Pontificia Universidad Javeriana

Asignatura:

Sistemas Operativos

Trabajo:

Análisis concurrente de un log usando la estrategia MapReduce

Estudiantes:

Alejandro Mayorga

Nicolás Miranda

Bogotá D.C. Mayo 23 de 2019

Objetivo general

Comparar el desempeño del programa buscador de registros en la implementación mediante hilos con la implementación utilizando procesos y finalizar comparando con la implementación de procesos usando pipes en vez de archivos.

El propósito es determinar cuál de las 3 es la mejor opción en términos de rendimiento a medida que el problema escala a proporciones mayores.

Objetivos específicos

1. Poner en práctica los conceptos teóricos vistos en la asignatura.
2. Contrastar la teoría con la práctica, mediante diferentes escenario de prueba que constate que una implementación es más eficiente que otra.
3. Mostrar los elementos más esencial del diseño del proyecto.
4. Justificar con argumento técnicos la implementación, presentando cuál es la mejor opción.

Decisiones de diseño y estructura

Debido a que uno de los objetivos específicos de este proyecto es aprender a usar los pipes para el traspaso de información entre procesos, a continuación se explicará la estructura básica que tiene nuestra implementación:

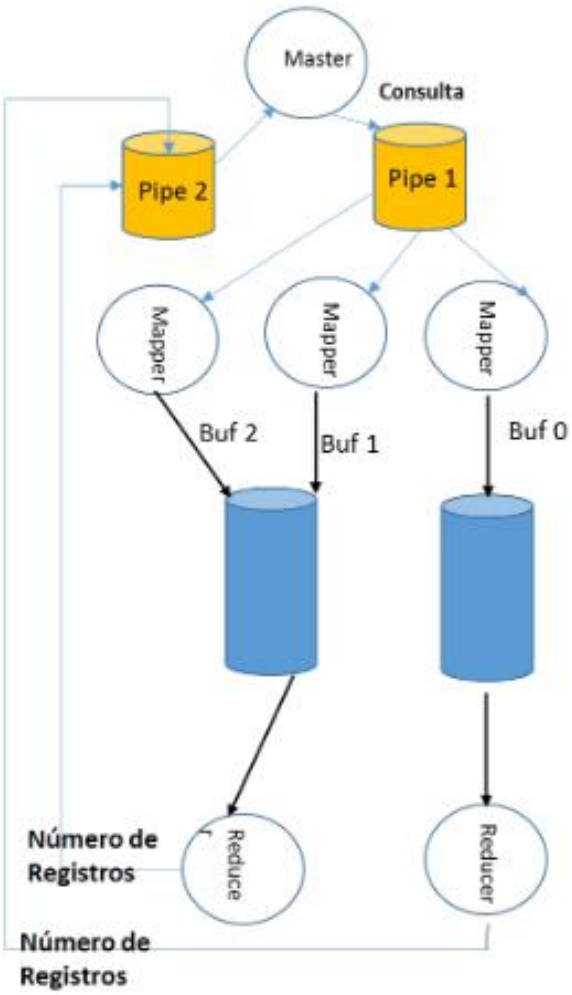

Figura 0. estructura de la implementación.

Como se puede ver en la figura 0, nuestra decisión fue conectar el proceso master con los mappers y reducers mediante un único pipe no nominal para cada uno, de esta forma nos ahorramos inconvenientes al conectar de forma simple el proceso padre con los hijos.

La conexión entre los procesos mappers y reducers se realizó mediante pipes nominales, donde los nombres son entregados por el proceso padre, dotando de esta forma a cada proceso reducer con su pipe personal al cual cada mapper escribirá tal como se ve en la parte inferior de la figura 0.

El flujo de información ocurre de la siguiente forma:

1. El proceso master recibe la información de la consulta deseada.
2. El proceso master envía la consulta y los rangos de operación en la matriz mediante el pipe de los mappers y, manda una señal para indicarle a cada uno que se le fue enviada información.

3. El proceso maper se activa y recibe la información, realiza sus cálculos y envía al reducer correspondiente, mediante el pipe correspondiente, los resultados.
4. El proceso reducer recibe la información de los diferentes mappers, realiza sus cálculos pertinentes y envía los resultados por el pipe no nominal al proceso master.
5. El proceso master recibe la información, realiza sus cálculos pertinentes y los muestra en pantalla.

Forma en la que se envio la informacion

Para el pipe no nominal que conecta al master con los mappers, lo que se envía a cada uno es simplemente una estructura que contiene los rangos de operación y la consulta a trabajar.

Para los pipes nominales que conectan a los mappers con los reducers lo que se envía por cada uno es una estructura que contiene la llave y el valor encontrado, por lo que se necesitan mandar n elementos por el pipe, siendo n, el número de registros que concuerden con lo requerido en la consulta.

Para el pipe no nominal que conecta al reducer con el master, lo que se manda es un número que representa el número de registros que esté encontró.

