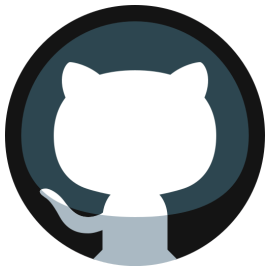
INFRAESTRUCTURAS PARALELAS Y DISTRIBUIDAS
Cardoso Forero Manuel Felipe (2027288)
Facultad de Ingeniería
Universidad del Valle, Cali Colombia
INFORME DE RESULTADOS (Parcial I)
octubre 2024

Enlaces de entregables del parcial: CardosoForero(ParcialPractico)



ManuhCardoso1501/ParcialTeoricoInfra

Imagenes procesadas

En este documento se relacionan los resultados del parcial práctico de Infraestructuras Paralelas y Distribuidas, donde el objetivo fue tomar los tiempos de ejecución de un programa en su versión secuencial y paralela, el cual se encarga de filtrar un conjunto de imágenes disponibles. Para lograr este análisis, se han considerado variables fundamentales como las siguientes:

1. **Tiempo de ejecución**: Medido para ambas versiones (secuencial y paralela).
2. **Speedup**: Relación entre el tiempo de ejecución secuencial y el tiempo de ejecución paralelo, que refleja la ganancia en rendimiento al utilizar múltiples hilos.
3. **Eficiencia**: Mide qué tan bien se utilizan los recursos disponibles al aumentar el número de hilos.

Resultados

Para la versión secuencial tenemos los siguientes resultados

Numero de ejecución	Tiempo (segundos)
1	20,935
2	20,863
3	~~21,393~~
4	20,297
5	~~19,895~~

Promedio	20,698

Antes de mostrar los resultados la forma en cómo se consiguió el paralelismo fue por medio de directivas **#pragma omp**. En la función **aplicarFiltro**, la directiva **#pragma omp parallel for num_threads(numHilos) collapse(2) schedule(dynamic)** permite dividir el trabajo de aplicar el filtro a las imágenes en bloques de filas y columnas, habilitando múltiples hilos para procesar en paralelo cada píxel de la imagen. La opción collapse(2) es especialmente útil aquí, ya que combina los dos bucles anidados (uno para las filas y otro para las columnas), lo que genera una carga de trabajo más uniforme entre los hilos y minimiza el tiempo de espera. Por otro lado, en la función **calcularSumaPixeles**, la directiva **#pragma omp parallel for num_threads(numHilos) reduction(+:suma) schedule(dynamic)** se utiliza para sumar los píxeles procesados de la imagen de manera paralela. La cláusula reduction(+:suma) es esencial, ya que asegura que cada hilo mantenga una copia local de la variable suma y, al final de su ejecución, combine estas copias para obtener el resultado final sin conflictos de acceso concurrente. Adicionalmente para ejecutar esta versión se tuvieron en cuenta dos casos:

1. Número de hilos igual número de núcleos de la máquina = **8**.
2. Número de hilos igual número de núcleos de la máquina = **8*2 = 16**.

Resultados para el caso 1:

Numero de ejecución	Tiempo (segundos)
1	18,420
2	~~18,246~~
3	19,155
4	~~22,120~~
5	18,726

Promedio	18,767

Resultados para el caso 2:

Numero de ejecución	Tiempo (segundos)
1	19,991
2	~~18,882~~
3	19,792
4	19,951
5	~~20,079~~

Promedio	19,911

SpeedUp

Para este caso se aplicó la siguiente fórmula y los resultados obtenidos fueron los siguientes:

$$Speedup = \frac{Tiempo\ Secuencial}{Tiempo\ paralelo}$$

Speedup (Caso 1)	1,1029
Speedup (Caso 2)	1,0395

Eficiencia

Para este caso se aplicó la siguiente fórmula y los resultados obtenidos fueron los siguientes:

Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz	
CPU family:	6
Model:	142
Thread(s) per core:	2
Core(s) per socket	4
Socket(s):	1
Stepping:	12
RAM = 12	

Eficiencia (8 hilos)	13,79%
Eficiencia (16 hilos)	6,50%

$$Eficiencia = \frac{Speedup}{número\ de\ hilos}$$

Comentarios

En el primer caso de ejecución paralela, donde se utilizaron 8 hilos, se alcanzó un tiempo promedio de 18,767 segundos. Este resultado se traduce en un speedup de 1,1029, indicando una mejora notable en el rendimiento al implementar la paralelización. La eficiencia en este caso, calculada en 13,79%, sugiere que aunque hubo un avance en el rendimiento, solo una fracción del potencial de los hilos fue utilizada de manera efectiva. Este nivel de eficiencia puede atribuirse a la sobrecarga inherente a la gestión de hilos y a la sincronización necesaria para combinar los resultados de cada hilo, así como a la naturaleza del algoritmo y las características de los datos procesados.

En el segundo caso, al aumentar el número de hilos a 16, el tiempo promedio de ejecución fue de 19,911 segundos, lo que resultó en un speedup de 1,0395. Este aumento en el número de hilos, aunque inicialmente puede parecer ventajoso, condujo a un ligero decremento en el speedup y una considerable caída en la eficiencia, que se situó en 6,50%. Es importante destacar que, aunque el número de hilos se basó en el número de núcleos de la máquina, al sobrepasar este límite, se introduce una creciente sobrecarga en la gestión de hilos. La competencia por recursos compartidos y el tiempo adicional requerido para la sincronización pueden afectar negativamente la capacidad de los hilos para trabajar de manera independiente y eficiente.

Algo que se observó es que la paralelización del filtro aunque se logre hacer de manera óptima se tiene un limitante en cuanto funciona el programa y está en el script pues al procesar cada imagen de manera secuencial, el script no aprovecha la paralelización que se podría implementar en el programa principal porque se observa que, aunque el programa en sí puede estar diseñado para ejecutarse en paralelo, el script limita la ejecución a un solo hilo por archivo, lo que puede resultar en un tiempo total de procesamiento considerablemente mayor. Por lo que se puede decir que el script que corre el programa en modo secuencial limita significativamente el potencial de rendimiento y eficiencia del sistema al no aprovechar la paralelización que el programa podría ofrecer. Dado que la función objetivo también incluyera paralelizar este proceso se podrían obtener mejores resultados y aprovechar el paralelismo.