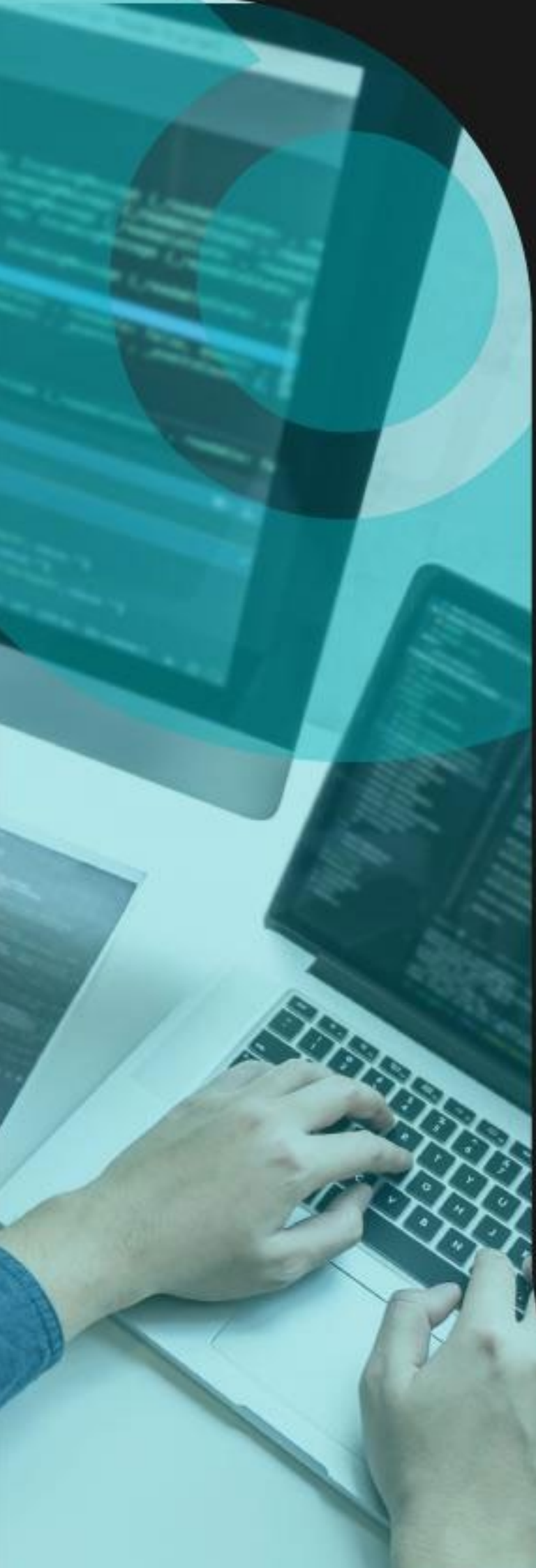


AWAKELAB

BASECAMP

Ciencia de Datos

Módulo: Fundamentos del Big Data

Aprendizaje Esperado

---

2. Manipular grandes volúmenes de datos utilizando objetos RDD, transformaciones y acciones para resolver un problema

---

Amazon AWS

Amazon Web Services (AWS) es una plataforma integral de computación en la nube proporcionada por Amazon. Las primeras ofertas de AWS se lanzaron en 2006 para proporcionar servicios en línea para sitios web y aplicaciones del lado del cliente. El servicio ya está disponible en 190 países de todo el mundo.

Para minimizar el impacto del tiempo de inactividad y garantizar la estabilidad del sistema, AWS se divide en regiones con diferentes centros de datos. Estas regiones tienen centros clave en América del Norte (región de EE. UU.: Este y Oeste), América del Sur (región de Sao Paulo, Brasil), Europa/Oriente Medio/África (región de la UE: Irlanda, Frankfurt) y Asia Pacífico (Asia Pacífico y Asia). región del Pacífico).China).

¿Qué es Apache Spark?

