Práctica 3.- Resolución de problemas y análisis de rendimiento con PThreads

Arquitectura y Computación de Altas Prestaciones. Universidad de Granada. Curso 2023/24

Objetivos: En esta práctica se pretende que los estudiantes apliquen los conocimientos adquiridos en el seminario de PThreads usando dicha librería para la resolución de problemas. De esta forma, gracias al estudio previo de MPI, ya serán capaces de explotar el paralelismo a dos niveles, procesos e hilos, para abordar problemas computacionales potencialmente costosos.

Bases: Esta práctica se realizará individualmente y los programas desarrollados estarán pensados para funcionar en un entorno Linux. Habrá que entregar los archivos de código fuente desarrollados y una memoria (en formato PDF) que explique el trabajo realizado. La entrega se realizará mediante la actividad que se creará en SWAD para tal efecto. Las entregas tardías supondrán una penalización en la nota (0.5 puntos por día hasta alcanzar el límite de 5).

Entorno: La mayor parte del trabajo se puede realizar en equipos personales. Sin embargo, según la actividad, habrá que acceder al cluster de prácticas ATCGrid. Cada estudiante tendrá un usuario y contraseña en ATCGrid. Serán asignados por el profesor de forma individual e intransferible. Tampoco se podrán cambiar las claves recibidas. El acceso al cluster de prácticas puede requerir estar conectado por VPN a la red de la universidad.

Ejercicio 1 (1 punto)

Escribe un programa que reciba por consola tres parámetros: un tamaño de vector, el número de hilos a crear, y un valor de entre {0, 1, 2}, asociado al tipo de reparto de carga. El programa creará entonces dos vectores de tipo “int” con el tamaño indicado y los rellenará con valores aleatorios en el rango [0, 100). Seguidamente, reservará espacio para un tercero, destinado a contener su suma. Llegados a este punto, creará tantos hilos como se haya indicado, y éstos calcularán concurrentemente la suma de los dos vectores previamente preparados. Usa el tercer parámetro para ajustar cómo se reparte la carga.

Si el tercer parámetro es 0, los hilos se moverán siguiendo un esquema “cíclico”, es decir, “saltándose”. Por ejemplo, para vectores de tamaño 7 y 3 hilos:

- Hilo 0 accede a {0, 3, 6}
- Hilo 1 accede a {1, 4}
- Hilo 2 accede a {2, 5}

Si el tercer parámetro es 1, los hilos se moverán por “bloques”, asignando el excedente al final. Por ejemplo, para vectores de tamaño 7 y 3 hilos:

- Hilo 0 accede a [0, 2)
- Hilo 1 accede a [2, 4)
- Hilo 2 accede a [4, 7)

Finalmente, si el tercer parámetro es 2, los hilos se moverán por “bloques balanceados”, repartiendo el excedente de la forma más equilibrada posible. Por ejemplo, para vectores de tamaño 7 y 4 hilos:

- Hilo 0 accede a [0, 2)
- Hilo 1 accede a [2, 4)
- Hilo 2 accede a [4, 6)
- Hilo 3 accede a [6, 7)

Cuando el resultado esté listo, si la longitud de los vectores es inferior a 10, se mostrarán por consola tanto los dos que se suman como su resultado. En cualquier caso, se comprobará siempre el resultado de forma secuencial. Esfuérzate por hacer un código lo mejor diseñado y compacto posible.

Ejercicio 2 (1 punto)

Escribe un programa que reciba 2 parámetros: un tamaño de vector y el número de hilos a crear. El programa creará entonces dos vectores de tipo “double” con el tamaño indicado y los rellenará con valores aleatorios, con parte decimal, en el rango [0, 100). Llegados a este punto, el programa creará tantos hilos como se haya especificado, y se dedicarán a calcular el producto escalar de ambos vectores de forma concurrente. Balancea la carga de forma que ningún hilo tenga asignada más de una tarea extra sobre los demás.

Cuando el resultado esté listo, si la longitud de los vectores es inferior a 10, se mostrarán por consola ambos y su producto escalar. En cualquier caso, incluye también el resultado que obtienes en una comprobación secuencial.

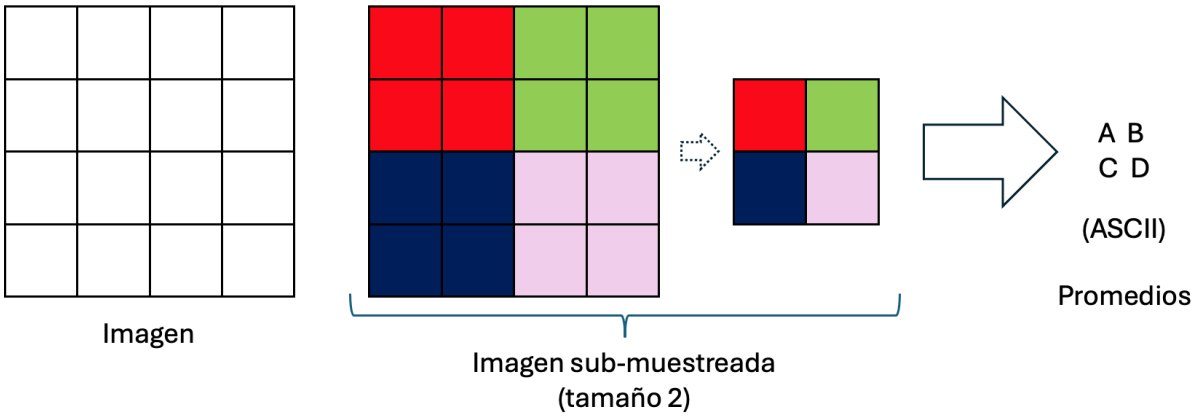
Para realizar el cálculo en paralelo, ¿has usado exclusión mutua? Si es así, ¿cómo podrías hacer este mismo cálculo sin usarla? Si no, ¿cómo la usarías? Diseña la función hebrada que usarías en esta situación alternativa.

