

كلية العلوم
ⵜⴰⵎⴻⵔⴰⵏⵜ ⵏ ⵜⴰⵎⴻⵔⴰⵏⵜ
FACULTÉ DES SCIENCES



UNIVERSITE IBN ZOHR FACULTE DES SCIENCES

Département Informatique

Master de recherche de science des données

Projet Hadoop et Spark

Présenté par :

- Ousayd Lahcen
- Bentaher Noura
- Elkhalfaoui Yasmina
- Saidi Souad

Encadré par : Mr.Kabbadj

Année universitaire 2021-2022

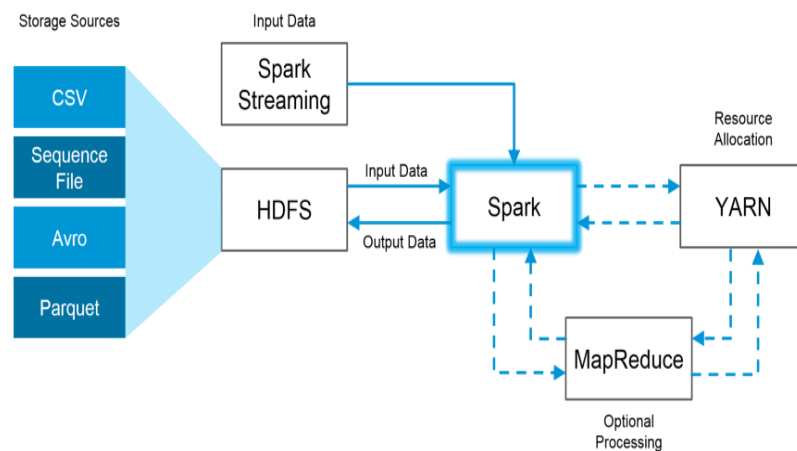
I- Introduction :

Apache Hadoop et Apache Spark sont tous les deux des Frameworks open-source pour le traitement des données volumineuses avec quelques différences clés.

Hadoop utilise MapReduce pour traiter les données, tandis que Spark utilise des ensembles de données distribués résilients (RDD). Hadoop dispose d'un système de fichiers distribués (HDFS), ce qui signifie que les fichiers de données peuvent être stockés sur plusieurs machines.

Le système de fichiers est évolutif, car des serveurs et des machines peuvent être ajoutés pour accueillir des volumes croissants de données. Spark ne fournit pas de système de stockage de fichiers distribué, il est donc principalement utilisé pour le calcul.

La bonne nouvelle est que Spark est entièrement compatible avec Hadoop et fonctionne sans problème avec Hadoop Distributed File System (HDFS). Ainsi, lorsque la taille des données est trop grande pour que Spark puisse les gérer en mémoire, Hadoop peut aider à surmonter cet obstacle grâce à sa fonctionnalité HDFS. Voici un exemple visuel de la façon dont Spark et Hadoop peuvent travailler ensemble :

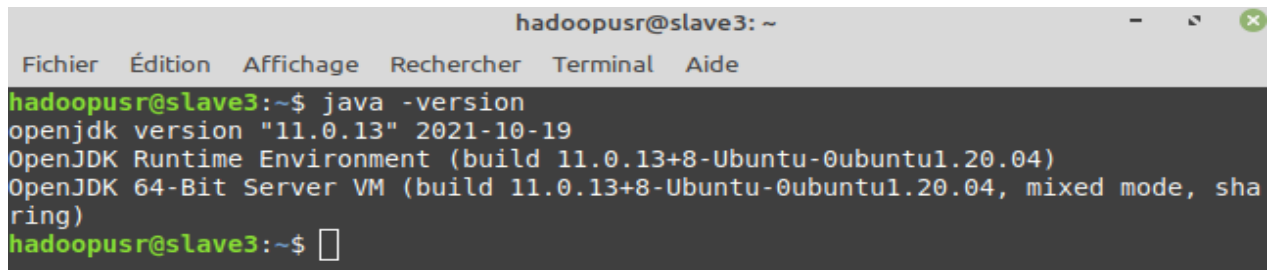


L'image ci-dessus montre comment Spark utilise les meilleures parties de Hadoop à travers HDFS pour la lecture et le stockage des données, MapReduce pour le traitement optionnel et YARN pour l'allocation des ressources.

