Native Cluster	Decision Tree	8:37 min	23.96	20:22 min	23.96	20:05 min	23.96
Native Cluster	Random Forest	11:50 min	23.96	35:37 min	24.82	35:53 min	24.82
Local Cluster	Linear Regression	2:33 min	20.7	7:40 min	20.7	7:36 min	20.7
Local Cluster	Decision Tree	5:30 min	16.49	11:35 min	24.03	13:05 min	24.03
Local Cluster	Random Forest	8:44 min	23.96	18:30 min	23.96	**26:28 min**	**6.86**

Table 2: Selected experimental results on the training models.

We have two important approaches, one is that Linear Regression is the fastest model, and Random Forest is the worst (What it isnt so important for our main purpose), and the other one is that clearly, with local clustering the machine invest much less time than with Native Spark training the same models respectively. Also, its better to use Clustering with several machines, but we couldnt really experiment with that.

5 Conclusion

Apache spark has designed in such a way that it can scale up from one to thousands of computer node. Moreover, the simplicity of APIs in spark and its processing speed makes Spark a popular framework

among data scientists. So, for the data exploration, Spark uses SQL shell. MLlib supports data analysis and Machine Learning. It lets the data scientist handle the problem with large data size.

Hence Spark provides a simple way to parallelize the applications across Clusters and hides the complexity of distributed systems programming, network developer, and fault tolerance.

References

[1] Douglas Eadline. *Hadoop 2 Quick-Start Guide: Learn the Essentials of Big Data Computing in the Apache Hadoop 2 Ecosystem*. Addison-Wesley Professional, 2015.

[2] Apache Software Foundation. Apache spark official site.

[3] Kaggle. New york city taxi fare prediction.

[4] Yandong Wang, Robin Goldstone, Weikuan Yu, and Teng Wang. Characterization and optimization of memory-resident mapreduce on hpc systems. In *2014 IEEE 28th International Parallel and Distributed Processing Symposium*, pages 799–808. IEEE, 2014.

[5] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J Franklin, Scott Shenker, and Ion Stoica. 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*, pages 2–2. USENIX Association, 2012.

[6] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets.