International Journal of Engineering Trends and Technology
ISSN: 2231 – 5381 /doi:10.14445/22315381/IJETT-V69I1P206
Volume 69 Issue 1, 35-42, January 2021

Implementation of a Big Data Architecture For The Realization of Predictive Models With Great Volumes of Data

Enrique Lee Huamaní[1], Avid Roman-Gonzalez[2]

[1,2]*Image Processing Research Laboratory (INTI-Lab), Universidad, de Ciencias y Humanidades, Lima, Perú*

[1]ehuamaniu@uch.edu.pe, [2]avid.roman-gonzalez@ieee.org

Abstract *- The research direction of the University of Sciences and Humanities has integrated a Big Data architecture to make predictive models with large volumes of data. Therefore it was implemented with the purpose that in future research, this architecture can be used efficiently. In this study, the theoretical concepts of Hadoop version 2.0 will be discussed, as well as the next scalability in a Beowulf cluster implemented in one of the University's laboratories and the configuration of Hadoop Spark and how they were able to work in conjunction. Finally, in the results section, tests will be carried out to validate that this architecture works perfectly.*

Keywords: *Big Data, Hadoop, Spark, Predictive models, HDFS.*

I. INTRODUCTION

Data science is changing, and with it, the amount of existing information in the world; data sizes have evolved over the years from a few kilobytes to exabytes [1] therefore, nowadays, it has become essential to deal with large volumes of information using new tools that we have at our disposal since traditional tools do not meet expectations due to their limitations. Consequently, at present, there are methods for dealing with large volumes of information, and this is how the terminology of Big Data was born, where the adjective "Big" refers to the great amount of data [2]. As an overview, when we talk about Big Data, we refer to data sets whose size, complexity, and growth speed make it difficult to capture, manage and process or analyze them using conventional technologies and instruments. When thinking about making predictive models, there is always the need to deal with large volumes of data. The greater the amount of data, the more accurate the prediction is; thus, the Big Data associated with data science is the most suitable to work with. What makes Big Data so useful is the reduction of cost, it is much faster and allows us to make better decisions, and with it, we can provide new products and services.

II. BACKGROUND

Research related to Big Data Hadoop architecture in Universities in Peru is an increasingly recurrent topic as we have the case of the National University of Engineering that has created a Hadoop cluster for performance testing using GPUs and CPUs [5] as well as work related to the vehicle system in Peru applying the machine Learning and the Big Data [6]. In addition, there have been investigations that have combined HPC and Big Data fields as there is one of the projects funded by the Peruvian National Council for Science, Technology and Technological Innovation (Concytec by its Spanish acronym) [7]. Big Data is becoming essential in Peru. There are works such as the University of the Pacific where a greater effort among all university students to obtain the knowledge to know how to use these types of architecture was given [8]. Regarding work performed abroad, there are many examples, among which we have the intensive analysis of Big Data under the Netflix platform, which performs an analysis of how its algorithm works and chooses the best options for the consumer [9], as well as there are investigations related to astronomy in charge of determining the movement of celestial bodies by applying intensive techniques with the MPI libraries and in obtaining information by applying the Big Data Hadoop [10].

III. METHODOLOGY

In this work, the implementation of a Big Data Hadoop architecture for the realization of predictive models using Spark will be discussed to carry out more complex research in the future. As shown in Fig. 1, we contemplate all that can be integrated using Hadoop 2.0. Moreover, in this work, we will concentrate on the distributed programming with Spark, following we will explain the general concepts of Hadoop and its elements for its operation.

