

Big Data

Stiven Andrés Hurtado Loaiza - shurtad5@eafit.edu.co

EAFIT

Palabras clave— k-means, dataset, cluster, spark, python, big data, etl.

I. RESUMEN

En el presente documento se plantea una solución al Proyecto 4 de la materia Tópicos Especiales en Telemática. Éste consiste en implementar un algoritmo distribuido en Big Data que permita agrupar diferentes documentos, específicamente se hablará del algoritmo **k-means**, que es una medida que ayuda a clasificar documentos relacionados entre sí.

La implementación de este algoritmo distribuido se realizara en Python 3 con Spark.

II. INTRODUCCIÓN

El mundo de hoy se puede decir que es inconcebible sin la tecnología, pues gracias a dispositivos como los computadores, celulares, entre otros, podemos buscar todo tipo de información que a diario fluye por medio de internet. Es tanta la información que se transmite por este medio, que resulta necesario para los creadores de buscadores como Google, Firefox, Microsoft Edge, entre otros, utilizar algoritmos que agrupen dicha información y de esta manera poder mostrarle al usuario lo más relacionado posible con la búsqueda realizada por éste.

III. MARCO TEÓRICO

Se cuenta con un conjunto de documentos desordenados, los cuales requieren ser organizados en un número de clusters dependiendo del grado de similitud entre estos documentos. El primer paso a realizar es determinar qué tan similares son cada uno de los documentos del dataset. Existen varios algoritmos que ayudan a determinar la similitud entre dos documentos, entre ellos están: *Distancia Euclidiana*, *Similitud Coseno*, *Coeficiente de Jaccard*, *Coeficiente de Correlación de Pearson*, *Divergencia de Kullback-Leibler*, *tf-idf (Term frequency – Inverse document frequency)*.

Luego de determinar qué tan similares son los documentos, se debe utilizar un algoritmo de agrupamiento como *k-means* para clasificarlos en un número de *k* clusters.

IV. ANÁLISIS y DISEÑO MEDIANTE ETL

El diseño del algoritmo distribuido se utiliza a través de **ETL**, el cual es un proceso que consiste en Extracción, Transformación y Carga de los datos para analizarlos con un fin específico, en este caso, clasificar documentos en clusters.

La **Extracción** consiste en extraer los datos desde algún sistema origen, ya sea una ruta local, desde HDFS, o cualquier otro sitio válido que contenga un dataset.

En la **Transformación** se aplican una serie de funciones a los datos extraídos para convertirlos en datos que serán cargados.

En la **Carga** se almacenan los datos ya transformados en el sistema destino, por ejemplo, una base de datos o simplemente una ruta válida donde se puedan guardar estos datos.

V. IMPLEMENTACIÓN

A continuación se muestra cada una de las librerías y funciones requeridas para llevar a cabo la implementación del algoritmo distribuido. **Cabe destacar que se hace uso del código proporcionado por Apache Spark en los ejemplos de kmeans :**

- Se importan las siguientes librerías:
 - *SparkContext* - es el punto de entrada principal para utilizar las funcionalidades que ofrece Spark, es decir, sirve para establecer una conexión a un cluster Spark. Entre las funcionalidades están: crear RDDs, acumuladores y variables de difusión (broadcast) en ese cluster.
 - *randint* - módulo para generar números random.
 - *KMeans* - librería de machine learning que ofrece Spark para organizar en clusters un conjunto de datos.
 - *KMeansModel* - sirve para guardar y cargar el modelo de KMeans.
 - *HashingTF* - mapea un conjunto de términos y retorna su correspondiente frecuencia.
 - *IDF* - computa la frecuencia inversa de documento, es decir, la frecuencia de ocurrencia

VI. CONCLUSIONES

Comparando HPC con Big Data se observan limitaciones principalmente sobre hardware, ya que si se desea escalar respecto a ello, resulta más costoso, es decir, se necesitan más computadores, mientras que con Big Data no necesariamente ocurre esto, ya que con un sólo computador con mejores especificaciones de memoria, almacenamiento, etc, se podría ejecutar de manera óptima el algoritmo.

Se pudo observar la mejora de un algoritmo distribuido frente a uno paralelo.

REFERENCIAS

- [1] Anna Huang. *Similarity Measures for Text Document Clustering*. Department of Computer Science. The University of Waikato, Hamilton, New Zealand.
- [2] Clustering - RDD-based API - Spark 2.1.1 Documentation. *K-means* Noviembre 17, 2017
<https://spark.apache.org/docs/2.1.1/mllib-clustering.html>
- [3] Apache Spark. *spark/k_means_example.py at master apache/spark*. Noviembre 17, 2017
https://github.com/apache/spark/blob/master/examples/src/main/python/mllib/k_means_example.py
- [4] Apache Spark. *spark/tf_idf_example.py at master apache/spark*. Noviembre 17, 2017
https://github.com/apache/spark/blob/master/examples/src/main/python/mllib/tf_idf_example.py
- [5] Mike Bernico *Using TF-IDF to convert unstructured text to useful features* Febrero 1, 2015
<https://www.youtube.com/watch?v=hXNbFNCgPfY>
- [6] missushiyana *super power - doing k-means clustering using Spark ML library* Diciembre 5, 2015
<https://www.youtube.com/watch?v=Lm1c2U8BmoA>
- [7] Extract, transform and load - Wikipedia, la enciclopedia libre. *Extract, transform and load* Octubre 19, 2017
https://es.wikipedia.org/wiki/Extract,_transform_and_load
- [8] Tf-idf - Wikipedia, la enciclopedia libre. *Tf-idf* Agosto 27, 2017
<https://es.wikipedia.org/wiki/Tf-idf>