

# Trabajo Práctico Módulo 4. Servicios gestionados en la nube para el procesamiento de datos masivos

Máster en Ingeniería y Ciencia de Datos  
Curso 2021/2022



Jesús Galán Llano  
[jgalan279@alumno.uned.es](mailto:jgalan279@alumno.uned.es)

## Índice

1. Introducción.....	1
2. Metodología de desarrollo.....	1
3. Recursos y servicios empleados .....	2
4. Desarrollo de la solución .....	3
5. Conclusiones.....	9
Referencias.....	10

## 1. Introducción

Este documento contiene la cuarta práctica de la asignatura “Infraestructuras Computacionales para el Procesamiento de Datos Masivos”. La misma consiste en diseñar y desplegar una infraestructura en la nube para realizar ciertas pruebas con el fin de medir la potencia computacional y el uso de recursos de la infraestructura creada.

La práctica puede ser desarrollada utilizando dos de los proveedores *cloud* más conocidos actualmente, que son Amazon Web Services (AWS) y Google Cloud Platform (GCP).

El documento está dividido en cinco apartados: el primer apartado es una introducción al mismo, el segundo apartado se incluye la metodología de desarrollo, en el tercero se explican los recursos y servicios empleados en crear la solución, en el cuarto se detalla el desarrollo de la solución y, por último, en el quinto apartado se incluyen las conclusiones del autor.

## 2. Metodología de desarrollo

En este apartado se describe el procedimiento seguido para el desarrollo de la práctica. En primer lugar, se ha realizado un análisis de las dos plataformas en la nube que se proponen en el enunciado de esta, AWS y GCP, con el fin de determinar qué proveedor ofrece los mejores servicios para implementar la solución.

En mi caso, he decidido elegir AWS debido a que cuento con cierta experiencia y con conocimientos previos sobre esta plataforma. Además, considero que los productos que ofrece permiten crear soluciones de mejor calidad que los de GCP debido a la cantidad de opciones de configuración que permite, tanto a nivel de recursos como de conexiones entre los componentes, y debido también a la facilidad de despliegue y monitorización de los productos.

Una vez seleccionado el proveedor de servicios de computación en la nube, el siguiente paso ha consistido en escoger una herramienta de creación de benchmarks que permitan evaluar el rendimiento de la solución. De entre todas las opciones propuestas en el enunciado se ha elegido la tercera opción, spark-bench, debido a varios motivos que se detallan a continuación. Esta herramienta ofrece una gran flexibilidad porque permite a los usuarios crear sus propias pruebas utilizando ficheros de configuración que pueden estar formados por una o más cargas de trabajo o *workloads* de Apache Spark. Cada carga de trabajo tiene sus propias características y permite probar un escenario concreto. Por ejemplo, se puede simular una carga de trabajo de un algoritmo de Machine Learning (ML), una carga de trabajo que requiera de una gran capacidad de cómputo o la generación de datos y la ejecución de consultas sobre estos.

Tras esto, los pasos siguientes han consistido en realizar un análisis de los productos necesarios para crear la infraestructura, además de su configuración, la ejecución de las pruebas de rendimiento y, por último, se ha realizado un análisis de los resultados con el fin de obtener conclusiones acerca de los recursos empleados. Estos pasos están explicados en los siguientes apartados.

### 3. Recursos y servicios empleados

En este apartado se enumeran y se describen los recursos y los servicios escogidos para llevar a cabo la solución. Los recursos que ofrece AWS consisten en un amplio abanico de posibilidades, pudiendo configurar desde máquinas virtuales, bases de datos, aplicaciones empresariales, Internet de las Cosas (IoT), servicios de seguridad e identidad y hasta servicios multimedia.

Uno de los puntos clave consiste en realizar un estudio exhaustivo acerca de los servicios que ofrecen este tipo de plataformas para asegurarnos de que sus funcionalidades se adaptan a nuestros requisitos. En base a esto, debemos de seleccionar como mínimo dos productos que formarán la infraestructura: uno de almacenamientos de datos que permita guardar archivos, datos de entrada y resultados de manera persistente y otro orientado a la computación de grandes volúmenes de datos. Después de analizar los productos que ofrece AWS se ha decidido escoger los siguientes:

- **Amazon S3:** Amazon Simple Storage Service (S3) es un servicio de almacenamiento de objetos que proporciona escalabilidad, disponibilidad, seguridad y rendimiento (Amazon, 2022). Estos objetos son la entidad fundamental de S3 y están formados por datos de objetos y metadatos que permiten describir al objeto. S3 guarda los datos dentro de *buckets*, que funcionan como contenedores. Cada objeto se identifica de forma única dentro de un *bucket* a través de su clave, su nombre, y un identificador de versionado. En cuanto a los beneficios que ofrece se debe destacar, además de los ya comentados, su facilidad de uso, que permite una gran variedad de transferencia de datos, y su coste ya que solamente se paga por su uso y no por su infraestructura.
- **Amazon EMR:** conocida anteriormente como Amazon Elastic MapReduce, es un clúster de equipos orientados al procesamiento de grandes volúmenes de datos distribuidos a gran escala (Amazon, 2022). Los clústeres se pueden configurar con las soluciones más empleadas en esta área como son Apache Hadoop, Apache Spark, Hive o Presto. Aparte del procesamiento de datos almacenados en bases de datos también puede ser empleado para el procesamiento de datos en tiempo real y para la creación de modelos en el área de Machine Learning. Entre sus ventajas se encuentran las siguientes: gran capacidad de cómputo, grandes opciones de configuración con las aplicaciones más recientes, alta disponibilidad y escalabilidad. Además, Amazon proporciona una lista de precios detallada para cada configuración (Amazon, 2022).

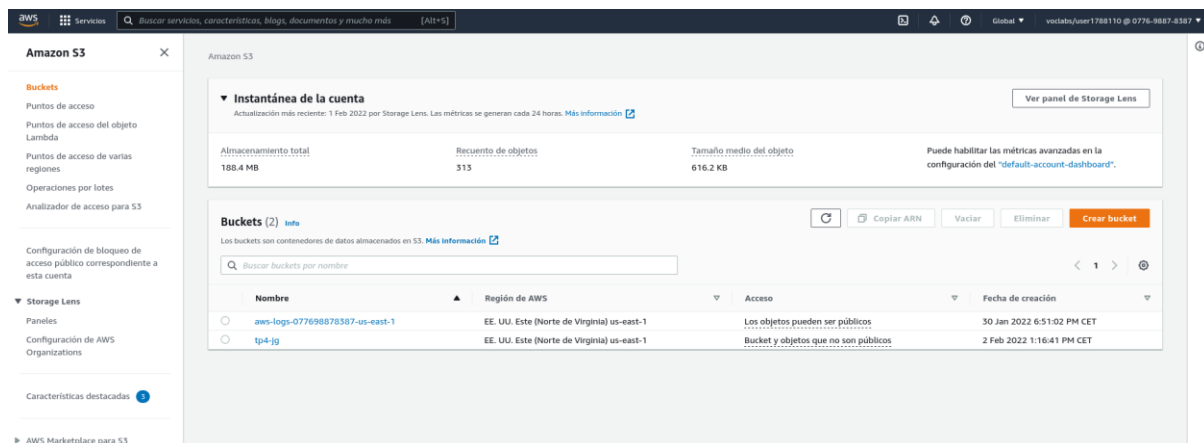
La conexión entre ambos productos permite crear de forma sencilla y rápida una infraestructura simple para el procesamiento de datos y para realizar pruebas de rendimiento sobre las distintas opciones de configuración que ofrece EMR. De esta forma S3 se encargará de toda la parte de almacenamiento de datos, tanto de entrada como de salida, mientras que EMR se encargará de procesar dichos datos y enviar los resultados devuelta a S3.