II- Installation du Hadoop :

Étape 1 : Installation de Java :

Java est le prérequis principal pour Hadoop. Tout d'abord, vous devez vérifier l'existence de java dans votre système en utilisant « java -version ».



```
hadoopusr@slave3: ~  
Fichier  Édition  Affichage  Rechercher  Terminal  Aide  
hadoopusr@slave3:~$ java -version  
openjdk version "11.0.13" 2021-10-19  
OpenJDK Runtime Environment (build 11.0.13+8-Ubuntu-0ubuntu1.20.04)  
OpenJDK 64-Bit Server VM (build 11.0.13+8-Ubuntu-0ubuntu1.20.04, mixed mode, sha  
ring)  
hadoopusr@slave3:~$
```

Étape 2 : Après la vérification de l'installation de java, nous avons besoin d'un utilisateur dédié pour l'installation de Hadoop avec le nom hadoopusr, on utilisant les commandes suivante :

- **sudo addgroup hadoop**
- **sudo adduser --ingroup hadoop hadoopusr**

Étape 3 : maintenant on ajoute l'utilisateur « hadoopusr » au groupe « sudo »

- **sudo adduser hadoopusr sudo**

Étape 4 : Une fois que vous avez téléchargé hadoop-2.9.0.tar.gz , maintenant nous extrayons ce fichier avec la commande ci-dessous :

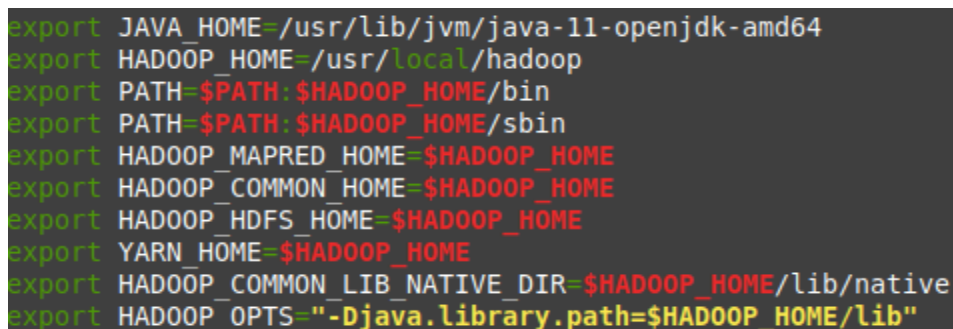
- **sudo tar -xvzf hadoop-2.9.0.tar.gz**

Étape 6: Maintenant, nous devons déplacer ce dossier extrait vers l'utilisateur hadoopusr et puis nous devons changer la propriété de ce dossier :

- **sudo mv hadoop /usr/local/hadoop**
- **sudo chown -R hadoopusr /usr/local**

Étape 7 : configurer notre fichier . /bashrc .

- **sudo nano ~/.bashrc**
- **Ajouter les lignes suivant :**



```
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64  
export HADOOP_HOME=/usr/local/hadoop  
export PATH=$PATH:$HADOOP_HOME/bin  
export PATH=$PATH:$HADOOP_HOME/sbin  
export HADOOP_MAPRED_HOME=$HADOOP_HOME  
export HADOOP_COMMON_HOME=$HADOOP_HOME  
export HADOOP_HDFS_HOME=$HADOOP_HOME  
export YARN_HOME=$HADOOP_HOME  
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native  
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
```

- **source ~/.bashrc**

Étape 8:

- `sudo nano /usr/local/hadoop/etc/hadoop/hadoop-env.sh`
- Ajouter `JAVA_HOME`

```
# The java implementation to use.
#export JAVA_HOME=${JAVA_HOME}
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
```

Étape 9 : Pour les autres fichiers de configuration de Hadoop , on va le configure dans la deuxième partie

III- Configuration multi-nœuds (master et slaves) :

1. Mettre à jour le nom d'hôte on master et slaves.

Machine master nommée master , les machines slaves nommée respectivement slave1, slave2 et slave3 dans notre situation.

