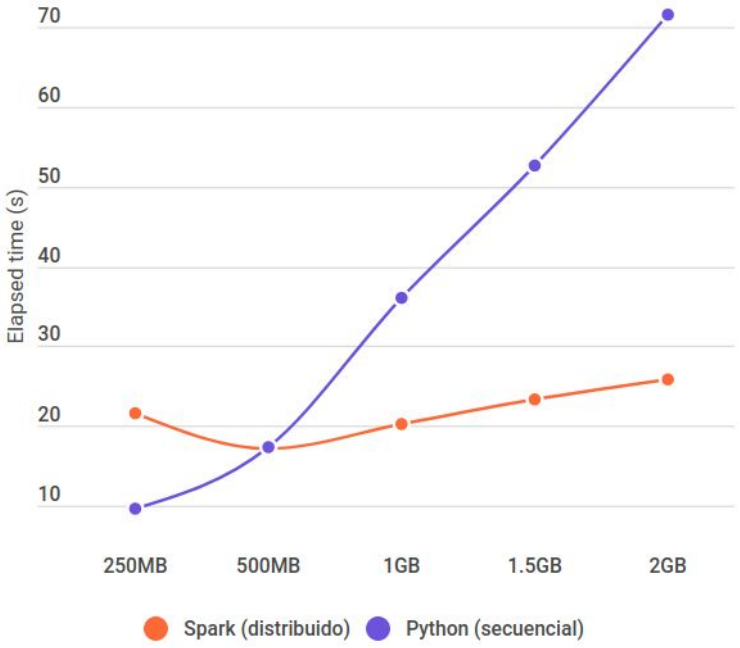
Análisis de datos con Spark y AWS

El objetivo de este proyecto es evaluar el rendimiento de Spark en función de la cantidad de datos de entrada. Para conocer el rendimiento y la mejora que aporta el uso de Spark, se tiene que comparar el tiempo de ejecución del mismo programa en versión secuencial y en versión distribuida con Spark.

El programa utilizado para la prueba es un código que retorna los 5 hashtags más usados de un conjunto de tweets descargados previamente.
Los dos programas serán ejecutados en máquinas de AWS ec2 con las mismas características. La versión secuencial en python utilizará solo una de estas máquinas, mientras que la versión Spark estará distribuida entre 4 máquinas. El modelo utilizado de AWS ec2 es **Ubuntu Server 18.04 LTS (HVM)** con instancias del tipo **m5.2xlarge**.

Para poder ver el escalamiento del tiempo de ejecución hace falta comparar el rendimiento con ficheros de datos de distintos tamaños, en este caso se utilizan ficheros de 250MB, 500MB, 1GB, 1.5GB y 2GB aproximadamente. Los ficheros están creados a partir de la concatenación de un fichero más pequeño de 14 MB aproximadamente.

Tamaño del fichero de entrada	Spark distribuido Elapsed time (s)	Python secuencial Elapsed time (s)
250MB	21.555	9.671
500MB	17.080	17.409
1GB	20.141	36.114
1.5GB	23.403	52.576
2GB	25.841	71.479



Tal como se puede ver en el gráfico el tiempo de ejecución de Spark es relativamente constante (tiene un crecimiento muy lento). El trabajo está distribuido entre 4 máquinas, por lo que la ejecución es mucho más rápida que en secuencial. La parte principal del tiempo utilizado es para desplegar la estructura de Spark entre las 4 máquinas, por lo que el tiempo necesario es poco dependiente del tamaño de los datos de entrada.
El tiempo de ejecución del código secuencial crece de una forma lineal en función de los datos de entrada. Esto es debido a que la mayoría del tiempo es utilizado a trabajar directamente sobre los datos, ya que no están distribuidos y una sola máquina tiene que hacer todo el trabajo. Como consecuencia, el tiempo necesario es directamente proporcional con el tamaño de los datos de entrada.

En resumen, para los ficheros de poco tamaño (menos de 500GB), no es útil utilizar Spark, ya que el tiempo necesario para desplegar toda su estructura es superior al tiempo utilizado por la solución secuencial. Para ficheros de datos grandes, Spark rápidamente se convierte en una solución mejor a la solución secuencial, ya que proporciona un muy buen escalamiento.

Scripts utilizados:
download.py : Utilizado para descargar el fichero original de tweets
hashtag_seq.py : Implementación secuencial con python
hashtag_spark.py : Implementación distribuida con pySpark