Informe Proyecto Estrategia Mappers-Reducer

PRESENTADO A: ING. MARIELA CURIEL

PRESENTADO POR:

NICOLAS CAMACHO
(nicolas-camacho@javeriana.edu.co)

JHONNIER CORONADO
(jhonnier.coronado@javeriana.edu.co)

BRAYAN GONZÁLEZ
(gonzalez-b@javeriana.edu.co)

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
SISTEMAS OPERATIVOS
BOGOTÁ D.C.
2019

Proyecto

Comparar la eficiencia del algoritmo MapReduce de manera concurrente por medio de tres implementaciones diferentes: procesos que se comunican con archivos, hilos que se comunican con buffers, e hilos productor/consumidor mediante mediante la implementación de semáforos. La comparación y la elección de diseño se basa en fundamentos teóricos y en comparar los resultados y desempeño de cada implementación con varios tamaños de logs.

Opciones a evaluar:

Procesos:
Es un programa en ejecución que contiene la actividad que está realizando actualmente la CPU y una información que es almacenada en el PC y en los registros, esto se conoce como: "Recursos" y "Planificación/Ejecución", que son un espacio de direcciones y datos protegidas por el SOP y una ruta de ejecución respectivamente. Cada proceso tiene un área de datos, un código y una pila. Toda esta información se llama el contexto y no es posible compartirla entre procesos, por este motivo la comunicación entre procesos existe mediante la creación, escritura y eliminación de archivos, es decir operaciones de Entrada/Salida que resultan en llamadas al sistema y cambios de contexto.

Hilos:
Existe la posibilidad de que en un SOP se traten los "Recursos" y la "Planificación/Ejecución" como dos propiedades independientes; a la unidad activa se le denomina hilo, mientras que a el componente propietario de los recursos se le llama tarea o proceso. Bajo este contexto, los procesos están asociados a un espacio de direcciones y cuentan con un acceso protegido a E/S, otros procesos, etc. Por otro lado, el hilo está asociado a un contexto, espacio de almacenamiento de datos locales, acceso a recursos y un estado de ejecución que comparten entre sí un contexto si estamos en un ambiente de multihilos. De esta forma, es posible que los hilos se comuniquen a través de la memoria principal sin operaciones E/S.

Hilos con Semáforos Productor/Consumidor:
Esta implementación utiliza hilos y realiza la creación de los mappers y reducers al tiempo que trabajen simultáneamente, esto conlleva problemas de concurrencia que son frenados mediante la implementación de Semáforos Productor/Consumidor. Estos últimos protegen la integridad de los datos para evitar condiciones de carrera y que los hilos alteren datos que otros están utilizando además de garantizar que las tareas se ejecuten en orden; el master genera una consulta, los mappers la ven y guardan coincidencias, los reducers cuentan coincidencias y producen un output que el master lee. Se utiliza espacios de memoria específicos creados entre Master-Mappers (logs), Mappers-Reducers (Buffers) y Reducers-Master (output) que son regulados por semáforos para evitar condiciones de carrera, las cuales son un acceso a un dato compartido que se puede modificar.

Elección final:
Se escogió la implementación de hilos concurrentes a través de semáforos porque la utilización de hilos es favorable en comparación a la de los procesos por el cambio de

contexto. Si se está trabajando con un número grande de Workers (Mappers y/o Reducers) va a existir una gran cantidad de operaciones de comunicación que requeriría de cierto overhead en la creación y administración de los pipes, mientras que en el caso de hilos, la lógica es más simple, simplemente se realiza sem_wait o sem_post. Además de esto, y la razón que más influyó en la decisión es que la implementación no implicaba muchos cambios con respecto a la de hilos sin semáforos, lo que implicaba un menor costo en tiempos y menor dificultad para lograr un desarrollo exitoso, además que el problema se podía dividir en problemas similares más pequeños y de una implementación más intuitiva; El master utilizaba un buffer para la consulta mediada por un semáforo, los buffers que comunican a los mappers y reducers generan una situación de productor consumidor, y los outputs que generaban los reducers con respecto a el master también generaba una situación idéntica de productor consumidor.

Funcionamiento del Programa:

Para la comunicación entre Mappers y Reducers se utilizó el mismo número de buffers que de Mappers. Mediante una lógica similar a la que se utilizó para distribuir los logs a los Mappers, se le distribuían los buffers a los productores.

