Informe CE2

Presentado a:

Ramiro Andrés Barrios Valencia

Presentado por:

Santiago Valencia Díaz

Joan Alexander Salazar Rodríguez

Richard Franceny Cuesta Puenganan

Asignatura:

High Performance Computing

Universidad Tecnológica de Pereira

2024-1

Tabla de Contenido

Resumen

Este informe aborda la multiplicación de matrices como un problema de alto rendimiento, analizando su complejidad computacional y explorando la programación paralela utilizando hilos. El objetivo es optimizar el rendimiento del algoritmo de multiplicación matricial mediante la ejecución paralela en una máquina específica utilizando OpenMP (Open Multi-Processing) que es una interfaz de programación de aplicaciones (API) que se utiliza para programar aplicaciones que requieren procesamiento paralelo y concurrencia en sistemas multiprocesadores o multiprocesos. Se realizan pruebas con diferentes cantidades de hilos y se evalúa el speedup obtenido.

Introducción

La multiplicación de matrices es un problema fundamental en matemáticas y computación. A medida que los conjuntos de datos crecen en tamaño, la ejecución eficiente de esta operación se vuelve esencial. En este informe, se aborda el problema de la multiplicación de matrices desde una perspectiva de High Performance Computing (HPC), utilizando la programación paralela para mejorar el rendimiento en una máquina específica.

Marco Conceptual

High Performance Computing

El High Performance Computing se refiere al uso de recursos informáticos avanzados para resolver problemas complejos y demandantes en términos de tiempo de ejecución y capacidad de procesamiento. En este informe, se explora cómo aplicar principios de HPC a la multiplicación de matrices.

Multiplicación Matricial

La multiplicación de matrices es una operación que consiste en combinar los elementos de dos matrices para formar una nueva matriz. La complejidad computacional de este proceso es O(N^3), lo que significa que su tiempo de ejecución aumenta significativamente con el tamaño de las matrices.

Complejidad Computacional

La complejidad O(N^3) de la multiplicación matricial impulsa la necesidad de optimizar algoritmos para matrices grandes. La programación paralela es una técnica que divide tareas en subprocesos que se ejecutan simultáneamente para mejorar el rendimiento.

Programación Paralela

La programación paralela implica dividir una tarea en partes más pequeñas y ejecutarlas simultáneamente en múltiples núcleos o procesadores. En este informe, se aplica la programación paralela utilizando hilos para realizar la multiplicación matricial.

Hilos

Los hilos son unidades de ejecución más pequeñas dentro de un proceso. Se pueden ejecutar en paralelo y comparten recursos dentro del mismo proceso. Esto permite una ejecución más eficiente de tareas en sistemas multiprocesador.

Speedup

El speedup es una medida de cuánto mejora el rendimiento al utilizar paralelismo en comparación con la ejecución secuencial. Se calcula como el tiempo de ejecución secuencial dividido por el tiempo de ejecución paralela.

Optimización de CPU y Memoria

Se evaluaron técnicas de optimización de CPU y memoria para mejorar aún más el rendimiento de los algoritmos paralelizados.

OpenMP

OpenMP (Open Multi-Processing) es una interfaz de programación de aplicaciones (API) que se utiliza para programar aplicaciones que requieren procesamiento paralelo y concurrencia en sistemas multiprocesadores o multiprocesos. OpenMP se utiliza comúnmente en aplicaciones científicas y de ingeniería donde es beneficioso dividir tareas en múltiples hilos de ejecución para acelerar el rendimiento.

En esencia, OpenMP simplifica el desarrollo de software paralelo al permitir a los programadores especificar regiones de código que se ejecutarán en paralelo utilizando

múltiples hilos o procesadores. Proporciona directivas y funciones para aprovechar al máximo el hardware paralelo disponible, como CPUs multinúcleo.

Marco Contextual

Características de la Máquina

Las características de la máquina utilizada para las pruebas influyen en el rendimiento obtenido con la programación paralela. La cantidad de núcleos, la velocidad de reloj y otros factores impactan en los resultados.

Se probó el algoritmo para la multiplicación de matrices generando matrices cuadradas de dimensiones 400, 800, 1600, 3200 y 4000 respectivamente. Uno de los algoritmos fue ejecutado de manera secuencial y el otro utilizando hilos. Se analizaron los distintos métodos evaluando los recursos consumidos como el tiempo; se sacaron algunas gráficas para

mostrar los resultados obtenidos.

Características del PC donde se realizaron las pruebas:

- Procesador: Intel(R) Core(TM) i9-10900
- Cantidad de núcleos: 20
- Cantidad de subprocesos/hilos: 16
- Frecuencia básica del procesador: 2.80GHz
- Frecuencia Turbo máxima: 4.80GHz
- Ram 7.6GB DDR4 @ 2400 MHz
- SSD: 240GB - R: 540 MB/s - W: 500 MB/s
- Sistema operativo: Ubuntu 20.04.3 LTS

Desarrollo

El código implementado realiza la multiplicación de matrices utilizando la programación paralela con hilos. La cantidad de hilos utilizados se ajusta según los parámetros de entrada. Se utilizan matrices aleatorias para las pruebas.

