Javier Guzmán Figueira Domínguez
Big Data
Máster en Investigación en Inteligencia Artificial, UIMP
23/04/2018

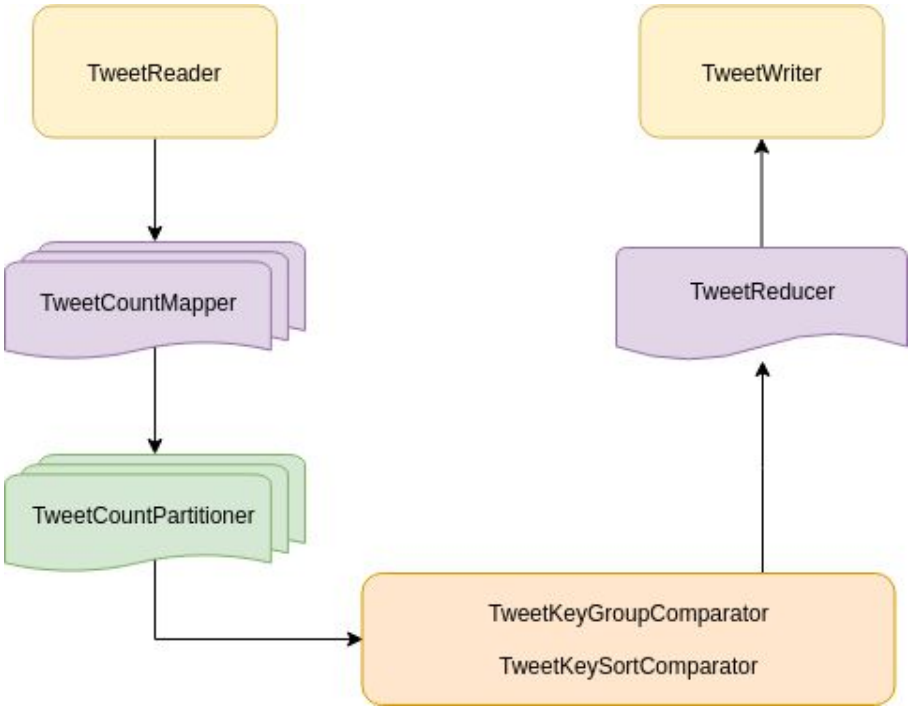
Práctica de Hadoop: Arquitectura

Modelo

- Tweet: clase utilizada para *mapear* un objeto JSON y mantener solo las propiedades requeridas en esta clase (id, usuario, número de retweets y timestamp)
- TweetKey: clase utilizada para encapsular las claves consistentes en pares usuario-timestamp. Aunque podría no ser obligatorio utilizar esta clase y únicamente emplear un *Text* con el usuario, el timestamp no será útil en la fase de ordenado para recibir en el *reduce* el conjunto de valores ordenados por *timestamp* ascendente.
- TweetKey: se utiliza como valor de salida de la fase *map*. Contiene información sobre el número de retweets, el timestamp y un campo que se utilizará como contador en el *reduce*.
- MappedTweetCollection: encapsula una colección de *MappedTweet* y las operaciones que se pueden realizar sobre la misma.
- UserStats: objeto valor de salida de la *reduce,* que contiene las métricas calculadas para cada usuario.

Entrada

Se ha definido una entrada personalizada para *parsear* los objetos JSON del archivo de entrada al programa. En la clase *TweetReader* se realiza dicho parseo y se definen los pares clave-valor que tomará el *mapper* como entrada. De esta manera, se define el número de línea del archivo como clave y un objeto que contenga la información necesaria del tweet, como valor.



Map

En la clase *TweetCountMapper*, para cada tweet se genera un par clave-valor, en el que la clave es un TweetKey y el valor un TweetKey. A ambos objetos se le asignan los valores que contiene el tweet y la propiedad *count = 1*.

TweetKeyPartitioner

Mediante esta clase particionamos la salida de la fase *map* de forma que enviemos a cada *reducer* las tuplas que referencian al mismo *username*. Por defecto, *hadoop* utiliza el *HashPartitioner*, por lo que se ha implementado esta clase para únicamente tener en cuenta la propiedad *username* del objeto *TweetKey*.

TweetKeyGroupComparator

Aquí se implementa un comparador de grupos personalizado para únicamente tener en cuenta la username del objeto TweetKey. Esto se debe a que, al salir de la fase map, cada clave incluye el timestamp de su valor asociado. Se realiza esta transferencia para poder disponer del conjunto de tweets de un mismo username ordenados por timestamp ascendente, en el reducer. Debido a eso, debemos especificar en este comparador, que únicamente se tenga en cuenta el username.

TweetSortComparator

Tal y como se ha mencionado en el apartado anterior, en esta clase se implementa un comparación para poder disponer del conjunto de tweets de un mismo username ordenados por timestamp ascendente, en el reducer.

TweetCountReducer

En esta etapa, recibimos una colección de objetos MappedTweet por cada TweetKey. Al usar los métodos definidos en esta clase, se calcula el número de tweets por usuario, el *volumeMomentum* de métricas y el *popularityMomentum*.

Salida

Se ha implementado un formato de salida personalizado, de forma que se cree un archivo HDFS por cada partición de salida de la fase *reduce*. En cada una de las particiones se realiza una escritura en formato CSV de la forma:

<username,tweetsCount,volumeMomentum,popularityMomentum>