2. Mettre à jour fichier `/etc/hosts` on master et slaves :

Ajouter les adresses IPs de machine master et slaves :

```
hadoopusr@slave3: ~
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
GNU nano 4.8 /etc/hosts
127.0.0.1    localhost
127.0.1.1    Yasmina-Linux
192.168.43.32 master
192.168.43.80 slave2
192.168.43.237 slave1
192.168.43.163 slave3
# The following lines are desirable for IPv6 capable hosts
::1        ip6-localhost ip6-loopback
fe00::0    ip6-localnet
ff00::0    ip6-mcastprefix
ff02::1    ip6-allnodes
ff02::2    ip6-allrouters
```

3. Configuration de SSH :

Install SSH dans toutes les machines, pour être capable de communiquer entre eux sans mot de passe :

- `sudo apt-get install openssh-server`

```
@slave1:~$ ssh master
Warning: Permanently added 'master,192.168.220.185' (ECDSA) to the list of known hosts.
Warning: Permanently added 'master,192.168.220.185' (ECDSA) to the list of known hosts.
@master:~$
```

4. Changer fichier **core-site.xml** :

```
GNU nano 4.8 core-site.xml

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://master:9000</value>
</property>
</configuration>
█
```

5. Changer fichier **hdfs-site.xml** :

➤ Dans Master machine :

```
hadoopusr@master: /usr/local/hadoop/etc/hadoop
File Edit View Search Terminal Help
GNU nano 4.8 hdfs-site.xml
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
<name>dfs.replication</name>
<value>3</value>
</property>
<property>
<name>dfs.namenode.name.dir</name>
<value>file:/usr/local/hadoop_tmp/hdfs/namenode</value>
</property>
</configuration>
█
```

➤ Dans Slaves machines :

```
GNU nano 4.8 hdfs-site.xml
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
<name>dfs.replication</name>
<value>3</value>
</property>

<property>
<name>dfs.datanode.data.dir</name>
<value>file:/usr/local/hadoop_tmp/hdfs/datanode</value>
</property>
</configuration>
█
```

6. Changer fichier **yarn-site.xml** :

➤ Dans Master et Slaves :

```
hadoopusr@slave1: /usr/local/hadoop/etc/hadoop
File Edit View Search Terminal Help
GNU nano 4.8 yarn-site.xml
] limitations under the License. See accompanying LICENSE file.
-->
<configuration>

<!-- Site specific YARN configuration properties -->
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
  <name>yarn.resourcemanager.resource-tracker.address</name>
  <value>master:8025</value>
</property>

<property>
  <name>yarn.resourcemanager.scheduler.address</name>
  <value>master:8035</value>
</property>

<property>
  <name>yarn.resourcemanager.address</name>
  <value>master:8050</value>
</property>
</configuration>
```

7. Changer les fichiers **masters et slaves** dans Master hôte seulement :

➤ Fichier masters :

```
hadoopusr@master: /usr/local/hadoop/etc/hadoop
File Edit View Search Terminal Help
GNU nano 4.8 masters
master
█
```

➤ Fichiers slaves :

```
hadoopusr@master: /usr/local/hadoop/etc/hadoop
File Edit View Search Terminal Help
GNU nano 4.8 slaves
slave1
slave2
slave3
█
```

8. Créer le dossier namenode uniquement dans master :

- `sudo rm -rf /usr/local/hadoop_tmp`
- `sudo mkdir -p /usr/local/hadoop_tmp/hdfs/namenode`
- `sudo chown hadoopusr : hadoop -R /usr/local/hadoop_tmp/`
- `sudo chmod 777 /usr/local/hadoop_tmp/hdfs/namenode`

9. Créer le dossier datanode uniquement dans slaves :

- `sudo rm -rf /usr/local/hadoop_tmp`
- `sudo mkdir -p /usr/local/hadoop_tmp/hdfs/datanode`
- `sudo chown hadoopusr : hadoop -R /usr/local/hadoop_tmp/`
- `sudo chmod 777 /usr/local/hadoop_tmp/hdfs/datanode`

10. Formater le Namenode (Master uniquement) :

- `hdfs namenode -format`

11. Start the dfs & Yarn (Master uniquement):

- `start-all.sh`

12. Jps dans master et slaves :

```
hadoopusr@master:/usr/local/hadoop/etc/hadoop$ jps
4482 Jps
4217 ResourceManager
3853 NameNode
4062 SecondaryNameNode
```

```
hadoopusr@slave1:~$ jps
6866 Jps
6084 NodeManager
5948 DataNode
```

<http://master:8088>

hadoop

All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved
0	0	0	0	0	0 B	24 GB	0 B	0	24	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
3	0	0	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

Show 20 entries

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCoers	Allocated Memory MB	Reserved CPU VCoers	Reserved Memory MB	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes
No data available in table																			