## 4. Desarrollo de la solución

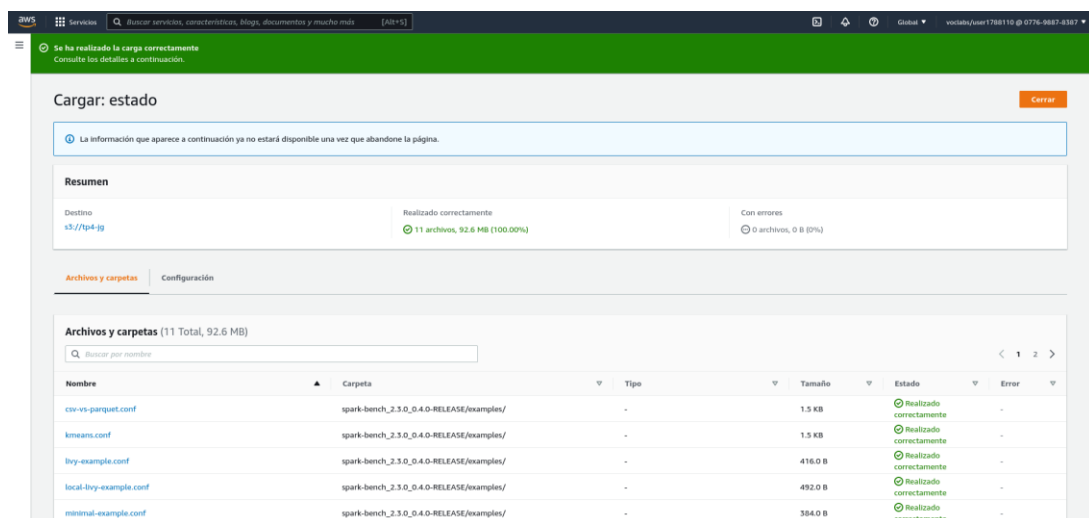
Este apartado contiene los pasos que se han seguido para el desarrollo de la práctica. Para cada paso se incluyen imágenes explicativas sobre la interacción con la interfaz de configuración de AWS.

Antes de nada, se ha accedido al curso de AWS Academy proporcionado por el equipo docente. Una vez dentro de la pestaña “Módulos” se selecciona la opción “Start Lab” que permite conectarnos a una máquina EC2 y acceder a la interfaz de configuración de AWS. Una vez hecho esto se han realizado los siguientes pasos:

1. **Creación del S3.** En primer lugar, se ha desplegado un nuevo *bucket* de S3 para el almacenamiento de los datos. Para ello hemos buscado este servicio en la interfaz y una vez dentro se ha seleccionado el botón “Crear bucket”. En esta página se ha incluido el nombre del *bucket* y su configuración destacando que se ha habilitado el control de versiones.



2. **Modificación de ficheros y subida a S3.** Una vez configurado el producto de almacenamiento se ha pasado a modificar los ficheros necesarios para realizar las pruebas de rendimiento para su posterior subida.



Las modificaciones que se han realizado consisten en establecer la ruta correcta de Spark en EMR y configurar las rutas de entrada y salida de datos dentro del *bucket* S3 antes creado. Para obtener la ruta de Spark en EMR se puede emplear el comando: “whereis spark”. Estos ficheros de configuración se incluyen dentro del archivo comprimido entregado. Además de estos ficheros también se han subido los archivos necesarios para ejecutar las pruebas según la documentación de spark-bench. Para subir los ficheros se ha seleccionado el *bucket* y una vez dentro se ha seleccionado la opción “Cargar”.

- 3. Creación del EMR.** La creación de un clúster EMR se realiza seleccionando el botón “Crear clúster”. Este botón nos lleva a la página de configuración del clúster donde podemos establecer un nombre y, lo más importante, su configuración de software y de hardware. Respecto a la configuración de software, se ha seleccionado la versión emr-5.34.0 que permite instalar la aplicación Spark 2.4.8 sobre Hadoop 2.120.1 YARN y Zeppelin 0.10.0. Esto se debe a que en la guía de uso de spark-bench se recomienda que la versión de Spark empleada sea una versión 2.x. A pesar de que existe la tercera versión de Spark este *benchmark* no está configurado para permitir en este momento su uso. En cuanto a la configuración de hardware, se ha seleccionado la instancia de tipo general m5.xlarge. Esta instancia cuenta con un volumen de EBS GP2 de 64GiB y 1 *core* físico (Amazon, 2022). Por último, se han creado un par de clases EC2 con el fin de acceder a la máquina de forma remota.

Crear clúster: Opciones rápidas [Ir a las opciones avanzadas](#)

**Configuración general**

Nombre del clúster:

☒ Registro

Carpeta S3:

Modo lanzamiento: ☒ Clúster ☐ Ejecución de pasos

**Configuración de software**

Versión:

Aplicaciones:

- ☐ Core Hadoop: Hadoop 2.10.1, Hive 2.3.8, Hue 4.9.0, Mahout 0.13.0, Pig 0.17.0, and Tez 0.9.2
- ☐ HBase: HBase 1.4.13, Hadoop 2.10.1, Hive 2.3.8, Hue 4.9.0, Phoenix 4.14.3, and ZooKeeper 3.4.14
- ☐ Presto: Presto 0.261 with Hadoop 2.10.1 HDFS and Hive 2.3.8 Metastore
- ☒ Spark: Spark 2.4.8 on Hadoop 2.10.1 YARN and Zeppelin 0.10.0