Apache Spark es un sistema de procesamiento distribuido de código abierto que se utiliza para grandes cargas de trabajo de datos. Utiliza el almacenamiento en caché en la memoria y la ejecución optimizada de consultas para consultas analíticas rápidas contra datos de cualquier tamaño. Proporciona API de desarrollo en Java, Scala, Python y R, y admite la reutilización de código en múltiples cargas de trabajo: procesamiento por lotes, consultas interactivas, análisis en tiempo real, aprendizaje automático y procesamiento de gráficos. Lo encontrará

utilizado por organizaciones de cualquier industria, incluso en FINRA, Yelp, Zillow, DataXu, Urban Institute y CrowdStrike. Apache Spark se ha convertido en uno de los marcos de procesamiento distribuido de big data más populares con 365 000 miembros de reuniones en 2017.

Apache Spark frente a Apache Hadoop

Aparte de las diferencias en el diseño de Spark y Hadoop MapReduce, muchas organizaciones han descubierto que estos marcos de big data son complementarios y los usan juntos para resolver un desafío comercial más amplio.

Hadoop es un marco de código abierto que tiene el sistema de archivos distribuidos de Hadoop (HDFS) como almacenamiento, YARN como una forma de administrar los recursos informáticos utilizados por diferentes aplicaciones y una implementación del modelo de programación MapReduce como motor de ejecución. En una implementación típica de Hadoop, también se implementan diferentes motores de ejecución, como Spark, Tez y Presto.

Spark es un marco de código abierto centrado en consultas interactivas, aprendizaje automático y cargas de trabajo en tiempo real. No tiene su propio sistema de almacenamiento, pero ejecuta análisis en otros sistemas de almacenamiento como HDFS u otras tiendas populares como Amazon Redshift, Amazon S3, Couchbase, Cassandra y otras. Spark on Hadoop aprovecha YARN para compartir un clúster y un conjunto de datos comunes como otros motores de Hadoop, lo que garantiza niveles de servicio y respuesta uniformes.

¿Cuáles son los beneficios de Apache Spark?

Hay muchos beneficios de Apache Spark para convertirlo en uno de los proyectos más activos en el ecosistema de Hadoop. Éstos incluyen:

- Rápido

A través del almacenamiento en caché en la memoria y la ejecución optimizada de consultas, Spark puede ejecutar consultas analíticas rápidas contra datos de cualquier tamaño.

- Apto para desarrolladores

Apache Spark es compatible de forma nativa con Java, Scala, R y Python, lo que le brinda una variedad de lenguajes para crear sus aplicaciones. Estas API facilitan las cosas a sus desarrolladores, ya que ocultan la complejidad del procesamiento distribuido detrás de operadores simples de alto nivel que reducen drásticamente la cantidad de código necesario.

- Múltiples cargas de trabajo

Apache Spark viene con la capacidad de ejecutar múltiples cargas de trabajo, incluidas consultas interactivas, análisis en tiempo real, aprendizaje automático y procesamiento de gráficos. Una aplicación puede combinar múltiples cargas de trabajo sin problemas.

Cargas de trabajo de Apache Spark

El marco Spark incluye:

- Spark Core como base de la plataforma
- Spark SQL para consultas interactivas
- Spark Streaming para análisis en tiempo real
- Spark MLlib para el aprendizaje automático
- Spark GraphX para el procesamiento de gráficos

Habilitación de Spark en AWS EMR

Para habilitar Spark en Amazon Web Services (AWS) Elastic MapReduce (EMR), siga los siguientes pasos:

1. Cree un clúster EMR: Para crear un clúster EMR, inicie sesión en la consola de AWS, seleccione EMR en el panel de servicios y haga clic en el botón "Crear clúster". En la página de creación de clúster,

seleccione la versión de EMR que desea usar, el tamaño del clúster, las opciones de configuración, etc. Asegúrese de seleccionar la opción "Spark" como motor de procesamiento.