Showing 0 to 0 of 0 entries

First Previous Next Last

http://master :50070

➤ Afficher Namenode :

Hadoop

Overview

Datanodes

Datanode Volume Failures

Snapshot

Startup Progress

Utilities -

Overview

'master:9000' (active)

Started:	Thu Jan 06 09:54:43 +0100 2022
Version:	2.9.0, r756ebc8394e473ac25feac05fa493f6d612e6c50
Compiled:	Mon Nov 13 23:15:00 +0000 2017 by arsureh from branch-2.9.0
Cluster ID:	CID-b2827030-0385-4cb0-a67d-ce747f84f418
Block Pool ID:	BP-1642398829-192.168.43.32-1641394638722

Summary

Security is off.
Safemode is off.
5 files and directories, 4 blocks = 9 total filesystem object(s).
Heap Memory used 66.16 MB of 124 MB Heap Memory. Max Heap Memory is 1000 MB.
Non Heap Memory used 62.64 MB of 65.55 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	25.93 GB
DFS Used:	1.05 MB (0%)
Non DFS Used:	11.63 GB
DFS Remaining:	12.96 GB (49.96%)
Block Pool Used:	1.05 MB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%

➤ Afficher datanode disponible :

Node	Http Address	Last contact	Last Block Report	Capacity	Blocks	Block pool used	Version
✓ slave1:50010 (192.168.43.237:50010)	http://slave1:50075	0s	13m	95.62 GB	0	32 KB (0%)	2.9.0
✓ slave2:50010 (192.168.43.80:50010)	http://slave2:50075	0s	13m	25.93 GB	2	84.14 KB (0%)	2.9.0
✓ slave3:50010 (192.168.43.163:50010)	http://slave3:50075	2s	13m	36.53 GB	2	94.12 KB (0%)	2.9.0

IV- Configuration du Spark :

1. Installation du Spark :

- Extraire le package:
tar -xvf spark-3.0.3-bin-hadoop2.7.tgz
mv spark-3.0.3-bin-hadoop2.7 /usr/local/spark
- Ajouter les ligne suivante au fichier ~/.bashrc :

```
export SPARK_HOME=/usr/local/spark
export PATH=$PATH:$SPARK_HOME/bin
```

- Les deux lignes suivantes utilisant pour intégrer Spark avec Yarn Hadoop :

```
export HADOOP_CONF_DIR=/usr/local/hadoop/etc/hadoop
export LD_LIBRARY_PATH=/usr/local/hadoop/lib/native:$LD_LIBRARY_PATH
```

- Mettre à jour les fichiers suivants :

1- spark-env.sh :

```
HADOOP_CONF_DIR="/usr/local/hadoop/etc/hadoop"
SPARK_YARN_QUEUE="default"
SPARK_MASTER_HOST=master
SPARK_DRIVER_MEMORY=10G
```

2- slaves :

```
# A Spark Worker will be started on each of the machines listed below.
master
slave1
slave2
slave3
```

2. Installation du jupyter et configure Pyspark :

Pour installation de jupyter on utilise command suivant :

- pip Install jupyter

PySpark est une interface pour Apache Spark en Python. Alors on va configurer le fichier ~/.bashrc pour pouvoir utiliser jupyter dans Spark .

```
export PYSARK_PYTHON=python3
export PYSARK_DRIVER_PYTHON=jupyter
export PYSARK_DRIVER_PYTHON_OPTS='notebook'
```

3. Start Spark (master uniquement) :

```
hadoopusr@master:~$ cd /usr/local/spark/sbin
hadoopusr@master:/usr/local/spark/sbin$ ./start-all.sh
starting org.apache.spark.deploy.master.Master, logging to /usr/local/spark/logs/spark-hadoop
usr-org.apache.spark.deploy.master.Master-1-master.out
slave3: starting org.apache.spark.deploy.worker.Worker, logging to /usr/local/spark/logs/spar
k-hadoopusr-org.apache.spark.deploy.worker.Worker-1-slave3.out
master: starting org.apache.spark.deploy.worker.Worker, logging to /usr/local/spark/logs/spar
k-hadoopusr-org.apache.spark.deploy.worker.Worker-1-master.out
slave2: /usr/local/spark/conf/spark-env.sh: line 73: SPARK_MASTER_HOST: command not found
slave2: starting org.apache.spark.deploy.worker.Worker, logging to /usr/local/spark/logs/spar
k-hadoopusr-org.apache.spark.deploy.worker.Worker-1-slave2.out
slave1: starting org.apache.spark.deploy.worker.Worker, logging to /usr/local/spark/logs/spar
k-hadoopusr-org.apache.spark.deploy.worker.Worker-1-slave1.out
hadoopusr@master:/usr/local/spark/sbin$
```

