REVISTA DE LA ESCUELA PROFESIONAL DE CIENCIA DE LA COMPUTACIÓN
UNIVERSIDAD NACIONAL DE SAN AGUSTIN DE AREQUIPA
VOL. 1, NO. 2, NOVIEMBRE 2024

Laboratorio del curso de Computación Paralela y Distribuida: Shared-Memory Programming with Pthreads - Programming assingments

Eduardo G. Ruiz[1]

[1] *Escuela de Ciencias de la Computación, Facultad de Ingeniería de Procesos, Universidad Nacional de San Agustín, Arequipa, Arequipa, Perú*

Resumen—En el siguiente laboratorio se resolvieron ejercicios propuestos del libro aplicando lo aprendido en el capítulo 4 a través de ejercicios de implementación, se entiende el funcionamiento de funciones como mutex, la creación de hilos y variables de condición, se analiza el comportamiento dentro de programas que resuelven ejercicios comunes como el poblema de Monte Carlo o el de escritores y lectores, llegando a la conclusión que la cantidad de hilos y la cantidad de ejecuciones afectan el rendimiento del programa, así como el orden en el que se llaman a los bloqueos dentro de los hilos. El poder analizar mejor las consecuencias de las variaciones de estos parámetros puede aplicarse mejor en futuros programas.

Palabras clave—Pthreads, mutex, bloqueos, prioridad, condiciones de carrera, problema de lectores y escritores

Abstract—In the following laboratory, proposed exercises from the book were solved by applying what was learned in chapter 4 through implementation exercises, the operation of functions such as mutex, the creation of threads and condition variables is understood, the behavior within programs that solve common exercises such as the Monte Carlo problem or the writers and readers problem is analyzed, reaching the conclusion that the number of threads and the number of executions affect the performance of the program, as well as the order in which the locks are called within the threads. Being able to better analyze the consequences of the variations of these parameters can be better applied in future programs.

Keywords—Pthreads, mutex, locks, priority, race conditions, reader-writer problem

OBJETIVO

El objetivo de este laboratorio es aprender sobre el uso de algunas funciones de la biblioteca Pthreads, a través de la implementación de diferentes programas que resuelven problemas comunes de la realidad, como el ingreso de datos a un histograma, el cálculo de pi mediante el método de Monte Carlo y el problema de lectores y escritores.

Este laboratorio permite el entendimiento de conceptos clave referentes a la biblioteca Pthreads, así como comprender dónde aplicar las alternativas que ofrece, en qué casos es mejor usar alguna estrategia o no y la aplicación de los conceptos explicados en el capítulo 4 del libro de Peter Pacheco. La implementación y ejecución de los ejercicios se encuentra en `https://github.com/EGRM23/CPD-2024/tree/main/Pthreads`

EJERCICIOS

Los ejercicios propuestos para el siguiente laboratorio son 3, a continuación se detallará las especificaciones dde cada uno.

El ejercicio 4.1 indica escribir un programa Pthreads que implemente el programa del histograma en el Capítulo 2. El programa base consiste en un programa serial que genera un histograma, se necesita decidir cuales serán los rangos de los contenedores, cuántos números contendrá cada uno e imprimir las barras del histograma. Como no está direccionado a una interfaz de entrad y salida, el resultado será el número de elementos que tiene cada contenedor.

EL ejercicio 4.2 indica suponer que se lanza dardos al azar a un tablero cuadrado, cuyo centro está en el origen y cuyos lados miden 2 pies de largo. Se supone también que hay un círculo inscrito en el tablero cuadrado. El radio del círculo es de 1 pie y su área es de π pies cuadrados. Si los puntos que son alcanzados por los dardos están distribuidos uniformemente (y siempre se alcanza el cuadrado), entonces el número de dardos que alcanzan el interior del círculo debería satisfacer aproximadamente la ecuación

$$\frac{\text{dardos en el círculo}}{\text{número total de dardos}} = \frac{\pi}{4}$$

ya que la relación entre el área del círculo y el área del cuadrado es $\frac{\pi}{4}$. Adicional a eso propone una fórmula para esti-

mar el valor de π mediante un generador de números aleatorios. Este método se denomina "Monte Carlo", ya que utiliza la aleatoriedad (los lanzamientos de dardos). El ejercicio indica escribir un programa Pthreads que utilice un método Monte Carlo para estimar π. El hilo principal debe leer el número total de lanzamientos e imprimir la estimación.

Por último, el ejercicio 4.6 indica escribir un programa Pthreads que utilice dos variables de condición y un mutex para implementar un bloqueo de lectura y escritura. Indica descargar un programa de lista enlazada en línea que utiliza bloqueos de lectura y escritura de Pthreads y modifícarlo para utilizar los bloqueos de lectura y escritura propios. Después comparar el rendimiento del programa cuando se les da preferencia a los lectores con el del programa cuando se les da preferencia a los escritores. Y pregunta si se puede hacer alguna generalización respecto a los resultados.