2. Configure el acceso a S3: Si desea acceder a los datos de S3 desde su clúster EMR, deberá configurar el acceso. Puede hacer esto utilizando IAM (Identidad y Acceso de AWS) para crear un rol que tenga permisos para acceder a los datos de S3. Luego, asigne ese rol al clúster EMR cuando lo cree.

3. Inicie sesión en el clúster EMR: Una vez que haya creado el clúster EMR, podrá iniciar sesión en él utilizando SSH. Deberá usar las credenciales de SSH proporcionadas por AWS para conectarse al clúster.

4. Ejecute aplicaciones de Spark: Una vez que haya iniciado sesión en el clúster EMR, podrá ejecutar aplicaciones de Spark utilizando los comandos de línea de comandos de Spark o un IDE como Jupyter o Zeppelin. Si está utilizando un IDE, deberá configurar la conexión a Spark en la interfaz del IDE.

Con estos pasos, habrá habilitado Spark en AWS EMR y podrá comenzar a procesar grandes conjuntos de datos utilizando Spark en la nube de AWS.

Creación de una instancia EC2

Para crear una instancia EC2 en Amazon Web Services (AWS), siga estos pasos:

1. Inicie sesión en la consola de AWS: Vaya a la página de inicio de sesión de AWS y utilice sus credenciales para iniciar sesión en la consola de AWS.

2. Navegue hasta el servicio EC2: Una vez que haya iniciado sesión, busque y seleccione el servicio EC2 en el panel de servicios de AWS.

3. Haga clic en "Lanzar instancia": En la página de inicio de EC2, haga clic en el botón "Lanzar instancia" para comenzar el proceso de creación de la instancia EC2.

4. Seleccione una imagen AMI: La primera decisión que debe tomar es seleccionar una imagen AMI (Amazon Machine Image) para su instancia EC2. La AMI es una plantilla que contiene la información necesaria para lanzar una instancia. Seleccione la AMI que mejor se adapte a sus necesidades.

5. Seleccione un tipo de instancia: En la página siguiente, seleccione un tipo de instancia EC2. Hay varios tipos de instancias disponibles, cada uno con diferentes especificaciones de CPU, memoria y almacenamiento. Seleccione el tipo de instancia que mejor se adapte a sus necesidades.

6. Configure la instancia: En la siguiente página, configure los detalles de su instancia EC2, como la cantidad de instancias que desea lanzar, el tamaño del almacenamiento, la VPC (Virtual Private Cloud) a la que se asignará la instancia, la red de seguridad, etc.

7. Configure el almacenamiento: En la siguiente página, configure el almacenamiento de su instancia EC2. Puede elegir entre varios tipos de almacenamiento, como EBS (Elastic Block Store) o almacenamiento local de instancia.

8. Configure las etiquetas: En la página siguiente, configure las etiquetas para su instancia EC2. Las etiquetas son una forma de asignar metadatos a su instancia para facilitar su seguimiento y gestión.

9. Configure el grupo de seguridad: En la página siguiente, configure el grupo de seguridad para su instancia EC2. El grupo de seguridad es una forma de controlar el tráfico de red hacia y desde su instancia.

10. Revise y lance la instancia: En la página final, revise la configuración de su instancia EC2 y haga clic en el botón "Lanzar" para lanzar la instancia.

Una vez que haya lanzado su instancia EC2, podrá conectarse a ella y comenzar a utilizarla para ejecutar aplicaciones o alojar su sitio web o aplicación.

Configuración de Putty para acceder a la instancia

Para conectarse a una instancia de Amazon EC2 utilizando Putty, siga estos pasos:

1. Descargue e instale Putty: Si aún no lo ha hecho, descargue e instale Putty desde el sitio web oficial: https://www.putty.org/

2. Abra Putty y configure la conexión: En la ventana de inicio de Putty, configure la conexión proporcionando la dirección IP de la instancia de EC2 y el puerto SSH (22) en la sección "Host Name" y "Port". Asegúrese de seleccionar la opción "SSH" en el tipo de conexión.

3. Configure las opciones de sesión: En la sección "Session" de la ventana de configuración de Putty, ingrese un nombre para la sesión y haga clic en "Guardar" para guardar la configuración.