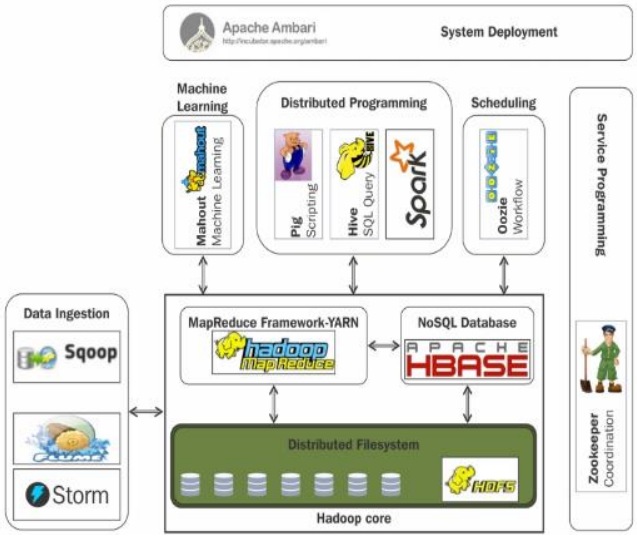

Fig. 1 Hadoop ecosystem

A. Hadoop Framework

One of the most widely used tools for handling large volumes of data is the Hadoop framework in its version 2.0, Java-based open-source software to store data and run applications on basic hardware clusters. This framework operates using two concepts: data storage in Hadoop Distributed File System (HDFS) and Yarn that will be a fundamental part of the Hadoop 2.0 ecosystem. Traditionally in version 1.0, MapReduce was used as the only process manager. However, in the new version, there are more options for processing. Figure 2 shows a comparison between the Hadoop 1 and 2 ecosystem showing that in version 2, we have many benefits because we can integrate it with some tools that had been used previously, such as the MPI, which was implemented in one of the laboratories of the Universidad de Ciencias y Humanidades.

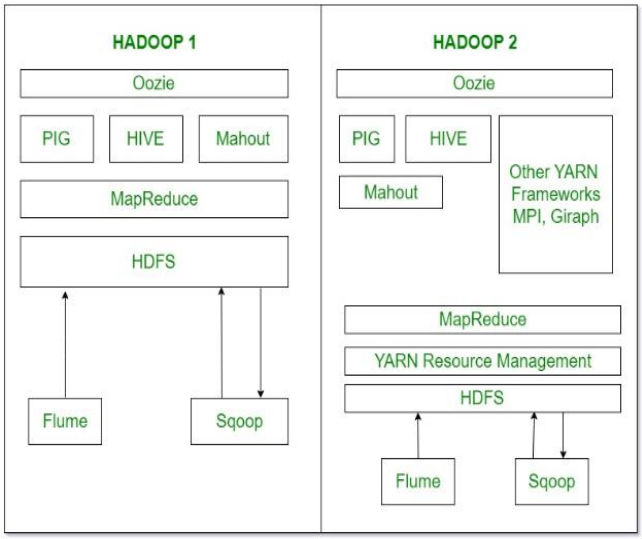

Fig. 2 Hadoop 1.0 and Hadoop 2.0 comparison

a) Hadoop Distributed File System: The Hadoop distributed File System with its acronym HDFS is a distributed file system from Hadoop, its capacity is to store files in a cluster of several machines [11]; this feature is important when trying to store large amounts of data as it is generally not possible to store hundreds of terabytes or petabytes on a single machine. The Hadoop cluster data is divided into smaller parts called blocks and then distributed throughout the cluster; blocks and block copies are stored on other servers. HDFS follows a policy to distribute the blocks in Hadoop [12] to achieve block distribution. There are 3 main elements; NameNode, DataNode, and the HDFS client [13]. Figure 3 illustrates the idea between the Name Node that will take the orchestrator computer's role and the Data Nodes where the processing and storage of information will occur.

