Practica 6: Multiplicación de matrices con hilos

Alumno(s):

Hernández Pérez Priscila Itzel
Méndez Gutiérrez Néstor Javier
Franco Tenorio Roberto Carlos

Profesor – M.I. Marcela Ortiz Hernández

01 de octubre de 2019

Semestre - *2019-2020/I*

Multiplicación de matrices con hilos.

1. Introducción

En la siguiente práctica aprendemos el uso de hilos, el objetivo es lograr la multiplicación de matrices (dos matrices de NxN que generan como resultado otra matriz de NxN) mediante hilos, donde generamos NxN hilos y trabajamos con ellos mediante las siguientes funciones pthread_create(), pthread_join(), pthread_self().

2. Datos

"no aplica"

3. Figuras, imágenes o gráficos

```
nestor@nestor-Ubuntu32:~/SistemasOperativos/Hilos/matrices_Hilos$ ./Mult_Matri mat11.txt mat22.txt
El tamaño de las matrices es el mismo y esta dentro del rango
El elemento: -49 no cumple con ser igual o mayor a 0 o menor o igual a 9
El elemento: -49 no cumple con ser igual o mayor a 0 o menor o igual a 9
Hilos: 9
Padre 7045
Matriz A:
0 1 2
0 1 2
0 1 2
Matriz B:
0 1 2
0 1 2
0 1 2
Matriz Resultante C:
0 3 6
0 3 6
0 3 6
nestor@nestor-Ubuntu32:~/SistemasOperativos/Hilos/matrices_Hilos$
```

4. Preguntas

1. Escriba el algoritmo del proceso padre, explicando como crea los hilos necesarios, como espera los resultados y como imprime el resultado de la multiplicación de las matrices.

    1. Se obtienen el tamaño de las matrices y este numero se eleva al cuadrado y el numero resultante será el numero de hilos a crear.
    2. Una vez que sabemos el numero de hilos declaramos dos arreglos con este dato, el primero es un arreglo de hilos (**pthread_t tids[iNumHilos]**), y después un arreglo que contiene estructuras (**struct Ren_Col_struct info[iNumHilos]**), estas estructuras, contendrán la información que requiere el hilo para realizar los cálculos.
    3. Se realizan dos ciclos **for** anidados en los cuales se inicializa la información de las estructuras para posteriormente llamar a la función pthread_create, la cual crea y ejecuta el hilo.
    4. Se realizan dos ciclos **for** en los cuales se espera hilo por hilo a que estos terminen su ejecución, como uno de los parámetros de la función pthread_join es el hilo el cual se

esta esperando, no necesitamos usar el identificador de cada hilo ya que los esperamos en el orden en cual fueron ejecutados.

5. Una vez que todos los hilos terminaron solo mandamos llamar la función imprime matriz para cada una de las tres matrices.

2. Explique el uso de las funciones pthread_create(), pthread_join() y pthread_self().

   pthread_create(): Función para crear un hilo, definida en la librería pthread.h y requiere cuatro argumentos. Como primer argumento se la pasa la direccción de la variable de tipo pthread_t, después le pasamos los atributos del hilo (al usar NULL el hilo se crea con los parámetros normales), en el tercer parámetro pasamos la función que queremos que el hilo ejecute y por último pasamos el valor para el argumento de la función del hilo (puede ser NULL).
   pthread_join(): Mecanismo de sincronización que permite esperar la terminación de un hilo.
   pthread_self(): Regresa el identificador de la hebra que le asignó el sistema operativo.

3. Explique el formato que utilizó en los archivos que contienen las matrices.

   El primer renglón del archivo de texto contiene un numero el cual indica la cantidad de renglones y columnas de la matriz, después del primer renglón se encuentra la matriz de números los cuales deben de ser iguales o mayores a cero y menores o iguales a 9.

4. Escriba cual es la orden en la línea de comandos para ejecutar su práctica. Indique y explique cada uno de sus argumentos.