4. Configure la autenticación: En la sección "Connection" de la ventana de configuración de Putty, seleccione "SSH" y luego "Auth". En la sección "Authentication parameters", seleccione la opción "Browse" para encontrar su archivo .pem de Amazon EC2 en su computadora y luego haga clic en "Open" para cargar el archivo.

5. Conéctese a la instancia: Haga clic en el botón "Open" para iniciar la conexión con la instancia de EC2. Se abrirá una ventana de terminal de Putty, donde deberá ingresar el nombre de usuario "ec2-user" (para instancias basadas en Amazon Linux, CentOS, RHEL y Fedora) o "ubuntu" (para instancias basadas en Ubuntu) y presione "Enter".

6. Comience a usar la instancia: Una vez conectado, tendrá acceso a la línea de comando de la instancia de EC2 y podrá comenzar a ejecutar comandos y configurar su servidor.

Con estos pasos, habrá configurado Putty para conectarse a una instancia de Amazon EC2 y podrá comenzar a administrar su servidor.

Instalación y Configuración de Jupyter Notebook en EC2

Configure Jupyter Notebook: Para configurar Jupyter Notebook, primero cree un archivo de configuración utilizando el siguiente comando en el terminal:

```
jupyter notebook --generate-config
```

Edite el archivo de configuración de Jupyter Notebook: Edite el archivo de configuración de Jupyter Notebook utilizando el siguiente comando en el terminal:

```
nano ~/.jupyter/jupyter_notebook_config.py
```

Agregue las siguientes líneas de código al archivo de configuración:

```
c = get_config()


# Permitir conexiones desde cualquier dirección IP

c.NotebookApp.ip = '0.0.0.0'
```

```
# Utilice un puerto de su elección (por ejemplo,
8888)

c.NotebookApp.port = 8888

# Permitir conexiones remotas

c.NotebookApp.open_browser = False
```

Guarde y cierre el archivo de configuración. Luego, inicie Jupyter Notebook: Inicie Jupyter Notebook en la instancia EC2 utilizando el siguiente comando en el terminal: **`jupyter notebook`**

Acceda a Jupyter Notebook desde su navegador: Acceda a Jupyter Notebook desde su navegador utilizando la dirección IP de la instancia de EC2 y el puerto especificado en la configuración (por ejemplo, http://[dirección IP de EC2]:8888).

Con estos pasos, habrá instalado Anaconda en su instancia EC2 y configurado Jupyter Notebook para que se ejecute en la instancia y se pueda acceder a él desde su navegador web.

Creación y configuración de una instancia EMR

Para crear y configurar una instancia de Amazon EMR (Elastic MapReduce), siga los siguientes pasos:

1. Inicie sesión en la consola de AWS: Inicie sesión en la consola de AWS con sus credenciales de Amazon.
2. Seleccione el servicio EMR: Una vez que haya iniciado sesión, seleccione el servicio EMR de la lista de servicios de AWS.
3. Haga clic en el botón "Create cluster": En la página principal de EMR, haga clic en el botón "Create cluster" para comenzar a crear una instancia de EMR.
4. Especifique los detalles de la instancia: En la página "Quick options" o "Advanced options", especifique los detalles de la instancia que

desea crear, como el número de instancias, el tipo de instancia, la versión de Hadoop, etc. También puede seleccionar la configuración predefinida si no desea configurar manualmente su instancia EMR.

5. Configure las opciones adicionales: Configure cualquier opción adicional que desee para su instancia EMR, como la configuración de seguridad y las opciones de red.
6. Haga clic en el botón "Create cluster": Una vez que haya especificado todos los detalles y opciones de su instancia EMR, haga clic en el botón "Create cluster" para comenzar a crear la instancia.
7. Espere a que la instancia se cree: Dependiendo de la configuración y las opciones que haya seleccionado, puede tomar algunos minutos o más para que se cree la instancia de EMR.
8. Conéctese a la instancia EMR: Una vez que se haya creado la instancia de EMR, puede conectarse a ella utilizando SSH o la consola de AWS.
9. Configure la instancia EMR: Una vez conectado a la instancia EMR, configure su instancia según sea necesario, instalando paquetes adicionales y software que desee usar, y ejecutando tareas de procesamiento de datos y análisis.