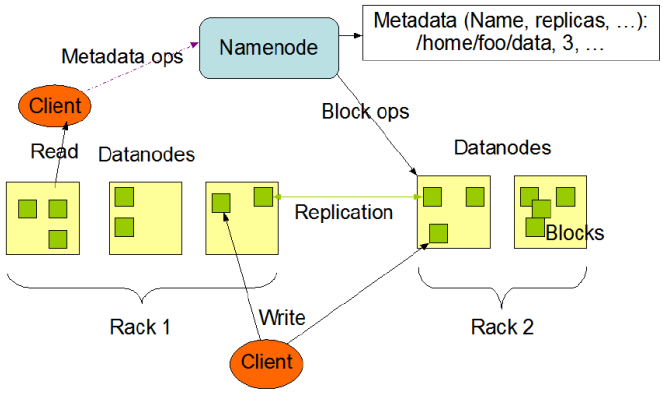

Fig. 3 HDFS Architecture

b) Another Resource Negotiator: It is also known as Yarn because of its acronym, it is a cluster management technology used since the second version of Hadoop. It is currently characterized as a large-scale distributed operating system for Big Data applications. Yarn allows Hadoop to support more varied processing approaches and a wider range of applications [14]. For example, Hadoop clusters can perform interactive queries and data application transmissions simultaneously with the MapReduce block jobs, as well as in the documentation extracted from the Cloudera website [15]. For example, Hadoop clusters can now make interactive queries and data application transmissions simultaneously with MapReduce's batch jobs, as well as in the documentation extracted from the Cloudera website, which gives us, as shown in Fig. 4, the processing managers that can be used. We have HPC OpenMpi, which in previous research used parallel processing for rendering farms, as can be seen in [16]. Likewise, with respect to other previous research, we have the use artificial intelligence using Machine Learning to predict terrorist attacks [17]. In this research, it was argued as future work to be able to use the large volumes of data in social networks to perform with a greater degree of accuracy on suspected cyber terrorism attacks, so concerning YARN would be useful to use Spark as it is easily adaptable to the model that has been developed.

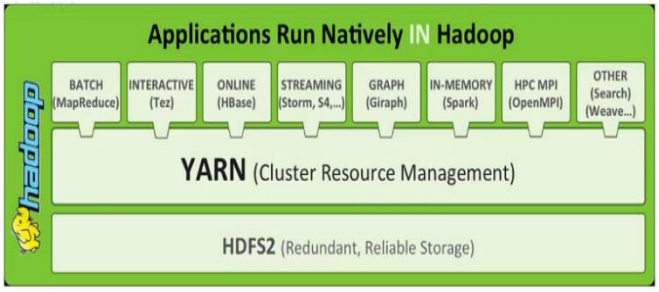

Fig. 4 Information delivery by HDFS

B. Hadoop Yarn Behavior

The yarn has become a fundamental part of the Hadoop ecosystem. It allows support for several execution engines, including MapReduce. YARN's main idea is to split the resource management and scheduling functionality as well as work supervision into separate daemons. As shown in Fig. 5, extracted from the research of [18], the functionality of Yarn is composed of a global ResourceManager (RM) and an ApplicationMaster (AM) per application, where one application is a single job. The ResourceManager and NodeManager form the data calculation framework. The ResourceManager is the highest authority that arbitrates resources between all the applications on the system. Simultaneously, the NodeManager is the frame agent per team responsible for the containers, monitoring the use of the resources (CPU, disk memory, network), and reporting it to the Resource Manager /Scheduler. The ApplicationMaster by Application is a framework-specific library. It has the task of negotiating the resources of the ResourceManager and working with the NodeManager to execute and monitor the tasks.

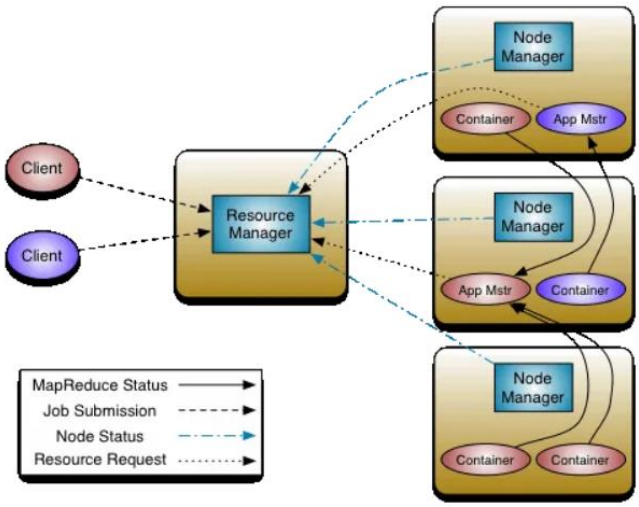

Fig. 5 Hadoop Yarn Architecture

ResourceManager contains two main components: Schedule and ApplicationManager.

***a) Schedule*:** It has a connectable policy responsible for dividing the resources of the cluster between various queues, applications, etc. The current programmers, like CapacityScheluder and Faircheduler, would be some examples of compliments.

***b) ApplicationManager*:** It is responsible for accepting job submissions, negotiating the first container to run the application-specific ApplicationMaster, and providing the service to restart the ApplicationMaster container in case of failure. The ApplicationMaster, per application, has the responsibility to negotiate appropriate resource containers from the Scheduler, track their status, and monitor their progress.