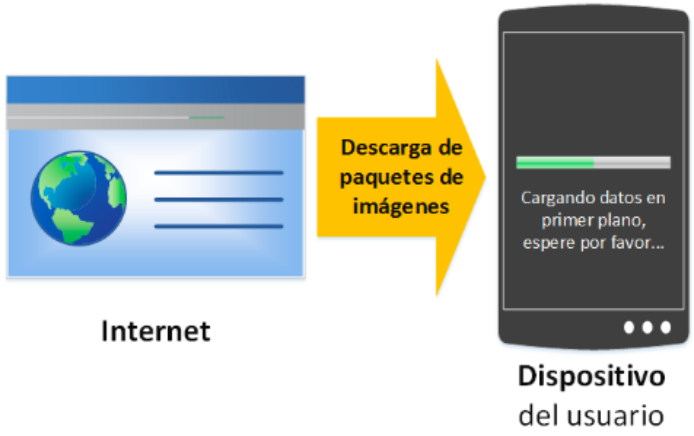
   ./Mult_Matri mat1.txt mat2.txt

   a. Primer parámetro: nombre del ejecutable
   b. Segundo parámetro: nombre del archivo que contiene la matriz A.
   c. Tercer parámetro: nombre del archivo que contiene la matriz B.

5. En general, ¿cuál es la principal diferencia entre usar procesos y usar hilos? Explique un escenario donde usar procesos es mejor que usar hilos. Explique otro escenario en el que usar hilos sea mejor que usar procesos.

Los hilos se distinguen de los tradicionales procesos en que los procesos son generalmente independientes, llevan bastante información de estados, e interactúan sólo a través de mecanismos de comunicación dados por el sistema. Por otra parte, muchos hilos generalmente comparten otros recursos directamente. En muchos de los sistemas operativos que proveen facilidades para los hilos, es más rápido cambiar de un hilo a otro dentro del mismo proceso, que cambiar de un proceso a otro. Este fenómeno se debe a que los hilos comparten datos y espacios de direcciones, mientras que los procesos al ser independientes no lo hacen. Al cambiar de un proceso a otro el sistema operativo (mediante el dispatcher) genera lo que se conoce como overhead, que es tiempo desperdiciado por el procesador para realizar un cambio de modo (mode switch), en este caso pasar del estado de Running al estado de Waiting o Bloqueado y colocar el nuevo proceso en Running. En los hilos como pertenecen a un mismo proceso al realizar un cambio de hilo este overhead es casi despreciable.

Ejemplo donde es mejor usar hilos que procesos:
Imaginemos que queremos ver un listado de 100 imágenes que se descargan desde Internet, como usuario. La mejor opción a implementar sería descargar las 100 imágenes, donde es usuario pueda ir viendo y usando las que ya se han descargado.



Segundo plano (o en inglés background): El segundo plano tiene la característica de darse en el mismo momento que el primer plano. Aquí los hilos deberían de llevar las ejecuciones pesadas de la aplicación. El segundo plano el usuario no lo ve, es más, ni le interesa, para el usuario no existe. Por lo que comprenderás, que el desarrollador puede moverse libremente por este segundo plano –dentro de unos límites. Aquí el desarrollador se puede resarcir y hacer que, por ejemplo, un método tarde horas en ejecutarse, ya que el usuario ni lo sentirá –aunque si está esperando sí que lo notará, con lo que un poco de cuidado también.

Ejemplo donde es mejor usar procesos que hilos:

Imagina que tienes 4 CPUs equivaldría a un proceso por cada una de estas (paralelo). Cada CPU tiene un sistema de memorias caches, y al correr en paralelo, sucede que si no se migra ninguno de los procesos a otra CPU distinta, al cambiar de contexto en cada CPU sea mejor ya que la cache coherency se mantiene. Pero quizá se pueda hacer este esquema sólo con threads y mantener la coherencia, entonces no sería mejor usar procesos aquí.

Conclusiones

En esta práctica se aplicaron los conocimientos obtenidos en clase acerca del funcionamiento de los hilos, como es que se crean, ejecutan, como obtener un resultado de estos y cuáles son las diferencias existentes entre hilos y procesos, y en general se puede decir que fue una práctica mas sencilla en comparación con la practica de procesos.