SPARK USANDO PYTHON

Juan M. Tirado

November 17, 2017

SPARK

QUÉ ES SPARK

- Framework para el desarrollo de aplicaciones Big Data
- Desarrollado por el AMPLab de UC Berkeley en 2009
- Disponible en código abierto bajo licencia Apache
- `http://spark.apache.org/`

QUÉ ES SPARK

Spark ofrece varias ventajas para el procesado de grandes volúmenes de datos

- Método de acceso unificado a los datos (RDD)
- Uso intensivo de memoria en lugar de disco
- Soporte para varios lenguajes (Java, Scala, R, Python, Clojure)
- Soporte para Map/Reduce
- Soporte para SQL queries

QUÉ ES SPARK

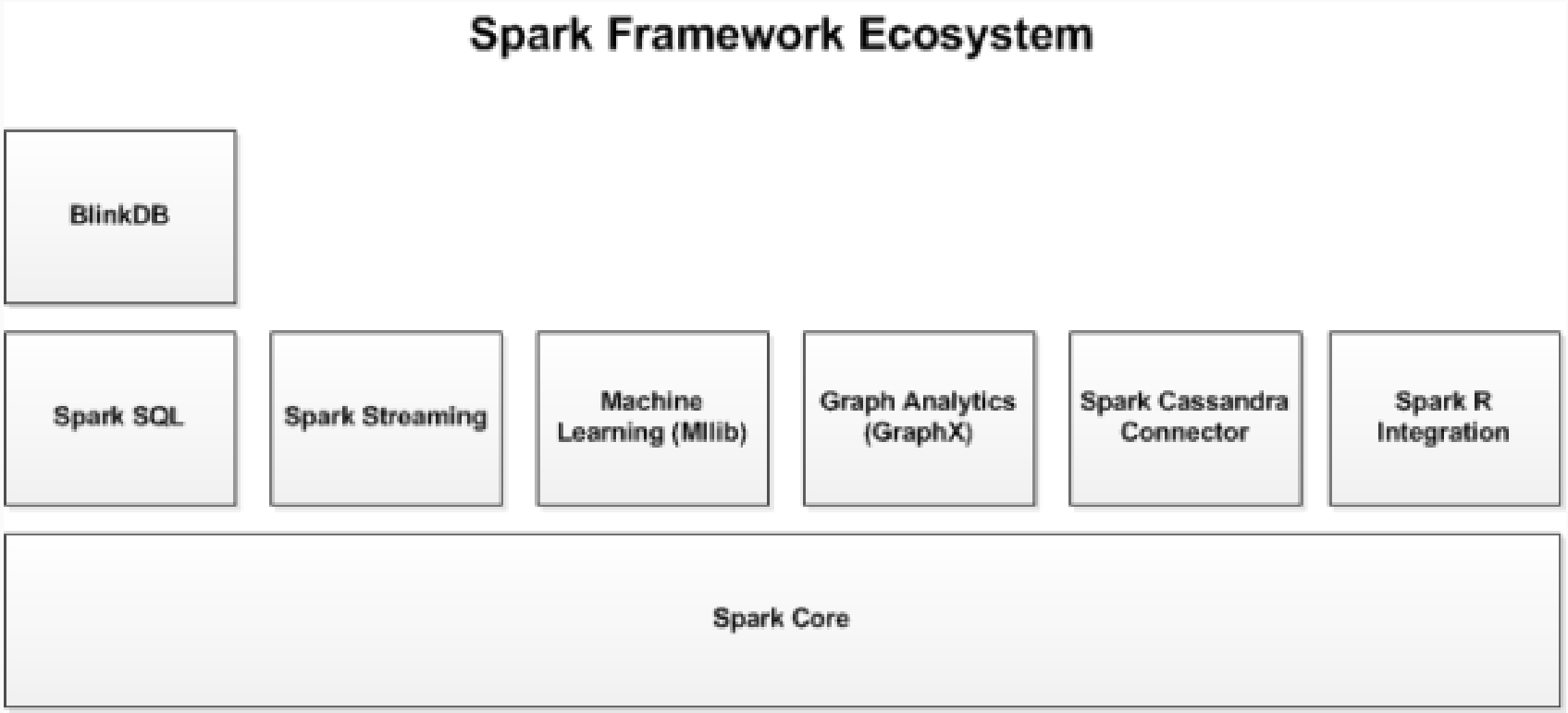
Spark ofrece un completo ecosistema de desarrollo