Elementos Básicos de Spark

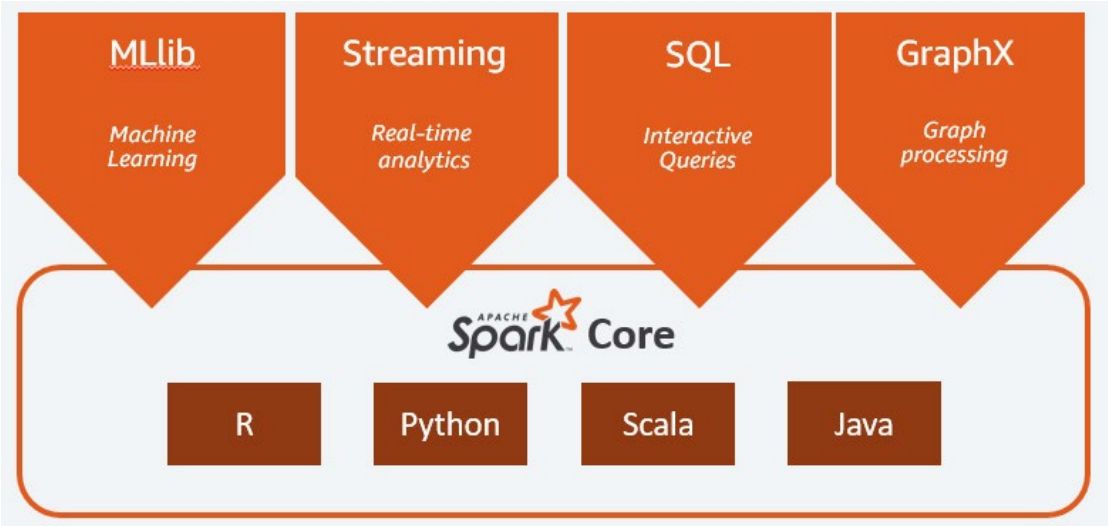
Spark es un framework de computación distribuida que permite procesar grandes volúmenes de datos en paralelo y de manera eficiente. Algunos de los elementos básicos de Spark son:

Configuración, Conexión y Contexto de Spark

Configuración de Spark: La configuración de Spark permite establecer propiedades y configuraciones específicas para la instancia de Spark que se va a utilizar. Estas configuraciones incluyen la cantidad de memoria y núcleos de CPU que se asignan a la instancia, la ubicación de los archivos de registro, entre otros.

Conexión a Spark: Para conectarse a Spark, se utiliza una API de programación de Spark que permite interactuar con el clúster de Spark y ejecutar tareas en paralelo. La API de programación de Spark se encuentra disponible en varios lenguajes de programación, como Java, Python y Scala.

Contexto de Spark: El contexto de Spark es un objeto que proporciona una conexión a un clúster de Spark y se utiliza para crear RDDs (Resilient Distributed Datasets), que son la unidad básica de procesamiento en Spark. El contexto de Spark también se encarga de gestionar la memoria y los recursos del clúster, y permite la ejecución de tareas en paralelo.



Resilient Distributed Dataset (RDD)

RDD fue la principal API orientada al usuario en Spark desde su creación. En esencia, un RDD es una colección distribuida inmutable de elementos de sus datos, divididos en nodos en su clúster que se pueden operar en paralelo con una API de bajo nivel que ofrece transformaciones y acciones.