☐ Usar el catálogo de datos de AWS Glue para metadatos de tabla

**Configuración de hardware**

Tipo de instancia:  El tipo de instancia seleccionado añade un volumen de EBS GP2 de 64 GiB predeterminado por instancia. [Más información](#)

Número de instancias:  Nodos maestros: (1) y Nodos principales: (2)

Cluster scaling: ☐ scale cluster nodes based on workload

Terminación automática: ☒ Habilitar la terminación automática [Más información](#)

Terminar el clúster si permanece inactivo después de:  horas

minutos

Una vez configurado el clúster se selecciona la opción “Crear clúster” y se espera a que el producto esté listo para poder trabajar con él, es decir, cuando su estado sea “Esperando”.

- 4. Configuración del EMR.** Una vez desplegado el EMR, hace falta añadir una serie de configuraciones antes de comenzar a trabajar con él. En primer lugar, se debe habilitar el puerto correspondiente al servicio SSH para poder acceder a la máquina de forma remota. Para ello vamos a la pantalla de detalle del clúster y en la sección “Seguridad y acceso” seleccionamos la ruta que aparece en el campo “Grupos de seguridad para

SSH

TCP

22

Mi IP

Q

85.56.243.145/32

X

Eliminar

Agregar regla


Cancelar

Previsualizar los cambios

Guardar reglas

```
ssh -i mykeypair.pem hadoop@ec2-3-87-154-206.compute-1.amazonaws.com
```

```
jesus@jesus-Aspire-A514-52:~/university/Infraestructuras_Computacionales/practicas/tp4$ ssh -l mykeypair.pem hadoop@ec2-44-193-80-133.compute-1.amazonaws.com
The authenticity of host 'ec2-44-193-80-133.compute-1.amazonaws.com (44.193.80.133)' can't be established.
ECDSA key fingerprint is SHA256:IbeI7cHQ3obqclps6s7BceTDgtZvX8nZeuhzWLSDA.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-44-193-80-133.compute-1.amazonaws.com,44.193.80.133' (ECDSA) to the list of known hosts.
```



```
Amazon Linux 2 AMI

https://aws.amazon.com/amazon-linux-2/
21 package(s) needed for security, out of 26 available
Run "sudo yum update" to apply all updates.

EEEEEEEEEEEEEEEEEE MMMMMMMM                MMMMMMMM RRRRRRRRRRRRRRRR
E:::EEEEEEEEEEEEEE M:::MM M:::MM R:::MM R:::MM
EE::EEEEEEEEEEEEEE M:::MM M:::MM R:::RRRRRR:::R
E:::E EEEEE M:::MM M:::MM RR:::R R:::R
E:::E M:::MM M:::MM M:::MM R:::R R:::R
E:::EEEEEEEEEE M:::M M:::M M:::M R:::RRRRRR:::R
E:::E M:::M M:::M M:::M R:::RRRRRR:::R
E:::EEEEEEEEEE M:::M M:::M R:::RRRRRR:::R
E:::E M:::M M:::M M:::M R:::R R:::R
E:::E EEEEE M:::M MMM M:::M R:::R R:::R
EE::EEEEEEEEEEEEEE M:::M M:::M R:::R R:::R
E:::E M:::M M:::M R:::R R:::R
EEEEEEEEEEEEEEEEEE MMMMMMMM MMMMMMMM RRRRRRR RRRRRR
```

```
[hadoop@ip-172-31-5-232 ~]$
```

```
aws s3 cp s3://tp4-jg/spark-bench_2.3.0_0.4.0-RELEASE/ ./ --recursive
```

```
chmod +x bin/spark-bench.sh
```

Por último, configuramos las variables de entorno necesarias para ejecutar spark-bench. Estas variables son SPARK\_HOME y SPARK\_MASTER\_HOST. La primera contiene la ruta la instalación local de Spark, que se puede obtener con el comando whereis spark, mientras que la segunda permite establecer el proceso maestro de Spark, que en este caso se corresponde con un proceso local.

