



ARQUITECTURA DE SISTEMES BIG DATA PEL RECORD LINKAGE DE XARXES SOCIALS

TREBALL FINAL DE GRAU – Informe de progrés II



2018-2019

ALEJANDRO GARCIA CARBALLO

1423957

Índex:

1. Introducció.....	3
2. Metodologia seguida durant el projecte.....	3
3. Planificació.....	4
4. Connexió client amb clúster.....	5
5. Interfície de dades	8
6. Testing	9
6.1. Test funcional.....	10
6.1. Test no funcional.....	11
7. Resultats obtinguts	12
8. Conclusions	15
9. Bibliografia	16

1 Introducció

Aquesta segona fita de progrés és l'última part del projecte dedicada al desenvolupament i, per tant, te com a objectiu finalitzar gran part de les tasques i extreure unes conclusions del projecte.

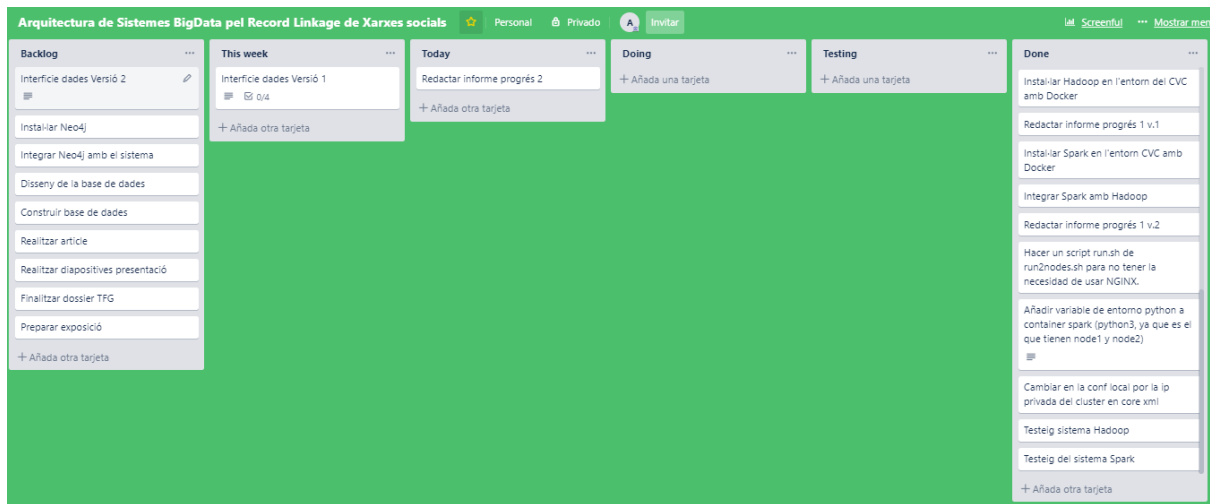
En aquesta fita he desenvolupat les tasques referents a les proves d'arquitectura, on he realitzat proves per garantir la correctesa de les dades i la seva correcta distribució, comprovant a la vegada que hi ha una comunicació exitosa entre els diferents elements de l'arquitectura. Aquestes proves son necessàries ja que les dades distribuïdes en el HDFS s'utilitzaran a l'hora de carregar les dades a través d'una interfície i retornar-les a l'usuari, i per tant, hem de garantir que les dades importades són correctes i no tenen cap diferència amb les dades originals. També he realitzat proves de rendiment per comprovar el temps d'execució amb diferents dades i poder analitzar com està funcionant el sistema i tractar de buscar possibles millores.

Com es va comentar a l'inici del projecte, aquest projecte està emmarcat dins del projecte de recerca XARXES i està pensat per a que pugui ser utilitzat per un estudiant de màster encarregat del tractament de les dades. A causa d'això, és necessari crear una interfície que faciliti la utilització d'aquesta arquitectura. Amb aquesta interfície, s'aconsegueix accedir a les dades distribuïdes i retornar-les en una estructura de dades de Python.

Per poder realitzar totes aquestes tasques també ha sigut necessari aconseguir una connexió entre el client i el clúster per aconseguir executar els scripts necessaris desde un client extern.

2 Metodologia seguida durant el projecte

Per realitzar els diferents objectius establerts durant el projecte, s'ha seguit la metodologia proposada a l'inici d'aquest, una metodologia Kanban que permet aconseguir una visualització del flux de treball i separar les tasques en fases per poder limitar el treball. En aquest projecte no calia una gran quantitat de gestió ja que és un projecte individual i, per tant, no tenia sentit realitzar reunions amb un equip de treball o plantejar algunes de les pràctiques comuns en altres metodologies.



II·lustració 1 - Tauler Kanban utilitzat durant el projecte

Gràcies a aquest tipus de metodologia també era fàcil adaptar-se a canvis o imprevistos, com ara el que ha sorgit en aquesta fita, on no es va contemplar com a necessari tenir en compte les tasques referents a la connexió de client i clúster. Només ha fet falta afegir al backlog les tasques que eren necessàries i limitar-les en el temps.

3 Planificació

En la planificació d'aquesta fita estava contemplat que es pogués dur a terme una fase de testeig, la realització de la interfície comentada en l'apartat d'introducció, que constava de dues versions, una primera versió on només era necessari aconseguir que la interfície retornés les dades del HDFS en una estructura de Python o Spark i una segona versió en la que si donava temps, es tractaria d'harmonitzar les dades. Com a última tasca, tot i que era difícil arribar a aquest punt, es volia realitzar una implementació d'una base de dades estructurada en graf per tal d'emmagatzemar les dades en aquest tipus d'estructura.

