

Memoria Trabajo Kafka

Carolina Giménez Arias y Pedro Pons Suñer

Mayo de 2019

1. Introducción

Apache Kafka es una plataforma de distribución en streaming, se trata de un sistema de mensajes “publish / subscribe” o “productor-consumidor” Open Source basado en una arquitectura P2P (arquitectura Peer to Peer) que consiste en una red en la que los aspectos funcionan sin clientes o servidores fijos, sino que hay una serie de nodos que se comportan como iguales entre sí. Su funcionamiento se basa en la existencia de un elemento “publisher” o productor que, al generar un dato o mensaje, no envía de forma directa a la “dirección” del consumidor o suscriptor, sino que es este quien se suscribe para poder recibir estos mensajes.

Para ello, se dispone de listas de temas o topics publicados específicos y un conjunto de consumidores. El productor trata de clasificar el mensaje en base a una tipología, lo pone en la lista de un tema específico y el receptor se suscribe a la listas para recibir ese tipo de mensajes.

En el presente trabajo se pretende ejemplificar el uso de esta plataforma durante la producción y consumición de tweets en tiempo real. Para llevarlo a cabo ha sido necesario crear una cuenta de desarrollador en la API de Twitter; de esta forma, se han obtenido unas credenciales o *tokens* que permiten recolectar los tweets.

A lo largo de esta memoria se indicarán los objetivos y algunos detalles sobre la estructura y el funcionamiento de nuestro código e instalación de kafka. Para más información relacionada con el funcionamiento del código, se adjuntan los tres notebooks de python con explicaciones más concretas sobre cada apartado.

2. Objetivos

Los principales objetivos que se han planteado para este proyecto han sido los siguientes:

- Implementar un sistema con Apache Kafka, en primera instancia, con un único broker.
- Desplegar un productor y un consumidor y conectarlos con Twitter para poder extraer información en streaming.
- Mostrar de forma gráfica algunos resultados en tiempo real.
- Analizar una temática que afecte a la sociedad actual y que sea tendencia en Twitter.
- Obtener alguna conclusión sobre la temática analizada.

3. Instalación y funcionamiento de Kafka

Para poder realizar este trabajo ha sido necesaria la instalación de Apache Kafka en nuestros respectivos sistemas operativos. De forma genérica, esta instalación se ha basado en descargar desde la página web de Apache kafka (<https://kafka.apache.org>) la última versión disponible. A continuación, ha sido necesario descomprimir el fichero descargado en el directorio elegido y, en caso de windows, se ha tenido que crear una variable de entorno denominada KAFKA_HOME, que haga referencia a la ruta/directorio de instalación.

Una vez instalado, se debe inicializar Zookeeper en una terminal y el servidor de Kafka en otra. Es esencial este paso, ya que Kafka requiere de Zookeeper activo para funcionar, pues es el encargado de manejar los brokers y ayudar en la elección de líderes para particiones.

Con la finalidad de que nuestro código sea tolerante a fallos y tenga un buen rendimiento, se deberían crear varios brokers. El número ideal sería 3, ya que en caso de que uno de ellos estuviese en mantenimiento y otro fallase por algún motivo, siempre quedaría alguno activo y el sistema podría seguir funcionando. A pesar de conocer la metodología necesaria para inicializar estos tres brokers, finalmente no ha sido posible debido a que este proyecto ha sido desplegado en local y se requerirían de otras condiciones algo más complejas de obtener, como el empleo de un servidor remoto.

En cuanto a los topics, se crearán dos: en el primero se almacenarán los tweets “en crudo”, para ser posteriormente consumidos para procesarlos y guardar los datos de interés en un segundo topic.