5 razones sobre cuándo usar RDD

1. Desea transformación y acciones de bajo nivel y control sobre su conjunto de datos;
2. Sus datos no están estructurados, como flujos de medios o flujos de texto;
3. Desea manipular sus datos con construcciones de programación funcional que con expresiones específicas de dominio;
4. No le importa imponer un esquema, como el formato de columnas mientras procesa o accede a atributos de datos por nombre o columna; y
5. Puede renunciar a algunos beneficios de optimización y rendimiento disponibles con DataFrames y Datasets para datos estructurados y semiestructurados.

Creación y carga de un RDD, Almacenamiento de un RDD y Pair-RDD

La creación y carga de un RDD (Resilient Distributed Dataset) en Spark se puede realizar de diferentes maneras, incluyendo la carga de datos desde un archivo, la generación programática de datos y la transformación de otro RDD existente. A continuación, se muestran algunos ejemplos de cómo crear y cargar RDDs en Spark:

Carga de datos desde un archivo con Python:

```
from pyspark import SparkContext

sc = SparkContext("local", "myAppName")

rdd = sc.textFile("ruta/al/archivo")
```

Generación programática de datos:

```
from pyspark import SparkContext

sc = SparkContext("local", "myAppName")
```

```
data = ["Hola", "Mundo", "en", "Spark"]

rdd = sc.parallelize(data)
```

Transformación de otro RDD:

```
from pyspark import SparkContext

sc = SparkContext("local", "myAppName")

data = ["Hola", "Mundo", "en", "Spark"]

rdd1 = sc.parallelize(data)

rdd2 = rdd1.map(lambda x: x.upper())
```

Una vez que se ha creado un RDD, se puede almacenar en memoria o en disco utilizando las funciones de persistencia de Spark. Las funciones de persistencia incluyen cache(), que almacena el RDD en memoria, y persist(), que permite almacenar el RDD en memoria o en disco.

```
rdd.persist()
```

Los RDDs Pair son una variante de los RDDs en Spark que contienen datos almacenados en pares de clave-valor. Los RDDs Pair se utilizan comúnmente en el procesamiento de datos estructurados y en la realización de operaciones de agregación. Algunas de las operaciones que se pueden realizar en los RDDs Pair incluyen groupByKey(), reduceByKey() y sortByKey().

```
from pyspark import SparkContext
```

```
sc = SparkContext("local", "myAppName")

data = [("apple", 3), ("banana", 2), ("pear", 1),
("apple", 1)]

rdd = sc.parallelize(data)

rddPair = rdd.reduceByKey(lambda x, y: x + y)
```

En este ejemplo, se crea un RDD Pair a partir de una lista de tuplas que contienen el nombre de una fruta y su cantidad. Luego, se utiliza la función reduceByKey() para agregar la cantidad de cada fruta y se genera un nuevo RDD Pair que contiene el nombre de la fruta y su cantidad total.

Transformaciones: Linaje y Lazy evaluation

Qué es una transformación

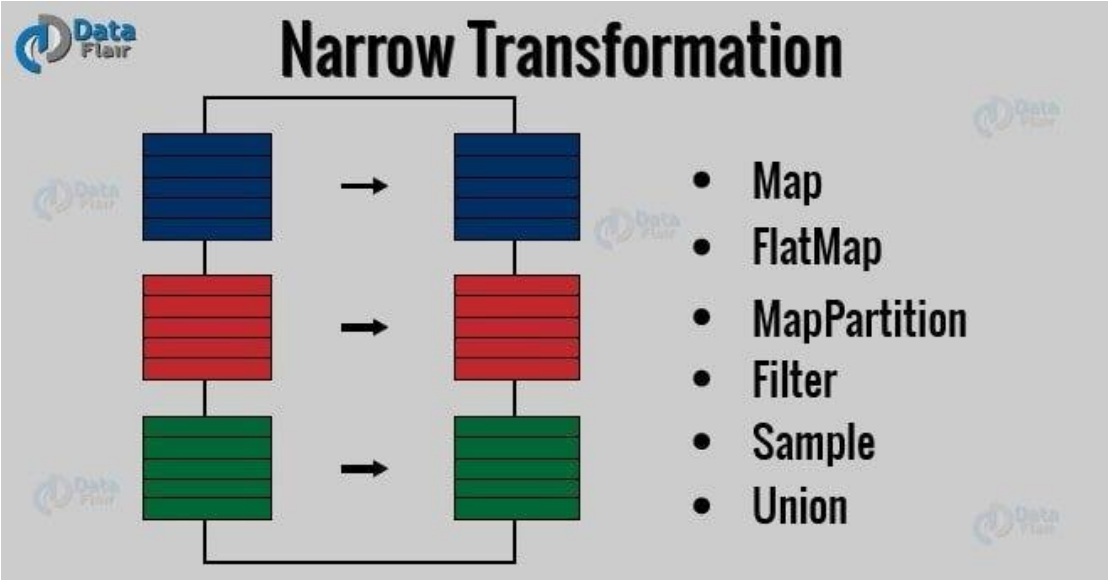
En Spark, las transformaciones son operaciones que se aplican a un RDD (Resilient Distributed Dataset) para generar un nuevo RDD. Las transformaciones no modifican el RDD original, sino que generan un nuevo RDD a partir de él.