- Spark Streaming
- Spark SQL
- Spark MLib
- Spark GraphX

También es fácilmente integrable con otras plataformas

- Cassandra
- BlinkDB
- Tachyon

QUÉ ES SPARK


Spark Framework Ecosystem
BlinkDB
Spark SQL
Spark Streaming
Machine Learning (Mllib)
Graph Analytics (GraphX)
Spark Cassandra Connector
Spark R Integration
Spark Core

RDD

RDD = Resilient Distributed Datasets

- Estructura de datos básica en Spark
- Distribuida
- Inmutable
- Puede ser reconstruida (resistencia a fallos)
- Permite realizar operaciones de manera distribuída
- Abstrae al desarrollador de la gestión y permite centrarse en el procesado

RDD

Un RDD puede considerarse una tabla donde cada columna puede ser un tipo de datos diferente: int, float, string, etc.

	Columna 0	Columna 1	...	Columna n-1
Fila 0	"Texto0"	0	...	{objeto:[1,2]}
Fila 1	"Texto1"	1	...	{objeto:[1,2,3]}
⋮		⋮		
Fila n-1	"Texto n-1"	100	...	{objeto:[1,2,100]}

Esto permite que podamos definir un programa como un conjunto de operaciones sobre una o varias tablas.

RDD

Las RDD soportan dos tipos de operaciones

- **Transformación** → operaciones que devuelven un nuevo RDD
  - map
  - filter
  - groupByKey
  - reduceByKey
  - ...
- **Acción** → operaciones que devuelven un nuevo valor
  - reduce
  - collect
  - count
  - ...

RDD

Check the API for more info:

- `spark.apache.org/docs/latest/programming-guide.html#transformations`
- `spark.apache.org/docs/latest/programming-guide.html#actions`

PRIMER PROGRAMA EN PYSPARK

Spark está desarrollado en Scala pero permite acceder a sus funcionalidades a través de PySpark
Para utilizar PySpark necesitamos:

- Python 2.6 o posterior
- Scala
- Instalar Spark

PRIMER PROGRAMA EN PYSPARK

Para instalar Spark

- Descargamos Spark
  `http://spark.apache.org/downloads.html`
- Descomprimir

```
tar -xvzf spark-2.1.0-bin-hadoop2.7.tgz
```

