

Cluster Apache Spark

Por Prof. Oscar H. Mondragón

1. Objetivos

- Implementar un Cluster Apache Spark en múltiples nodos.

2. Herramientas a utilizar

- Vagrant
- VirtualBox
- Apache Spark

3. Introducción

En esta práctica de laboratorio, se implementará un cluster de Apache Spark en múltiples nodos. El objetivo es que los estudiantes comprendan cómo funciona un cluster de Spark y cómo se pueden aprovechar las ventajas de procesamiento distribuido para realizar análisis y procesamiento de grandes conjuntos de datos.

En esta práctica, los estudiantes aprenderán a configurar y administrar un cluster de Spark, a lanzar una aplicación en el cluster y a analizar los resultados. Además, se continuarán profundizando en conceptos clave de Spark, como RDDs (Resilient Distributed Datasets), transformaciones y acciones, y cómo estos conceptos se aplican en un entorno distribuido.

4. Desarrollo de la práctica

4.1. Configuración de Vagrant

Esta práctica la desarrollaremos usando boxes de Ubuntu 22.04 en Vagrant. El Vagrantfile que usaremos es el siguiente (le agregamos más cpus y memoria al servidor con el que venimos trabajando):

```
# -*- mode: ruby -*-  
# vi: set ft=ruby :
```

```
Vagrant.configure("2") do |config|
```

```
  if Vagrant.has_plugin? "vagrant-vbguest"  
    config.vbguest.no_install = true  
    config.vbguest.auto_update = false
```

```

    config.vbguest.no_remote = true
end

config.vm.define :clienteUbuntu do |clienteUbuntu|
  clienteUbuntu.vm.box = "bento/ubuntu-22.04"
  clienteUbuntu.vm.network :private_network, ip: "192.168.100.2"
  clienteUbuntu.vm.hostname = "clienteUbuntu"
end

config.vm.define :servidorUbuntu do |servidorUbuntu|
  servidorUbuntu.vm.box = "bento/ubuntu-22.04"
  servidorUbuntu.vm.network :private_network, ip: "192.168.100.3"
  servidorUbuntu.vm.provider "virtualbox" do |v|
    v.cpus = 3
    v.memory = 2048
  end
end
end

```

4.2. Instalación de Apache Spark en servidorUbuntu

Instalar Java en Ubuntu 22.04 (Puede revisar <https://linuxhint.com/install-java-ubuntu-22-04/>)

```
vagrant@servidorUbuntu:~$ sudo apt update
```

```
vagrant@servidorUbuntu:~$ sudo apt install -y openjdk-18-jdk
```

```
vagrant@servidorUbuntu:~$ cat <<EOF | sudo tee /etc/profile.d/jdk18.sh
export JAVA_HOME=/usr/lib/jvm/java-1.18.0-openjdk-amd64
export PATH=$PATH:$JAVA_HOME/bin
EOF
```

```
vagrant@servidorUbuntu:~$ source /etc/profile.d/jdk18.sh
```

Una vez instalado Java puede verificar la instalación obteniendo la versión instalada:

```
vagrant@servidorUbuntu:~$ java -version
openjdk version "18.0.2-ea" 2022-07-19
OpenJDK Runtime Environment (build 18.0.2-ea+9-Ubuntu-22.04)
OpenJDK 64-Bit Server VM (build 18.0.2-ea+9-Ubuntu-22.04, mixed mode, sharing)
```

Descargar y descomprimir Spark

Verifique en

<https://dlcdn.apache.org/spark/>

la última versión de spark y proceda a descargarla usando wget.

En mi caso utilizaré la versión 3.3.2, la cual es la última disponible al momento de escribir esta guía, es posible que deba usar una versión posterior, por lo cual deberá cambiar los números de la versión en el comando de descarga.

```
vagrant@servidorUbuntu:~$ mkdir labSpark  
vagrant@servidorUbuntu:~$ cd labSpark/  
vagrant@servidorUbuntu:~/labSpark$ wget https://dlcdn.apache.org/spark/spark-3.3.2/spark-3.3.2-bin-hadoop3.tgz  
vagrant@servidorUbuntu:~/labSpark$ tar -xvzf spark-3.3.2-bin-hadoop3.tgz
```

4.3. Descargar un dataset de prueba en servidorUbuntu y en clienteUbuntu (*Usar las mismas rutas*)

Utilizaremos el dataset **eCommerce Events History in Cosmetics Shop** que contiene información de comportamiento de 20 millones de usuarios almacenados durante 5 meses en una tienda de cosméticos online.

Cada fila del archivo representa un evento. Todos los eventos están relacionados con productos y usuarios. Cada evento es como una relación de muchos a muchos entre productos y usuarios.