C. Hadoop scalability in a Beowulf cluster

A Beowulf cluster was implemented in a previous investigation regarding [19], performing highly complex scientific processes using the Open Mpi libraries through the SHH protocol. However, due to loss of communication, operating system failure, or other errors in one of the nodes, the process may be unfinished; therefore, Hadoop will be used with the Beowulf architecture as a proposal for future integration since the version of Hadoop 2.0 has the yarn to be able to use the MPI and have a high availability Beowulf cluster. Concerning this research, related to predictive models with large volumes of data, we will also take advantage of it because we will have a greater amount of data node for scalable storage and distribution process, to know more about the architecture and power of the Beowulf cluster. Fig. 6 shows the physical architecture in which we work, and Fig. 7 shows a graph extracted from previous research. With this architecture, Beowulf has planned to scale the current Big Data architecture in the future; in Table I, we can see the characteristics of each of the computers that are performing the process.

Fig.6 Beowulf Cluster of the University of Sciences and Humanities

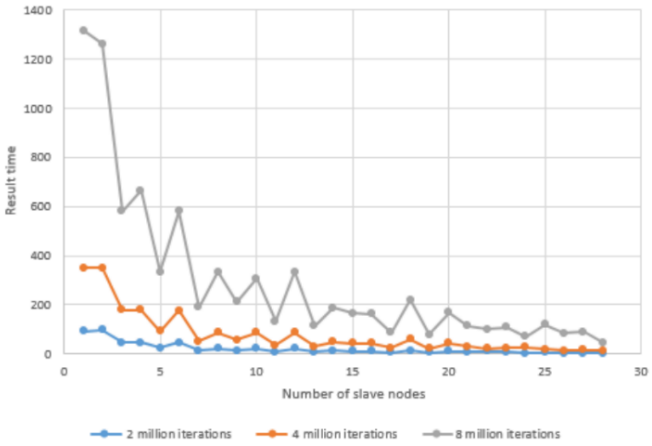

Fig. 7 Performance Test Comparison

Table 1. Hardware characteristics of the beowulf cluster

Components	Description
Modell	HP EliteDesk 800 G1 SFF
HDD	HDD 1 TB
RAM	RAM 8 GB
Processor	Intel(R) Core (TM) i7-8700
Total Cores	12
Type of Operating System	64-bit
Operative System	Ubuntu 18.04

D. Hadoop framework implementation

To operate the Hadoop cluster, configurations will have to be made using the Hadoop packaging extracted from its official website. Then the following traditional ones will have to be made to their configuration.

***a) Installation of the Java JDK package*:** The origin of Hadoop is Java so it is installed with an apt-get update for missing ubuntu updates as shown in (1) (2) (3).

apt-get update (1)

apt-get install degault-jdk (2)

update-alternative -config java (3)

In this the Java JDK installation was placed in the file /usr/lib/jvm/java-11-openjdk-amd64 to then introduce "export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64" with the following command from the terminal (4) finally using the reboot command.

vi /etc/profile.d/java.sh (4)

***b) Hadoop user creation*:** Before proceeding with the configuration, it is recommended to create a new user to handle the predictive models with Hadoop (5).

adduser hadoop (5)

***c) RSA key generation*:** When a Hadoop cluster is assembled initially, it is necessary to generate an RSA key, but since this work generated a unique key for using the Beowulf cluster, this key will be reiterated for the use of the Big Data architecture.

d) Hadoop framework installation: After downloading and unzipping the Hadoop, the file must be moved to the path /usr/local/Hadoop, then the property (6) so that the Hadoop can be operated normally.

vi /etc/profile.d/hadoop.sh (6)

The following content will be included in the Hadoop. Sh as shown in Fig. 8.

```
#/bin/bash
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export YARN_HOME=$HADOOP_HOME
export PATH="$PATH:${HADOOP_HOME}/bin:${HADOOP_HOME}/sbin"
```

Fig. 8. Hadoop. sh configuration

Following the Hadoop configuration, you must set the environment variable in path (7) by entering the code line (8) to reboot the system finally.

vi /usr/local/hadoop/etc/hadoop/hadoop-env.sh (7)

export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64 (8)

e) Hadoop node configuration : Then, the environment of the hadoop framework must be configured in the corresponding files, which are in xml format.