Spark utiliza una estrategia de evaluación perezosa (Lazy evaluation) para optimizar la ejecución de las transformaciones. Esto significa que las transformaciones no se ejecutan inmediatamente después de ser llamadas, sino que se almacenan en el grafo de transformaciones (también conocido como DAG) para su ejecución posterior.

El grafo de transformaciones es un registro de todas las transformaciones que se han aplicado a un RDD, desde su creación hasta su transformación final. El grafo de transformaciones permite a Spark optimizar la ejecución de las operaciones, ya que puede reordenar y agrupar las transformaciones para minimizar la cantidad de datos que se transfieren a través de Stop generating

Principales Métodos

Transformación estrecha: en la *transformación estrecha* , todos los elementos necesarios para calcular los registros en una partición única viven en la partición única del RDD principal. Se utiliza un subconjunto limitado de partición para calcular el resultado. *Las transformaciones estrechas* son el resultado de *map(), filter().*



- **Map:** La función de mapa itera sobre cada línea en RDD y se divide en un nuevo RDD. Usando la transformación **map()** tomamos cualquier función, y esa función se aplica a cada elemento de RDD. En el mapa, tenemos la flexibilidad de que la entrada y el tipo de retorno de RDD pueden diferir entre sí. Por ejemplo, podemos tener el tipo RDD de entrada como String, después de aplicar el función map (), el RDD de retorno puede ser booleano.

- **FlatMap:** Con la ayuda de la función **flatMap()** , para cada elemento de entrada, tenemos muchos elementos en un RDD de salida. El uso más simple de flatMap() es dividir cada cadena de entrada en palabras. Map y flatMap son similares en la forma en que toman una línea de la entrada RDD y aplican una función en esa línea. La diferencia clave entre map() y flatMap() es que map() devuelve solo un elemento, mientras que flatMap() puede devolver una lista de elementos.

- **MapPartition:** MapPartition convierte cada *partición del* **RDD** de origen en muchos elementos del resultado (posiblemente ninguno). En mapPartition(), la función map() se aplica en cada partición simultáneamente. MapPartition es como un mapa, pero la diferencia es que se ejecuta por separado en cada partición (bloque) del RDD.

- **Union:** Con la función **union()** , obtenemos los elementos de ambos RDD en un nuevo RDD. La regla clave de esta función es que los dos RDD deben ser del mismo tipo. Por ejemplo, los elementos de **RDD1** son (Spark, Spark, Hadoop, Flink) y los de **RDD2** son (Big data, Spark, Flink), por lo que el ***rdd1.union(rdd2) resultante*** tendrá elementos (Spark, Spark, Spark, Hadoop , Flink, Flink, Big data).

- **Filter:** La función Spark RDD **filter()** devuelve un nuevo RDD, que contiene solo los elementos que cumplen con un predicado. Es una *operación estrecha* porque no mezcla datos de una partición a muchas particiones. Por ejemplo, supongamos que RDD contiene los primeros cinco números naturales (1, 2, 3, 4 y 5) y el predicado busca un número par. El RDD resultante después del filtro contendrá solo los números pares, es decir, 2 y 4.