**5. Configuración y ejecución de los benchmark en EMR.** Una vez finalizado el proceso de configuración del EMR el siguiente paso es la ejecución de las pruebas de rendimiento. Cada prueba está formada por distintas cargas de trabajo de las que se ha decidido seleccionar las siguientes:

- **SparkPI.** Esta carga de trabajo realiza un cálculo aproximado del valor de Pi generando números aleatorios entre los puntos (0,0) y (1,1) (spark-bench, 2022). Sirve principalmente para probar la capacidad de cómputo sin que se vea afectada por operaciones de lectura o escritura de datos. Permite configurar el parámetro “slices” que se corresponde con la cantidad de números aleatorios que se generan.
- **Data Generator - KMeans.** Esta carga de trabajo permite generar una cantidad de datos para ser procesados posteriormente por el algoritmo KMeans. Sus argumentos permiten establecer el número de filas y de columnas que se van a generar. Además, permite indicar la ruta final del fichero, lo que aporta mucha flexibilidad y nos permite, por ejemplo, guardar los datos en nuestra instancia de S3.
- **SQL.** Esta carga de trabajo permite realizar una consulta sobre los datos de entrada (spark-bench, 2022). La ruta de los datos de entrada puede especificarse como una ruta de un fichero en S3. También se puede almacenar el resultado final modificando el parámetro output.

Respecto a las pruebas de rendimiento, comentar que se han creado los dos siguientes:

- **pi.conf.** Esta prueba de rendimiento utiliza la carga de trabajo SparkPi. Está configurado para generar 50.0000 números aleatorios para sacar el máximo partido de los recursos.
- **sql.conf.** Esta prueba de rendimiento utiliza las cargas de trabajo Data Generator - Kmeans y SQL. Consiste en generar dos archivos de datos en formato csv y en formato parquet y realiza una consulta SQL. En total se generan 20.000 filas de 24 columnas cada una. Los datos generados se guardan en una ruta de S3, al igual que los resultados de las consultas.

Tras configurar los *benchmark* simplemente se deben de ejecutar con los comandos:

```
./bin/spark-bench.sh examples/pi.conf
```

```
./bin/spark-bench.sh examples/sql.conf
```



6. **Análisis de los resultados.** Una vez que han finalizado los *benchmarks* se ha pasado a analizar los resultados de su ejecución. Estos resultados pueden ser accedidos desde el enlace “Spark history server” que se encuentra dentro de la pestaña “Historial de aplicaciones”. Una vez dentro podemos observar todos los trabajos, tanto los finalizados como los no completados:

**History Server**

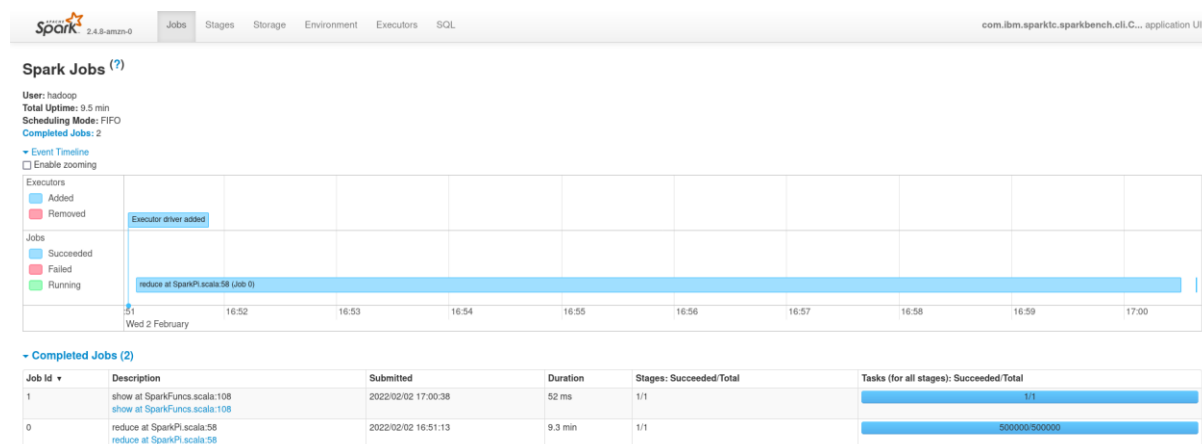
Event log directory: s3a://prod-us-east-1-appinfo.sroj.ZYKRSPL0T5K/sparklogs  
Last updated: 2022-02-02 19:45:01  
Client local time zone: Europe/Madrid

Search:

App ID	App Name	Started	Completed	Duration	Spark User	Last Updated	Event Log
local-1643826882928	com.ibm.sparktc.sparkbench.cli.CLICKickoff	2022-02-02 19:34:41	2022-02-02 19:35:07	26 s	hadoop	2022-02-02 19:35:21	<a href="#">Download</a>
local-1643825821979	com.ibm.sparktc.sparkbench.cli.CLICKickoff	2022-02-02 19:17:00	2022-02-02 19:18:21	1.4 min	hadoop	2022-02-02 19:19:21	<a href="#">Download</a>
local-1643825691183	com.ibm.sparktc.sparkbench.cli.CLICKickoff	2022-02-02 19:14:49	2022-02-02 19:15:18	30 s	hadoop	2022-02-02 19:15:21	<a href="#">Download</a>
local-1643825667898	com.ibm.sparktc.sparkbench.cli.CLICKickoff	2022-02-02 19:14:25	2022-02-02 19:14:31	6 s	hadoop	2022-02-02 19:15:21	<a href="#">Download</a>
local-1643825463471	com.ibm.sparktc.sparkbench.cli.CLICKickoff	2022-02-02 19:11:01	2022-02-02 19:11:27	26 s	hadoop	2022-02-02 19:13:21	<a href="#">Download</a>
application_1643814360288_0001	com.ibm.sparktc.sparkbench.cli.CLICKickoff	2022-02-02 19:02:39	2022-02-02 19:03:50	1.2 min	hadoop	2022-02-02 19:05:21	<a href="#">Download</a>
local-1643824813862	com.ibm.sparktc.sparkbench.cli.CLICKickoff	2022-02-02 19:00:11	2022-02-02 19:00:38	27 s	hadoop	2022-02-02 19:01:21	<a href="#">Download</a>
local-1643824768991	com.ibm.sparktc.sparkbench.cli.CLICKickoff	2022-02-02 18:59:27	2022-02-02 18:59:36	9 s	root	2022-02-02 19:01:21	<a href="#">Download</a>
local-1643824713758	com.ibm.sparktc.sparkbench.cli.CLICKickoff	2022-02-02 18:58:31	2022-02-02 18:58:40	8 s	hadoop	2022-02-02 18:59:21	<a href="#">Download</a>
local-1643821635320	com.ibm.sparktc.sparkbench.cli.CLICKickoff	2022-02-02 18:07:13	2022-02-02 18:07:38	25 s	hadoop	2022-02-02 18:09:21	<a href="#">Download</a>
local-1643821372846	com.ibm.sparktc.sparkbench.cli.CLICKickoff	2022-02-02 18:02:51	2022-02-02 18:03:16	26 s	hadoop	2022-02-02 18:03:21	<a href="#">Download</a>
local-1643820668577	com.ibm.sparktc.sparkbench.cli.CLICKickoff	2022-02-02 17:51:06	2022-02-02 18:00:38	9.5 min	hadoop	2022-02-02 18:01:21	<a href="#">Download</a>
local-1643820409546	com.ibm.sparktc.sparkbench.cli.CLICKickoff	2022-02-02 17:46:47	2022-02-02 17:48:21	1.6 min	hadoop	2022-02-02 17:49:21	<a href="#">Download</a>

Showing 1 to 13 of 13 entries  
[Show incomplete applications](#)

El primer *benchmark* arroja los siguientes resultados:



En la imagen anterior se puede observar que el tiempo total de ejecución ha sido de 9.5 minutos. Si accedemos a la pestaña “Executors” podemos obtener métricas adicionales:

**Executors**

Show Additional Metrics  
Select All  
On Heap Memory  
Off Heap Memory

Summary

	RDD Blocks	Storage Memory	On Heap Storage Memory	Off Heap Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Blacklisted
Active(1)	0	0.0 B / 956.6 MB	0.0 B / 956.6 MB	0.0 B / 0.0 B	0.0 B	4	0	0	500001	500001	39 min (43 s)	0.0 B	0.0 B	0.0 B	0
Dead(0)	0	0.0 B / 0.0 B	0.0 B / 0.0 B	0.0 B / 0.0 B	0.0 B	0	0	0	0	0	0 ms (0 ms)	0.0 B	0.0 B	0.0 B	0
Total(1)	0	0.0 B / 956.6 MB	0.0 B / 956.6 MB	0.0 B / 0.0 B	0.0 B	4	0	0	500001	500001	39 min (43 s)	0.0 B	0.0 B	0.0 B	0