1) Namenode configuration: To configure the namenode for the Hadoop cluster operation, you will have to configure the path (9) and edit it as shown in Fig. 9.

vi /usr/local/hadoop/etc/hadoop/core-site.xml (9)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<property>
        <name>fs.default.name</name>
        <value>hdfs://localhost:9000</value>
        <description>The default file system URI</description>
</property>
</configuration>
```

Fig. 9. Hadoop Namenode Configuration

2) HDFS storage configuration: Afterwards, the path where the HDFS storage will be done must be configured by editing the following configuration (10) in Hadoop. In Fig. 10, a replication value of 3 is shown; this means that the data volume is divided into replication blocks into 3 different containers. In the future, when the Beowulf cluster architecture is used, the amount of replication will increase.

vi /usr/local/hadoop/etc/hadoop/hdfs-site.xml (10)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<property>
      <name>dfs.replication</name>
      <value>3</value>
   </property>

   <property>
      <name>dfs.name.dir</name>
      <value>file:///hadoop/hdfs/namenode</value>
   </property>

   <property>
      <name>dfs.data.dir</name>
      <value>file:///hadoop/hdfs/datanode</value>
   </property>
</configuration>
```

Fig. 10. HDFS Configuration

3) MapReduce and Yarn configuration: Since we are working with Hadoop version 2.0, it must be indicated that the MapReduce will be integrated with the YARN. Therefore the following path is entered modifying the xml file (11), and the following configuration will be added as shown in Fig. 11. The YARN configuration is edited in the following path (12), modifying the xml as in Fig. 12.

vi /usr/local/hadoop/etc/hadoop/mapred-site.xml (11)

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<property>
  <name>mapreduce.framework.name</name>
   <value>yarn</value>
 </property>
</configuration>
```

Fig. 11. XML MapReduce Configuration

vi /usr/local/hadoop/etc/hadoop/yarn-site.xml (12)

```
<?xml version="1.0"?>
<configuration>
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
<!-- Site specific YARN configuration properties -->

</configuration>
```

Fig. 12. XML Yarn Configuration

4) Creation of directories, formatting, and starting the Hadoop framework: To complete the Hadoop configuration, 2 important directories must be created for the information storage, these are the data node and the namenode, with the following command and declaring permissions for its storage from the terminal (13) (14) (15).

mkdir -p /hadoop/hdfs/namenode (13)

mkdir -p /hadoop/hdfs/datanode (14)

chown -R hadoop:hadoop /hadoop (15)

When all configurations are done to the XML, the terminal must enter the path (16) and format it with the command ./hdfs namenode -format to finally start the services entered in the path (17) using the commands ./start-dfs.sh and ./start-yarn.sh to display from the web. Fig13 shows that Hadoop works successfully to make predictive models using large volumes of data.

cd /usr/local/hadoop/bin (16)

cd $HADOOP_HOME/sbin/ (17)

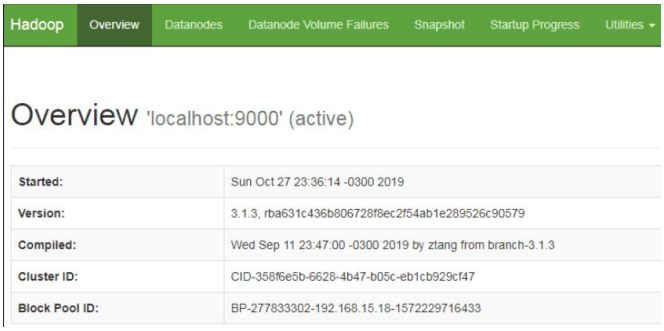

Fig. 13. Hadoop in progress

E. Spark framework implementation

To make predictive models with large volumes of data, we can easily use it using the Hadoop framework without an additional framework. However, this research intends to get the maximum results with tools that provide results in the shortest time possible. Therefore, Hadoop, through YARN,

will use Apache Spark, which is an open-source cluster computing framework that offers the advantage of having a fast and general engine used to analyze large scale data, in addition to the fact that one of its most relevant characteristics is that the data is stored in memory and not in the disk, making it much faster than the traditional Hadoop [20]. In this study, Apache Spark can mention many things, but the main idea is to integrate it with Hadoop. Therefore, below we show Spark's configuration with Hadoop to be stored in HDFS in each of the data nodes.

a) Scale installation: Spark worked under the scale language; therefore should be downloaded at [21], which corresponds to the official site, which will have a file scala-2.13.4.tgz and then decompress it using the command in console tar xvf scala-2.12.7.tgz. Once the file has been decompressed, it is moved to the file /usr/local/, and then an environment variable is created in .bashrc, and the following line of code is added: export SCALA_HOME=/usr/local/scala-2.13. 4, with this line, you indicate where Scala is installed. Following, inside the PATH line, all the binary files inside the scala/home path should be shown with the following sentence in the same bashrc export PATH=$PATH:$SCALA_HOME/bin. As we already have the Hadoop installed, we add code export PATH=$PATH:$HADOOP_HOME/bin:$SCALA_HOME/bin. To finish this process, the environment is updated with the data entered with the following command from the terminal source ~/.bashrc.

b) Spark installation: Once Scala has been installed, we will proceed to install Spark using the following link [22], then you must decompress it with tar xvf ./spark-2.4.0-bin-hadoop2.7.tgz and move it to /usr/local, then edit bashrc one more time and add export SPARK_HOME=/usr/local/spark-2.4.0-bin-hadoop2.7. In addition, we indicate where the Spark executables are by adding a PATH with the following export sentence PATH=$PATH:$HADOOP_HOME/bin:$SCALA_HOME/bin:$SPARK_HOME/bin. Finally, we updated the environment with source ~/.bashrc, and to verify that the spark is operational, we head to the spark-submit –version. Having made all configurations, we can use Spark to use predictive models with large data volumes using Hadoop's HDFS as a storage medium since we are having a communication with the YARN for distributed processing.

IV. RESULTS

To determine how functional the Spark will be using Hadoop, predictive models will be made, and the Apache Spark Mlib will be used to create an automatic learning application. The application being tested will perform predictive analysis of an open data set from Spark's built-in self-learning libraries, in this example using logistic regression for classification. In this case, the development will be worked by notebook Jupyter, as shown below.