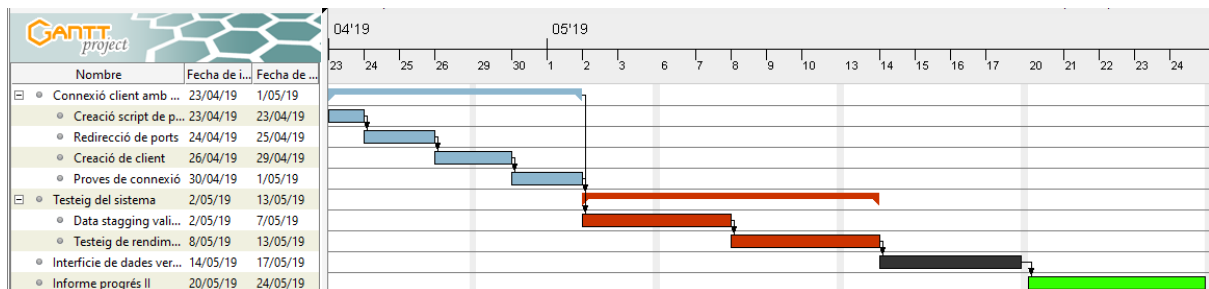


II·lustració 2 - Planificació inicial fita de progrés II

En la planificació no hi era contemplada una fase dedicada a la connexió del client amb el clúster que ha ocupat gran part del temps. A causa de no haver tingut en compte aquesta fase i el fet d'haver tingut dificultats a l'hora d'aconseguir resoldre les tasques pertanyents a aquesta, degut a una sèrie d'errors de connectivitat, el temps per realitzar els objectius de la segona fita s'han vist bastant reduïts.

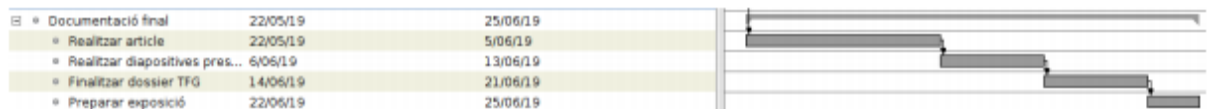
Finalment, en aquesta fita he modificat part dels objectius planificats per tal d'incorporar aquestes noves tasques, i gràcies a la metodologia comentada anteriorment, ha sigut fàcil realitzar la incorporació d'aquestes.

Les tasques que he realitzat finalment en aquesta fita han sigut realitzar la connexió entre el client i el clúster, així com la realització de proves en el sistema i finalment la primera versió de la interfície de dades. La idea es que es pogués haver arribat a aconseguir una versió de la interfície que retornés les dades harmonitzades, però no ha sigut possible a causa dels endarreriments comentats anteriorment.



II-lustració 3 - Distribució real de les tasques en aquesta fita de progrés II

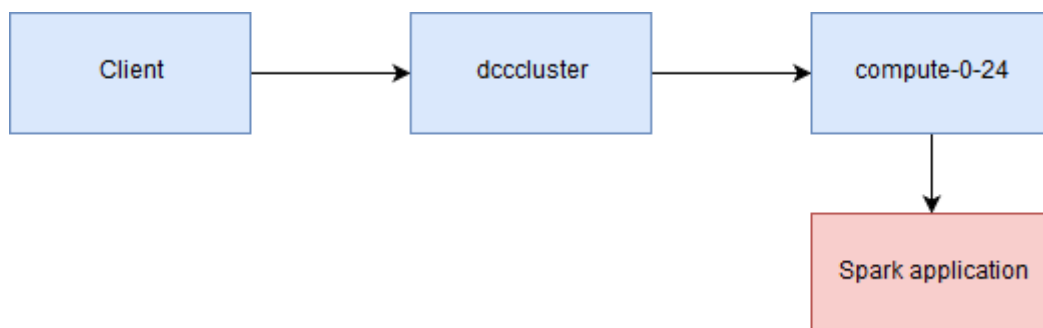
Com que aquesta era l'última fase de progrés, les següents tasques a realitzar seran aquelles que tenen relació amb la finalització de la documentació final.



II-lustració 4 - Planificació per a la següent fita

4 Connexió client amb clúster

Per a poder realitzar les proves i la interfície de dades el primer pas és aconseguir que el client sigui capaç d'executar el seu codi en mode clúster, és a dir, ha de poder connectar-se al clúster i mitjançant la configuració especificada en els fitxers de configuració llançar una aplicació Spark [6] amb YARN.



II-lustració 5 - Objectiu a aconseguir

El primer pas realitzat per poder aconseguir la connexió del client amb el clúster ha sigut fer un conjunt de redireccions en els ports [1][3] de manera que, al estar treballant amb Docker, per a que els ports puguin ser escoltats desde fora de la VLAN, primerament he hagut d'exposar tots els ports necessaris. Una vegada exposats, he realitzat un conjunt de redireccions de manera que quan s'accedeix a un port concret del node compute-0-24 de dcccluster, aquest a la seva vegada accedeix al port que ens interessa dels diferents serveis

que pot oferir Hadoop o Spark, els quals estan connectats a través d'una xarxa Docker. El rang de ports que poden ser utilitzats treballant en el node compute-0-24 de dcccluster van desde el port 52470 fins al 52480. La redirecció de ports s'ha realitzat a través del mateix script encarregat d'executar les comandes docker per aixecar els containers especificant una redirecció a través del paràmetre -p de docker run.