Ejercicio 3 (3 puntos)

Imagina ahora que tienes un programa paralelo C que usa PThreads, pero no va del todo bien. Funciona lento y no siempre da el resultado que se esperaría. Todo apunta a la función hebrada, que se muestra bajo este párrafo. ¿Qué dos problemas tiene? Explícalos y diseña una función que los corrija.

```c
void* funcionHebrada(void* arg){
    tarea_hilo* task = (tarea_hilo*) arg;
    for(i = task->ini; i<task->end; i++){
        pthread_mutex_lock(&mutex);
        resultadoGlobal += pow(vector[i], 5);
        pthread_mutex_unlock(&mutex);
    }
    return 0;
}
```

Bien, ahora que manejas con solvencia los PThreads, haz un nuevo conversor de imágenes PGM a ASCII. El enfoque es igual al ya planteado en la práctica anterior: Reserva el paralelismo únicamente para la conversión, siendo el número de hilos a usar un parámetro. No obstante, vas a incluir una importante mejora: capacidad de submuestreo. Concretamente, vas a soportar un tamaño de ventana desde 1 (sin submuestreo) hasta 3. ¿Qué quiere decir esto? Que la conversión no se hace píxel a píxel, sino sobre el valor promedio de los píxeles que quedan dentro de la ventana especificada. Ésta se desplaza además, saltándose a sí misma. Descarta los excedentes (empezando por la esquina superior izquierda).



Ejercicio 4 (5 puntos)

Elabora un programa híbrido (MPI-PThreads) que calcule el producto de dos matrices de tamaño arbitrario. Hará falta entonces trabajar a dos niveles: Gran división del problema entre los procesos disponibles, seguida finalmente de una subdivisión dentro ya de cada proceso, mediante hilos.

Las matrices serán de tipo "*double*" y tendrán valores en el mismo rango que el ejercicio 2. El proceso 0 recibe como parámetro el tamaño de las matrices (filas de A, columnas de A y columnas de B), las inicializa, y finalmente se realiza el cálculo entre todos los procesos disponibles debiendo quedar el resultado final en el proceso 0. Este mostrará siempre el tiempo real que ha tardado la operación. Es decir, debes **medir** desde que el **proceso 0**, tras reservar e inicializar las matrices, **comienza a repartir** los datos, **hasta que ya tiene** en su memoria **el resultado** final.

Además, si las matrices son pequeñas (p.ej., hasta 7x7), se mostrarán tanto las matrices multiplicadas como la resultante de dicho producto.

Teniendo en cuenta que el proceso 0 debe recibir también el número de hilos que se pueden crear por proceso, aquí se muestra un ejemplo de formato de llamada que lanzaría 2 procesos y 4 hilos por proceso para multiplicar dos matrices, la primera de tamaño 7x5, y la segunda de tamaño 5x7:

mpirun -n 2 ./programa 7 5 7 4

Cuando tu programa esté listo, fija un tamaño de matrices de 1000x1000 y realiza un estudio computacional del tiempo que tarda para varias unidades de ejecución. En concreto, realiza una gráfica de aceleración con respecto a una versión secuencial (que mida sólo el cálculo también). En el eje X trabaja en términos de hilos desplegados en total. Llega, por lo menos, a 16 unidades de ejecución (hilos) en total (p.ej., 2, 4, 8 y 16), promediando cada caso con 5 mediciones. Puedes servirte de ATCGrid para lograrlo.

¡Concurso! Si te has esforzado y crees que tu programa es competitivo, calcula su aceleración en ATCGrid con 2 procesos (2 máquinas) y 8 hilos por proceso. El valor de aceleración debe estar promediado tras 10 ejecuciones, quitando el mejor y peor tiempo. Se dará con 2 decimales. Es importante que sea realista: Si el valor se desvía significativamente cuando el profesor lo compruebe, el estudiante estará descalificado (sin perjuicio de penalizaciones en la nota en caso de detectar intento de engaño).

El profesor escogerá los 5 mejores resultados y los comparará personalmente. El primer, segundo y tercer clasificado ganarán 3, 2 y 1 puntos, respectivamente, para asignarlos a prácticas. En caso de acumular excedente (p. ej., alguien con un 10 en todas las prácticas y que ya no tenga margen de mejora), los puntos podrán añadirse a la nota del examen de teoría multiplicados por 0.5.

Nota: Las condiciones del concurso, incluso su propia realización, quedan bajo entera disposición del profesor. Éste, aunque actuará siempre de buena fé, puede tomar cualquier decisión al respecto, pudiendo incluso cancelarlo por problemas de agenda.