Sistemas Operativos

Taller de Evaluación

Brayan Fajardo

Jorge Andrés Fortich Ordosgoitia

Francisco Guzman

Juan Camilo Alba

Santiago Botero

21/10/2024

Introducción

Este proyecto tiene como objetivo evaluar el rendimiento de dos algoritmos diferentes para la multiplicación de matrices de gran tamaño: el método clásico y el método de la transpuesta. Ambos algoritmos se han implementado en C utilizando la biblioteca POSIX Threads (Pthreads) para aprovechar el paralelismo y mejorar los tiempos de ejecución.

El proyecto consta de los siguientes componentes:

1. **`mm_clasico.c`**: Implementación del algoritmo de multiplicación de matrices clásico utilizando hilos.
2. **`mm_transpuesta.c`**: Implementación del algoritmo de multiplicación de matrices utilizando la transpuesta de una de las matrices.
3. **`lanza.pl`:** Script en Perl que automatiza la ejecución de los programas, variando el tamaño de las matrices y el número de hilos, y almacenando los resultados en archivos de datos.

La motivación de este proyecto es comprender cómo se comportan estos dos algoritmos de multiplicación de matrices en términos de rendimiento cuando se aplica paralelismo a través del uso de múltiples hilos. Además, se busca determinar en qué condiciones (tamaño de matriz, número de hilos) cada algoritmo presenta un mejor desempeño.

Algoritmos de Multiplicación de Matrices

Método Clásico

La multiplicación de matrices de forma clásica se basa en el cálculo del producto punto entre cada fila de la matriz A y cada columna de la matriz B. Cada elemento de la matriz C se obtiene mediante la suma de los productos de los elementos correspondientes de la fila de A y la columna de B.

En el código **`mm_clasico.c`**, la lógica de paralelización se implementa de la siguiente manera:

1. Se divide el rango de filas de la matriz C entre el número de hilos, de modo que cada hilo se encarga de calcular una parte de las filas.
2. Cada hilo itera sobre su rango de filas asignado y, para cada fila, calcula el producto punto entre esa fila de A y cada columna de B, almacenando el resultado en la matriz C.
3. Se utiliza un mutex (**`MM_mutex`**) para sincronizar el acceso a la escritura en la matriz C y evitar condiciones de carrera.

Esta estrategia de paralelización permite que varios hilos trabajen simultáneamente en la multiplicación de matrices, lo que debería mejorar el rendimiento en comparación con una implementación secuencial.

Método de la Transpuesta

El método de la transpuesta aprovecha la propiedad conmutativa de la multiplicación de matrices. En lugar de calcular **`C = A * B`**, se calcula **`C = A * B^T`**, donde **`B^T`** es la transpuesta de la matriz B.

Esta estrategia permite que cada hilo calcule una parte de las filas de C, de forma similar al método clásico. Sin embargo, en este caso, los accesos a la matriz B se realizan a través de columnas, lo que puede mejorar el rendimiento en algunos casos.

En el código **`mm_transpuesta.c`**, la lógica de paralelización es muy similar a la del método clásico:

1. Se divide el rango de filas de la matriz C entre el número de hilos.
2. Cada hilo itera sobre su rango de filas asignado y, para cada fila, calcula el producto punto entre esa fila de A y la columna correspondiente de B, almacenando el resultado en la matriz C.
3. Se utiliza un mutex (**`MM_mutex`**) para sincronizar el acceso a la escritura en la matriz C y evitar condiciones de carrera.

La diferencia clave entre los dos métodos es la forma de acceder a los elementos de la matriz B. En el método clásico, se accede por filas, mientras que en el método de la transpuesta, se accede por columnas. Esto puede tener un impacto significativo en el rendimiento, especialmente cuando se trabaja con matrices de gran tamaño.

Automatización de Ejecuciones

Para facilitar la evaluación y comparación de los dos algoritmos, se ha desarrollado un script en Perl llamado **`lanza.pl`**. Este script se encarga de automatizar la ejecución de los programas de multiplicación de matrices, variando el tamaño de las matrices **`(@Size_Matriz)`** y el número de hilos (**`@Num_Hilos`**).

El script realiza los siguientes pasos:

1. Obtiene el directorio de trabajo actual y lo almacena en la variable **`$Path`**.
2. Define los arreglos **`@Nombre_Ejecutable, @Size_Matriz y @Num_Hilos`** que contienen, respectivamente, los nombres de los programas a ejecutar, los tamaños de las matrices y los números de hilos a utilizar.
3. Itera sobre cada combinación de ejecutable, tamaño de matriz y número de hilos.
4. Para cada combinación, crea un nombre de archivo **`$file`** que se utilizará para almacenar los resultados.
5. Repite la ejecución del programa correspondiente **`($Nombre_Ejecutable)`** 30 veces (**`$Repeticiones`**) con los parámetros actuales, redirigiendo la salida al archivo **`$file`**.
6. Imprime en la terminal el comando que se está ejecutando.

Este script facilita enormemente la recopilación y el análisis de los tiempos de ejecución para las diferentes configuraciones, lo que permite evaluar y comparar el rendimiento de ambos algoritmos de una manera sistemática y automatizada.

