UNIVERSIDAD REY JUAN CARLOS

Master en Data Science

Sistemas distribuidos de procesamiento de datos

ANALISIS DE SENTIMIENTOS CON HADOOP

Autor: Alejandro Pérez Barreiro

Móstoles, Diciembre 2016

Índice de contenidos

1. Objetivo de la práctica.

El análisis de sentimientos es un área de aplicación que está cobrando bastante importancia en los últimos años. La idea es utilizar un programa que automáticamente extraiga emociones de un texto, para poder obtener actitudes y sentimientos existentes en la red, ya sea en blogs, comentarios, foros, Twitter etc… Así se pueden rastrear opiniones sobre productos, marcas, personas para determinar si están bien o mal vistos en la red o hasta establecer la sensación o actitud de un segmento de la red.

Para poder obtener un análisis realista se suele trabajar sobre un gran volumen de información de forma que se cuente con un número elevado de datos sobre los que realizar el análisis. En este sentido suele ser bastante interesante utilizar técnicas de paralelismo escalables, cómo MapReduce, para su ejecución. En concreto, en esta práctica se abordará el problema de implementar un programa de análisis de sentimientos que utilice datos de Twitter y trabaje con ellos utilizando el algoritmo MapReduce.

Más en detalle, el objetivo será el de extraer tweets de Estados Unidos y hacer un análisis de sentimientos sobre los diferentes estados generando así una clasificación en función a la "felicidad" de cada estado comprando las palabras de esos tweets con los de una lista en la que un serie de palabras tienen asignado un valor entre -5 y 5. Las palabras que no estén en esa lista sumaran 0 al valor del tweet.

2. Obtención de datos.

Para llevar la práctica a cabo será necesario obtener una gran cantidad de tweets para analizar, estos tweets los obtendremos accediendo al stream de los últimos tweets publicados. Para ello se ejecuta el programa *twitterstream.py* habiendo indicado previamente en el código del programa nuestro consumer key, consumer secret, access token, y access secret que nos proporciona Twitter cuando nos creamos una app desde https://dev.twitter.com/apps. Dejamos corriendo el

programa unas horas y guardamos la salida en un archivo json (*tweets.json*), de esta manera ya tenemos los datos listos para ejecutar el MapReduce.

3. Descripción del código.

El código consta de tres partes, la creación de un diccionario con las palabras y sus valores, la parte del Map y la parte del Reduce.

Para la creación del diccionario (*diccionario.txt*) bastará con ejecutar el script *diccionario.py* que convertirá la lista de palabras con sus valores del archivo *AFINN-111.txt* a un diccionario python.