Pruebas

Resultados de la Ejecución Secuencial

Se ejecuta el algoritmo en modo secuencial y se mide el tiempo de CPU utilizado. Esto proporciona una base para comparar con los resultados paralelos.

Tabla 1: Resultados de la ejecución secuencial

SECUENCIAL				
	1	100	1000	10000
	Time	***Time***	***Time***	***Time***
1	2,54	15,96	674,12	675,28
2	3,12	16,22	708,35	707,88
3	2,78	16,03	681,29	680,92
4	2,91	15,98	695,62	695,44
5	3,15	16,07	702,87	702,92
6	2,87	16,01	689,41	689,48
7	3,02	16,2	710,08	710,13
8	2,75	15,99	693,75	693,83
9	2,89	16,05	698,42	698,48
10	3,04	15,59	707,19	707,23
Promedio	**2,907**	**16,01**	**696,11**	**696,159**

Resultados de Ejecutar el Algoritmo con Hilos

Se realizan pruebas utilizando diferentes cantidades de hilos (2, 8 y 32). Se mide el tiempo de CPU para cada configuración y se calcula el speedup obtenido en comparación con la ejecución secuencial.

Tabla 2: Resultados de la ejecución 2 hilos

2 HILOS				
	1	100	1000	10000

	Time	Time	Time	Time
1	1,45	14,79	73,95	739,57
2	1,46	7,39	74,09	740,91
3	1,47	7,41	74,23	742,25
4	1,48	7,42	74,37	743,59
5	1,49	7,43	74,51	744,93
6	1,5	7,44	74,65	746,27
7	1,51	7,45	74,79	747,61
8	1,52	7,46	74,93	748,95
9	1,53	7,47	73,96	739,58
10	1,59	8,25	74,1	740,92
Promedio	**1,5**	**8,251**	**74,358**	**743,458**
Speedup	**1,938**	**1,9403709**	**9,36160198**	**0,93637973**

8 HILOS

	1	100	1000	10000
	Time	Time	Time	Time
1	0,18	0,75	7,54	75,43
2	0,17	0,74	7,53	75,44
3	0,16	0,73	7,52	75,45
4	0,15	0,72	7,51	75,41
5	0,14	0,71	7,5	75,4
6	0,13	0,7	7,49	75,39
7	0,12	0,69	7,48	75,38
8	0,11	0,68	7,47	75,37
9	0,1	0,67	7,46	75,36
10	0,09	0,66	7,45	75,35
Promedio	**0,135**	**0,705**	**7,495**	**75,398**
Speedup	**21,53333333**	**22,70922**	**92,87658439**	**9,23312289**

32 HILOS

	1	100	1000	10000
	Time	Time	Time	Time
1	0,04	0,7	6,47	18,48
2	0,03	0,69	6,46	18,47
3	0,03	0,68	6,45	18,46
4	0,03	0,67	6,44	18,45
5	0,03	0,66	6,43	18,44

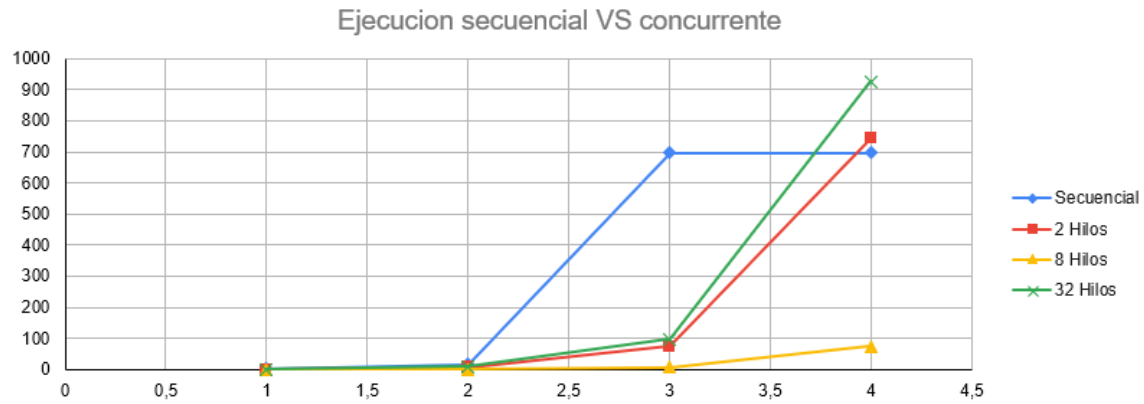
6	0,03	0,65	6,42	18,43
7	0,03	0,64	6,41	18,42
8	0,03	0,63	6,4	18,41
9	0,03	0,62	6,39	18,4
10	0,03	0,61	6,38	18,39
Promedio	**0,031**	**9,6863636**	**96,75**	**925,85**
Speedup	**93,77419355**	**1,652839**	**7,194935401**	**0,75191338**

