Universitat Politècnica de Catalunya

Facultad de Informática de Barcelona

MapReduce

Búsqueda y Análisis de Información Masiva

Autores

Nicolas Llorens

Ricardo Lopez

12/12/2024

Contents

1 Implementación

MRKmeans es una implementación paralelizada del algoritmo de clustering *k-means* utilizando el modelo *MapReduce*. Este algoritmo agrupa datos en k clusters optimizando iterativamente las posiciones de los centroides.

El algoritmo general que utiliza MRKmeans es el siguiente

1. **Asignación a prototipos**: Se asigna cada dato al cluster más cercano según una medida de similitud (como Jaccard en nuestro caso).
2. **Actualización de prototipos**: Se recalculan los prototipos como el promedio ponderado de los datos asignados a cada cluster.

Hemos realziado las siguientes tareas para implementar `MRKmeansSteps`:

- **Definición de funciones clave:** Implementamos los métodos para calcular la similitud de Jaccard y para gestionar la asignación y actualización de clusters en formato MapReduce.
- **Paralelización:** Aprovechamos la biblioteca `mrjob` para distribuir las tareas de cálculo en múltiples núcleos, acelerando significativamente el procesamiento de datos.
- **Automatización de iteraciones:** Finalmente utilizamos el script `MRKmeansSteps` para ejecutar iterativamente, verificando la convergencia de los clusters.

2 Experimentación

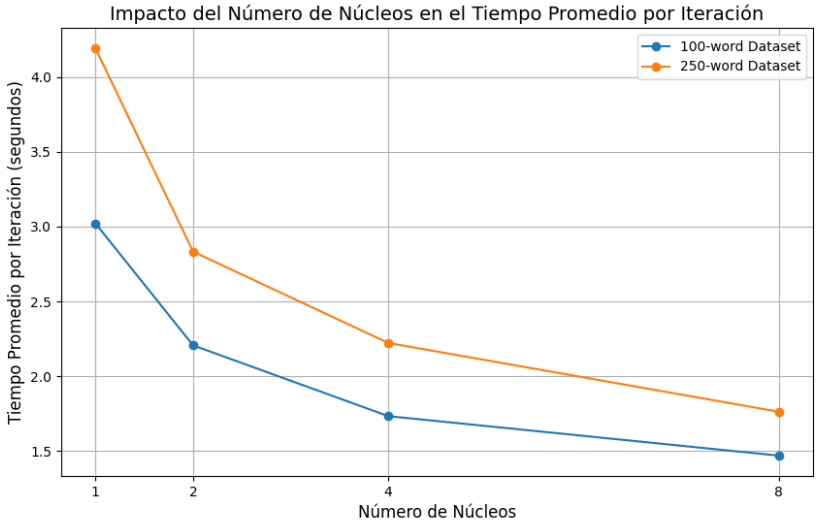
Para evaluar el rendimiento y los resultados de MRKmeans, realizamos dos experimentos principales: el impacto del número de núcleos utilizados y el impacto del número de clusters.

2.1 Impacto del Número de Núcleos

Este experimento medimos cómo afecta la paralelización al tiempo promedio por iteración utilizando los siguientes números de core (1, 2, 4, 8).

2.1.1 Resultados del Tiempo

- **Dataset de 100 palabras:** El tiempo promedio por iteración disminuyó de 3.02 segundos con 1 núcleo a 1.47 segundos con 8 núcleos.
- **Dataset de 250 palabras:** De manera similar, el tiempo promedio pasó de 4.19 segundos a 1.76 segundos al aumentar los núcleos.



La gráfica muestra que el beneficio de paralelización disminuye al aumentar los núcleos.

2.2 Impacto del Número de Clusters

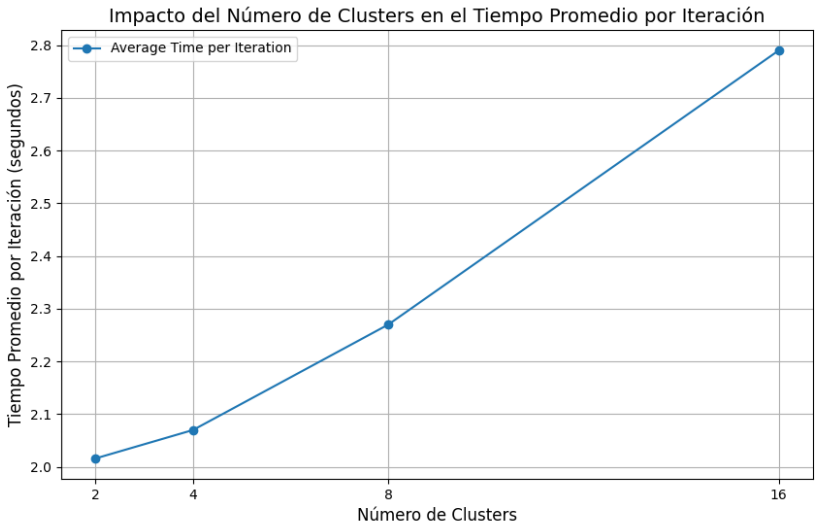
En este experimento evaluamos cómo afecta el número de clusters (k) al tiempo promedio por iteración y también la calidad de los clusters generados.