4. Jps après start Spark :

```
hadoopusr@slave3:~$ jps
3478 Jps
2568 NodeManager
2876 Worker
2431 DataNode
```

5. http://master :8080 :

 Spark Master at spark://master:7077

URL: spark://master:7077
Alive Workers: 4
Cores in use: 24 Total, 0 Used
Memory in use: 24.2 GiB Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (4)

Worker id	Address	State	Cores	Memory	Resources
worker-20220103135540-192.168.43.32-36235	192.168.43.32:36235	ALIVE	4 (0 Used)	6.7 GiB (0.0 B Used)	
worker-20220103135610-192.168.43.163-43697	192.168.43.163:43697	ALIVE	12 (0 Used)	6.2 GiB (0.0 B Used)	
worker-20220103135613-192.168.43.80-40303	192.168.43.80:40303	ALIVE	4 (0 Used)	6.7 GiB (0.0 B Used)	
worker-20220103135623-192.168.43.237-34839	192.168.43.237:34839	ALIVE	4 (0 Used)	4.7 GiB (0.0 B Used)	

Running Applications (0)

6. Start Pyspark shell application :

```
hadoopusr@master:/usr/local/spark/sbin$ pyspark --master spark://master:7077
[I 16:09:51.094 NotebookApp] Serving notebooks from local directory: /usr/local/
spark/sbin
[I 16:09:51.094 NotebookApp] Jupyter Notebook 6.4.6 is running at:
[I 16:09:51.094 NotebookApp] http://localhost:8888/?token=b92c5feee599bbf1750a50
95c9676fc419c26c8abb8729e8
[I 16:09:51.094 NotebookApp] or http://127.0.0.1:8888/?token=b92c5feee599bbf175
0a5095c9676fc419c26c8abb8729e8
[I 16:09:51.094 NotebookApp] Use Control-C to stop this server and shut down all
kernels (twice to skip confirmation).
[C 16:09:51.202 NotebookApp]

To access the notebook, open this file in a browser:
file:///home/hadoopusr/.local/share/jupyter/runtime/nbserver-7921-open.h
tml
Or copy and paste one of these URLs:
http://localhost:8888/?token=b92c5feee599bbf1750a5095c9676fc419c26c8abb8
729e8
or http://127.0.0.1:8888/?token=b92c5feee599bbf1750a5095c9676fc419c26c8abb8
729e8
```

Running Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20220106143923-0000	(kill) PySparkShell	8	1024.0 MiB		2022/01/06 14:39:23	hadoopusr	RUNNING	30 s

V- Implémentation :

1- Copie les Datasets de local à HDFS :

Les datasets adoptant dans cette implémentation sont Titanic et Banque dataset.

Alors on va copier ces deux fichier csv dans notre HDFS, en utilisant la commande suivante : **hdfs dfs -put**

```
hadoopusr@master:/usr/local/spark$ hdfs dfs -put /usr/local/titanic.csv /titanic.csv
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authentication.util.KerberosUtil (file:/usr/local/hadoop/share/hadoop/common/lib/hadoop-auth-2.9.0.jar) to method sun.security.krb5.Config.getInstance()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.security.authentication.util.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
```

```
hadoopusr@master:~$ hdfs dfs -put /usr/local/bank.csv /bank.csv
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authentication.util.KerberosUtil (file:/usr/local/hadoop/share/hadoop/common/lib/hadoop-auth-2.9.0.jar) to method sun.security.krb5.Config.getInstance()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.security.authentication.util.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
```

2- Load data from HDFS to Spark jupyter:

Spark Session est le point d'entrée pour programmer Spark avec le jeu de données.