La asignación de buffers para los reducers se da mediante una división entera entre el número de mappers y el número de reducers, siendo el resultado entero de esta división el número de buffers que va a tomar el reducer. Debido a que es una división de enteros se puede presentar un residuo de la división el cual se sumará al número de buffers que tiene el último reducer, por consiguiente el último reducer por lo general tendrá mayor cantidad de buffers asignados.

Para evitar que haya condiciones de carrera y corrupción de datos que se comparten por los hilos al acceder al mismo tiempo, se plantea el uso de semáforos; para ello utilizamos 8 vectores de semáforos. 3 de los vectores tendrán la función de controlar la producción el consumo y la exclusión del buffer para cada mapper. Otros 3 cumpliran con la regulacion de la producción, exclusión y consumo del output que es otra región crítica en el reducer y los últimos 2 vectores son del máster para evitar que el mapper y reducer trabajen sin haber siquiera iniciado una consulta.

La escritura de los logs entre Master y Mapper se dará mediante una división del número de líneas entre la cantidad de Mappers, el cual resultara como un intervalo en el archivo para cada Mapper. El Mapper leerá desde el inicio de su intervalo hasta el final de su intervalo, cada mapper utilizara sus semáforos correspondientes del vector de semáforos de mapper para producir y mandarlo al buffer correspondiente.

La escritura del buffer entre Mapper y Reducer se da mediante una búsqueda de los logs que cumplan la condición de consulta ingresada en el programa mediante el uso de los semáforos correspondientes al buffer en modo consumidor. Esto dará como resultado un conjunto de datos definido por el número de log y el valor que satisface la condición.

La escritura de los outputs entre Reducer y Master se da mediante un conteo de la cantidad de datos guardados en los buffers correspondientes a cada Mapper. El reducer utilizará el conjunto de semáforos designado para el output con el cual producirá el conteo y evitará que se corrompa los datos ya que solo habrá un reducer cambiando el output a la vez,

Al finalizar el Master me dará el tiempo en el que se demoró todo este proceso junto con el resultado de la consulta.