```
from pyspark.ml import Pipeline
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.feature import HashingTF, Tokenizer
from pyspark.sql import Row
from pyspark.sql.functions import UserDefinedFunction
from pyspark.sql.types import *
def csvParse(s):
import csv
from StringIO import StringIO
sio = StringIO(s)
value = csv.reader(sio).next()
sio.close()
return value
inspections=sc.textFile('/HdiSamples/HdiSamples/FoodInspectionData/Food_Inspections1.csv')\ .map(csvParse)
inspections.take(1)
```

Developing the code shows the following result and indicates that everything until this point is functioning properly.

```
[['707413',
'CENTRAL PARK  ACADEMY',
 'CENTRAL PARK DAY CARE',
'2049789',
"Children's Services Facility",
'Rocks 1 (Batch)',
'1850 FOREST DAY VALLAGE ',
'NEW YERSEY',
'XL',
'15254',
 '08/20/2009',
'License Copernicus',
fail,
'Comment none. ',
'42.78349412541566',
'-40.26896247521587',
'('42.78349412541566', -40.26896247521587)']]
```

As shown in the code, the 4 columns of interest that will be used are id, name, results, and violations; in the last line of code, it was indicated that it shows the first 5 results, so we

will have the results shown in Table II.

Table 2. The first 5 data for prediction with spark

id	name	result	violation
707413	CENTRAL PARK ACADEMY	fail	15. CLEAN THE FLOOR
142562	HERMIT COFFEE	fail	2. FACILITIES TO ...
159854	QUICK PIZZA	fail	
859657	PREMIERE HOSTEL	fail	
358658	OP BURGER	fail	

To predict the results, the data set must be properly understood. Therefore the data frame already analyzed must be based on a model that corresponds to our data set and is constructed according to the violations during food inspection. Therefore, we will perform a logistical regression that is a binary classification method, and the results data must be grouped into two categories: Failed and Approved. Afterward, a code is executed to convert the existing data frame (df) into a new one, and each inspection will be presented as a pair of labels-villations. Label 0.0 represents an error, label 1.0 represents success, and label -1.0 will represent some results in addition to these two results.

```
def labelForResult(s):

if s == 'Pass Condition or s == 'Pass':

        return “1.0”

elif s == 'Fail':

    return “0.0”

else:

    return “-1.0”

label = UserDefFunction(labelForResult, DoubleType())

labeledData = df.select(label(df.result).alias('label'),

df.violation).where('label >= 0')
```

A logistic regression model is then created from the input data frame.