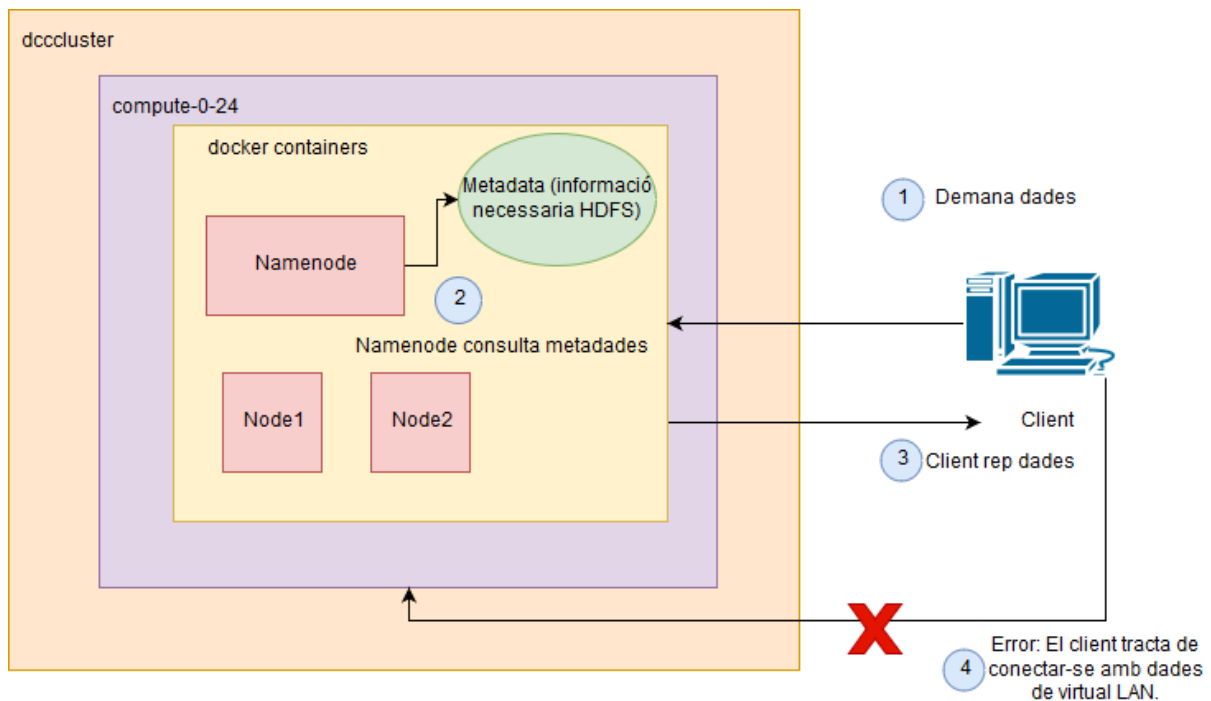
Nom configuració	Port original	Redirecció del port
fs.default.name	9000	52475
yarn.resourcemanager.address	8032	52476
yarn.resourcemanager.webapp.address	8088	52472
yarn.nodemanager.webapp.address	8042	52479
dfs.datanode.address (node1)	9866	52479
dfs.datanode.address (node2)	9866	52478
spark.ui.port	4044	52473

Taula 1 - Redireccions de ports en els serveis Hadoop i Spark

El següent pas era comprovar el funcionament de l'arquitectura a l'hora d'executar un script en python desde fora de la virtual LAN. En un inici, les proves per comprovar el funcionament es van realitzar amb un script simple, que s'encarregava de comptar el nombre de caràcters que contenia un String. Per evitar que els errors que poguessin sorgir tinguessin relació amb el codi que havia realitzat, finalment les proves es van realitzar executant un script de mostra que està inclòs per defecte en spark i d'aquesta manera assegurar-me que el codi executat era funcional. Primerament les execucions es van realitzar amb spark submit ja que l'objectiu principal era testear si l'execució en mode clúster funcionava. Per realitzar l'execució és necessari indicar a Spark on són els fitxers de configuració per tal de poder realitzar l'execució amb la configuració establerta. Això es pot aconseguir creant dues variables d'entorn anomenades HADOOP_CONF_DIR i YARN_CONF_DIR que apunten al directori que conté els fitxers de configuració (hdfs-site.xml, core-site.xml ...).

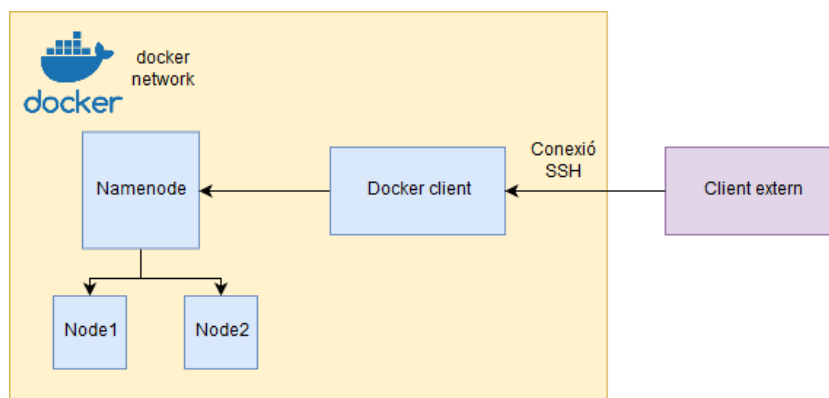
A l'hora d'executar el script va aparèixer un error indicant que no s'havien pogut realitzar les rèpliques a l'hora de realitzar el staging de spark (procés en el qual l'aplicació es puja al HDFS). El motiu d'aquest error és molt ampli ja que no hi ha una causa única per la qual pugui aparèixer, per tant, trobar la seva solució pot arribar a ser un procés complicat.

L'explicació que tenia mes sentit degut a la descripció de l'error [2] i com estava estructurada l'arquitectura, consistia en que el client estava intentant connectar-se al clúster a través de les dades privades de la VLAN en la que està muntada l'arquitectura. A través de les redireccions establertes anteriorment per poder fer possible la comunicació, el client demanava al namenode la informació pertanyent als datanodes. El namenode no es comunicava directament amb els datanodes, sinó que retornava les dades que emmagatzemava en les metadades al client per a que aquest s'encarregués de realitzar la comunicació. Per tant, quan el client rebia les dades, estava rebent les dades pertanyents a la virtual LAN i al tractar de comunicar-se amb aquestes dades sorgia un error.



