Apache Spark

Autor:

Kevin Cárdenas.

2023

Índice general

Introducción

Apache Spark es en terminos generales un motor unificado, y un conjunto de bibliotecas para el procesamiento de datosparalelo en un *cluster* de computadores. Spark soporta multiples lenguajes de programcación, entre ellos Python, Java, R, etc.

La filosofía de Apache Spark es la de ser un motor de procesamiento de datos unificado, es decir, que sea capaz de realizar procesamiento de datos en tiempo real, procesamiento de datos por lotes, procesamiento de datos de tipo *streaming*, procesamiento de datos de tipo *machine learning*, procesamiento de datos de tipo *graph processing*, etc.

Definición 1 (Cluster de computadores)**.** Un *cluster* de computadores es un conjunto de computadores que trabajan en conjunto para realizar una tarea en común.

¿A qué nos referimos con unificado? A que Spark es capaz de realizar todas las tareas mencionadas anteriormente, y muchas más, en un solo motor de procesamiento de datos.

La naturaleza unificada de Spark hace estas tareas mucho más sencillas de realizar, ya que no es necesario utilizar diferentes motores de procesamiento de datos para realizar diferentes tareas, como por ejemplo, utilizar un motor de procesamiento de datos para realizar procesamiento de datos en tiempo real, y otro motor de procesamiento de datos para realizar procesamiento de datos por lotes. Spark provee APIs consistentes que se pueden utilizar para construir aplicaciones de mosulos más pequeños o de biblitecas ya existentes.

El problema de Big Data

El problema de *Big Data* es un problema que surge cuando se tiene una gran cantidad de datos que no pueden ser procesados por un solo computador. En estos casos, es necesario utilizar un *cluster* de computadores para procesar estos datos.

Descarga e instalación de Spark en Local (Linux)

Supondremos que tiens Docker, y un conocimiento básico de Docker. Si no es así, te recomiendo que leas la documentación oficial de Docker.

Ejecuta el para descargar e iniciar un contenedor de Docker con Spark

```
docker run --name spark -p 8080:8080 -p 4040:4040 -d bitnami/spark:
latest
```

Puedes verificar que el contenedor de Docker se ha iniciado correctamente con el siguiente comando

```
docker ps
```

puedes ver que se ha iniciado el contenedor de Docker de Spark correctamente con el siguiente comando

```
docker ps
```

y ver los logs del contenedor de Docker de Spark con el siguiente comando

```
docker logs spark
```

Ahora, puedes acceder a la interfaz web de Spark en la siguiente dirección: http://localhost:8080

Uso de Spark con python

A partir de la verisón 2.2 de Spark, Spark soporta Python 3.5 o superior. Para utilizar Spark con Python, es necesario instalar la biblioteca de Spark para Python, llamada *PySpark*.

Para instalar PySpark, es necesario instalar la biblioteca de Spark para Python, llamada *PySpark*. Ejecuta el siguiente comando para instalar PySpark

```
pip install pyspark
```

Arquitectura de Spark

Como definimos antes, un *cluster* de computadores es un conjunto de computadores que trabajan en conjunto para realizar una tarea en común. Pero, un grupo de maquinas no es suficiente para realizar una tarea en común, es necesario que estas maquinas estén conectadas entre sí, y que puedan comunicarse entre sí. Además, es necesario que estas maquinas estén conectadas a un sistema de almacenamiento de datos, para que puedan leer y escribir datos. Spark hace justo esto, administrar un *cluster* de computadores, y administrar la comunicación entre estos computadores, y la lectura y escritura de datos.

El cluster que Spark usa para realizar una tarea es administrado por un *cluster manager*, que es un sistema que administra el *cluster* de computadores. Spark soporta varios *cluster managers*, como por ejemplo, YARN, o Mesos.

Aplicaciones de Spark

Una aplicación de Sparl consiste en un *driver process*, y un conjunto de *executor processes*. El *driver process* es el proceso principal de la aplicación (la función `main()`) y los *executor processes* son los procesos que realizan el trabajo "real" de la aplicación. Los *executor processes* se ejecutan en los nodos del *cluster* de computadores. Vea la figura 1.

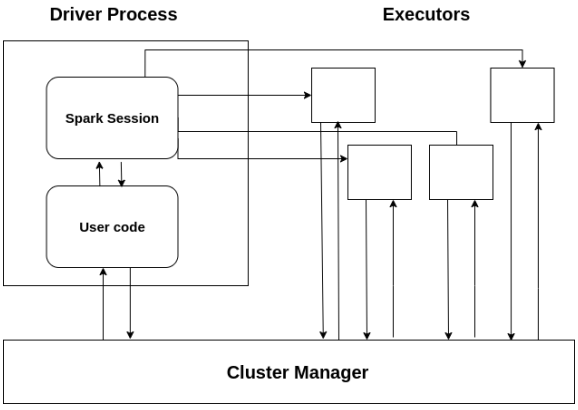

Figura 1: Aplicación de Spark

En la figura 1, podemos la relación entre el *driver process* y los *executor processes*. En local, el *driver process* y los *executor processes* se ejecutan en el mismo computador como un *hilo de ejecución* de la aplicación.

Algunos puntos clave para entender Spark son:

- Spark emplea un *cluster manager* que administra los recursos disponibles.
- El *driveer process* es responsable de la ejecución de la aplicación hasta que se completa la tarea.

Los *executor processes*, en su mayor parte corren codigo Spark. Sin empargo, el driver puede "conducir" diferentes lenguajes.

Api de lenguaje de Spark

La Api de lenguaje de Spark hace posible que los usuarios de Spark escriban aplicaciones en diferentes lenguajes de programación, como por ejemplo, Python, Java, Scala, R, etc. EN general, Spark presenta algnos conceptos centrales en todo lenguaje, estos conceptos son traducidos internamentre a un lenguaje de programación específico que corra en el cluster. Con las apis estructuradas, se puede esperare un performance similar de todos los lenguajes.