METODOLOGÍA

Ejercicio 4.1

El ejercicio 4.1 se resolvió en base a 5 contenedores en el histograma, 4 hilos de ejecución, 22 datos a clasificar (preestablecidos en el mismo programa) de tipo flotante con un mínimo de 0.0 y un máximo de 5.0, y un mutex. Adicional a esto se declaró un struct que junta los argumentos que se pasarán a la función que ejecutarán los hilos, los argumentos que incluye son el id del hilo, el índice de inicio y el índice del final (en otras palabras el rango de índices dentro de los datos que analizará el hilo). Por otro lado también hay 2 funciones que se usan durante la ejecución del programa, una llamada find_bin_index, la cual devuelve el número de contenedor al que pertenece un valor determinado, y la otra función llamada compute_histogram, la función que ejecutarán los hilos, recibe el struct de argumentos y procede a iterar entre los índices que le toca evaluar, clasifica los valores usando la otra función y usa un mutex para aumentar el número de datos en los contenedores del histograma.

El flujo del programa comienza inicializando un arreglo de hilos y un arreglo de argumentos de hilo, calcula el número de elementos que analizará cada hilo e inicializa el mutex. Luego procede a iterar entre el número de hilos, asignando los argumentos al struct que le corresponde a cada uno, después crea el hilo con sus argumentos y con la función que ejecutará. Finalmente espera que todos los hilos terminen su ejecución e imprime el histograma, la impresión muestra el número de valores por contenedor.

Ejercicio 4.2

El ejercicio 4.2 se resolvió haciendo uso de 4 variables globales: el total de lanzamientos, el total de lanzamientos que dieron en el círculo, el número de hilos y un mutex; además de la función que ejecutarán los hilos, llamada monte_carlo, la cuál recibe un número de lanzamientos y por cada uno cálcula una coordenada aleatoria dentro del área del cuadrado (el cuál mide 2 unidades por lado).

El flujo del programa comienza recibiendo el número de lanzamientos a probar con el número de hilos que los ejecutarán, crea un arreglo de hilos, calcula el número de elementos por hilo e inicializa el mutex. Luego procede crear los hilos y ejecutar la función monte_carlo en cada uno, espera que todos los hilos terminen y calcula el valor de π multiplicando el número de dardos en el círculo por 4 y diviéndolo entre el total de lanzamientos, finalmente imprime el valor hallado. La pruebas se realizaron sobre una base de un millón de lanzamientos distribuidos en 10 hilos.

Ejercicio 4.6

El ejercicio 4.6 se resolvió en 3 fases, esto debido a las especificaciones que tiene.

Primero se hizo el programa Pthreads que usa 2 variables de condición y un mutex, este programa se llama ejer4.6-idea.c, consta de 4 funciones de inicio de lectura, fin de lectura, inicio de escritura y fin de escritura, cada una usa el mutex cuando se invocan y tienen diferentes lógicas de acuerdo a su propósito, como la función de inicio de lectura que antes de leer verifica si hay alguien escribiendo, sino usa la condición de lectores, lo mismo para otras funciones, este programa está orientado a dar preferencia a los escritores.

La segunda fase se refiere al programa de lista enlazada que usa rwlock para organizar sus permisos, como decía el problema este programa fue descargado de internet, del siguiente repositorio `https://github.com/Rajind/LinkedLists-in-C/blob/master/linked_list_one_rwlock.c`, sigue la lógica de una lista enlazada convencional, y para organizar las operaciones entre hilos usa el rwlock. Este programa se encuentra en el repositorio con el nombre de ejer4.6-base.c.

La tercera fase consistió en usar el programa Pthreads de la primera fase en el programa de la segunda fase en ves del uso de rwlock, se añadió las funciones de inicio de lectura, fin de lectura, inicio de escritura y fin de escritura en el programa base juntos con las varibles que usa, luego se cambió el contenido de la función Thread_Function quitando las llamadas a rwlock y en ves de eso usar el mutex propio junto con la llamada a las funciones de inicio y fin de lectura y escritura según sea el caso. Sin embargo, debido a las especificaciones del ejercicio, hay 2 versiones finales, un programa que da prioridad a los escritores y otro que da prioridad a los lectores, las únicas diferencias entre ambos, son en las condiciones del bucle while de la función de inicio de lectura y en las condiciones del bucle while de la función de fin de escritura, estas condiciones hace que varíe el comportamiento. Las pruebas se realizaron sobre una base de 500 nodos en la lista enlazada y 10000 operaciones, donde había un 65 por ciento de probabilidad que la operación fuera de lectura y el resto de escritura.