Il·lustració 6 - Descripció del problema a l'hora d'executar codi en mode clúster

La solució a aquest problema, tenint en compte que el projecte havia de continuar i una gran quantitat del temps d'aquesta fita s'ha dedicat a entendre per què la connexió estava fallant, ha sigut aixecar un container que actuï com a client dins de la virtual LAN de Docker i d'aquesta manera poder realitzar les execucions.



Il·lustració 7 - Solució proposada per a solucionar el problema

Aquest container té la configuració necessària de SSH i les redireccions necessàries per a poder ser accedit desde qualsevol client i executar els scripts directament desde el client de Docker. Per aconseguir-ho, s'han realitzat algunes modificacions en el Dockerfile de Spark per a generar el contenidor que actuarà com a client. En el Dockerfile s'han afegit algunes línies de codi encarregades d'instal·lar i configurar el servidor SSH [7].

```

#creating SSH server
RUN apt-get update && apt-get install -y openssh-server
RUN mkdir /var/run/ssh
RUN echo 'root:testtest' | chpasswd
RUN sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd_config

# SSH login fix. Otherwise user is kicked off after login
RUN sed 's@session\s*required\s*pam_loginuid.so@session optional pam_loginuid.so@g' -i /etc/pam.d/ssh

ENV NOTVISIBLE "in users profile"
RUN echo "export VISIBLE=now" >> /etc/profile

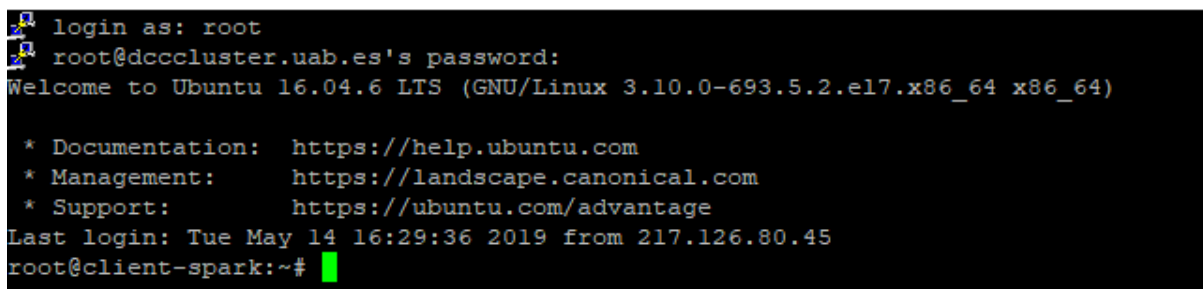
#copy spark config
ADD configs/spark-defaults.conf $SPARK_HOME/conf/

# expose various ports
EXPOSE 4044 22

```

II·lustració 8 - Línies afegides al Dockerfile del client per a que pugui ser accessible amb SSH

Aquesta imatge és molt similar a la de Spark, ja que conté Spark per a poder executar les aplicacions amb spark submit, però amb la diferencia de tenir un servidor SSH instal·lat. Executant aquesta imatge es crea un contenidor amb Spark al qual podem accedir amb l'usuari de SSH 'root' i la password 'testtest'.



```

login as: root
root@dcccluster.uab.es's password:
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 3.10.0-693.5.2.el7.x86_64 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
Last login: Tue May 14 16:29:36 2019 from 217.126.80.45
root@client-spark:~#

```

II·lustració 9 - Connexió SSH amb el contenidor del client

5 Interfície de dades

Les dades distribuïdes en el HDFS han de poder ser accedides fàcilment, sense que l'usuari que utilitzi les dades tingui la necessitat d'entendre on són aquestes dades ni com accedir-hi. Aquestes dades estan pensades per a ser utilitzades per un estudiant de màster, el qual està creant uns mòduls en Python que s'encarreguen del tractament de les dades per a un posterior anàlisis. La interfície de dades que he desenvolupat té com a objectiu accedir a les dades i retornar-les, de manera que el col·laborador encarregat de la part del tractament de les dades s'encarregui d'utilitzar-les. Per a facilitar la compatibilitat entre la seva part i la interfície creada, he decidit utilitzar com a llenguatge de programació Python i he retornat les dades en una estructura de dades compatible amb aquest.

Actualment disposem de les dades de tres localitzacions diferents: Sant Feliu, Castellví Rosanes i Santa Coloma. El codi està pensat per a que es pugui seleccionar qualsevol de les tres localitzacions i retornar totes les dades emmagatzemades i distribuïdes en el HDFS pertanyents a aquestes localitzacions. Revisant el codi proporcionat pel col·laborador encarregat del tractament de les dades, vaig observar que hi havia un conjunt de camps dins de les dades que li eren de més interès. Amb aquestes dades crea una estructura per a

cadascuna de les localitzacions, les quals contenen les dades principals amb les que treballa.

```
CensusLocation.SANT_FELIU: {  
  CensusFields.FIRST_NAME : 'Noms_harmo',  
  CensusFields.SURNAME_1 : 'cognom_1',  
  CensusFields.SURNAME_2 : 'cognom_2',  
  CensusFields.ID_INDIVIDUAL : 'id_padro_individu',  
  CensusFields.ID_HOUSEHOLD : 'id_padro_llar',  
  CensusFields.CENSUS_YEAR : 'any_padro',  
  CensusFields.DNI : 'DNI',  
  CensusFields.YOB : 'cohort',  
  CensusFields.CIVIL_STATUS : 'estat_civil',  
  CensusFields.RELATION : 'parentesc_har',  
  CensusFields.OCCUPATION : 'ocupacio_hisco',  
  CensusFields.GENDER : None,  
  CensusFields.DOB : 'data_naix',  
  CensusFields.AGE : 'edat'  
},
```

Il·lustració 10 - Estructura de dades utilitzada per el col·laborador de màster

Per tant, a part de retornar totes les dades de cadascuna de les localitzacions, també vaig crear unes funcions encarregades de retornar exclusivament les dades seleccionades pel col·laborador per a cadascuna de les localitzacions i d'aquesta manera facilitar l'accés a les dades.