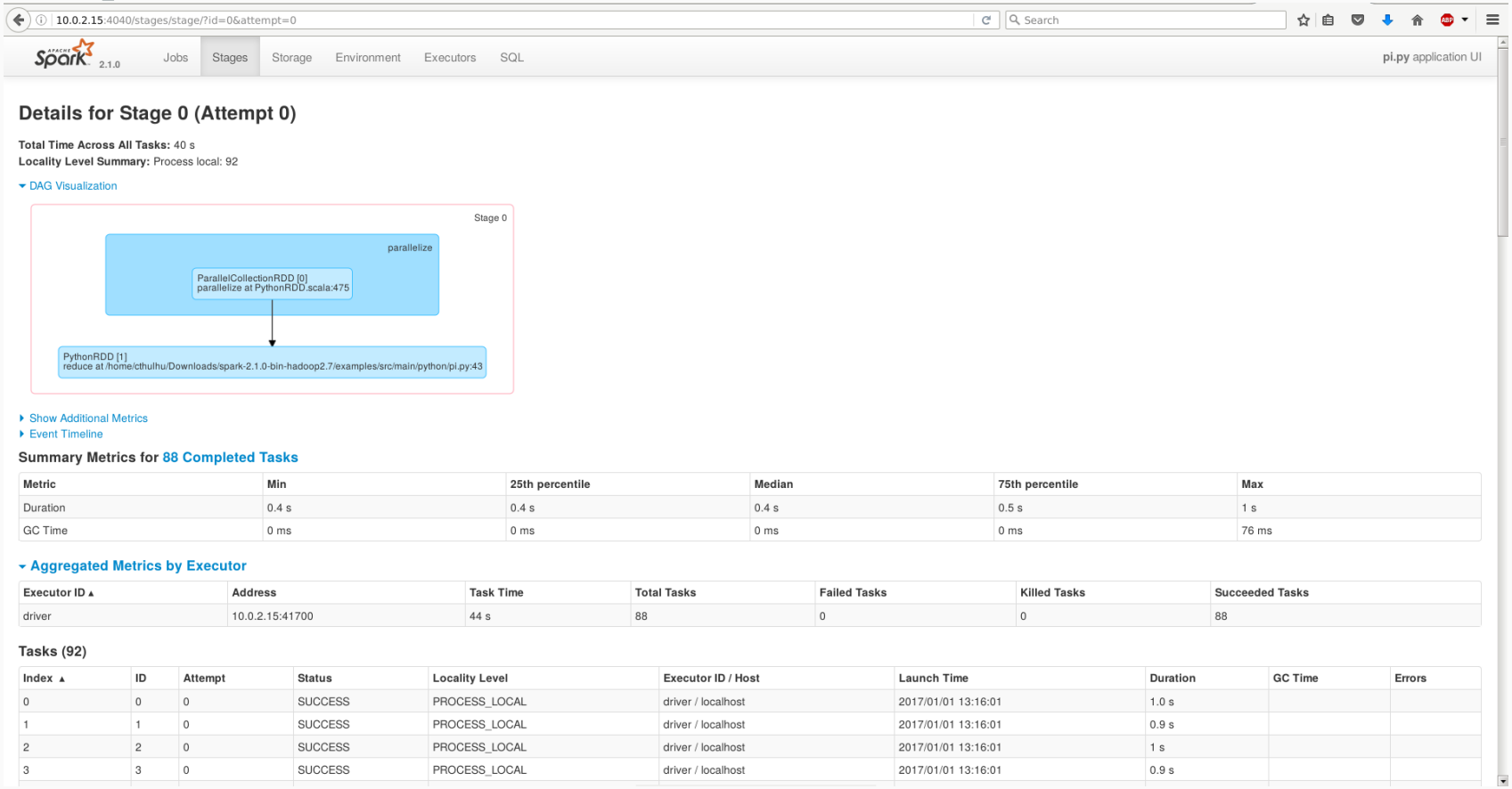
- Lanzamos un test

```
./bin/spark-submit examples/src/main/python/pi.py 1000
```

PRIMER PROGRAMA EN PYSPARK

Podemos supervisar el estado del sistema en

`http://127.0.0.1:4040`



PRIMER PROGRAMA EN PYSPARK

Ahora podemos ejecutar el mismo trabajo utilizando más recursos para reducir el tiempo

- 1 procesador

```
./bin/spark-submit examples/src/main/python/pi.py 10000
```

- 2 procesadores

```
./bin/spark-submit --master local[2] \
 examples/src/main/python/pi.py 10000
```

- 4 procesadores

```
./bin/spark-submit --master local[4] \
examples/src/main/python/pi.py 10000
```

PRIMER PROGRAMA EN PYSPARK

¿Pero qué hemos ejecutado?

PRIMER PROGRAMA EN PYSPARK

```
# Importamos dependencias
import sys
from random import random
from operator import add
from pyspark.sql import SparkSession

# Creamos una sesion de Spark
spark = SparkSession.builder.appName("PythonPi")\
        .getOrCreate()

partitions = int(sys.argv[1]) \
        if len(sys.argv) > 1 else 2
```

PRIMER PROGRAMA EN PYSPARK

```
# Funcion auxiliar
def f(_):
    x = random() * 2 - 1
    y = random() * 2 - 1
    return 1 if x ** 2 + y ** 2 < 1 else 0

# Creamos el dataset a procesar
n = 100000 * partitions
dataset = range(1, n + 1)
```

PRIMER PROGRAMA EN PYSPARK

```
# Creamos nuestro RDD
rdd = spark.sparkContext\
      .parallelize(dataset,partitions)

# Definimos nuestro pipeline

# 1) Aplicar f a todos los elementos
res_mapeo = rdd.map(f)

# 2) Sumamos los elementos del RDD resultante
res_reduce = res_mapeo.reduce(add)
```