Escenario de prueba

1. **Condiciones de la consulta:**
    a. Número de Jobs que usaron 2 o más procesadores en su ejecución: *5 , >, 1*
    b. Mappers: *6*
    c. Reducers: *2*
    d. Valor intermedio: *0* (para las soluciones con procesos)

2. **Especificaciones de hardware:**
    Las pruebas fueron realizadas en los computadores de la sala de bases de datos de la Universidad Javeriana y cuentan con las siguientes características:
    a. Sistema operativo: *Ubuntu 18.04.2 LTS*
    b. Memoria: *7.7 GiB*
    c. Procesador: *Intel® Core™ i7-6700T CPU @ 2.80GHz × 8*
    d. Gráficos: *Intel® HD Graphics 530 (Skylake GT2)*
    e. Gnome: *3.28.2*
    f. Tipo de SO: *64 bits*

g. Disco: *102.8 GB*

Con respecto a las aplicaciones que estaban consumiendo recursos de memoria y podían interferir de cierta manera en los resultados del tiempo en cada consulta, se tenían abiertas 3 instancias de la terminal, una para cada prueba (con procesos y archivos, con hilos y con procesos y pipes) y también, se tenían 2 instancias de Google Docs en Mozilla Firefox para ir registrando los resultados en el informe.

Tabla tiempo total de ejecución de cada implementación

	10000 registros	**20000 registros**	**30000 registros**	**40000 registros**
Procesos a través de archivos (µseg)	6.376	6.250	7.625	9.190
Hilos (µseg)	1.455	1.720	1.950	2.098
Procesos a través de pipes (µseg)	6.895	8.806	10.003	12.044