6 Testing

Per entendre el conjunt de proves [8][12] que s'han realitzat hem d'entendre quins són els diferents passos que es realitzen per processar el Big Data en una arquitectura basada en Hadoop i Spark.

Podem separar les fases per les quals passa la informació en tres passos principals:

1. **Carregar fitxers de dades en el HDFS:** Dades de moltes fonts diverses són carregades en el HDFS i són dividides en múltiples fitxers. Aquestes dades es distribueixen en diferents nodes a partir dels quals podem accedir a les dades.
2. **Realitzar el processament amb Spark:** Una vegada les dades són carregades en el HDFS, es realitzen operacions per al processament d'aquestes i poder obtenir els resultats desitjats. Podem utilitzar diferents frameworks de processament, en aquest cas, utilitzem Spark.
3. **Extracció de dades i/o resultats:** En aquest últim pas es tracta de extreure les dades de sortida resultants del processament i carregar-les en alguna estructura com ara un data warehouse per tal de generar reports analítics. En aquest projecte aquest últim pas no era un dels objectius, per tant, les dades resultants es guarden en logs propis del sistema Hadoop.



Il·lustració 11 - Àrees on enfocar-se per realitzar testing en Big Data

En aquest tipus de sistemes el fet de treballar amb una quantitat de dades molt gran i estar executant el codi en diferents nodes fa que sigui molt probable trobar problemes en cadascuna d'aquestes tres fases principals. És a causa d'això que l'enfocament del testing en Big Data és centra en cadascuna d'aquestes tres fases.

6.1 Test funcional

Algunes de les proves que s'han realitzat es basen en tests funcionals, un tipus de proves on el sistema es testeja per tal de comprovar que el sistema compleix els requisits funcionals. Les proves de test funcionals es basen en observar quins són els outputs o resultats donada una informació d'entrada i comprovar que el sistema compleix correctament els requisits.

Dins de les proves funcionals he realitzat les següents validacions:

- **Validació de data staging:** Alguns dels problemes que podem trobar durant la primera fase del procés on movem les dades de l'origen al sistema de Hadoop són: que hi hagi una incorrecta captura de la informació, un incorrecte emmagatzematge de la informació o bé un emmagatzematge on les dades són incompletes, o també pot succeir que hi hagi una replicació incorrecta. Les validacions que podem realitzar en aquesta fase poden ser comprovar que les dades són carregades correctament en el HDFS, respectant la destinació on em indicat que es guardin les dades, comparar que les dades originals són iguals que les que hem pujat al HDFS, i per últim validar que les dades són replicades correctament respectant el factor de replicació indicat, de manera que les dades també estiguin disponibles en diferents nodes i que es respecten els tamanyes de blocs.
- **Validació del processament Spark:** Els possibles problemes que podem trobar a l'hora del processament de Spark poden ser problemes a l'hora d'executar el codi a causa d'alguna incompatibilitat o de codi no funcional, problemes a l'hora d'executar codi en mode clúster quan en client funciona correctament o un problema a l'hora de retornar el output. Per tant, s'ha validat que el procés es completa satisfactòriament, tant en mode client com en mode clúster, de manera que el output que obtenim sigui el desitjat.
- **Validació extracció resultats:** Com he comentat anteriorment, en aquest projecte no està contemplada l'extracció dels resultats en cap tipus d'estructura, per tant, només s'ha comprovat que els logs amb els resultats són accessibles.

6.2 Test no funcional

Les proves no funcionals tracten de validar els requisits que no són funcionals, aquells que tenen més a veure amb característiques com ara el rendiment del sistema o bé els recursos que aquest utilitza. En aquest cas, he realitzat una prova de rendiment comparant el temps d'execució d'un script en mode client i en mode clúster. Per realitzar les proves he utilitzat un conjunt de datasets [12] de diferents mides sobre registres de vendes. Els fitxers contenen des de 100 registres fins a un màxim de 1.500.000 registres i tenen un tamany de de 13 KB fins a 182 MB. Les proves s'han realitzat amb els diferents tamanyos de fitxers per poder comprovar com afecta el tamany dels fitxers i el nombre de registres i poder tractar de trobar a partir de quin límit es comença a notar la diferència en el temps d'execució.

Les proves es realitzaran executant el script mostrat en la il·lustració 12, que s'encarrega de llegir els fitxers .csv i realitzar un sumatori de tots els elements d'una de les columnes amb un RDD (Resilient Distributed Dataset). Per realitzar la prova s'han utilitzat les funcions map i reduce per tal de tractar de paral·lelitzar el codi.

```
from pyspark.sql import SparkSession
import time

if __name__ == "__main__":
    start = time.time()
    spark = SparkSession.builder.appName("Testing Data Ingestion").getOrCreate()
    df = spark.read.format("csv").option("header", "true").option("inferSchema", "true") \
        .load("hdfs:///user/root/testFiles/100SalesRecords.csv")
    print(type(df))
    df.printSchema()
    print(df.rdd.map(lambda x: (1,x[1])).reduceByKey(lambda x,y: x + y).collect()[0][1])
    print("Tiempo total:")
    end = time.time()
    print(end-start)
```

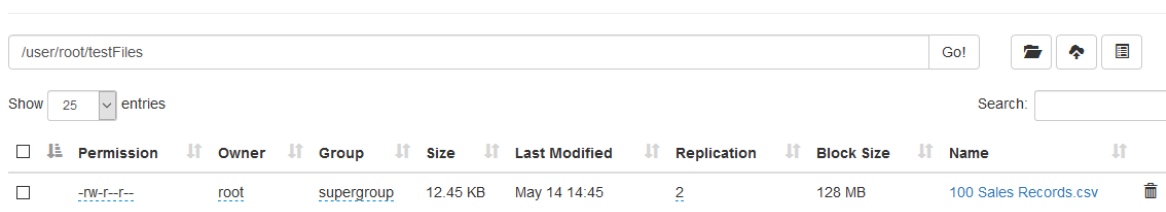
Il·lustració 12 - Script per realitzar proves de rendiment

7 Resultats obtinguts

Els resultats mostrats en aquest apartat es basen en les diferents validacions que s'han proposat en l'apartat de testing, de manera que he comentat quins resultats s'han obtingut per a cadascuna de les proves i he interpretat els resultats.