RESULTADOS

Fig. 1: Ejecución del ejercicio 4.1

```
egrm23@egrm23-Victus-by-HP-Gaming-Laptop-15-fb2xxx:/media/egrm23/Datos/UNSA/CIENC
IAS DE LA COMPUTACION/8vo semestre/CPD-2024/Pthreads$ ./ejer4.2 1000000 10
Estimación de π con 1000000 lanzamientos y 10 hilos: 3.140304088592529
egrm23@egrm23-Victus-by-HP-Gaming-Laptop-15-fb2xxx:/media/egrm23/Datos/UNSA/CIENC
IAS DE LA COMPUTACION/8vo semestre/CPD-2024/Pthreads$ ./ejer4.2 10000000 10
Estimación de π con 10000000 lanzamientos y 10 hilos: 3.141127586364746
egrm23@egrm23-Victus-by-HP-Gaming-Laptop-15-fb2xxx:/media/egrm23/Datos/UNSA/CIENC
IAS DE LA COMPUTACION/8vo semestre/CPD-2024/Pthreads$ ./ejer4.2 100000000 10
Estimación de π con 100000000 lanzamientos y 10 hilos: 3.141483068466187
```

Fig. 2: jecución del ejercicio 4.2

```
IAS DE LA COMPUTACION/8vo semestre/CPD-2024/Pthreads$ ./ejer4.6-writers 2
0.0390750000
egrm23@egrm23-Victus-by-HP-Gaming-Laptop-15-fb2xxx:/media/egrm23/Datos/UNSA/CIENC
IAS DE LA COMPUTACION/8vo semestre/CPD-2024/Pthreads$ ./ejer4.6-writers 4
0.0572380000
egrm23@egrm23-Victus-by-HP-Gaming-Laptop-15-fb2xxx:/media/egrm23/Datos/UNSA/CIENC
IAS DE LA COMPUTACION/8vo semestre/CPD-2024/Pthreads$ ./ejer4.6-writers 6
0.0676030000
egrm23@egrm23-Victus-by-HP-Gaming-Laptop-15-fb2xxx:/media/egrm23/Datos/UNSA/CIENC
IAS DE LA COMPUTACION/8vo semestre/CPD-2024/Pthreads$ ./ejer4.6-writers 8
0.0758460000
egrm23@egrm23-Victus-by-HP-Gaming-Laptop-15-fb2xxx:/media/egrm23/Datos/UNSA/CIENC
IAS DE LA COMPUTACION/8vo semestre/CPD-2024/Pthreads$ ./ejer4.6-writers 10
0.0863280000
egrm23@egrm23-Victus-by-HP-Gaming-Laptop-15-fb2xxx:/media/egrm23/Datos/UNSA/CIENC
IAS DE LA COMPUTACION/8vo semestre/CPD-2024/Pthreads$ ./ejer4.6-writers 12
0.1095360000
```

Fig. 3: Ejecución del ejercicio 4.6, preferencia a writers

```
IAS DE LA COMPUTACION/8vo semestre/CPD-2024/Pthreads$ ./ejer4.6-readers 2
0.0432750000
egrm23@egrm23-Victus-by-HP-Gaming-Laptop-15-fb2xxx:/media/egrm23/Datos/UNSA/CIENC
IAS DE LA COMPUTACION/8vo semestre/CPD-2024/Pthreads$ ./ejer4.6-readers 4
0.0680760000
egrm23@egrm23-Victus-by-HP-Gaming-Laptop-15-fb2xxx:/media/egrm23/Datos/UNSA/CIENC
IAS DE LA COMPUTACION/8vo semestre/CPD-2024/Pthreads$ ./ejer4.6-readers 6
0.0734600000
egrm23@egrm23-Victus-by-HP-Gaming-Laptop-15-fb2xxx:/media/egrm23/Datos/UNSA/CIENC
IAS DE LA COMPUTACION/8vo semestre/CPD-2024/Pthreads$ ./ejer4.6-readers 8
0.0747130000
egrm23@egrm23-Victus-by-HP-Gaming-Laptop-15-fb2xxx:/media/egrm23/Datos/UNSA/CIENC
IAS DE LA COMPUTACION/8vo semestre/CPD-2024/Pthreads$ ./ejer4.6-readers 10
0.0755800000
egrm23@egrm23-Victus-by-HP-Gaming-Laptop-15-fb2xxx:/media/egrm23/Datos/UNSA/CIENC
IAS DE LA COMPUTACION/8vo semestre/CPD-2024/Pthreads$ ./ejer4.6-readers 12
0.0784070000
```