4. Detalles del funcionamiento del código

El objetivo de nuestro proyecto es recoger en tiempo real tweets a partir de la API de Twitter; en concreto, la palabra clave escogida será “26M” ya que resulta interesante conocer la opinión política de los usuarios de Twitter sobre las elecciones del 26 de mayo. Una vez nuestro productor ha recogido dichos tweets, el productor-consumidor los procesará para convertir aquellas palabras relacionadas con los nombres de cada partido político, como por ejemplo los alias de Twitter de cada uno, en los nombres de sus respectivos partidos políticos para que, finalmente, estos sean consumidos por los distintos consumidores y analizados en tiempo real. Para todo ello, contamos con cuatro scripts:

- **Productor**, que recoja todos los tweets con el hashtag escogido y los produzca en el topic *tweetsRaw*.
- **Consumidor-Productor**, que obtenga los tweets recogidos en el topic anterior, los procese y los vuelva a reinsertar en un segundo topic (*tweetsProcessed*). Utilizaremos este paso intermedio para sustituir ciertas palabras clave, como los *@user* de cada partido, por el nombre de los mismos, con el fin de hacer las comparaciones más equitativas.
- **Consumidor Gráfico de Barras**, que lea los tweets procesados de *tweetsProcessed* y utilice los datos para representarlos en tiempo real de dos formas: un gráfico de barras que muestre la suma acumulada de menciones para cada partido y una serie temporal que muestre el número de menciones a cada partido en distintas ventanas temporales.
- **Consumidor Serie Temporal**, que también lea los tweets procesados y utilice los datos para obtener una gráfica en tiempo real que muestre la evolución temporal del número de menciones por partido.

5. Ejecución del código

5.1. Resultados del script

Para poder demostrar el correcto funcionamiento lo ideal sería ejecutar el código proporcionado, ya que el punto más interesante de este proyecto reside en la capacidad de poder visualizar gráficamente la actividad de Twitter en tiempo real. A continuación, se muestran algunas capturas sobre el correcto funcionamiento de nuestra aplicación:



Figura 1: Número de tweets por partido.

La Figura 1 muestra el número de tweets en tiempo real que hablan de cada partido. Se aprecia como, durante el intervalo de tiempo que nuestro código estuvo en funcionamiento, la mayoría de los tweets hablaban sobre VOX y, en segundo lugar, se hablaba sobre Ciudadanos.

A su vez nos pareció interesante estudiar la evolución temporal de la popularidad de cada partido político con la finalidad de detectar algunas tendencias o picos de popularidad en el tiempo. Para ello, utilizando un segundo consumidor, se hizo un análisis a modo de serie temporal de la evolución del número de menciones en el tiempo de cada partido (ver Figura 2):

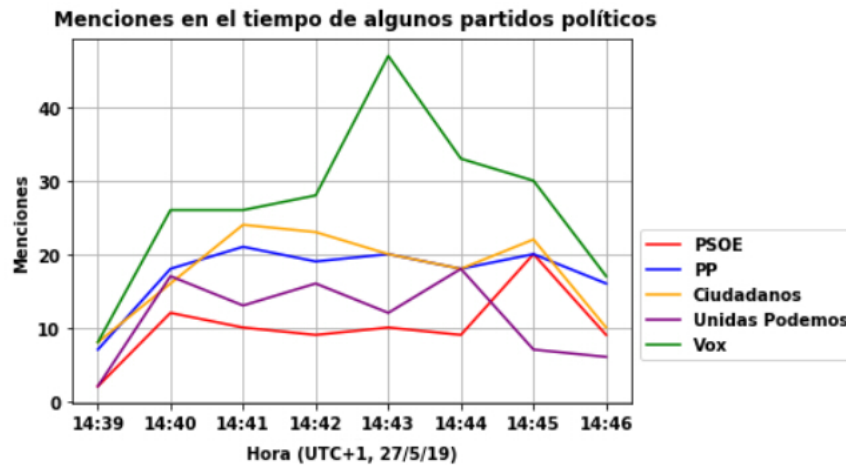
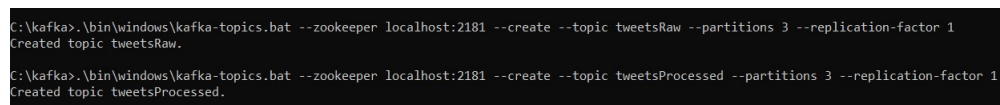


Figura 2: Evolución temporal de la popularidad de cada partido.

