SparkContext y esqueleto de una aplicación con pyspark

Spark Context es el objeto principal del API de Spark.

SparkContext se trata del contexto básico de Spark, desde donde se crean el resto de variables que maneja el framework. Sólo un **SparkContext** puede estar activo por JVM.
En la shell interactiva de Spark es creada automáticamente bajo el nombre **sc**, mientras que en otros entornos es necesario instanciarlo explícitamente. La configuración de **SparkContext** puede ser definida mediante un bundle específico llamado SparkConf.

Sobre este objeto realizaremos todas las operaciones. Se puede utilizar para crear RDD, acumuladores y variables. Es el principal objeto que disponemos en pyspark que contiene toda la información sobre cómo acceder al cluster y realizar las diferentes operaciones sobre el mismo.

Para utilizar las funcionalidades pySpark necesitamos instanciar un objeto especial llamado SparkContext. Éste objeto le dice a Spark cómo acceder al clúster y contiene algunos parámetros específicos de la aplicación. En el IPython Notebook suministrado en la máquina virtual, esta variable ya está disponible y se llama sc (es la opción por defecto cuando se inicia un IPython Notebook)

En el caso del intérprete de pyspark,el objeto se crea automáticamente al inicio y puede ser accedido mediante la **variable sc.**

sc._conf.getAll()

Out:[(u'spark.rdd.compress', u'True'),
(u'spark.master', u'yarn-client'),
(u'spark.serializer.objectStreamReset', u'100'),
(u'spark.yarn.isPython', u'true'),
(u'spark.submit.deployMode', u'client'),
(u'spark.executor.cores', u'2'),
(u'spark.app.name', u'PySparkShell')]

Objeto que “sabe cómo interactuar con el cluster” y permite crear RDDs a partir de ficheros o colecciones de python.

Spark se centra en torno a los RDDS, una colección de datos resistente, distribuido se trata de una colección de alta disponibilidad de elementos que puedan ser explotados en forma paralela. Los datos son inmutables, tolerante a fallos.

Por ejemplo si queremos **leer un fichero** que tenemos en nuestra máquina:

```
fichero= sc.textFile(“ruta_fichero”)

#devuelve el número de líneas del fichero
fichero.count()

#obtiene la primera línea del fichero
fichero.first()
```

Para inicializar spark en python se usan las clases **SparkConf** y **SparkContext** pertenecientes al paquete pyspark.

```
from pyspark import SparkConf,SparkContext
conf = SparkConf().setMaster("local").setAppName("My app")
sc = SparkContext(conf=conf)
```

Otra forma de inicializar es directamente con la clase SparkContext sin necesidad de usar la clase SparkConf

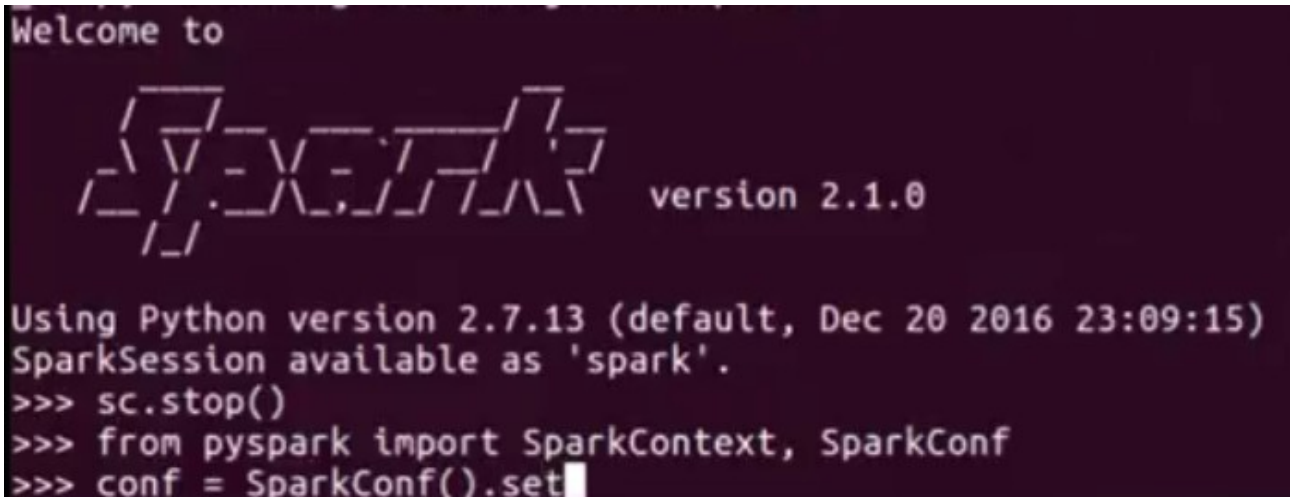
```
sc = SparkContext("local[*]","My app")
```

Si creamos un script desde cero hay que iniciarlo de esta forma.
Si usamos el intérprete de pyspark esta inicialización la hace por defecto.

La cadena local[*] indica que se ejecute con tantos hilos como procesadores en la máquina local

Shell de Python con Spark

Para usar el shell de Python con Spark, simplemente ejecute el comando **./bin/pyspark.** Al igual que el shell Scala, el objeto SparkContext de Python debe estar disponible como la variable sc de Python. Debería ver una salida similar a la mostrada en esta captura de pantalla:



Esqueleto de un programa spark

Un programa Spark ejecuta los siguientes pasos:

1. **Cargar datos de distintas fuentes**
2. **Aplicar transformaciones sobre las RDD creadas**
3. **Cachear las RDD necesarias**
4. **Ejecutar acciones sobre los RDD para persistir los datos**