El dataset es de fuente abierta y está disponible en:

<https://www.kaggle.com/datasets/mkechinov/ecommerce-events-history-in-cosmetics-shop>

Cree un directorio para el dataset en `/home/vagrant/labSpark/dataset`

```
vagrant@servidorUbuntu:~/labSpark$ mkdir datasetCluster  
vagrant@servidorUbuntu:~/labSpark$ cd datasetCluster/  
vagrant@servidorUbuntu:~/labSpark/datasetCluster$ pwd
```

```
/home/vagrant/labSpark/datasetCluster
```

Descargue el dataset desde

<https://www.kaggle.com/datasets/mkechinov/ecommerce-events-history-in-cosmetics-shop>

Copielo a su directorio de trabajo de vagrant y despues puede hacer uso de los directorios sincronizados y copiarlo a `/home/vagrant/labSpark/dataset`

```
vagrant@servidorUbuntu:~/labSpark/datasetCluster$ cp /vagrant/archive.zip .  
vagrant@servidorUbuntu:~/labSpark/datasetCluster$ ls  
archive.zip
```

Descomprima el dataset

```
vagrant@servidorUbuntu:~/labSpark/datasetCluster$ sudo apt install zip unzip  
  
vagrant@servidorUbuntu:~/labSpark/datasetCluster$ unzip archive.zip  
Archive:  archive.zip  
  inflating: world_population.csv
```

4.4. Configuración máquina clienteUbuntu

Repita los pasos de las secciones 4.2 y 4.3 en la máquina clienteUbuntu.

Asegúrese que las versiones utilizadas de Spark en servidorUbuntu y ClienteUbuntu sean las mismas.

4.5. Configure un Cluster Spark

Configuraremos un cluster de spark en múltiples nodos, con el master y worker corriendo en máquinas diferentes.

Si no ha hecho esto antes en el master (probablemente lo hizo en una práctica anterior):

Dirijase al directorio de configuración de Spark

```
vagrant@servidorUbuntu:~/labSpark$ cd spark-3.3.2-bin-hadoop3/conf/  
vagrant@servidorUbuntu:~/labSpark/spark-3.3.2-bin-hadoop3/conf$ pwd  
/home/vagrant/labSpark/spark-3.3.2-bin-hadoop3/conf
```

Haga una copia del archive de configuracion de variables de entorno de Spark

```
cp spark-env.sh.template spark-env.sh
```

Edite y agregue al final las configuraciones de SPARK_LOCAL_IP y SPARK_MASTER_HOST así:

```
SPARK_LOCAL_IP=192.168.100.3  
SPARK_MASTER_HOST=192.168.100.3
```

Diríjase a **sbin** e inicie el master:

```
vagrant@servidorUbuntu:~/labSpark/spark-3.3.2-bin-hadoop3/sbin$ ./start-  
master.sh  
starting org.apache.spark.deploy.master.Master, logging to  
/home/vagrant/labSpark/spark-3.3.2-bin-hadoop3/logs/spark-vagrant-  
org.apache.spark.deploy.master.Master-1-servidorUbuntu.out
```

Para verificar abra en un browser <http://192.168.100.3:8080/> para ver la interfaz de administración del master.

Spark Master at spark://192.168.100.3:7077

URL: spark://192.168.100.3:7077
 Alive Workers: 0
 Cores in use: 0 Total, 0 Used
 Memory in use: 0.0 B Total, 0.0 B Used
 Resources in use:
 Applications: 0 Running, 0 Completed
 Drivers: 0 Running, 0 Completed
 Status: ALIVE

▼ Workers (0)

Worker Id	Address	State	Cores	Memory	Resources
-----------	---------	-------	-------	--------	-----------

▼ Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	------------------------	----------------	------	-------	----------

▼ Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	------------------------	----------------	------	-------	----------

Configurar e Iniciar el worker en la máquina clienteUbuntu

Dirijase al directorio de configuración de Spark

```
vagrant@servidorUbuntu:~/labSpark$ cd spark-3.3.2-bin-hadoop3/conf/
vagrant@servidorUbuntu:~/labSpark/spark-3.3.2-bin-hadoop3/conf$ pwd
/home/vagrant/labSpark/spark-3.3.2-bin-hadoop3/conf
```

Haga una copia del archive de configuracion de variables de entorno de Spark

```
cp spark-env.sh.template spark-env.sh
```

Edite y agregue al final las configuraciones de SPARK_LOCAL_IP y SPARK_MASTER_HOST así:

```
SPARK_LOCAL_IP=192.168.100.2
```

```
SPARK_MASTER_HOST=192.168.100.3
```