Ici nous permet de lire et sauvegarder les données à partir de HDFS

```
In [1]: import numpy as np # linear algebra
import pandas as pd

In [1]: from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('titanic').getOrCreate()

In [2]: df = spark.read.csv('hdfs:///train.csv', header = True, inferSchema=True)
df_test = spark.read.csv('hdfs:///test.csv', header = True, inferSchema=True)

In [3]: df.printSchema()

root
 |-- PassengerId: integer (nullable = true)
 |-- Survived: integer (nullable = true)
 |-- Pclass: integer (nullable = true)
 |-- Name: string (nullable = true)
 |-- Sex: string (nullable = true)
 |-- Age: double (nullable = true)
 |-- SibSp: integer (nullable = true)
 |-- Parch: integer (nullable = true)
 |-- Ticket: string (nullable = true)
 |-- Fare: double (nullable = true)
 |-- Cabin: string (nullable = true)
 |-- Embarked: string (nullable = true)
```

```
In [5]: df.show()
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen ...	male	22.0	1	0	A/5 21171	7.25	null	
2	1	1	Cumings, Mrs. Joh...	female	38.0	1	0	PC 17599	71.2833	C85	
3	1	3	Heikkinen, Miss. ...	female	26.0	0	0	STON/O2. 3101282	7.925	null	
4	1	1	Futrelle, Mrs. Ja...	female	35.0	1	0	113803	53.1	C123	
5	0	3	Allen, Mr. Willia...	male	35.0	0	0	373450	8.05	null	
6	0	3	Moran, Mr. James	male	null	0	0	330877	8.4583	null	
7	0	1	McCarthy, Mr. Tim...	male	54.0	0	0	17463	51.8625	E46	
8	0	3	Palsson, Master. ...	male	2.0	3	1	349909	21.075	null	
9	1	3	Johnson, Mrs. Osc...	female	27.0	0	2	347742	11.1333	null	
10	1	2	Nasser, Mrs. Nich...	female	14.0	1	0	237736	30.0708	null	
11	1	3	Sandstrom, Miss. ...	female	4.0	1	1	PP 9549	16.7	G6	
12	1	1	Bonnell, Miss. El...	female	58.0	0	0	113783	26.55	C103	

3-Appliquer Random Forest sur Titanic dataset :

Random Forest

```
In [29]: # Train a RandomForest model.
# Import packages
from pyspark.ml import Pipeline
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.feature import StringIndexer, VectorIndexer, OneHotEncoder, VectorAssembler, IndexToString
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.sql.functions import *
rf = RandomForestClassifier(labelCol="Survived", featuresCol="features")

In [30]: model = rf.fit(trainingData)

[Stage 12:> (0 + 1) / 1]

In [33]: rf_prediction = model.transform(testData)
rf_prediction.select("prediction", "Survived", "features").show()
evaluator = MulticlassClassificationEvaluator(labelCol="Survived", predictionCol="prediction", metricName="accuracy")

lr_accuracy = evaluator.evaluate(rf_prediction)
print("Accuracy of Random forest is = %g" % (lr_accuracy))
print("Test Error of Random forest = %g " % (1.0 - lr_accuracy))
```

prediction	Survived	features
1.0	0	[1.0, 24.0, 0.0, 0.0...
1.0	0	[1.0, 24.0, 0.0, 1.0...
1.0	0	[1.0, 31.0, 0.0, 0.0...
1.0	0	[1.0, 31.0, 1.0, 0.0...
0.0	0	[1.0, 33.0, 0.0, 0.0...
1.0	0	[1.0, 36.0, 1.0, 0.0...
1.0	0	[1.0, 37.0, 0.0, 1.0...
0.0	0	(5, [0, 1], [1.0, 40.0])
1.0	0	[1.0, 46.0, 0.0, 0.0...
1.0	0	[1.0, 47.0, 0.0, 0.0...
0.0	0	[1.0, 47.0, 0.0, 0.0...
1.0	0	[1.0, 52.0, 1.0, 1.0...
1.0	0	[1.0, 54.0, 0.0, 1.0...
1.0	0	[1.0, 64.0, 1.0, 4.0...
0.0	0	[1.0, 71.0, 0.0, 0.0...
1.0	1	[1.0, 16.0, 0.0, 1.0...
1.0	1	[1.0, 18.0, 0.0, 2.0...
1.0	1	[1.0, 19.0, 0.0, 2.0...
1.0	1	[1.0, 21.0, 2.0, 2.0...
1.0	1	[1.0, 24.0, 0.0, 0.0...

only showing top 20 rows

```
Accuracy of Random forest is = 0.666667
Test Error of Random forest = 0.333333
```


4- Applique Logistique Régression sur Titanic dataset :

```
In [9]: from pyspark.ml import Pipeline
log_reg = LogisticRegression(featuresCol = 'features', labelCol = 'Survived')