2.2.1 Resultados del Tiempo

El tiempo promedio por iteración aumenta a medida que crece el número de clusters, como se muestra a continuación:

- **Cluster 2:** Tiempo promedio: 2.01 segundos.
- **Cluster 4:** Tiempo promedio: 2.07 segundos.

- **Cluster 8:** Tiempo promedio: 2.27 segundos.
- **Cluster 16:** Tiempo promedio: 2.79 segundos.



2.2.2 Calidad de Clusters

La calidad de los clusters se evaluó analizando las palabras clave representativas obtenidas de los prototipos finales.

A continuación, se presenta una tabla con los valores clave obtenidos para cada configuración de clusters:

Table 1: Palabras clave constantes y sus pesos representativos en diferentes configuraciones de clusters.

Número de Clusters	Cluster	Palabras Clave y Pesos Representativos
2	CLASS1	propos (0.264), state (0.263), perform (0.260), function (0.259), relat (0.255)
4	CLASS1	propos (0.264), state (0.263), perform (0.260), function (0.259), relat (0.255)
8	CLASS7	propos (0.264), state (0.263), perform (0.260), function (0.259), relat (0.255)
16	CLASS15	propos (0.264), state (0.263), perform (0.260), function (0.259), relat (0.255)

En este caso podemos ver que lso resultados para un diferente número de cluster es le mismo, enetemos que es porque se utiliza el mismo diccionario, es decir la frequencia de las palabaras es la misma para cada cluster. Creemos que en caso de que cambie la frequencia de las palabras al crear el diccionario, los valores que obtenemos por cluster será diferente

Table 2: Palabras clave constantes y sus pesos representativos en diferentes configuraciones de clusters para 100 palabras.

Número de Clusters	Cluster	Palabras Clave y Pesos Representativos
2	CLASS1	also (0.418), studi (0.398), have (0.395), two (0.391), time (0.389)
4	CLASS1	also (0.418), studi (0.398), have (0.395), two (0.391), time (0.389)
8	CLASS7	also (0.418), studi (0.398), have (0.395), two (0.391), time (0.389)
16	CLASS4	also (0.418), studi (0.398), have (0.395), two (0.391), time (0.389)

En este caso hemos modificado los valores de la frecuencia y tambien el tamaño de las palabras de diccionario. Pero podemos ver que producimos el mismo resultado independientemente de los clusters utilizados. Creemos que es por el mismo motivo que antes. Que el prototype tiene el mismo diccionario.

3 Conclusiones

Los experimentos realizados con MRKmeans han permitido evaluar el impacto de diferentes configuraciones en el rendimiento y la calidad de los clusters. A continuación, se resumen las conclusiones principales:

- **Eficiencia del Algoritmo**:
  - Podemos ver que la paralaleizacion disminuye el tiempo de ejecución considerablemente, especialmente al aumentar el numero de cores. Pero también podemos apreciar que a partir de 4 núcleos el tiempo los beneficios son marginales, como lo predice la Ley de Amdahl.
  - A nivel de tiempo también podemos ver que el tiempo aumenta a medida que aumenta el número de clusters
- **Calidad de Clusters**:
  - Con un número reducido de clusters, los resultados son más generales, lo que puede ser útil para análisis de alto nivel. Esto lo podemos ver con los resultados entre 100 y 250, dado que las palabras que obtenemos en el 100 son mucho más generales que no con un diccionario de 250

4 Dificultades

A nivel general creemos que para obtener un mejor resultado deberíamos tener una experimentación mucho mas exhaustiva. Deberiamos comprobar que pasa cuando se modifica la frequencia de las palabras pero no el tamañao del diccionario. Seguramente modificaría considerablemente el resultado final según el número de cluster utilizados.

También utilizar diferentes procesadores para el mismo número de clusters, para averiguar cual es el número mínimo de procesadores en el cual no mejora el tiempo de ejecución del algoritmo.

Una de las dificultades también ha sido entender el algoritmo y poder plasmarlo en el lenguaje.