PRIMER PROGRAMA EN PYSPARK

```
# Hemos calculado pi
print("Pi es %f" % (4.0 * res_reduce / n))
# Paramos la sesion
spark.stop()
```

USANDO PYSPARK

Podemos decir que cualquier programa en PySpark contiene:

- Creación de sesión
- Definición del dataset a utilizar
- Procesado
  - Definición de funciones auxiliares si fuese necesario
  - Definición del pipeline de procesado
- Obtención de resultados
- Cierre de sesión

OBJETO SESIÓN

Creamos un entorno con el que poder acceder al API de Spark

```
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("myApp")\
        .getOrCreate()
```

Indicamos el nombre de nuestra aplicación ("myApp") para poder hacer un seguimiento posterior

CREANDO UN RDD

En general definiremos un RDD de dos formas:

- Tomando una estructura de datos existentes y usando *parallelize* para distribuirla
- Leyendo el contenido desde algún medio de almacenamiento

CREANDO UN RDD

Creando un RDD a partir de un array:

```
#Array en Python
myarray = ['entry1', 'entry2', 'entry3']

#RDD a partir del array
my_rdd = spark.sparkContext.parallelize(myarray)

print('El array tiene %d entradas'%my_rdd.count())
```

CREANDO UN RDD

Un programa completo:

```
from pyspark import SparkContext


sc = SparkContext('local', 'MyApp')
numbers = [0,1,2,3,4,5]
# Get an RDD
rdd = sc.parallelize(numbers)
# Sum them up using reduction
result = rdd.reduce(lambda a,b : a + b)
print('Total: %d'%result)
# Close the context
sc.stop()
```

CREANDO UN RDD

SparkContext permite seleccionar configuración adicional:

- Indicar dónde está el nodo maestro de Spark
- local indica la máquina actual como maestra
- Cambiando a local[4] seleccionamos cuatro threads
- En entornos reales tendremos que indicar la URL del nodo maestro

CREANDO UN RDD

Podemos acceder a ficheros en almacenamiento secundario

- Cada fila del fichero será una fila del RDD
- Por defecto las entradas serán texto

```
#Leemos el fichero ubicado en /tmp/test.txt
my_rdd = spark.sparkContext\
         .textFile('/tmp/test.txt')

print('El fichero tiene %d lineas'%my_rdd.count())
```

CREANDO UN RDD

Generalmente una línea contendrá varios elementos.

Para un fichero que contenga:

user1, u1@email.com
user2, u2@email.com
user3, u3@email.com

```
my_rdd=spark.sparkContext.textFile('/tmp/test.txt')
          .map(lambda line: line.split(','))

print('El fichero tiene %d lineas'%my_rdd.count())
```

CREANDO UN DATAFRAME

Un dataframe es una extensión de un RDD

- Ofrece la funcionalidad de un RDD
- Permite el tipado de columnas
- Añade mejoras de rendimiento
- Permite la utilización de objetos

En general intentaremos utilizar dataframes

Por compatibilidad un dataframe puede transformarse en un RDD y viceversa

CREANDO UN DATAFRAME

user,	email,	score
user1,	u1@email.com,	100
user2,	u2@email.com,	30
user3,	u3@email.com,	15

```
from pyspark.sql.types import *
# Definimos la estructura
schema = StructType(
    [StructField("user", StringType(), True),
     StructField("email", StringType(), True),
     StructField("score", IntegerType(), True)])
# Leemos el csv
my_df = spark.read.csv('test.csv', header=True,
        sep=',',schema=schema)
```

OPERACIONES BÁSICAS

Algunas operaciones básicas de un RDD:

collect()	Devuelve todo el dataset en una lista
count()	Número de elementos
distinct()	Número de elementos distintos
first()	Primer elemento
isEmpty()	*True* cuando no hay contenido
max()	Valor máximo

OPERACIONES BÁSICAS

Algunas operaciones básicas de un RDD[1]:

mean()	Valor medio
min()	Valor mínimo
sum()	Suma de todas las entradas
take(num)	Toma las primeras num entradas