¿Qué son las acciones?

Las acciones de RDD son operaciones de PySpark que devuelven los valores al programa controlador. Cualquier función en RDD que devuelva algo que no sea RDD se considera una acción en la programación de PySpark.

Las funciones de acción activan las transformaciones para ejecutar. Como se mencionó en Transformaciones RDD , todas las transformaciones son evaluaciones perezosas, lo que significa que no se ejecutan de inmediato y la acción las activa para que se ejecuten.

- **Collect**
  `collect()` - Devuelve el conjunto de datos completo como una matriz.

- **Take, takeOrdered, takeSample**
  `take()` – Devuelve el primer número de elementos del conjunto de datos.
  `takeOrdered()`– Devuelve el primer número (el más pequeño) de elementos del conjunto de datos y esto es lo opuesto a la acción take().

  Nota: use este método solo cuando la matriz resultante sea pequeña, ya que todos los datos se cargan en la memoria del controlador.

  `takeSample()`– Devolver el subconjunto del conjunto de datos en un Array.

  Nota: use este método solo cuando la matriz resultante sea pequeña, ya que todos los datos se cargan en la memoria del controlador.

- **Top**
  `top()`– Devuelve los n elementos principales del conjunto de datos.

  Nota: use este método solo cuando la matriz resultante sea pequeña, ya que todos los datos se cargan en la memoria del controlador.

Job Spark

En Spark, un job es un conjunto de transformaciones y acciones que se aplican a uno o varios RDDs para obtener un resultado final. Un job se ejecuta en un clúster de Spark y se divide en tareas (tasks), que se ejecutan en diferentes nodos del clúster.

La ejecución de un job en Spark se inicia cuando se llama a una acción (action) sobre un RDD. Cuando se llama a una acción, Spark genera el grafo de transformaciones (DAG) correspondiente a las transformaciones aplicadas al RDD y lo divide en tareas que se distribuyen en el clúster. Cada tarea se ejecuta en un nodo del clúster y procesa una parte de los datos. Cuando todas las tareas se han completado, el resultado final se devuelve al cliente que ha iniciado la acción.

Es importante tener en cuenta que la ejecución de un job en Spark es perezosa, lo que significa que las transformaciones no se ejecutan inmediatamente al ser llamadas, sino que se agregan al grafo de transformaciones. La evaluación de las transformaciones se produce cuando se llama a una acción sobre un RDD, lo que activa la ejecución del grafo de transformaciones correspondiente.

En resumen, un job en Spark es un conjunto de transformaciones y acciones que se aplican a uno o varios RDDs para obtener un resultado final. El job se ejecuta en un clúster de Spark y se divide en tareas que se ejecutan en diferentes nodos del clúster. La ejecución de un job se inicia cuando se llama a una acción sobre un RDD y se genera el grafo de transformaciones correspondiente.

Referencias

[1] AWS Amazon

https://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-spark-configure.html

[2] La nube de Amazon

https://sostenibilidad.aboutamazon.es/medio-ambiente/la-nube?energyType=true

[3] Configuración Spark

https://docs.aws.amazon.com/es_es/emr/latest/ReleaseGuide/emr-spark-configure.html

[4] RDD – Resilient Distributed Databas (Con video)

https://www.linkedin.com/learning/python-para-data-science-y-big-data-esencial/que-son-los-rdd-resilient-distributed-databases?autoplay=true

[5] Evaluación perezosa (Con video)

https://thatcsharpguy.github.io/tv/lazy-evaluation/

[6] Job Spark

https://spark.apache.org/docs/latest/job-scheduling.html

Complementario

[1] Servicios Amazon AWS

https://www.youtube.com/watch?v=3_ZNHe4PZsY&t=1s

[2] Qué son los RDD

https://youtu.be/6rTiRp982wE


AWAKELAB