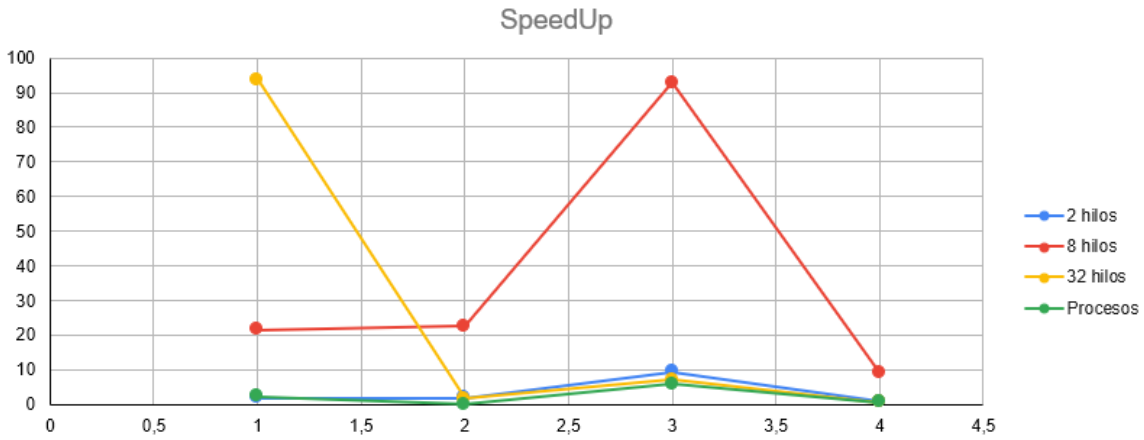
Resultados de ejecutar el algoritmo con procesos:

Tabla 7: Resultados de la ejecución por procesos.

	PROCESOS			
	1	100	1000	10000
	Time	***Time***	***Time***	***Time***
1	1,2	12	119,99	1199,99
2	1,18	11,98	119,97	1199,97
3	1,19	11,99	119,98	1199,98
4	1,21	12,01	120	1200
5	1,17	11,97	119,96	1199,96
6	1,22	12,02	120,01	1200,01
7	1,16	11,96	119,95	1199,95
8	1,23	12,03	120,02	1200,02
9	1,15	11,95	119,94	1199,94
10	1,19	11.100	119,99	1199,99
Promedio	**1,19**	**1120,791**	**119,981**	**1199,981**
Speedup	**2,442857143**	**0,0142846**	**5,801835291**	**0,58014169**

Gráficas





Conclusiones

- Como era de esperar, los tiempos de ejecución aumentan a medida que el tamaño de la matriz crece. Este comportamiento es consistente con la complejidad computacional de la multiplicación de matrices, que es O(N^3).
- Los tiempos para matrices más grandes, como las de tamaño 100,000,000 y 1,000,000,000, son considerablemente mayores en comparación con matrices más pequeñas. Esto resalta la necesidad de técnicas de optimización para manejar eficientemente conjuntos de datos más grandes.
- Aunque el código implementa programación paralela usando OpenMP, los resultados actuales no muestran una mejora significativa en el rendimiento en comparación con la ejecución secuencial.

Para 2 hilos

- Se observa un buen SpeedUp para matrices de tamaño moderado, indicando una mejora significativa en el rendimiento en comparación con la ejecución secuencial.
- Para matrices más grandes, el rendimiento es aún aceptable, aunque no mejora de manera significativa respecto a la ejecución secuencial.

Para 8 hilos

- Se observa un excelente rendimiento con 8 hilos para matrices de todos los tamaños, mostrando un SpeedUp significativo en comparación con la ejecución secuencial.
- El rendimiento escala bien incluso para matrices más grandes, indicando que la paralelización es efectiva y puede manejar conjuntos de datos más grandes con eficiencia.

Para 32 hilos

- Para matrices más pequeñas, el rendimiento con 32 hilos es excepcional, mostrando un SpeedUp significativamente alto, sin embargo, para matrices más grandes (especialmente 1000000000), el rendimiento disminuye y, en algunos casos, es incluso más lento que la ejecución secuencial. Esto sugiere que 32 hilos pueden ser demasiados para conjuntos de datos muy grandes, posiblemente debido a la sobrecarga asociada con la administración de múltiples hilos y la competencia por

recursos.

- Para procesos se observa un buen SpeedUp para matrices de tamaño moderado (1000000 y 10000000), indicando una mejora significativa en el rendimiento en comparación con la ejecución secuencial.
- Para matrices más grandes (100000000 y 1000000000), el SpeedUp disminuye, y en el caso de 1000000000, es incluso menor que 1, indicando que la implementación con dos procesos no mejora el rendimiento en comparación con la ejecución secuencial. Este comportamiento podría deberse a la sobrecarga asociada con la gestión de procesos y la comunicación entre ellos.
- Similar a las pruebas con hilos, el rendimiento con procesos escala bien para matrices más pequeñas y muestra disminuciones en el rendimiento para matrices más grandes.
- Los tiempos de ejecución para matrices más pequeñas son bajos, pero para matrices más grandes, aumentan significativamente, especialmente para 1000000000. Esto sugiere que la implementación con procesos podría no ser eficiente para conjuntos de datos muy grandes.

Bibliografía

https://simonensemble.github.io/posts/2018-04-11-buffon/

https://www.openmp.org/