Hola! Antes de empezar de lleno con el curso, debemos de ver como desarrollaremos el curso , es decir con que herramienta trabajaremos así del porqué.

Además, quiero introducir el primer compente que usaremos en todas las clases y que au que a veces no lo comente debes de tenerlo de en mente al momento de trabajar. ¿Sí?. Eso esperp

Antes de empezar el curso debemos de tener en mente que existen, principalmente, dos formas de ejecutar un proceso de Spark.

Uno, el clasico, es construir todo nuestro código en un editor de texto como Visual Studio, Sublime, VIM,etc y ejecutarlo. Si bien esta es la forma tradicional, esto se realiza cuando tenemos un código que sabemos que posee los estándares de calidad mínimos para ponerlo en un servidor productivo por ejemplo.

Sin embargo al momento de desarrollar es conveniente poder ver los estados por los que pasa nuestro proceso y más si estamos aprendiendo este faboluso framework. Es por lo que herramientas como Zeppelin o Jupyter nos auxilian.

La forma en que nos auxilian es mediante la ejecución por bloques del código que tenemos, pudiendo así ver errores de ejecución en vivo. En Spark se hace complicado el debugear una aplicacion ya que el formato lazy que posee no nos ayuda mucho. Esto lo platicaremos en clases siguientes, no te preocupes.

Por otro lado, el desarrollar con ayuda de notebook permite a colaboradores visualizar lo que estamos tratando de realizar y en un ambiente profesional, podemos el equipo encargado de desplegar nuestra applicación poder hacer cambios según muestren los resultados de monitoreo entre etapas.

Para empezar un programa con Spark debemos crear un objeto maestro que nos permitirá interactuar con toda el API, para esto, como podrán ver en códigos de ejemplo a lo largo de la red tenemos popularmente dos:Spark Context y Spark Session. Claro que existen mas como HiveContext o SQLContext sin embargo vamos a enforarnos en los primeros dos para poder comprender los demás

[En jupyter importamos SparkContext y SparkSession]

Como puedes ver, aun que ambos contextos vienen del modulo principal, pypsark, el segundo deriba del submodulo sql. Esto es debido a que SparkContext en las versioens previas a Spark 2.0 es el punto de etnrada para cualquier aplicación y junto con sparkConf se configuran ciertos parámetros adicionales.

En un primer estado, podemos cerar RDDs, que como hemos visto, son la estructura base, pero para otros componentes debemos de crear contextos específicos como SQLContext para Sql, HiveContext para Hive, etc.

En diferencia SparkSession, es una convinación de todos los contextos anteriores. De manera interna, SparkSesison crea un SparkContext con todas las operacones y contextos comentados.

Para crear una session debemos de hacer lo siguiente:

[En un segundo bloque construimos una session]

spark = (SparkSession

.builder

.appName(“cursoSparkPlatzi”)

.getOrCreate())

Con esto daremos de alta una nueva sesión en spark. El ejemplo que acabo de darte es lo mínimo que debes de agregar al momento de crear una sesión.

Parametros adicionales, ejemplo, es seleccionar cuantos nucleos de procesamineto se usarán, configuraciones de sparkConf o incluso si soportará Hive. Esto es por dar un ejemplo de lo poderoso que es un SparkSession.

Debido a que SparkSession es un componente nuevo y este framework se preocupa por poder trabajar con versiones anteriores. Podemos desde un SparkConext generar un SarpkSession

Antes, debemos de terminar la session actual con el método stop(). Esto es un símil a cuando en C reservamos memoria para un proceso dinámico. Debemos de liberar, en nuestro caso terminar, el proceso que estamos ejecutando.

[En un tercer bloque contruimos na session heredada]

spark.stop()

spark = SparkContext(master="local", appName="Proyecto")

spark2 = SparkSession(spark)

Como ya te comenté, la importancia de SparkSession es la de unificar los difentes contexto que existen. Pero además, la sesion permite tener multiples usuarios usando el mismo contexto.

En versiones previas a Spark 2.0, la solución era tener un ambiente aislado por operacion. Lo cual aumenta el costo, al neceistas una maquina virtual de java por ejecución.

sparkSession2 = spark2.newSession()

spark

spark2

Veamos que amboas son sessiones en memoria y particuarlmente son contexto como ya hemos platicado. Apate de mostrarnos la version de spark que usamos, el nodo donde vive y el nombre del proyecto nos muestra una liga llamada SparkUI, vamos a darle click

Vemos que nos abre una pestaña con ubicación el el puerto 4040 y podemos ver los trabajos que acualmetne estan en ejecución. SparkUI es un sistema de monitoreo que nos permite ver el estado de los jobs que actualemte estan corriendo en nuestro servidor.

Si vamos a las otras pestañas, no veremos mucho, ya que como tal no tenemos operaciones en ejecución. Pero en los siguientes modulos estaremos usando este monitor y veremos particularidades que llegamos a tener cuando va cambiando el estado de las etapas de nuestra aplicación.

Es imporante mencionar que solo mientras tengamos una sessión o contexto en ejecución este monitor estará disponible

[lanza un stop y recarga la pagina]

Vamos a la pestaña de ejecutres y vemos que tenemos un ejecutor vivo en este momento. Ese es el proceso que tenemos lanzado de momento.

Finalmente, y como agregado, vamos a la pestaña de enviroment. Aquí podemos ver una descripción de la versión de spark y de java que estamos usando.

Así como configuracioens adicionales que aun que nosotros no hicimos, son elementales para que Spark pueda funcionar.

Regresando al tema de ejecutar una aplicacion.

Para poder ejecutarla por terminal deberemos usar spark-submit

[muestra contenido de un programa, y lo ejecuta]

Como vemos, nos lanza a la terminal el estado actual en el que se encuentra, lo sé, no parece, pero esto que vemos es una bitácora de ejecución y el buen estado del prorgama

[ubica la salida]

Aqui vemos la salida del codigo que ejecutamos, que es solamente un contador de palabras. Es por esto que no usaremos la terminal a lo largo del curso, ya que aparte de no poder ver paso a paso lo que haremos, la salida se vuelve un poco complicada. Claro en un ambiente productivo todo debería correct con spark-submit.

No olvides terminar la sesión que tienes si has seguido paso a paso esta clase y nos vemos en la siguiente para crear nuestro primer RDD