Spark Streaming y teoría de Grafos

Alumno: Daniel Rodriguez Amezaga

ACTIVIDAD LECCIÓN 7

Objetivos:

- Realizar una pequeña memoria explicando la teoría de Grafos y sus utilidades en Big Data.
- Realizar un script usando Spark Streaming

Contenido correspondiente a lección 7:

1. Spark y su ecosistema.
2. Spark SQL y DataFrame.
3. Spark RDD

Actividad Spark Streaming

En este desafiante ejercicio de Apache Spark Streaming, he instalado todas las dependencias necesarias y configurado el entorno para el procesamiento de flujos en tiempo real. Aunque encontré algunos desafíos a lo largo del camino, estoy orgulloso del esfuerzo que invertí y de todo lo que aprendí en el proceso. Quiero expresar mi gratitud al profesor/mentor por su valiosa ayuda y orientación en esta práctica. Gracias a su apoyo, pude completar exitosamente el ejemplo y fortalecer mis habilidades en el análisis de datos en tiempo real con Spark.

En mi caso cree un entorno virtual con todo lo necesario para que funcioné apache spark y laze este código comentado desde “jupyter lab”.

In [1]:

```
# Librerías necesarias
from pyspark import SparkConf, SparkContext
from pyspark.streaming import StreamingContext
import socket

# Creación de una configuración Spark
conf = SparkConf().setAppName("StreamApp")

# Creación del contexto Spark
sc = SparkContext(conf=conf)
sc.setLogLevel("WARN")
```

```
# Creación del contexto de streaming con intervalo de 2 segundos
ssc = StreamingContext(sc, 2)
ssc.checkpoint("checkpoint_App")

# Creación del flujo de datos desde el socket
dataStream = ssc.socketTextStream("localhost", 9009)
obj = socket.socket()

# Conexión con el servidor. Parámetros: IP (puede ser del tipo 192.168.1.1 o localhost), Puerto
obj.connect(("localhost", 9009))
print("Conectado al servidor")

# Bucle para retener la conexión
while True:
    # Entrada de datos para enviar mensajes
    mensaje = input("Mensaje desde Cliente a Servidor >> ")
    mens = mensaje.encode('utf-8')
    obj.send(mens)

    if mensaje == "fin":
        break
# En este paso pide al usuario introducir los mensajes hasta que decida escribir la palabra “fin”.

# Cierre de la instancia del objeto servidor
obj.close()

# Mensaje de "Conexión cerrada" al cerrar la conexión
print("Conexión cerrada")
```

A continuación la parte del receptor el cual muestra los mensajes escritor por el emisor:

```
import socket

# Configuración del socket
TCP_IP = "localhost"
TCP_PORT = 9009

# Creación del socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((TCP_IP, TCP_PORT))
s.listen(1)

print("Esperando conexión TCP...")
conn, addr = s.accept()
print("Conexión establecida... Comenzando a recibir mensajes.")
```

```
while True:
    # Recibimos los datos del cliente
    data = conn.recv(1024)

    # Si no hay datos, se ha cerrado la conexión
    if not data:
        break

    # Mostramos la dirección IP y el mensaje recibido
    print("Recibido de la IP: {} Puerto: {}".format(addr[0], addr[1]))
    print(data.decode())

    # Comprobamos si se recibió el mensaje de finalización
    if data.decode() == "fin":
        break

# Cerramos la conexión del socket cliente y servidor
conn.close()
s.close()

print("Conexiones cerradas")
```

```
Esperando conexión TCP...
Conexión establecida... Comenzando a recibir mensajes.
Recibido de la IP: 127.0.0.1 Puerto: 37162
Lección 7 BigData
Recibido de la IP: 127.0.0.1 Puerto: 37162
Apache Spark Streaming
Recibido de la IP: 127.0.0.1 Puerto: 37162
Daniel Rodriguez Amezaga
Recibido de la IP: 127.0.0.1 Puerto: 37162
Reto superado!!!
Recibido de la IP: 127.0.0.1 Puerto: 37162
fin
Conexiones cerradas
```

Adjunto los archivos de jupyter además de este pdf.