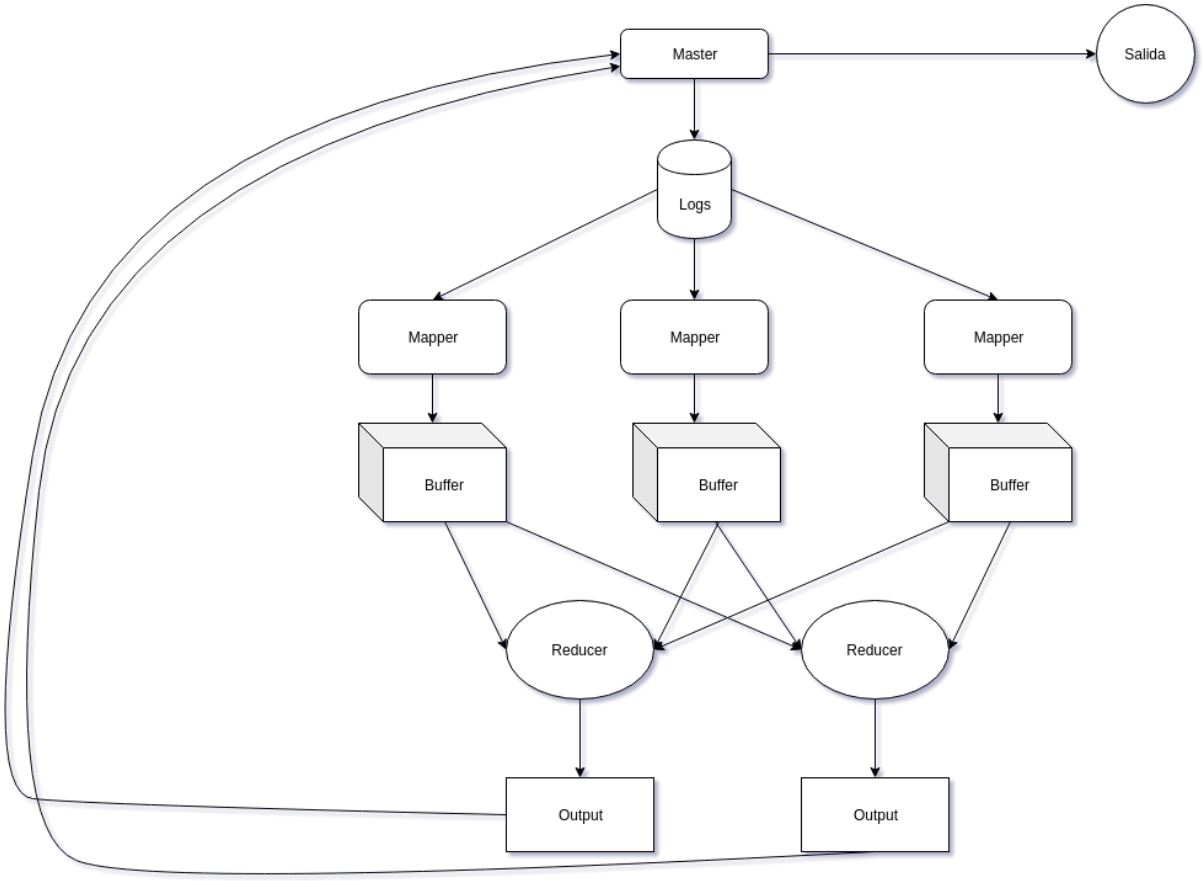

Figura 1: Diagrama de flujo del algoritmo Mapper-Reducer

Desempeño:

Las pruebas se realizaron utilizando la consulta “5,>,1” que muestra el total de Jobs que utilizaron dos o más procesadores en su ejecución. Esta consulta se utilizó en todos los casos pero variando los archivos con los logs. En total se utilizaron cinco archivos con 1000, 10000, 20000, 30000 y 40000 registros en los archivos.
Las comparaciones se realizaron entre la implementación de procesos que se comunican

mediante archivos, hilos, e hilos con semáforos Productor/Consumidor.

Tabla 1: Toma de datos

	1000 Registros	10000 Registros	20000 Registros	30000 Registros	40000 Registros
Procesos	2357	3732	4788	5237	6654
Hilos	557	823	1007	1110	1186
Hilos con semáforos	189	224	335	553	627

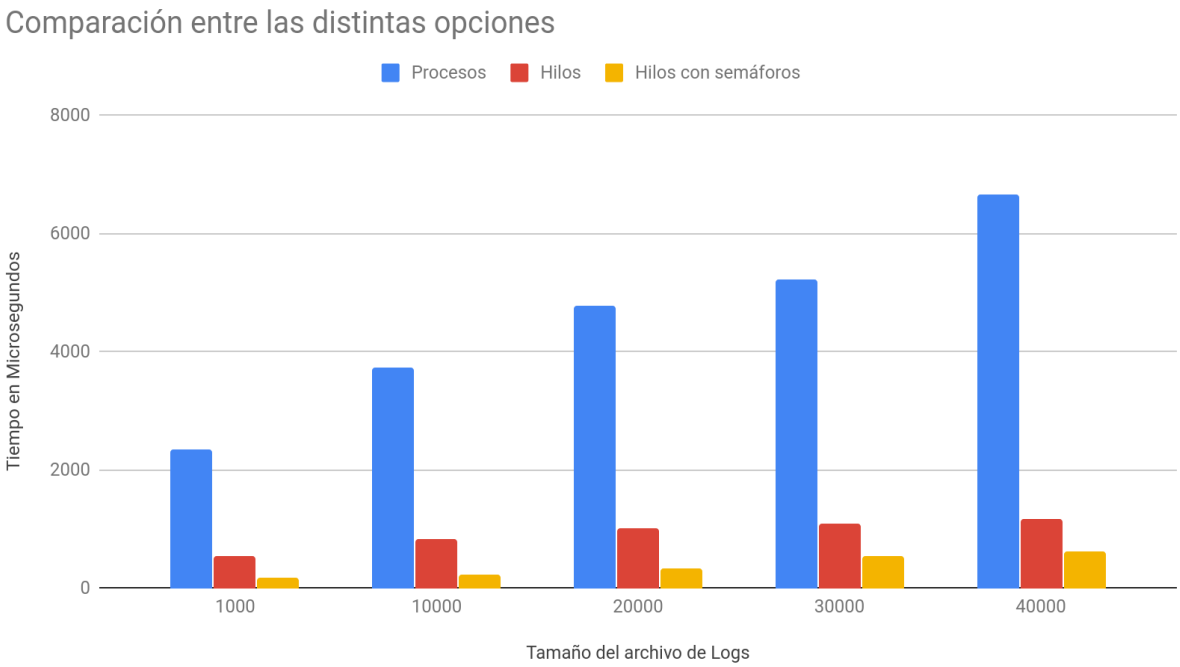

Figura 2: Comparación de tiempos.