Dirijase a **sbin** e inicie un worker en clienteUbuntu (debe usar la URL que aparece en la interfaz de administración del master):

```
vagrant@clienteUbuntu:~/labSpark/spark-3.3.2-bin-hadoop3/sbin$ ./start-  
worker.sh spark://192.168.100.3:7077  
starting org.apache.spark.deploy.worker.Worker, logging to  
/home/vagrant/labSpark/spark-3.3.2-bin-hadoop3/logs/spark-vagrant-  
org.apache.spark.deploy.worker.Worker-1-clienteUbuntu.out
```

Puede verificar en el log que se muestra en la salida del anterior comando si el worker se conectó correctamente o si hubo algún error, en mi caso:

```
vagrant@clienteUbuntu:~/labSpark/spark-3.3.2-bin-hadoop3/sbin$ cat  
/home/vagrant/labSpark/spark-3.3.2-bin-hadoop3/logs/spark-vagrant-  
org.apache.spark.deploy.worker.Worker-1-clienteUbuntu.out  
Spark Command: /usr/lib/jvm/java-1.18.0-openjdk-amd64/bin/java -cp  
/home/vagrant/labSpark/spark-3.3.2-bin-  
hadoop3/conf/:/home/vagrant/labSpark/spark-3.3.2-bin-hadoop3/jars/* -Xmx1g  
org.apache.spark.deploy.worker.Worker --webui-port 8081  
spark://192.168.100.3:7077  
=====
```

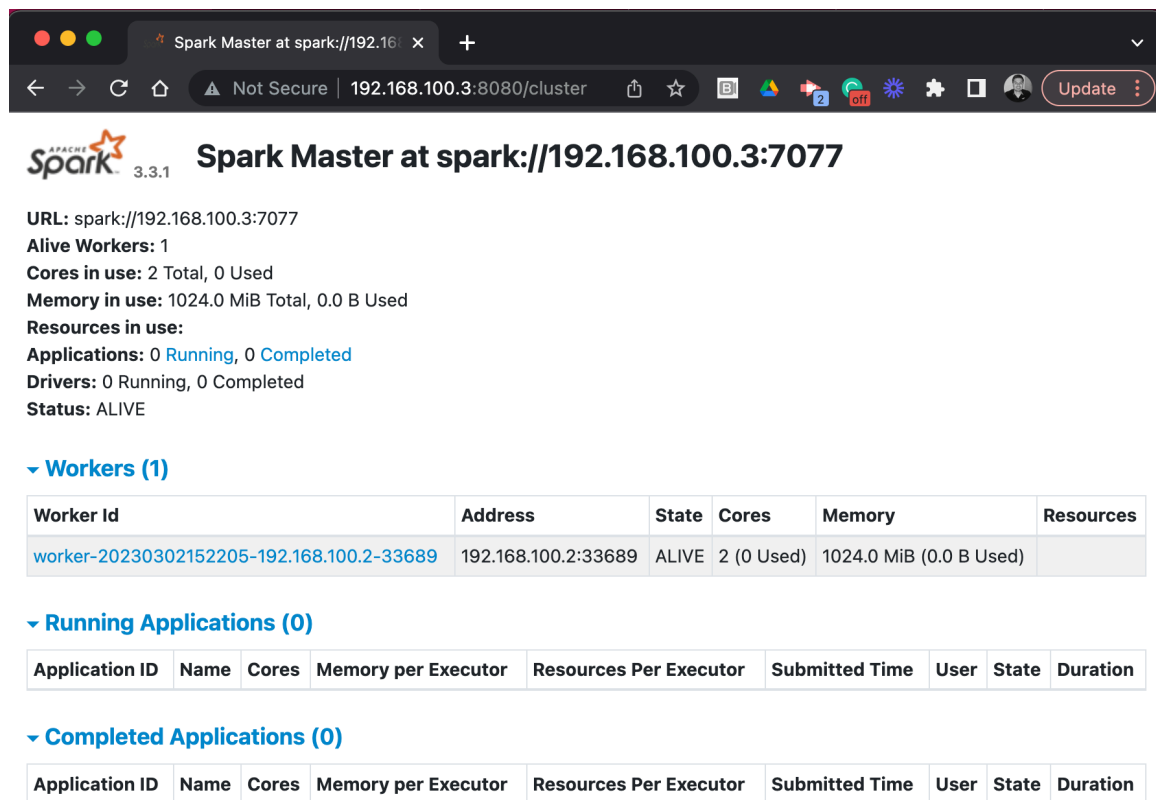
```
Using Spark's default log4j profile: org/apache/spark/log4j2-  
defaults.properties  
23/03/02 15:22:04 INFO Worker: Started daemon with process name:  
7319@clienteUbuntu  
23/03/02 15:22:04 INFO SignalUtils: Registering signal handler for TERM  
23/03/02 15:22:04 INFO SignalUtils: Registering signal handler for HUP  
23/03/02 15:22:04 INFO SignalUtils: Registering signal handler for INT  
23/03/02 15:22:05 WARN NativeCodeLoader: Unable to load native-hadoop library  
for your platform... using builtin-java classes where applicable  
23/03/02 15:22:05 INFO SecurityManager: Changing view acls to: vagrant  
23/03/02 15:22:05 INFO SecurityManager: Changing modify acls to: vagrant  
23/03/02 15:22:05 INFO SecurityManager: Changing view acls groups to:  
23/03/02 15:22:05 INFO SecurityManager: Changing modify acls groups to:  
23/03/02 15:22:05 INFO SecurityManager: SecurityManager: authentication  
disabled; ui acls disabled; users with view permissions: Set(vagrant); groups  
with view permissions: Set(); users with modify permissions: Set(vagrant);  
groups with modify permissions: Set()  
23/03/02 15:22:05 INFO Utils: Successfully started service 'sparkWorker' on  
port 33689.  
23/03/02 15:22:05 INFO Worker: Worker decommissioning not enabled.  
23/03/02 15:22:06 INFO Worker: Starting Spark worker 192.168.100.2:33689 with 2  
cores, 1024.0 MiB RAM  
23/03/02 15:22:06 INFO Worker: Running Spark version 3.3.2  
23/03/02 15:22:06 INFO Worker: Spark home: /home/vagrant/labSpark/spark-3.3.2-  
bin-hadoop3
```

```

23/03/02 15:22:06 INFO ResourceUtils:
=====
23/03/02 15:22:06 INFO ResourceUtils: No custom resources configured for
spark.worker.
23/03/02 15:22:06 INFO ResourceUtils:
=====
23/03/02 15:22:06 INFO Utils: Successfully started service 'WorkerUI' on port
8081.
23/03/02 15:22:06 INFO WorkerWebUI: Bound WorkerWebUI to 192.168.100.2, and
started at http://192.168.100.2:8081
23/03/02 15:22:06 INFO Worker: Connecting to master 192.168.100.3:7077...
23/03/02 15:22:06 INFO TransportClientFactory: Successfully created connection
to /192.168.100.3:7077 after 35 ms (0 ms spent in bootstraps)
23/03/02 15:22:06 INFO Worker: Successfully registered with master
spark://192.168.100.3:7077