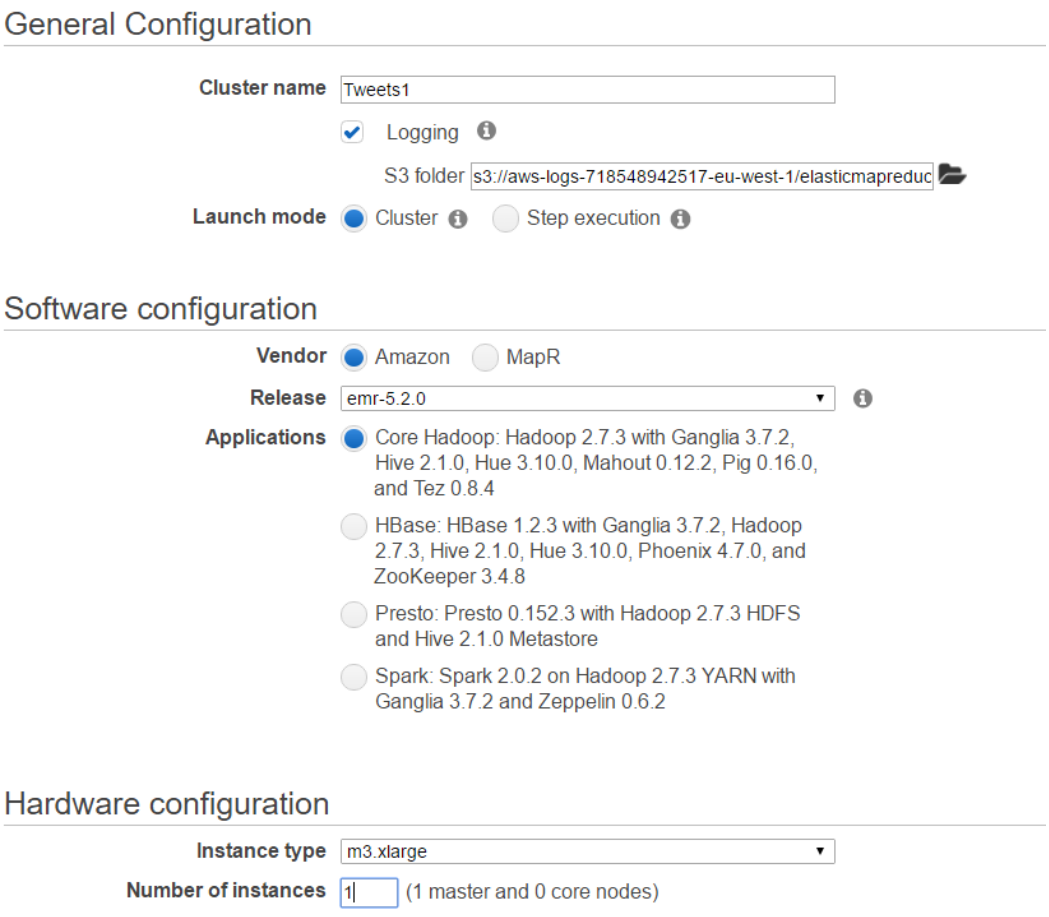
En el Map en primer lugar se importan los módulos necesarios para la ejecución del mismo (*map_tweet.py*) que son los módulos *sys, json, re y string* y añado la ruta del directorio donde quiero que busque los archivos adicionales que vaya a cargar. Después de los módulos se carga el diccionario (*diccionario.txt)* donde se encuentra cada palabra con su correspondiente valor. Hecho esto se puede proceder a hacer la selección de los tweets que nos interesan, en nuestro caso lo procedentes de Estados Unidos. Para ello leo los tweets uno por uno del archivo de entrada *tweets.json* y me quedo con los que tengan el campo de texto y de lugar y que además tengan como código de país “US” que es el correspondiente a Estados Unidos. Si cumplen estas condiciones tomo como clave el segundo valor del campo *full_name* dentro de *place* ya que será el nombre del estado donde se escribió el tweet. Por último, para calcular el valor del tweet lo divido en palabras y voy comparando cada una de ellas con las del diccionario, asignándoles su valor correspondiente. Si la palabra no aparece en el diccionario le asigno el valor 0. Previamente a la comparación con el diccionario elimino los signos de puntuación del texto para que la comparación se más precisa. Los datos de salida para pasarle al Reduce serán de la forma clave-valor donde la clave es el estado en el que se escribió el tweet y el valor es el valor total del tweet tras sumar la puntuación de cada una de sus palabras.

En cuanto al Reduce es sencillo, únicamente voy creando un diccionario donde las claves son los diferentes estados y el valor será la suma total de todos los tweets de ese estado. Para ello recorro todas las líneas de la entrada y miro si ese estado ya está en el diccionario, si es así le sumo el valor de ese tweet al ya existente, en caso de no estar añado ese estado al diccionario con el valor de ese tweet. Por ultimo emito pares clave-valor con el nombre del estado y el resultado del análisis de sentimiento de los tweets de ese estado.