En quant als requisits funcionals, seguint les validacions del data staging, el primer que s'ha comprovat és que les dades carregades en el HDFS es carreguessin de manera correcta i en el directori indicat. Al pujar les dades a través de la comanda “put” de Hadoop indicant el directori on volem que es carreguin, si anem a la UI del HDFS, podem comprovar que les dades s'han carregat correctament.

Browse Directory



The screenshot shows the HDFS Browse Directory interface. At the top, there is a text input field containing the path "/user/root/testFiles" and a "Go!" button. To the right of the input field are three icons: a folder, a document, and a list. Below the input field, there is a "Show" dropdown menu set to "25" and the text "entries". To the right of this is a "Search:" input field. Below these elements is a table with columns: Permission, Owner, Group, Size, Last Modified, Replication, Block Size, and Name. The table contains one entry for the file "100 Sales Records.csv".

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	root	supergroup	12.45 KB	May 14 14:45	2	128 MB	100 Sales Records.csv

Il·lustració 13 - Dades carregades en el directori indicat del HDFS

Per validar que aquestes dades estiguessin disponibles en diversos nodes, podem accedir a la informació addicional del fitxer carregat, on trobem un camp anomenat “availability” que ens mostra els nodes on estan disponibles aquestes dades. En aquest cas les dades son disponibles en els dos nodes creats, per tant, la distribució del fitxer és correcta. Les dades es distribueixen en tots els nodes ja que la quantitat de dades a emmagatzemar no és molt gran, i per tant, només necessitem dos nodes. En el cas de que tinguéssim més informació a emmagatzemar i, per tant, més nodes, no seria una bona idea que els fitxers es distribueixin en cadascun dels nodes ja que això suposaria un cost addicional i un malbaratament d'espai innecessari.



The screenshot shows the "File information - 100 Sales Records.csv" page. At the top, there are three links: "Download", "Head the file (first 32K)", and "Tail the file (last 32K)". Below these links is a green header bar with the text "Block information --" and a dropdown menu showing "Block 0". The main content area displays the following information: Block ID: 1073741825, Block Pool ID: BP-1731354512-192.168.128.4-1557839554919, Generation Stamp: 1001, Size: 12744, and Availability. The Availability section is highlighted with a red box and contains a list of nodes: node2 and node1.

Block information -- Block 0

Block ID: 1073741825
Block Pool ID: BP-1731354512-192.168.128.4-1557839554919
Generation Stamp: 1001
Size: 12744
Availability:
• node2
• node1

Il·lustració 14 - Disponibilitat de les dades en els diferents nodes

La següent validació és comprovar que es respecten els tamany de bloc. En la mateixa UI del HDFS podem trobar informació com ara el tamany de bloc i el nombre de blocs creats per a cadascun dels fitxers. Obtenim el tamany de bloc per defecte en Hadoop de 128 MB, que és el que vam tenir en compte des d'un inici del projecte.

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	-rw-r--r--	root	supergroup	119.01 MB	May 15 23:09	2	128 MB	1000000SalesRecords.csv	
<input type="checkbox"/>	-rw-r--r--	root	supergroup	11.91 MB	May 15 23:09	2	128 MB	100000SalesRecords.csv	
<input type="checkbox"/>	-rw-r--r--	root	supergroup	1.19 MB	May 15 23:09	2	128 MB	10000SalesRecords.csv	
<input type="checkbox"/>	-rw-r--r--	root	supergroup	122.08 KB	May 15 23:09	2	128 MB	1000SalesRecords.csv	
<input type="checkbox"/>	-rw-r--r--	root	supergroup	12.45 KB	May 14 18:49	2	128 MB	100SalesRecords.csv	
<input type="checkbox"/>	-rw-r--r--	root	supergroup	178.51 MB	May 15 23:09	2	128 MB	1500000SalesRecords.csv	
<input type="checkbox"/>	-rw-r--r--	root	supergroup	59.51 MB	May 15 23:09	2	128 MB	500000SalesRecords.csv	
<input type="checkbox"/>	-rw-r--r--	root	supergroup	5.95 MB	May 15 23:09	2	128 MB	50000SalesRecords.csv	
<input type="checkbox"/>	-rw-r--r--	root	supergroup	608.53 KB	May 15 23:09	2	128 MB	5000SalesRecords.csv	

Il·lustració 15 - Tamany de bloc en tots els fitxers

Quan un dels blocs sobrepassa el tamany de 128 MB, si accedim a la informació addicional del fitxer, trobem que el nombre de blocs creats són els necessaris, respectant així el tamany de bloc indicat.

