



ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS
INGENIERÍA DE SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

NOMBRE DE ESTUDIANTE: Díaz Padilla Danny Sebastián

Laboratorio de:

ANALÍTICA DE DATOS – BIG DATA

Práctica de la clase 21

Tema: Databricks

Objetivos:

1. Utilizar la herramienta Databricks community edition
2. Crear un cluster en la herramienta Databricks Community Edition.
3. Identificar cada función utilizada en un código de muestra y explicarlo

Marco teórico:

Apache Spark

Es el estándar abierto para el procesamiento flexible de datos en memoria que permite análisis por lotes, en tiempo real y avanzados en la plataforma Apache Hadoop. A través de One Platform Initiative, [1]



Databricks

Es una plataforma web para trabajar con Spark, aportando administración de los clústeres de Spark y notebooks al estilo IPython. [2]





ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS
INGENIERÍA DE SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

Desarrollo de la práctica:

A. Creamos nuestra cuenta y creamos un clúster

The image shows the Databricks home dashboard. On the left is a sidebar with navigation links: Home, Workspace, Recents, Data, Clusters, Jobs, and Search. The main area has three large cards: 'Explore the Quickstart Tutorial', 'Import & Explore Data', and 'Create a Blank Notebook'. Below these are sections for 'Common Tasks' (New Notebook, Create Table, New Cluster, New Job, New MLflow Experiment, Import Library), 'Recents' (Notebook_ABD, ABD_Ejercicios), and 'What's new in v3.11'. A user menu is open in the top right, showing the user is signed in as 'meliodasfistry@gmail.com' with options for User Settings, Admin Console, Manage Account, Log Out, and Workspaces.

Create Cluster

New Cluster

Cancel

Create Cluster

0 Workers: 0.0 GB Memory, 0 Cores, 0 DBU
1 Driver: 6.0 GB Memory, 0.88 Cores, 1 DBU

Cluster Name

bda

Databricks Runtime Version

Runtime: 6.2 (Scala 2.11, Spark 2.4.4)

New This Runtime version supports only Python 3.

B. Ingresamos al Notebook y cargamos el archivo de Python proporcionado en clase

Notebook_ABD (Python)

The image shows the Databricks Notebook interface. At the top, there's a toolbar with icons for Detached, File, View: Code, Permissions, Run All, Clear, Publish, Comments, Runs, and Revision history. Below the toolbar, a message says 'Notebook detached cluster not found'. The main content area has the title 'Spark. Curso Análisis de BIG DATA.' and a paragraph: 'Este cuaderno se basa en tutoriales realizados por Databricks. Databricks is a leading provider of the commercial and enterprise supported version of Spark.' A link to 'Databricks' is provided at the bottom.



ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS
INGENIERÍA DE SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

C. Realizamos un análisis de las funciones e instrucciones del código.

Python

Comandos

1. `print()`

Función interna de python que muestra un mensaje en la pantalla

2. `format()`

Función interna de python que permite la sustitución múltiple de valores y su formato. Permite encadenar elementos dentro de un string por medio de índices de formato.

3. `re.search(EXPRESION, TEXTO)`

Pertenece a la librería de expresiones regulares de Python, realiza una búsqueda en el TEXTO que coincida con la EXPRESIÓN y devuelve su posición inicial.

4. `re.search(EXPRESION, TEXTO).group(0)`

Pertenece a la librería de expresiones regulares de Python, `group(i)` retorna la parte del TEXTO que coincide con la EXPRESIÓN.

5. `datetime.datetime.now()`

Función interna de python, la función interna de python que permite obtener la fecha y hora actuales.

6. `type(sc)`

Función interna de python que retorna el tipo de variable de "Spark context"

sc es la abreviación para "spark context", el cual representa al cliente que está en ejecución actualmente.

7. `dir(sc)`

Función interna de python que retorna una lista de atributos para la variable "Spark context".

8. `help(sc)`

Función interna de python que retorna información extra de ayuda desde la documentación otorgada para la variable "Spark context".

9. `sc.version`

Imprime la versión de Spark que se está ejecutando.



**ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS
INGENIERÍA DE SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN**

10. `range(inferior, superior)`

Función interna de python que retorna una lista con números desde “inferior”(inclusive) hasta “superior” (exclusive).

11. `sc.parallelize(data, 8)`

Paraleliza datos usando 8 particiones. Esta operación es una transformación de datos en un RDD. Spark utiliza una evaluación diferida, por lo que no se ejecutan trabajos de Spark en este momento

12. `sc.parallelize(data, 8).id()`

Cada RDD obtiene una identificación única que puede ser usada a conveniencia.

13. `sc.parallelize(data, 8).setName(“Nuevo nombre”)`

Permite nombrar los RDD con el nombre especificado en el parámetro.

14. `sc.parallelize(data, 8).toDebugString()`

Muestra el conjunto de transformaciones del RDD

15. `sc.parallelize(data, 8).getNumPartitions()`

Muestra el número de particiones en que se dividirá el RDD

16. `def sub(value):`

Es una manera de declarar una función en Python enviándole un parámetro llamado value.