Tabla 1. Registro de los valores con las 3 implementaciones

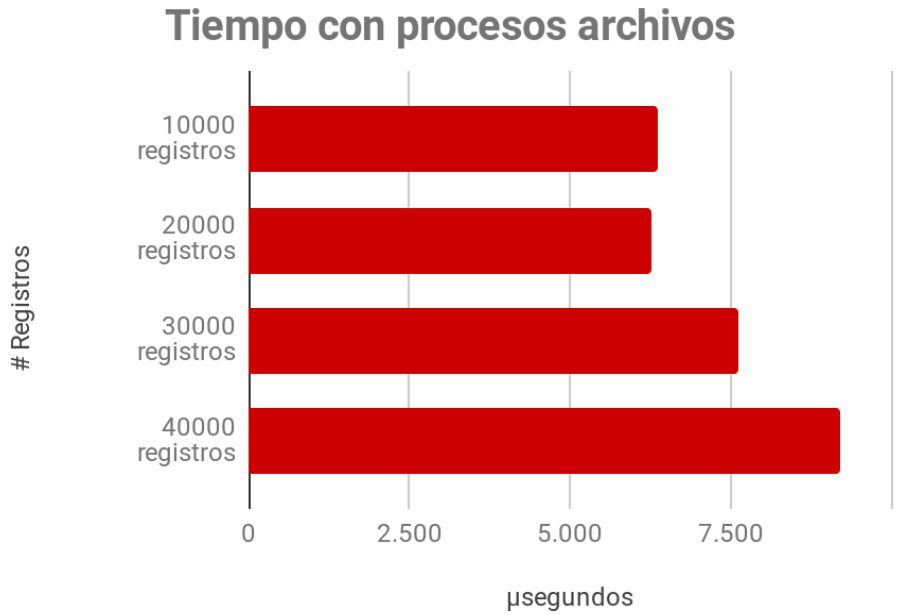

Figura 1. Tiempo con procesos archivos

Figura 2. Tiempo con hilos

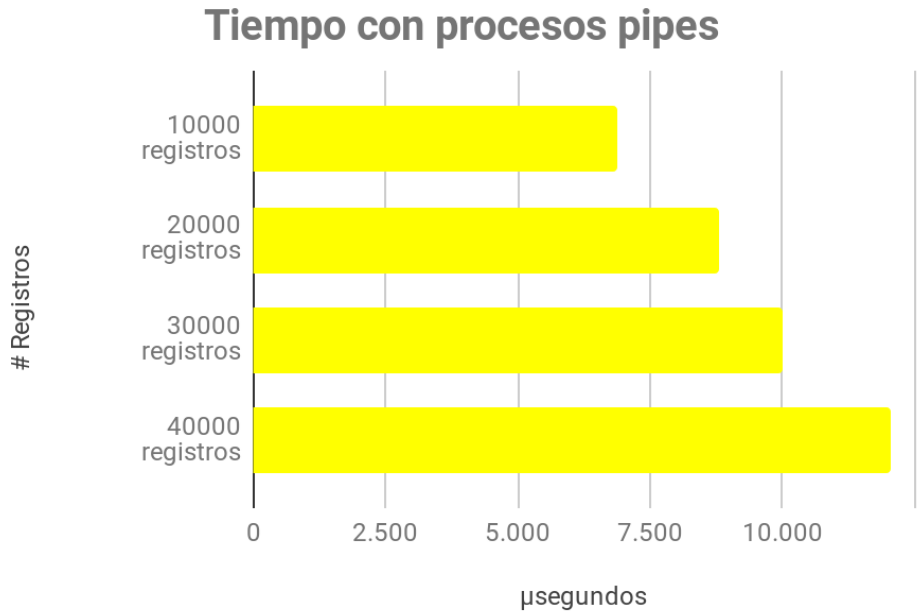

Figura 3. Tiempo con procesos archivos

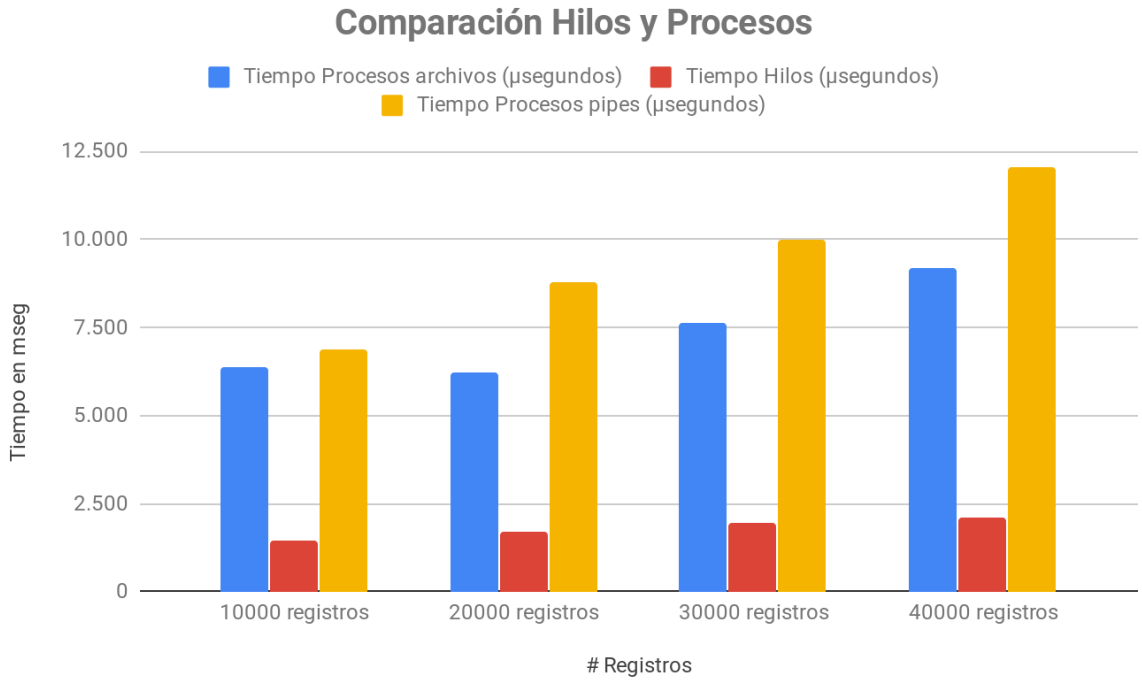

Figura 4. Comparación tiempo de hilos y procesos (archivos y pipes)

3. Análisis:

Como se observa en la figura 4, donde se compara el desempeño entre los hilos y los procesos usando archivos y usando pipes, se puede evidenciar

claramente que el que tiene el mejor tiempo son los hilos. Lo anterior se debe a múltiples factores, entre ellos que los hilos comparten un solo espacio de direccionamiento en memoria, cada uno de los hilos se ejecuta de forma secuencial y maneja su propio contador de programa y pila [1]. Agregando a lo anterior, los hilos ganan rendimiento debido a que no tienen que reemplazar el PCB activo cuando intercalan la ejecución de sus diferentes hilos [1] y finalmente, son más rápidos porque comparten espacio de memoria sin tener que establecerlo a través de mecanismos de comunicación entre procesos [1]. Ahora, comparando los procesos con archivos versus los procesos mediante pipes, se puede evidenciar que hay una ligera entre ellos y el que se demora un poco más son los procesos que se comunican mediante pipes, esto se debe a que se tiene que crear que los pipes (un archivo especial) que se almacena en el buffer hasta que alguien lo lea y que adicionalmente a esto, se debe tener un mecanismo de sincronización porque la salida de cada elemento de la cadena es la entrada del otro, esto hace que sea un poco más lento que los procesos mediante archivos.

4. Conclusiones:

En conclusión, el uso procesos e hilos es muy importante en los temas de sistemas operativos porque realizan operaciones a este nivel y ayuda a los desarrolladores a programar de una mejor forma usando mejor la memoria y previendo errores de seguridad que pueden ser letales como el desbordamiento del buffer. Hablando del rendimiento de cada uno, es relativo porque depende de lo que queramos realizar, en la mayoría de casos el uso de hilos es mejor que el de procesos, porque se comparte la memoria y permite la comunicación de una manera más eficiente, sin embargo, en otros casos puede que no tenga tan buen rendimiento. Por otro lado, la programación y el uso de ciertas estructuras y tipos de datos, hace que el programa se comporte de una manera no deseada o no tenga el rendimiento que se desea.

5. Referencias:

[1]. G. Wolf, *Fundamentos de sistemas operativos*, 1st ed. Ciudad de México: Universidad Autónoma de México, 2015, pp. 72-80.