Laboratorio 01 - Big Data

Map Reduce

Selene Barrios Cornejo

1 Código

La función *mapper* realiza la tarea de contar las ocurrencias de las palabras en un fragmento de texto y devuelve un diccionario que contiene las palabras y sus respectivos recuentos en ese fragmento.

```
def mapper(text):
    word_counts = {}
    words = text.split()
    for word in words:
        if word in word_counts:
            word_counts[word] += 1
        else:
            word_counts[word] = 1
    return word_counts
```

La función *reducer* toma una lista de resultados parciales de los mappers, combina las ocurrencias de las palabras y devuelve un diccionario que contiene las palabras y sus recuentos combinados.

```
def reducer(results):
    word_counts = {}
    for result in results:
        for word, count in result.items():
            if word in word_counts:
                word_counts[word] += count
            else:
                word_counts[word] = count
    return word_counts
```

El volumen de la data se tomará por porciones del 25%, 50% y 100%, para lo cual se utiliza la función *process_file*, la cuál se muestra a continuación:

```
def process_file(file_name, data_percentage):
    input_path = '/home/pejelagarta/Desktop/BIG DATA/MapReduce/Data'
    file_path = os.path.join(input_path, file_name)
    with open(file_path, 'r') as file:
```

```
        data = file.read()

    # Obtener una fraccion del texto para procesar
    data_size = len(data)
    data_to_process = data[:int(data_size * data_percentage)]

    # Dividir los datos en fragmentos para los mappers
    num_threads = 1  # Se utiliza un solo hilo dentro de cada proceso
    fragment_size = len(data_to_process) // num_threads
    fragments = [data_to_process[i:i+fragment_size] for i in range(0, len(
    data_to_process), fragment_size)]

    # Aplicar la funcion mapper a cada fragmento de texto
    mapper_results = map(mapper, fragments)

    return reducer(mapper_results)
```

1.1 Experimentos

El experimento se realizará con threads que serán equivalentes al número de procesadores de mi laptop, así que se usarán 4 threads.

1.2 Resultados

Hilos	% Data	Tiempo
4	0.25	44.2054
4	0.50	74.7615
4	1	114.1977

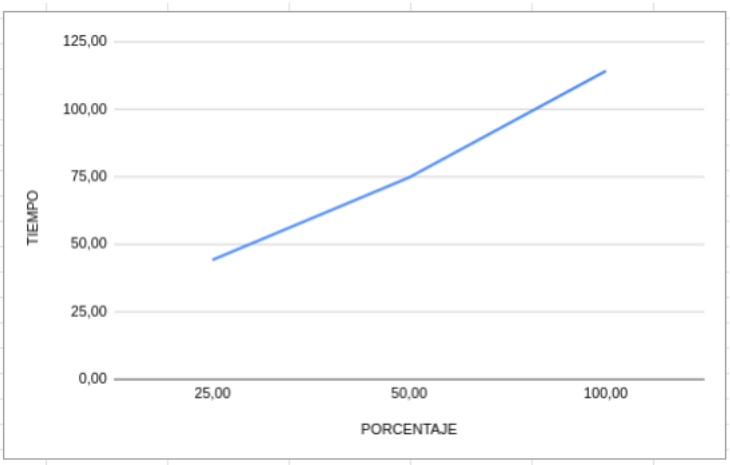

Figure 1: Tiempo frente a porcentaje de data

Cuando se implementa un contador de palabras utilizando el paradigma MapReduce usando 4 hilos, el porcentaje de la data tiene un impacto en el tiempo de procesamiento. A medida que se aumenta el porcentaje de la data, es decir, se procesa una mayor cantidad de texto, el tiempo de procesamiento generalmente aumenta.

Esto ocurre porque, al aumentar el porcentaje de la data, se procesa más información en paralelo utilizando los 4 hilos disponibles. Sin embargo, también se incrementa la cantidad total de datos que deben ser procesados, lo que implica una mayor carga de trabajo para los hilos y puede aumentar el tiempo total de ejecución.

Por otro lado, si se reduce el porcentaje de la data, se procesará menos información en paralelo, lo que puede disminuir el tiempo de procesamiento. Sin embargo, es importante tener en cuenta que si el porcentaje de la data es demasiado bajo, el impacto en el tiempo de procesamiento puede no ser significativo, ya que el tiempo necesario para la configuración y coordinación de los hilos puede ser más relevante en comparación con el tiempo de procesamiento real.

2 Conclusiones

Aumentar el porcentaje de la data puede incrementar el tiempo de procesamiento debido a la mayor carga de trabajo, mientras que reducir el porcentaje de la data puede disminuir el tiempo de procesamiento, pero hay un punto óptimo donde se equilibra el tamaño de la data y el rendimiento del procesamiento paralelo. Este equilibrio puede variar según el hardware, la implementación específica y las características del problema.