Executors

Show 20 entries

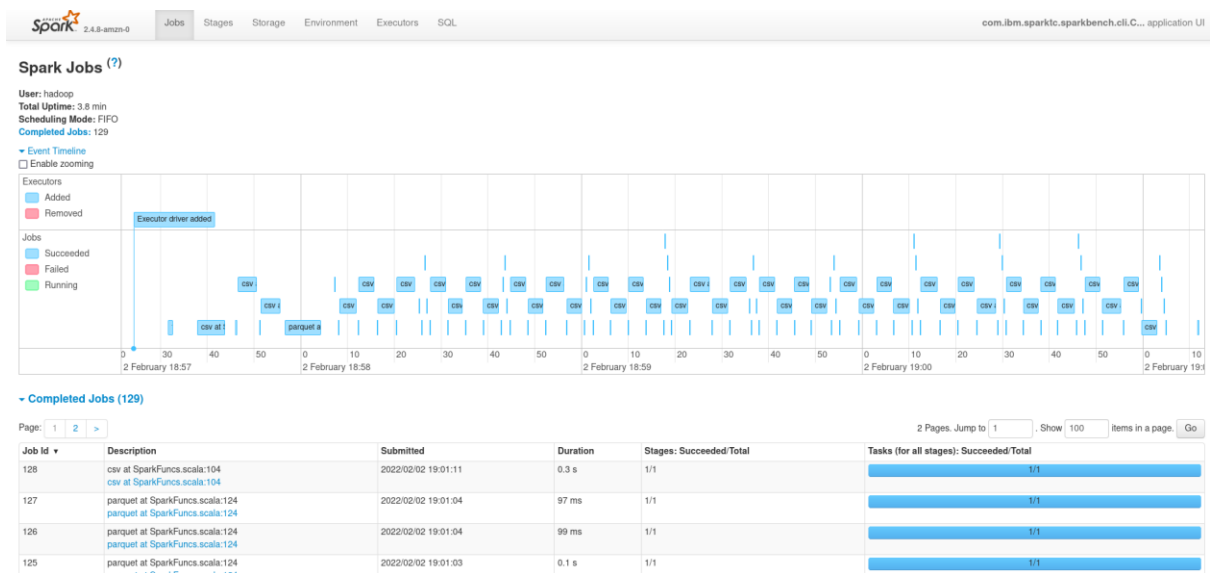
Executor ID	Address	Status	RDD Blocks	Storage Memory	On Heap Storage Memory	Off Heap Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write
driver	ip-172-31-5-232.ec2.internal:41959	Active	0	0.0 B / 956.6 MB	0.0 B / 956.6 MB	0.0 B / 0.0 B	0.0 B	4	0	0	500001	500001	39 min (43 s)	0.0 B	0.0 B	0.0 B

Showing 1 to 1 of 1 entries

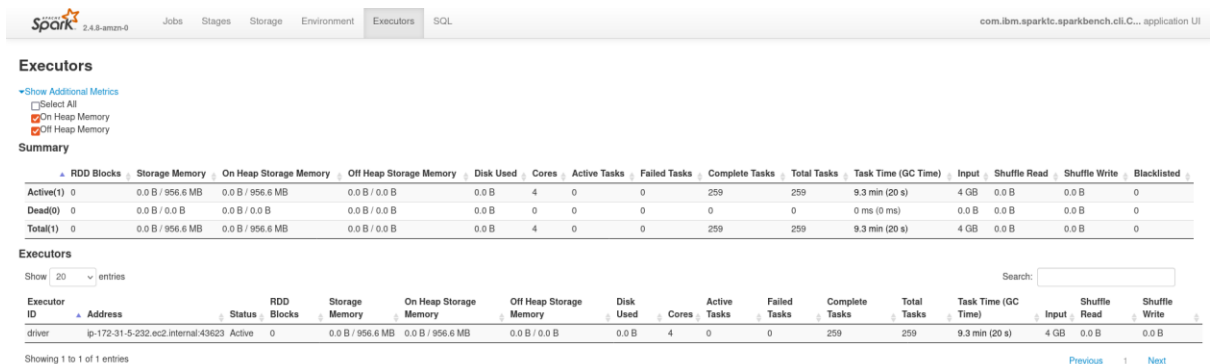
Previous 1 Next

En la imagen anterior se puede observar que la memoria total empleada por el *benchmark* ha sido de 956,6 MB.