File information - 1500000SalesRecords.csv
Download
Head the file (first 32K)
Tail the file (last 32K)

Block information

Block 0
Block 0
Block 1

Block ID: 10731418
Block Pool ID: BP-660753791-192.168.128.4-1557849963263
Generation Stamp: 1035
Size: 134217728
Availability:

- node2
- node1

Il·lustració 16 – Diferents blocs creats al pujar el fitxer en el HDFS

Tenint en compte que el tamany dels fitxers de les dades de les diferents localitzacions no són massa grans i que no sobrepassen el tamany de bloc, podem adonar-nos que realment el tamany de bloc utilitzat per a les dades que tenim es massa gran. Les dades que utilitzem sobre les poblacions més que ser grans fitxers, està pensat que siguin una gran diversitat de fitxers, però que aquests no siguin de un gran tamany, per tant, possiblement hauria

sigut una bona idea reduir el tamany de bloc de Hadoop per tal d'adaptar-ho a les nostres dades. D'aquesta manera possiblement hauríem obtingut a la vegada un millor rendiment en el sistema i s'estalviaria espai.

En quant a la validació de comparació dels fitxers de dades d'entrada i els emmagatzemats en el HDFS, fent ús de l'eina fc de Windows que s'encarrega de comparar dos fitxers, he guardat el resultat en un .txt anomenat output per comprovar si existeix alguna diferència entre els dos fitxers. Com a resultat obtenim que no hi ha cap diferència entre els dos fitxers, per tant, podem concloure que els dos fitxers contenen la mateixa informació.

Comparando archivos 100SalesRecords.csv y 100SALESRECORDSHDFS.CSV
FC: no se han encontrado diferencias

Il·lustració 17 - Output de la comparació entre el fitxer original i el carregat al HDFS

La següent validació realitzada ha sigut la de garantir que Spark executa una aplicació de manera satisfactòria, tant en mode clúster com en mode client. Al executar una aplicació, es rep un final status SUCCEEDED indicant que l'aplicació s'ha executat correctament (sempre i quan no hi hagin errors en el codi).

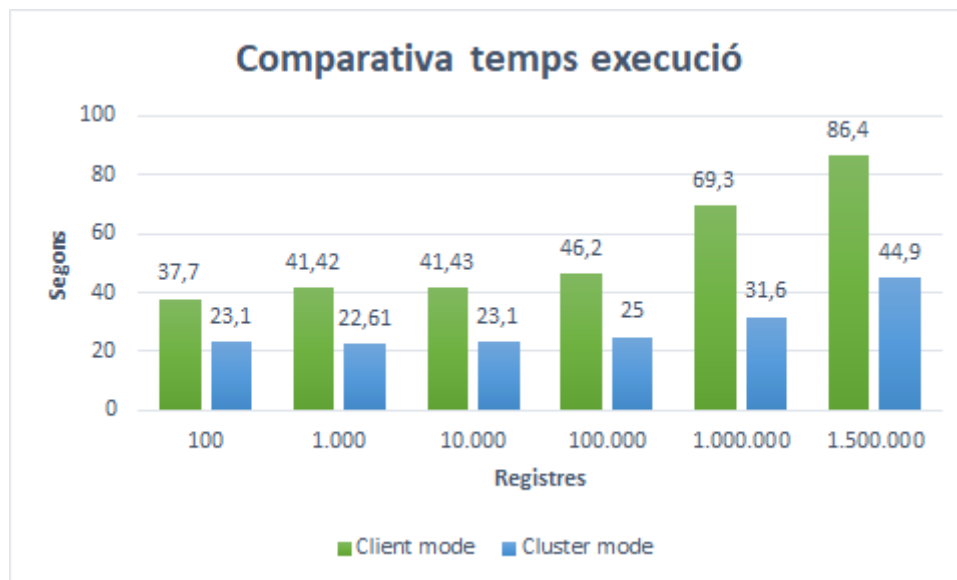
```
2019-05-12 18:02:31 INFO Client:54 -  
  client token: N/A  
  diagnostics: N/A  
  ApplicationMaster host: node2  
  ApplicationMaster RPC port: 46135  
  queue: default  
  start time: 1557684118412  
  final status: SUCCEEDED  
  tracking URL: http://namenode:8088/proxy/application_1557595668802_0001  
/  
  user: root
```

Il·lustració 18 - Execució satisfactòria d'una aplicació en mode clúster

Al executar el script en mode client obtenim l'output dels resultats directament en la consola de comandes on s'ha executat, ja que el programa s'executa en aquesta mateixa màquina, en canvi, al executar en mode clúster, el output s'obté en els logs dels nodes on s'ha executat, els quals també són accessibles a través de la UI del NameNode. Si fos necessari, es podria especificar més concretament on volem que s'emmagatzemin els outputs.

Pel que fa a la interfície de dades, a través de l'execució d'aquesta s'ha obtingut una estructura de dades de tipus Dataframe, amb els camps necessaris, o bé, el conjunt de totes les dades, que podrien ser utilitzades directament pel usuari.

Finalment, en la realització de les proves comentades en l'apartat de test no funcional, s'ha realitzat l'execució dels diferents fitxers tant en mode client com en mode clúster per tal de poder comparar els diferents resultats.



II-lustració 19 - Gràfica comparativa de rendiment entre una execució en mode client i en mode clúster

Quan executem el script que treballa amb 100 registres en mode client obtenim un temps d'execució de 37'7 segons, en comparació amb l'execució en clúster que obtenim 23'1 segons, pot semblar que la diferencia entre les dues execucions no es massa notòria. En canvi, a mida que el fitxer de dades va creixent i tenim més registres, la diferencia es va fent mes notable, obtenint pràcticament la meitat del temps d'execució en mode clúster en el script de que conté 1.500.000 registres. Això sembla indicar que el sistema esta dissenyat per a funcionar amb grans quantitats de dades i que per tant, la seva utilitat radica en utilitzar-ho per a aquests cassos.

8 Conclusions

La realització d'aquest projecte principalment tenia com a objectiu millorar la qualitat del projecte global de recerca XARXES, augmentant la seva eficiència i facilitant l'escalabilitat. Per aconseguir aquests objectius es plantejava com a solució la implementació d'un sistema big data.