```

Y ahora debe aparecer en la sección de workers de <http://192.168.100.3:8080/>



The screenshot shows the Spark Master web UI at <http://192.168.100.3:8080/>. The page title is "Spark Master at spark://192.168.100.3:7077". The status is "ALIVE". The URL is "spark://192.168.100.3:7077". The page shows 1 alive worker, 2 total cores (0 used), 1024.0 MiB total memory (0.0 B used), 0 running applications, and 0 completed applications. The "Workers (1)" section shows a table with one worker.

Worker Id	Address	State	Cores	Memory	Resources
worker-20230302152205-192.168.100.2-33689	192.168.100.2:33689	ALIVE	2 (0 Used)	1024.0 MiB (0.0 B Used)	

4.6. Lanzar una aplicación

Para probar nuestro clúster lanzaremos una aplicación que haga uso del dataset descargado. Esta aplicación ya la creamos en una práctica anterior, si ya la creó no es necesario escribirla de nuevo.

app1.py

Cree un archivo app1.py en un directorio app, así:

```
vagrant@servidorUbuntu:~/labSpark/resultsRDD$ cd ..
vagrant@servidorUbuntu:~/labSpark$ mkdir app
vagrant@servidorUbuntu:~/labSpark$ cd app/
vagrant@servidorUbuntu:~/labSpark/app$ touch app1.py
```

Agregue el siguiente contenido (verifique que la indentación esté correcta):

```
#app1.py

import sys
from pyspark.sql import SparkSession

spark=SparkSession.builder.getOrCreate()

df=spark.read.options(header='True',inferSchema='True').csv(sys.argv[1])

def myFunc(s):
    if s["brand"]=="riche" and s["event_type"]=="cart":
        return[(s["product_id"],1)]
    return[]

lines=df.rdd.flatMap(myFunc).reduceByKey(lambda a,b:a+b)
lines.saveAsTextFile(sys.argv[2])
```