Fig. 4: Ejecución del ejercicio 4.6, preferencia a readers

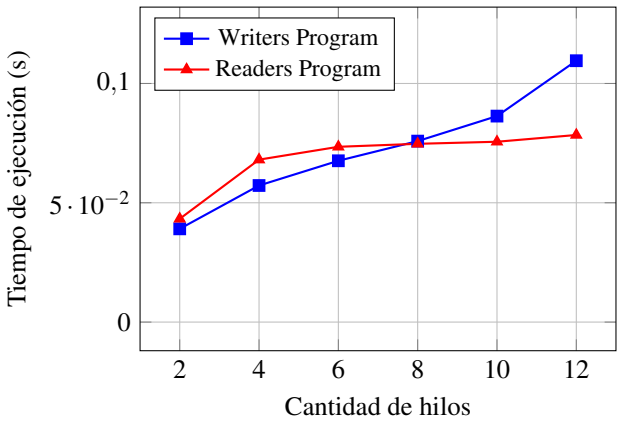

Fig. 5: Comparación del tiempo de ejecución para Writers y Readers Program

ANÁLISIS

En el ejercicio 4.1 se puede observar que el resultado es correcto, no hay mayor lógica, el uso de mutex para evitar errores en la ejecución de las funciones dentro de cada hilo hizo que el programa no se cuelgue, sin embargo, se podría mejorar el programa para que pueda clasificar muchos más datos y ver si el comportamiento sigue siendo el mismo.

En el ejercicio 4.2, además de probarse sobre un millón de lanzamientos también se probó sobre 10 y 100 millones de estos, el resultado muestra que el número sí se acerca al valor de π, sin embargo, se esperaba que fuese más exacto en los decimales, el número se mantiene en 3.14 a pesar de que el número de lanzamiento fuera alto y se acerca muy poco conforme se aumenta el valor, como se puede apreciar el valor más cercano es el obtenido cuando se hacen 100 millones de lanzamientos, quizás se puede acercar mucho más si se aumentan los hilos o la cantidad de lanzamientos, pero dentro de todo el resultado es aceptable.

Por último, en el ejercicio 4.6 se puede ver el tiempo de ejecución en ambas variaciones del programa usando diferentes cantidad de hilos. Analizando el resultado de la ejecución del programa que da prioridad a los escritores, se aprecia que el tiempo de ejecución aumenta conforme se aumenta el número de hilos, esto puede deberse a que al ser más hilos, habrán más bloqueos, sin embargo, se puede entender mejor este comportamiento de acuerdo a la operación que se da prioridad, y como en este caso son los escritores, se entiende que mientras 1 hilo está activo, los restantes estarán esperando, por lo que el tiempo aumenta de acuerdo al número de hilos que están trabajando. Por otro lado, el resultado de la ejecución del programa que da prioridad a los lectores, muestra que en un inicio va aumentando progresivamente, al igual que el otro programa (demorando un poco más de tiempo que el otro), pero a partir de cierto punto (cuando usa 8 núcleos) la variación deja de ser mucha y empieza a ser estable, llegando incluso a ser menor que el tiempo de ejecución del otro programa. Esto puede deberse a los factores de prioridad y posibilidad de las operaciones, como en este programa la prioridad es lectura, esta operación no bloquea todos los demás hilos, sino que trabaja en conjunto con todas las otros hilos que realicen la misma operación, además tomando en cuenta que es la operación que tiene más posibilidad de realizarse (65 por ciento), se puede decir que el problema primero se encarga de estas operaciones y cuando tiene que hacer las de escritura no tiene que bloquear a todos los otros hilos, sino simplemente los que queden. Una explicación adicional en cuanto a porque el tiempo de ejecución es mayor cuando se tiene pocos hilos, podría deberse a que cuando son menos hilos bloquear poco hilos es lo mismo que bloquear casi todo.

CONCLUSIONES

Los ejercicios propuestos ayudan a entender mejor algunas de las principales funciones de la biblioteca Pthreads, en especial el funcionamiento de mutex, la creación de hilos, el paso de los argumentos y un poco de variables de condición. Por otro lado, el poder conocer que el orden importa (durante los bloqueos en las secciones críticas) ayuda a comprender las condiciones de carrera que se dan cuando se ejecutan varios procesos al mismo tiempo. Aunque en general los resultados de los ejemplos son aceptables, no hay que negar que se pudo hacer una evaluación más específica con ciertos parámetros, como la cantidad de hilos o el número de ejecucuciones, de esa manera se hubiera podido encontrar mejores razones a ciertos comportamientos en los resultados, eso puede aplicarse a trabajos futuros.

REFERENCIAS