En cuanto al segundo *benchmark* obtenemos los siguientes resultados tras una ejecución de 4,2 minutos:



En la pestaña “Executors” obtenemos el siguiente resultado:



De los resultados obtenidos podemos afirmar lo siguiente: la ejecución del primer *benchmark* ha necesitado de más recursos y de tiempo para finalizar que la segunda prueba. Esto se debe principalmente a que el número de iteraciones del primero es mucho mayor que el segundo. En términos computacionales, el primer *benchmark* ha necesitado en total 9,5 minutos para procesar 500.000 trabajos, que es un tiempo más que razonable teniendo en cuenta el gran número de trabajos. Respecto al segundo *benchmark*, se ha demostrado que las consultas realizadas sobre los archivos en formato parquet han necesitado de un tiempo menor que aquellas realizadas sobre los archivos con formato csv. También, se debe mencionar que en este test el tiempo de los trabajos es muy pequeño en comparación con el primer test, ya que las consultas realizadas no han consumido excesivamente recursos debido a que son sencillas al consistir únicamente en una operación de filtrado y una proyección.

## 5. Conclusiones

Esta práctica me ha resultado muy interesante y me ha permitido poner en práctica los conocimientos adquiridos en el cuarto tema de la asignatura.

A nivel personal, esta práctica ha supuesto un reto porque ha sido la primera vez que he trabajado con soluciones en la nube desde su análisis, su configuración, su despliegue y su uso y, a pesar de que he tenido varias complicaciones al principio del trabajo, una vez que he practicado con los productos empleados me ha resultado bastante sencillo y de una gran utilidad. Respecto a los recursos económicos empleados, comentar que solamente se han gastado 4,9\$ de los 100\$ disponibles. Esto pone demuestra que con un presupuesto bajo se pueden desplegar infraestructuras sencillas con cierta capacidad de cómputo y de almacenamiento. Sin duda hacer la práctica me ha supuesto tener más curiosidad sobre este tema y sobre las múltiples soluciones que se pueden crear utilizando estas infraestructuras.

Otro punto positivo ha sido que la documentación de AWS es muy extensa y hay un gran número de recursos públicos, como ejemplos y vídeos explicativos. Esto me ha permitido ampliar mis conocimientos en este campo e identificar qué es lo que tenía que hacer. Sin embargo, los apuntes de la asignatura para este tema me han parecido insuficientes porque son muy escasos y no contienen mucha información relevante. Me hubiera gustado que hubiera más vídeos explicativos que enseñaran a utilizar varios productos de AWS siguiendo el mismo formato de los vídeos publicados en el aula virtual sobre EC2.

Sin embargo, he de decir que me ha resultado complejo encontrar una herramienta para generar *benchmarks* para poder utilizarla en AWS. De los cuatro enlaces propuestos solamente he sido capaz de configurar correctamente uno, spark-bench, lo cual me ha llevado bastante tiempo. Sería recomendable que en futuras prácticas se indicase en el enunciado de la práctica qué *benchmarks* utilizar, así como detallar sus posibles configuraciones.

En definitiva, a pesar de las complicaciones esta práctica me ha resultado de gran utilidad, muy entretenida y ha hecho que aumentara mi curiosidad sobre el área de computación en la nube consultando más recursos y realizando prácticas adicionales sobre esta infraestructura.

## Referencias

- Amazon. (1 de 2 de 2022). *¿Qué es Amazon EMR?* Obtenido de [https://docs.aws.amazon.com/es\\_es/emr/latest/ManagementGuide/emr-what-is-emr.html](https://docs.aws.amazon.com/es_es/emr/latest/ManagementGuide/emr-what-is-emr.html)
- Amazon. (1 de 2 de 2022). *Núcleos físicos de tipo de instancia Amazon EC2*. Obtenido de <https://aws.amazon.com/es/ec2/physicalcores/>
- Amazon. (1 de 2 de 2022). *Precios de Amazon EMR*. Obtenido de <https://aws.amazon.com/es/emr/pricing/>
- Amazon. (1 de 2 de 2022). *What is Amazon S3?* Obtenido de <https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html>
- spark-bench. (1 de 2 de 2022). *SparkPi*. Obtenido de <https://codait.github.io/spark-bench/workloads/sparkpi/>
- spark-bench. (1 de 2 de 2022). *SQL*. Obtenido de <https://codait.github.io/spark-bench/workloads/sql/>