[1] http://spark.apache.org/docs/latest/api/python/pyspark.html#pyspark.RDD

OPERACIONES BÁSICAS

```
#Funciones para el RDD
print('Hay %d elementos'%my_rdd.count())
print('El primer elemento es %d'%my_rdd.first())
print('Los primeros 5 elementos %s'%my_rdd.take(5))
print('La suma es %d'%my_rdd.sum())

#Transformamos de RDD --> Lista en Python
my_list = my_rdd.collect()
print('La lista tiene %d elementos'%len(my_list))
```

OPERACIONES BÁSICAS

Los dataframes ofrecen funciones adicionales[2]:

columns	Nombre de las columnas
corr(col1,col2)	Correlación entre dos columnas
cov(col1,col2)	Covarianza entre dos columnas
describe(cols*)	Estadísticas descriptivas
rdd()	Devuelve el contenido en un RDD
select(cols*)	Seleccionar columnas
show(n)	Imprime n filas en la consola
selectExpr(expr*)	Selección usando expresiones
toPandas()	Transformación en Pandas dataframe

[2] http://spark.apache.org/docs/latest/api/python/pyspark.sql.html?highlight=dataframe#pyspark.sql.DataFrame

OPERACIONES BÁSICAS

```
print('Las columnas son %s'%my_df.columns)
# Imprimir el contenido
my_df.show()
# Imprimir el nombre del usuario con score 100
result=my_df\
        .filter(my_rdd.score==100)
        .first()
print('Usuario con score 100 %s'%result['user'])
```

OPERACIONES BÁSICAS

```
# Seleccionar dos columnas
fragment = my_df.select('user','score')
fragment.show()

# Lista de usuarios y su servidor de correo
aux = my_rdd.rdd.map(lambda x: \
      [x['user'],x['email'].split('@')[0]]).toDF()
aux.show()

# Creando un dataframe renombrando las columnas
renamed = aux.withColumnRenamed('_1','User')\
              .withColumnRenamed('_2','Email')
aux.show()
```

EJERCICIOS

EL DATASET

EL DATASET

Descargar el dataset de `https://github.com/juanmanuel-tirado/musicbrainz/`

BandName,	**Country**,	**Genre**,	**SongName**,	**AlbumName**,	**Duration**,	**Score**,	**Year**
Frank Zappa,	US,	rock,	"You Are What You Is",	"Teen-Age Wind,"	192000,	98,	1981
The White Stripes,	US,	alternative rock,	”Icky Thump",	”Icky Thump",	254000,	100,	2007
⋮							

EL DATASET

Cada línea contiene información sobre una canción:

- **BandName**: nombre de la agrupación
- **Country**: país origen de la agrupación
- **Genre**: género músical
- **SongName**: nombre de la canción
- **AlbumName**: nombre del álbum donde aparece la canción
- **Duration**: duración de la canción en milisegundos
- **Score**: puntuación de la canción entre 0 y 100
- **Year**: año de publicación

EL DATASET

Utilizando PySpark encontrar:

- El número de canciones que contiene el dataset
- El número de bandas
- El número de canciones de The Beatles
- Grupo con la canción más larga
- Grupo con la canción más corta
- Duración media de una canción por año
- Duración media de una canción por género
- Número de canciones del género rock por año desde 1965
- (Extra) Calcular la correlación entre la longitud del título de una canción y su duración

EL DATASET

El esquema a utilizar:

```
schema = StructType([
StructField("BandName", StringType(), True),
StructField("Country", StringType(), True),
StructField("Genre", StringType(), True),
StructField("SongName", StringType(), True),
StructField("AlbumName", StringType(), True),
StructField("Duration", IntegerType(), True),
StructField("ScoreGenre", IntegerType(), True),
StructField("Year", IntegerType(), True)])
```

EL DATASET

Un ejemplo:

Calcular el número de bandas en el dataset

```
# Cargar el dataset
df = spark.read.csv(
 'recordings.csv', '/tmp/recordings.csv',
 header=False, sep=',', schema=schema)
# Seleccionar el nombre de las bandas,
# eliminar repetidos y contar
num_bands=df.select('BandName').distinct().count()
print('The number of bands is %d'%num_bands)
```