Como se puede evidenciar en la tabla y gráfica, es evidente la diferencia gigantesca entre las distintas implementaciones. Resultó que el más eficiente fue la implementación de MapReduce por medio de hilos concurrentes con semáforos aplicando el algoritmo de productor/consumidor. Esto se debe a lo que se mencionó anteriormente, los hilos no requieren un cambio de contexto ni de llamadas al sistema para comunicarse, y siendo este un proceso que se basa en comunicación , la diferencia es impresionante.

Por otro lado, como en la implementación de semáforos, todos los workers se crean una vez y siempre están trabajando, la concurrencia aumenta ya que el máster no debe estar creando los hilos cada vez que vayan a trabajar y del mismo modo no debe esperarlos, además de que los reducers pueden empezar a trabajar tan pronto como sus Mappers asignados produzcan un

buffer. Todo esto lleva a que se aumente la concurrencia y por lo tanto sus ventajas que se ven evidenciadas al comparar hilos e hilos con semáforos.

En la siguiente gráfica, se encuentra la optimización de Hilos e Hilos con semáforos tomando como referencia el tiempo que se demora en la ejecución del proceso map-reduce con diferentes tamaños de log en Procesos.

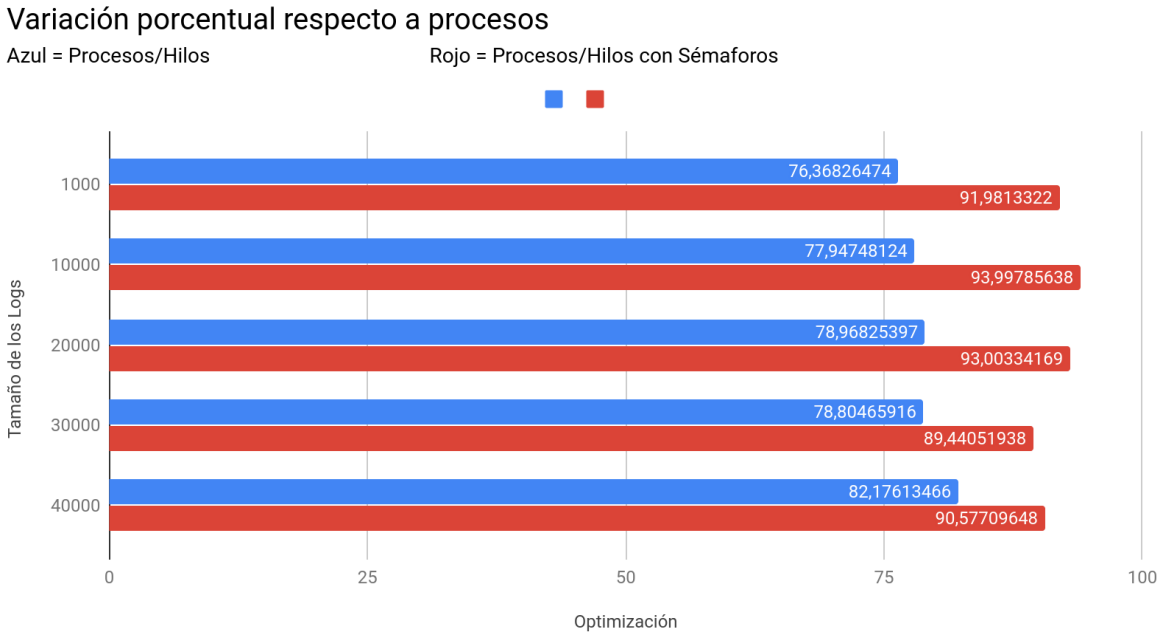

Figura 3: Optimización