Aconseguir part dels objectius ha comportat un procés de recerca sobre les tecnologies utilitzades per a la implementació del sistema i la utilització d'altres eines com ara Docker les quals inicialment tenia un coneixement molt bàsic, i gràcies a la realització d'aquest projecte he pogut aprofundir-hi més, posar-les en pràctica i aprendre a utilitzar-les.

Una gran part dels objectius principal s'han pogut dur a terme, aconseguint una arquitectura big data funcional en el clúster de dcccluster. Tot i que el projecte es funcional com a tal, i els diferents components estan executant-se en el node-0-24 de dcccluster, hi ha un conjunt de millores que es podrien realitzar abans de posar aquest sistema en producció, com ara modificar el tamany dels blocs, que podrien passar de 128 MB a 64 MB, reduint així l'espai ocupat innecessàriament, o bé, aconseguir que la connexió des d'un client pogués realitzar-se sense la necessitat de connectar-se a una Shell dins de la VLAN, sinó aconseguir que els clients externs a la VLAN puguin connectar-se directament, potser tractant d'utilitzar els hostnames en comptes de les direccions IP en la comunicació amb els datanodes [5].

També amb la disposició de més temps, es podria finalitzar la part de la interfície referent a l'harmonització de les dades, per tal de retornar les dades ja harmonitzades i estalviar aquest procés a l'usuari.

La utilització d'una metodologia i la realització d'una planificació en un projecte d'una llargada considerable han sigut molt útils per poder anar contemplant quin seria l'abast del projecte, poder fent una re-planificació i establir uns objectius en un límit de temps. Gràcies a això i als conceptes que he pogut anar aprenent al llarg d'aquests anys de carrera, conjuntament amb tot allò que he après a l'hora de realitzar aquest projecte, ha estat possible realitzar aquest projecte.

9 Bibliografia

- [1] S. Lippens, «Hadoop 3 default ports», 16 Novembre 2018. [En línia]. Disponible a: <https://www.stefaanlippens.net/hadoop-3-default-ports.html>. [Últim accés: 10 Maig 2019].
- [2] Hadoop Team, «One Of Several Explanations To “could only be replicated to 0 nodes” Error», 17 Febrer 2014. [En línia]. Disponible a: <http://www.hadoopinrealworld.com/could-only-be-replicated-to-0-nodes/>. [Últim accés: 5 Maig 2019].
- [3] R. Tang, «Default Ports Used by Hadoop Services (HDFS, MapReduce, YARN)», 2017. [En línia]. Disponible a: <https://kontext.tech/docs/DataAndBusinessIntelligence/p/default-ports-used-by-hadoop-services-hdfs-mapreduce-yarn>. [Últim accés: 6 Maig 2019].
- [4] F. C. Ospina, «submit local spark job to emr», 8 Febrer 2019. [En línia]. Disponible a: <https://stackoverflow.com/questions/54598950/submit-local-spark-job-to-emr>. [Últim accés: 10 Maig 2019].
- [5] Apache Hadoop, «HDFS Support for Multihomed Networks», 17 Març 2017. [En línia]. Disponible a: <https://hadoop.apache.org/docs/r2.8.0/hadoop-project-dist/hadoop-hdfs/HdfsMultihoming.html>. [Últim accés: 11 Maig 2019].
- [6] Apache Spark, «Apache Spark™ is a unified analytics engine for large-scale data processing.», [En línia]. Disponible a: <https://spark.apache.org/>. [Últim accés: 2 Maig 2019].
- [7] Docker Team, «Dockerize an SSH service», [En línia]. Disponible a: https://docs.docker.com/engine/examples/running_ssh_service/. [Últim accés: 11 Maig 2019].
- [8] Guru99, «Big Data Testing Tutorial: What is, Strategy, How to test Hadoop», [En línia]. Disponible a: <https://www.guru99.com/big-data-testing-functional-performance.html>. [Últim accés: 13 Maig 2019].
- [9] Intellipaat, «Big Data Testing Tutorial | Big Data Hadoop Testing YouTube Video | Intellipaat», 15 Setembre 2017. [En línia]. Disponible a: https://www.youtube.com/watch?v=2FutPlcU_zA. [Últim accés: 13 Maig 2019].
- [10] Tariq, «Upload data to HDFS running in Amazon EC2 from local non-Hadoop Machine», 18 Abril 2013. [En línia]. Disponible a:

<https://stackoverflow.com/questions/16073831/upload-data-to-hdfs-running-in-amazon-ec2-from-local-non-hadoop-machine>. [Últim accés: 5 Maig 2019].

[11] L. Gu and H. Li, "Memory or Time: Performance Evaluation for Iterative Operation on Hadoop and Spark," *2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing*, Zhangjiajie, 2013, pp. 721-727.

doi: 10.1109/HPCC.and.EUC.2013.106

keywords: {cache storage;distributed processing;iterative methods;performance evaluation;iterative operation;Hadoop framework;Spark framework;data-intensive applications;data reloading;cache mechanism;distributed machines;memory consumption;Sparks;Generators;Computers;Distributed databases;Iterative methods;Central Processing Unit;Twitter;Hadoop;Spark;System Performance;Iterative Operation},

URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6831988&isnumber=6825516>

[12] Eforexcel, «Downloads 18 - Sample CSV Files / Data Sets for Testing (till 1.5 Million Records) - Sales», 26 Agost 2017. [En línia]. Disponible a:

<http://eforexcel.com/wp/downloads-18-sample-csv-files-data-sets-for-testing-sales/>. [Últim accés: 14 Maig 2019]

[13] Gudipati, M., Rao, S., Mohan, N. D., & Gajja, N. K. (2013). Big data: Testing approach to overcome quality challenges. *Big Data: Challenges and Opportunities*, 11(1), 65-72.

URL: <https://pdfs.semanticscholar.org/e0a1/e310956085428e04c93ec2e3b15dffe112.pdf>