5.2. Pruebas de funcionamiento

Inicialmente se pretendía que el código estuviese desarrollado para ser tolerante a fallos en caso de que uno de los brokers diese error. Como finalmente se ha decidido emplear un único broker, las pruebas de funcionamiento que se realizarán comprobando que se han creado correctamente los topics, el número de replicasiones, el número de particiones y el broker.

Para crear los diferentes temas o topics, será necesario usar el comando “Kafka-topics” en la ventana de comandos tal y como se aprecia en la Figura 3. Zookeeper debe encontrarse en la misma máquina (localhost) desde la que ejecutemos el comando.



```
C:\kafka>.bin\windows\kafka-topics.bat --zookeeper localhost:2181 --create --topic tweetsRaw --partitions 3 --replication-factor 1
Created topic tweetsRaw.

C:\kafka>.bin\windows\kafka-topics.bat --zookeeper localhost:2181 --create --topic tweetsProcessed --partitions 3 --replication-factor 1
Created topic tweetsProcessed.
```

Figura 3: Creación de los topics.

Es conocido que, en caso de enviar mensajes a un topic que no existiese, Kafka nos enviaría un *warning* indicando que no existe un broker líder para ese topic, pero nos deja enviar los mensajes. Eso se debe a que Kafka por defecto crea el topic con los valores por defecto en esos casos. En este caso, el valor por defecto sería de 1 partición, y nos interesa tener 3 por topic, razón por la cual se ha decidido crearlos previamente desde la consola de comandos. Para este proyecto se han creado dos topics distintos: *tweetsRaw*, donde produce los datos el script productor y *tweetsProcessed*, topic al que envía los datos procesados el consumidor-productor.

En nuestro caso, el factor de replicación elegido es de 1. Es importante recordar que teniendo un solo broker, el máximo factor de replicación es 1, ya que no podemos tener un factor de replicado superior al número de brokers que tenemos.

El número de particiones que se definen para cada topic será de 3, ya que se considera que con esta distribución bastaría para repartir la carga de mensajes. Esta opción debe especificarse al crear el topic, como puede verse en la Figura 3.

En la Figura 4 se observa cómo efectivamente se han definido correctamente las particiones para cada topic.

```

C:\kafka>.bin\windows\kafka-topics.bat --zookeeper localhost:2181 --describe --topic tweetsRaw
Topic: tweetsRaw PartitionCount:3 ReplicationFactor:1 Configs:
Topic: tweetsRaw Partition: 0 Leader: 0 Replicas: 0 Isr: 0
Topic: tweetsRaw Partition: 1 Leader: 0 Replicas: 0 Isr: 0
Topic: tweetsRaw Partition: 2 Leader: 0 Replicas: 0 Isr: 0

C:\kafka>.bin\windows\kafka-topics.bat --zookeeper localhost:2181 --describe --topic tweetsProcessed
Topic: tweetsProcessed PartitionCount:3 ReplicationFactor:1 Configs:
Topic: tweetsProcessed Partition: 0 Leader: 0 Replicas: 0 Isr: 0
Topic: tweetsProcessed Partition: 1 Leader: 0 Replicas: 0 Isr: 0
Topic: tweetsProcessed Partition: 2 Leader: 0 Replicas: 0 Isr: 0

```

Figura 4: Descripción de los topics.

6. Conclusiones

Con este trabajo hemos podido comprender el funcionamiento de los principales conceptos de la plataforma Apache Kafka: Zookeeper, Broker, Topic, Productor, Productor-Consumidor y Consumidor. Esto es algo que ayuda a comprender mejor las características y funcionamientos que ya se detallaron en la teoría vista durante la asignatura de Big Data.

Como línea de futura mejora se podría plantear la incorporación de varios brokers para permitir la redundancia de la información y hacer este proyecto más tolerante a fallos; para ello, se requerirá el despliegue de un cluster de brokers.

Con respecto a los resultados, tras hacer un breve análisis de las gráficas obtenidas por los distintos consumidores, se observa como el día 27 de mayo en Twitter se hablaba sobre todo del partido VOX, probablemente debido a la fuerza que ha tomado en España tras las elecciones del 26M.