4. Ejecución del código

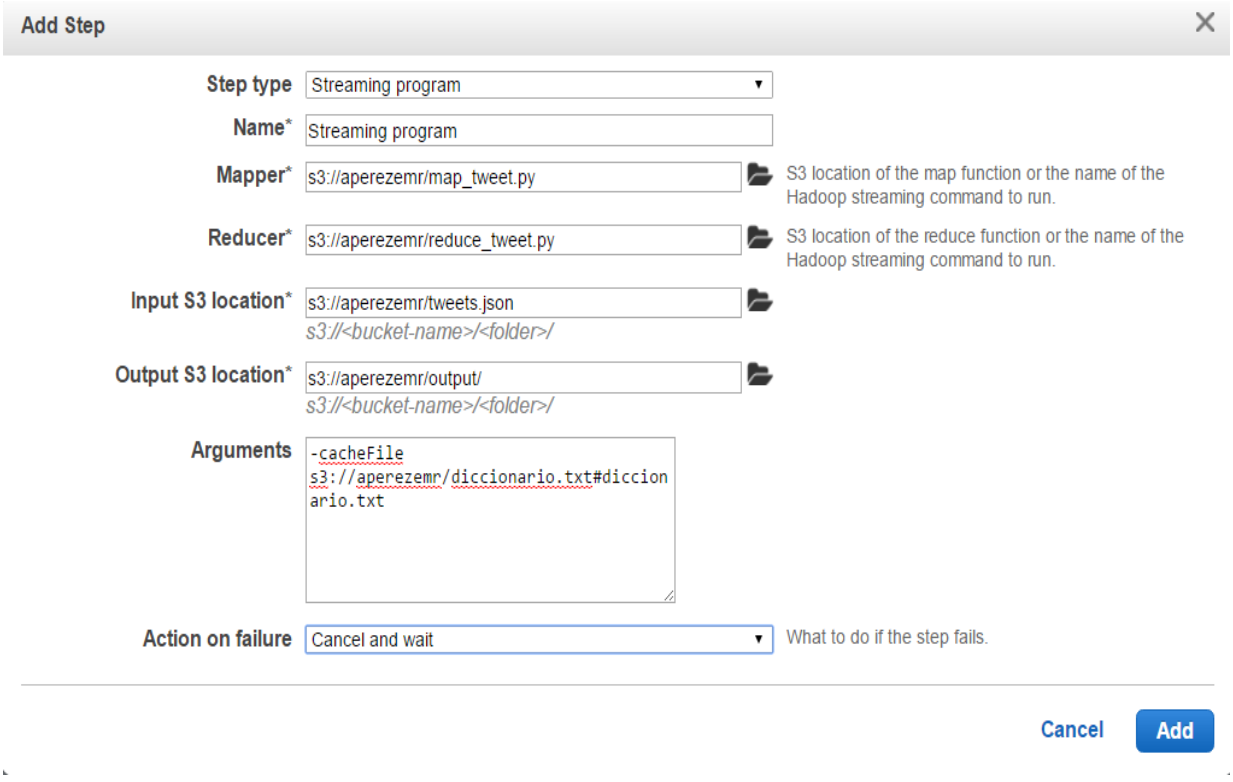
La ejecución del código la voy a llevar a cabo en Amazon Elastic MapReduce.

En primer lugar vamos a crear un clúster con las propiedades que se ven en la imagen. Más adelante estudiaremos que número y que tipo de instancias es más apropiado.



Una vez creado el clúster se procede a añadirle el paso stream que será el MapReduce. Para crear el paso se hace en la casilla *Add step* y se cubren las celdas con la ubicación de nuestro map (*map_tweet.py*), reduce (*reduce_tweet.py*), datos de entrada (*tweets.json*), carpeta para la salida del MapReduce (que no debe existir) y los argumentos que queramos añadir a nuestro paso. En este caso en argumentos hay que indicar que suba a todos los nodos del clúster el fichero *diccionario.txt* ya que el map necesita abrirlo para ejecutarse. Esto se hace con la orden *–cacheFile s3://aperezemr/diccionario.txt#diccionario.txt* que indica que suba al clúster el fichero *s3://aperezemr/diccionario.txt* con el nombre *diccionario.txt.*

Como se observa en la siguiente imagen es necesario que todos los ficheros estén subidos a S3.



Una vez hecho esto se añade el paso y se obtienen los resultados del MapReduce en la carpeta output indicada antes.

5. Escalabilidad de la solución

En este apartado únicamente vamos a ver como varían los tiempos de ejecución en función del número de nodos del clúster y del tipo de instancias que se utilicen.
Los resultados se resumen en la siguiente tabla:

Instancia	Master nodes	Core nodes	Creación clúster (min.)	Ejecución (min.)
m3.xlarge	1	0	10	2
m3.xlarge	1	1	10	2
m3.xlarge	1	2	11	1
m3.2xlarge	1	0	11	1
m3.2xlarge	1	1	9	1

Se observa como los tiempos no varían mucho. Esto se debe a que los datos de entrada únicamente ocupan 1.5 GB y por tanto los tiempos son bastante bajos dando poco margen a su mejora.

6. Comentarios personales

Problemas encontrados:

Uno de los problemas a la hora de hacer la práctica fue que al hacer la selección de los tweets con las condiciones mencionadas apenas se consiguen 10 MB de datos sobre un archivo de 1.3 GB lo que es una proporción muy baja.

Además dentro de esos tweets que pasan el filtro muchos no tienen los campos cubiertos de forma correcta ya que si se ve la salida del MapReduce se puede observar cómo pone a Manhattan o USA como dos estados más de Estados Unidos.

Evaluación del tiempo dedicado:

En mi caso el tiempo dedicado a la práctica fue bastante elevado debido a que mi formación es matemática, lo que hace que tenga que dedicar mucho más tiempo que si tuviese algún conocimiento previo de informática. Aun así opino que es

tiempo perfectamente invertido ya que la práctica hace que entiendas perfectamente el funcionamiento del MapReduce al igual que la utilización de servicios de AWS como EMR o S3 y como es obvio te va dando soltura en cuanto a programación en Python.