```
tokenizer=Tokenizer(inputCol="violations",outputCol="words")

hashingTF = HashingTF(inputCol=tokenizer.getOutputCol(), outputCol="features")

lr = LogisticRegression(maxIter=10, regParam=0.01)

pipeline = Pipeline(stages=[tokenizer, hashingTF, lr])

 model = pipeline.fit(labeledData)

testData=sc.textFile('wasbs:///HdiSamples/HdiSamples/FoodInspectionData/Food_Inspections2.csv')\

 .map(csvParse) \ .map(lambda l: (int(l[0]), l[1], l[12], l[13]))

testDf=spark.createDataFrame(testData, schema).where("results = 'Fail' OR results = 'Pass' OR results = 'Pass w/ Conditions'")

predictionsDf = model.transform(testDf)

predictionsDf.registerTempTable('Predictions')
```

Lastly, to visualize a prediction, the following sentence is placed predictionsDf.take(1) from which the output will show us in the following way.

“There were 9315 inspections, and there were 8087 successful predictions.”

“This is an 86.8169618894% success rate.”

As a visual representation of a prediction, we will use the following syntax shown in Fig.14 to finally use the following code fraction to generate a graphic using Matplotlib. The result is shown in Fig. 15.

```
%%sql -q -o true_positive SELECT count(*) AS cnt FROM Predictions WHERE prediction = 0 AND results = 'Fail'

%%sql -q -o false_positive SELECT count(*) AS cnt FROM Predictions WHERE prediction = 0 AND (results = 'Pass' OR results = 'Pass w/ Conditions')

%%sql -q -o true_negative SELECT count(*) AS cnt FROM Predictions WHERE prediction = 1 AND results = 'Fail'