17. `sc.parallelize(data, 8).map(sub)`

Se aplica una transformación usando un mapa con la subfunción “sub”

18. `sc.parallelize(data, 8).map(sub).collect() ;
sc.parallelize(data, 8).map(sub).filter(ten).collect();
*.collect()`

Permite mostrar los datos de forma común en Python, es decir, convierte el objeto de spark a una lista en Python.

19. `sc.parallelize(data, 8).map(sub).count();
sc.parallelize(data, 8).map(sub).collect().count();
*.count()`

Cuenta el número de elementos que posee la lista o el arreglo de Spark.



**ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS
INGENIERÍA DE SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN**

20. `sc.parallelize(data, 8).map(sub).filter(ten)`

Aplica un filtro en los datos, es decir, una transformación, por lo que no se ejecutan tareas. El filtro a aplicar se encuentra en la función “ten” y selecciona los datos que otorguen verdadero.

21. `lambda`

Es un método para declarar funciones en una línea dentro del lenguaje Python.

22. `sc.parallelize(data, 8).map(sub).filter(ten).first()`

Selecciona el primer elemento de los datos filtrados.

23. `sc.parallelize(data, 8).map(sub).filter(ten).take(4)`

Selecciona los primeros 4 elementos de los datos filtrados.

24. `sc.parallelize(data, 8).map(sub).filter(ten).takeOrdered(3)`

Toma los tres elementos más pequeños de los datos filtrados.

25. `sc.parallelize(data, 8).map(sub).filter(ten).top(5)`

Toma los tres elementos más largos del conjunto de datos filtrados.

26. `sc.parallelize(data, 8).map(sub).filter(ten).reduce(lambda)`

Realiza una reducción sobre el conjunto de datos que cumpla con la función especificada por parámetro.

27. `sc.parallelize(data, 8).map(sub).filter(ten).repartition(4).reduce(lambda)`

Toma una parte en el índice 4 del conjunto de datos

28. `sc.parallelize(data, 8).map(sub).filter(ten).takeSample(,)`

Toma una muestra de datos (ya sea reusando o no, eso se especifica por parámetro).

29. `sc.parallelize([1, 2, 3, 1, 2, 3, 1, 2, 1, 2, 3, 3, 3, 4, 5, 4, 6]).countByValue()`

Genera un diccionario clave valor con el conteo de la frecuencia de cada número único.

30. `sc.parallelize(wordsList, 4).flatMap(lambda)`

Similar a una función de mapeo pero cada elemento de entrada se puede asignar a 0 o más elementos de salida.

31. `sc.parallelize([('a', 1), ('a', 2), ('b', 1)]).groupByKey().mapValues(lambda)`

Se usa para mejorar el formato de impresión



**ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS
INGENIERÍA DE SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN**

32. `sc.parallelize(wordsList, 4).mapPartitions(lambda)`

`mapPartitions` toma una función que toma un iterador y devuelve un iterador

33. `join()`

Función interna de Python que permite intercambiar un separador por otro.

34. `list()`

Función interna de Python que permite transformar un tipo de dato a una lista

35. `sc.parallelize(data, 8).map(sub).filter(ten).cache()`

Pone al conjunto de datos en caché.

36. `sc.parallelize(data, 8).map(sub).filter(ten).unpersist()`

Si hemos terminado con el RDD, podemos eliminarlo para que su memoria pueda recuperarse

37. `sc.parallelize(data, 8).map(sub).filter(ten).getStorageLevel()`

Nivel de almacenamiento para un RDD no almacenado en caché

Análisis de resultados:

La estrategia MapReduce se logró hacer en pocas líneas de código respecto a Apache Spark en una máquina virtual de Cloudera que requería varios pasos adicionales.

La paralelización sigue siendo una tarea ultra importante al trabajar con datos sumamente grandes.

Conclusiones y recomendaciones:

- Se creó una cuenta en Databricks community edition y se logró familiarizarse con el entorno.
- Se creó un cluster llamado bda. Con recursos limitados debido a la versión gratuita utilizada.
- Se explicó las funciones y términos usados en el código proporcionado evitando la redundancia. Ejemplo: la función `print`, `count`, `collect` solo se explicó una vez aunque aparecieron varias veces en el código.
- Se puede realizar un procesamiento en paralelo de forma automática utilizando Databricks.
- La ejecución paralela mejora el rendimiento la mayoría de las veces.



ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS
INGENIERÍA DE SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

- Solo si existe buena experiencia es recomendable montar un propio clúster, de otra forma es mucho más fácil contratar un servicio web debido a la confiabilidad y disponibilidad.
- Preparar una estrategia de paralelización en orden de resolver un problema complejo de forma eficiente.

Bibliografía

[1]"Apache Spark | Cloudera", *Cloudera*. [Online]. Available: <https://www.cloudera.com/products/open-source/apache-hadoop/apache-spark.html>. [Accessed: 21- Jan- 2020].

[2]"Databricks - Unified Data Analytics", *Databricks*. [Online]. Available: <https://databricks.com/>. [Accessed: 21- Jan- 2020].