Para enviar el trabajo, diríjase al directorio bin de spark y utilice el siguiente comando (la URL del master es la que aparece en el dashboard, ver sección 4.2):

```
vagrant@servidorUbuntu:~/labSpark/spark-3.3.1-bin-hadoop3/bin$ ./spark-submit -
-master spark://192.168.100.3:7077 --conf spark.executor.memory=1g
/home/vagrant/labSpark/app/app1.py "/home/vagrant/labSpark/dataset/*csv"
"/home/vagrant/labSpark/resultsCluster"
23/02/16 22:01:55 INFO SparkContext: Running Spark version 3.3.1
23/02/16 22:01:55 WARN NativeCodeLoader: Unable to load native-hadoop library
for your platform... using builtin-java classes where applicable
23/02/16 22:01:55 INFO ResourceUtils:
=====
23/02/16 22:01:55 INFO ResourceUtils: No custom resources configured for
spark.driver.
```

```
23/02/16 22:01:55 INFO ResourceUtils:
```

```
=====
23/02/16 22:01:55 INFO SparkContext: Submitted application: app1.py
23/02/16 22:01:55 INFO ResourceProfile: Default ResourceProfile created,
executor resources: Map(cores -> name: cores, amount: 1, script: , vendor: ,
memory -> name: memory, amount: 1024, script: , vendor: , offHeap -> name:
offHeap, amount: 0, script: , vendor: ), task resources: Map(cpus -> name:
cpus, amount: 1.0)
...
```

Verifique los resultados en resultsCluster (en clienteUbuntu), Ejemplo:

```
vagrant@clienteUbuntu:~/labSpark$ cd resultsT2/
vagrant@clienteUbuntu:~/labSpark/resultsT2$ ls
_temporary
vagrant@clienteUbuntu:~/labSpark/resultsT2$ cd _temporary/
vagrant@clienteUbuntu:~/labSpark/resultsT2/_temporary$ ls
0
vagrant@clienteUbuntu:~/labSpark/resultsT2/_temporary$ cd 0
vagrant@clienteUbuntu:~/labSpark/resultsT2/_temporary/0$ ls
task_202303022114558911610408487536158_0021_m_000000
task_202303022114558911610408487536158_0021_m_000010
task_202303022114558911610408487536158_0021_m_000001
task_202303022114558911610408487536158_0021_m_000011
task_202303022114558911610408487536158_0021_m_000002
task_202303022114558911610408487536158_0021_m_000012
task_202303022114558911610408487536158_0021_m_000003
task_202303022114558911610408487536158_0021_m_000013
task_202303022114558911610408487536158_0021_m_000004
task_202303022114558911610408487536158_0021_m_000014
task_202303022114558911610408487536158_0021_m_000005
task_202303022114558911610408487536158_0021_m_000015
task_202303022114558911610408487536158_0021_m_000006
task_202303022114558911610408487536158_0021_m_000016
task_202303022114558911610408487536158_0021_m_000007
task_202303022114558911610408487536158_0021_m_000017
task_202303022114558911610408487536158_0021_m_000008
task_202303022114558911610408487536158_0021_m_000018
task_202303022114558911610408487536158_0021_m_000009 _temporary
vagrant@clienteUbuntu:~/labSpark/resultsT2/_temporary/0$ cd
task_202303022114558911610408487536158_0021_m_000000/
vagrant@clienteUbuntu:~/labSpark/resultsT2/_temporary/0/task_202303022114558911
610408487536158_0021_m_000000$ ls
part-00000
vagrant@clienteUbuntu:~/labSpark/resultsT2/_temporary/0/task_202303022114558911
610408487536158_0021_m_000000$ cat part-00000
(5844571, 20)
(5842234, 5)
(5842215, 10)
```

Verifique la aplicación lanzada en el dashboard :

<http://192.168.100.3:8080/cluster>

5. Ejercicio

1. Descargue el dataset disponible en <https://www.kaggle.com/datasets/iamsouravbanerjee/world-population-dataset>
2. Escriba una aplicación en PySpark para procesar el archivo CSV del dataset, contar los países de Sudamérica con una población superior a 30 millones de habitantes y guardar los resultados en un archivo de texto.
3. Lance la aplicación en el cluster montado y compruebe su funcionamiento.

6. Entregables y Evaluación

- Sustentación de la práctica

7. Bibliografía

- Instalar Java en Ubuntu 22.04. <https://linuxhint.com/install-java-ubuntu-22-04/>
- Apache Spark Cluster Setup. <https://www.tutorialkart.com/apache-spark/how-to-setup-an-apache-spark-cluster/>
- DataScience con PySpark I: Apache Spark, Python, DataFrames y RDDs. <https://youtu.be/iMOgTbaDJXc>