%%sql -q -o false_negative SELECT count(*) AS cnt FROM Predictions WHERE prediction = 1 AND (results = 'Pass' OR results = 'Pass w/ Conditions')
```

Fig. 14. Hadoop in progress

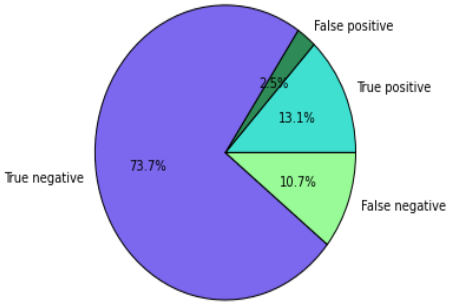

Fig. 15. Prediction result using Matplotlib

V. DISCUSSION AND CONCLUSIONS

Developing a Hadoop architecture to make predictive models using Spark is a proposal that will innovate in the direction of research at the University of Sciences and Humanities as each year the institution's research is increasingly intensive. As explained in the methodology section, this work can be improved if a Beowulf cluster is established; the idea is to use the Hadoop architecture with all the computers connected to the same network in the future. This would provide a powerful architecture to provide services and apply for funding projects that exist in Peru and make the University a pioneer in having Big Data architectures. In the short term, this work is intended to integrate other technologies to do various functions, such as extracting information in real time and including software to see the state of the Hadoop architecture. One of the projects planned is Ambari, so this work is the beginning of many jobs that will be performed.

We can conclude that the integration of the Big data Hadoop architecture was a success and that it could be integrated with Spark to make predictive models. In this case, we used pyspark, which was highly useful for creating predictive models that are easy to understand, as has been shown in this work. We can also indicate that by using existing resources from the University of Sciences and Humanities, the cost of the integration was low, in contrast to buying a pre-installed Hadoop service or being able to use the AWS cloud, which is of great benefit to researchers who seek to use this architecture.

REFERENCES

[1] T. D. Wemegah and S. Zhu, Big data challenges in transportation: A case study of traffic volume count from massive Radio Frequency Identification(RFID) data, Conf. Proc. - 2017 Int. Conf. Front. Adv. Data Sci. FADS (2017) 58–63.

[2] T. J. Barnes, Big data, a little history, Dialogues Hum. Geogr., 3(3)(2013) 297–302.

[3] G. Sharma and A. Ganpati, Performance evaluation of fair and capacity scheduling in Hadoop YARN, Proc. 2015 Int. Conf. Green Comput. Internet Things, ICGCIoT (2015),904–906.

[4] A. Wakde, P. Shende, S. Waydande, S. Uttarwar, and G. Deshmukh, ,Comparative Analysis of Hadoop Tools and Spark Technology, Proc. - 2018 4th Int. Conf. Comput. Commun. Control Autom. ICCUBEA(2018) 1–4.

[5] N. M. Lapa Romero, J. A. Fiestas Iquira, A. Tenorio Trigoso, and Y. Nuñez Medrano, Pruebas de rendimiento sobre el Clúster de CPUs y GPUs empleando simulación N-body, (2018) 19–21.

[6] G. Bravo-Rocca, P. Torres-Robatty, and J. Fiestas-Iquira, Sparkmach: A distributed data processing system based on automated machine learning for big data, Commun. Comput. Inf. Sci., 898(2019) 121–128.

[7] I. Ocampo and L. Exequiel, INTRODUCCIÓN A LA SUPERCOMPUTACIÓN EN EL PERU, 39(5)(2017).

[8] M. Nunez-del-Prado, M. Rodriguez, and Ieee, Big Data Analytics Labs in the Cloud Spaces for Teamwork, 2017 7th World Eng. Educ. Forum, (2017) 499–503.

[9] S. Maddodi and K. P. K,Netflix Bigdata Analytics- The Emergence of Data-Driven Recommendation, SSRN Electron. J., 3(2)(2019) 41–51.

[10] J. Fiestas, O. Porth, P. Berczik, and R. Spurzem, Evolution of growing black holes in axisymmetric galaxy cores, Mon. Not. R. Astron. Soc., 419(2012) 57–69.

[11] A. Siretskiy and O. Spjuth, HTSeq-Hadoop: Extending HTSeq for massively parallel sequencing data analysis using Hadoop, Proc. - 2014 IEEE 10th Int. Conf. eScience, eScience (2014),1,317–323.

[12] A. Shah and M. Padole, Load Balancing through Block Rearrangement Policy for Hadoop Heterogeneous Cluster, 2018 Int. Conf. Adv. Comput. Commun. Informatics, (2018) 230–236.

[13] C. Verma and R. Pandey, Comparative analysis of GFS and HDFS: Technology and architectural landscape, Proc. - 2018 10th Int. Conf. Comput. Intell. Commun. Networks, CICN,(2018) 54–58.

[14] T. Subbulakshmi and J. S. Manjaly,A comparison study and performance evaluation of schedulers in Hadoop YARN,Proc. 2nd Int. Conf. Commun. Electron. Syst. ICCES (2018)-Janua, no. Icces, (2018) 78–83.

[15] I. Hortonworks,Data access and data management.[Online]. Available: https://docs.cloudera.com/HDPDocuments/HDP2/HDP-2.1.2/bk_getting-started-guide/content/ch_hdp2_data-access-mgt.html.

[16] E. L. Huamaní, P. Condori, B. Meneses-Claudio, and A. Roman-Gonzalez, “Render farm for highly realistic images in a Beowulf cluster using distributed programming techniques,Int. J. Adv. Comput. Sci. Appl., 10(11)(2019) 407–411.

[17] E. L. Huamaní, A. M. Alicia, and A. Roman-Gonzalez,Machine Learning Techniques to Visualize and Predict Terrorist Attacks Worldwide using the Global Terrorism Database, Int. J. Adv. Comput. Sci. Appl.,11,(4)(2020) 562–570.

[18] H. Geng, Internet of things and data analytics handbook.(2017).

[19] E. L. Huamaní, P. Condori, and A. Roman-Gonzalez, “Implementation of a Beowulf Cluster and Analysis of its Performance in Applications with Parallel Programming,Int. J. Adv. Comput. Sci. Appl., 10(8)(2019) 522–527.

[20] A. V. Hazarika, G. Jagadeesh Sai Raghu Ram, and E. Jain, Performance Comparision of Hadoop and spark engine, Proc. Int. Conf. IoT Soc. Mobile, Anal. Cloud, I-SMAC (2017) 671–674.

[21] Scala, Scala Downloads, [Online]. Available: https://scala-lang.org/files/archive/.(2020).

[22] T. A. S. Foundation, The Apache Software Foundation. [Online]. Available: https://spark.apache.org/downloads.html.