In [10]: pipeline = Pipeline(stages = [gender_indexer, embark_indexer,
gender_encoder, embark_encoder,
assembler, log_reg])

In [11]: train, test = final_data.randomSplit([0.7, 0.3])

In [12]: fit_model = pipeline.fit(train)

In [13]: results = fit_model.transform(test)

In [14]: results.select('prediction', 'Survived').show(3)

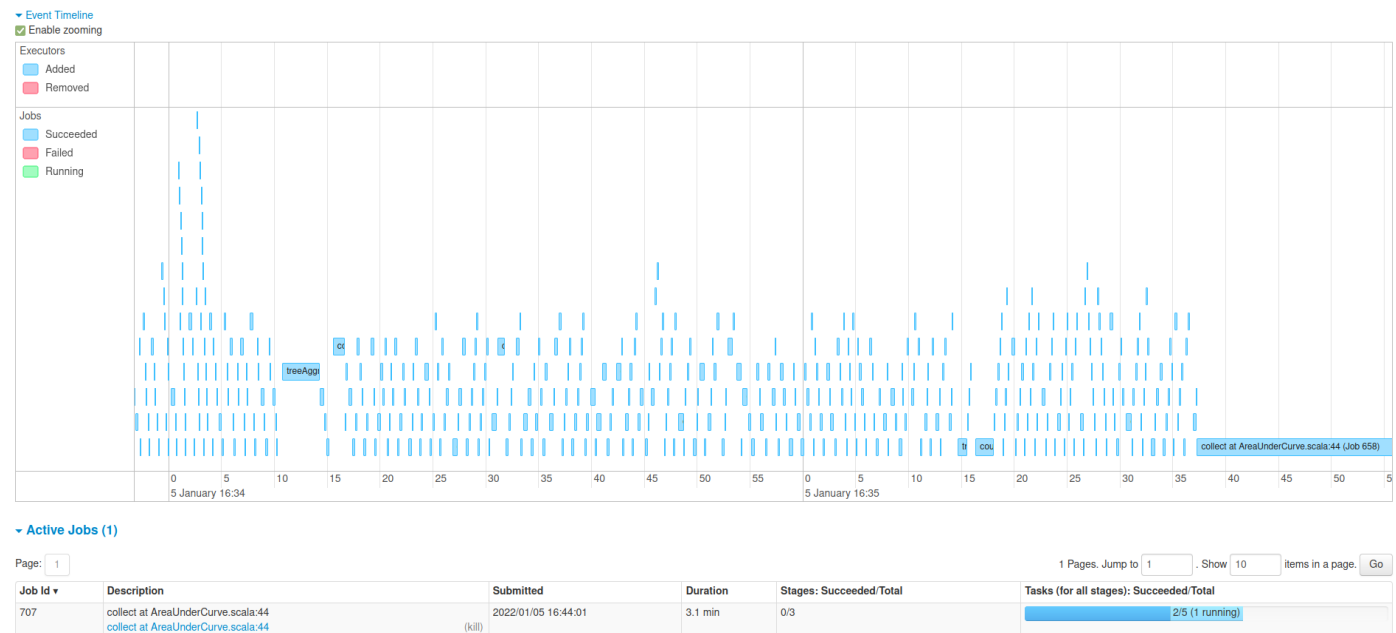
+-----+-----+
|prediction|Survived|
+-----+-----+
|         1.0|         0|
|         1.0|         0|
|         1.0|         0|
+-----+-----+
only showing top 3 rows

In [16]: from pyspark.ml.evaluation import BinaryClassificationEvaluator

evalu = BinaryClassificationEvaluator(rawPredictionCol = 'rawPrediction', labelCol = 'Survived')
AUC = evalu.evaluate(results)
AUC

Out[16]: 0.8512512512512513
```

Dans <http://master:4040> on peut observer les jobs exécutent sur pyspark application :



VI- Conclusion :

La Réalisation de ce travail nous a poussés de pratiquer et d'appliquer sur terrain toutes nos acquisitions et nos connaissances théoriques et d'une autre part, on a bien touché la notion de synchronisation entre les nœuds d'un cluster afin d'être capable d'y stocker des données et d'y exécuter des programmes.

Au cours de notre travail, on rencontre pas mal des problèmes et des difficultés dans la plupart de temps, ils sont liés à la configuration soit de Hadoop ou Spark, le grand (principal) problème qu'on rencontrée est liée l'installation de pig et spark.