Resultados y Análisis

Una vez ejecutados los programas a través del script **`lanza.pl`**, se generan múltiples archivos de datos que contienen los tiempos de ejecución para cada combinación de tamaño de matriz y número de hilos. Estos archivos de datos pueden ser posteriormente procesados y analizados para extraer conclusiones sobre el rendimiento de cada algoritmo.

El análisis de los resultados debe considerar métricas como:

- Tiempo de ejecución promedio para cada configuración
- Speedup obtenido al aumentar el número de hilos
- Eficiencia de paralelización (speedup / número de hilos)
- Diferencias de rendimiento entre el método clásico y el método de la transpuesta

Estos análisis permitirán determinar en qué condiciones (tamaño de matriz, número de hilos) cada algoritmo presenta un mejor desempeño, y cómo se ve afectado el rendimiento al variar estos parámetros.

RESULTADOS TRANSPUESTO

					TRANSPUESTA					
	800									
Promedio	2281989,7	1417959,1	979354,77	Promedio	1867260,2	1217860,9	825499,06	Promedio	1970116,5	1240858,7	885415,61
	1000									
Promedio	3893564,8	2294910,9	1333581,6	Promedio	3187353	1842806,7	1181081,4	Promedio	3204889,9	1816468,9	1313901,8
	1200									
Promedio	5872563,5	3437758,2	2047090,4	Promedio	5066569	2851166,6	1735877,9	Promedio	5198221,7	2904915,5	1830733,6
	1400									
Promedio	5872563,5	3437758,2	2047090,4	Promedio	5066569	2851166,6	1735877,9	Promedio	5198221,7	2904915,5	1830733,6
	1600									
Promedio	12786757	6903406,3	4068376,7	Promedio	11422274	5981859,2	3324041,1	Promedio	11858510	6301158,3	3818840,4
	1800									
Promedio	17695599	9317323,8	5352517,2	Promedio	16219579	8520889,2	4380964,9	Promedio	16598286	8541181	5241129
	2000									
Promedio	23547782	12488117	6913714,7	Promedio	22333004	11539972	6012318	Promedio	22967759	11886738	6938646,8
	2200									
Promedio	30877315	16373141	8867309	Promedio	29900660	15182212	7900311,4	Promedio	30353438	15420935	9173423,4
	2400									
Promedio	39901244	20917286	11101067	Promedio	38655889	19607571	10067341	Promedio	39439992	20025214	11564119
	2600									
Promedio	50395869	26309298	13957605	Promedio	49201867	24894050	12765808	Promedio	49524558	24912894	14673962
	2800									
Promedio	62756421	32432736	16668208	Promedio	61637472	31237015	15872286	Promedio	61708484	31109917	18146361

RESULTADOS CLÁSICO

					Clasico						
	800										
Promedio	2046123,71	1231838,84	848292,161	Promedio	2498679,48	1370459,81	1038172,32	Promedio	2542798,94	1583121,39	1019884,61
	1000										
Promedio	3217203,26	2022012,9	1257024,71	Promedio	4131111,9	2454155,29	1468507,97	Promedio	4547251,07	2592826,32	1557043,77
	1200										
Promedio	5845337,29	3116633,77	1798623,19	Promedio	6403962,74	3704797,45	2097521,29	Promedio	7721572,87	4075052,74	2165963,13
	1400										
Promedio	9441367,29	4906698,39	2650731,03	Promedio	9562880,61	5422470,26	3088983,58	Promedio	12056024	6204999,77	3061214,32
	1600										
Promedio	15746747,3	7996759,61	4351217,65	Promedio	14136840,2	7644819,9	4261505,29	Promedio	17797421,1	8946708,03	4497294,97
	1800										
Promedio	24730436,3	12631847,6	6225506,52	Promedio	21569155,5	11021694,2	5895296	Promedio	25286513,1	12851793,9	6469349,48
	2000										
Promedio	35904148	18336133,4	9399526,03	Promedio	30565808,1	15986450,5	8046779,61	Promedio	36582511	17930562,9	8909109,52
	2200										
Promedio	48669070,9	24634606,4	12663284,5	Promedio	40105608,8	20485692,9	10520714,1	Promedio	48462059,3	23859208,7	11549640,6
	2400										
Promedio	62303769	29793853,1	15024807,3	Promedio	52575046,4	26940697,2	13676651,8	Promedio	62139066,7	29912303,3	14197722,3
	2600										
Promedio	82215628,7	41690272,6	20905218,3	Promedio	66231933,2	33731029,9	16907842,6	Promedio	78491362,3	38113242,9	17835145,1
	2800										
Promedio	104633915	51351059,7	25879126,1	Promedio	85913478,5	43806413,8	21762854,1	Promedio	100673567	48674154,6	22703267,5

Al observar los datos de las imágenes, podemos identificar varios patrones generales en el comportamiento de los tiempos de ejecución:

1. **Crecimiento del tiempo con el tamaño de las matrices**:
    - En ambos métodos (clásico y transpuesta), los tiempos de ejecución aumentan conforme el tamaño de la matriz crece, lo cual es esperado porque la cantidad de operaciones necesarias para la multiplicación es proporcional al cubo del tamaño de las matrices.
2. **Impacto del número de hilos**:
    - Para matrices pequeñas, como de tamaño 800 o 1000, el uso de múltiples hilos no siempre mejora significativamente el tiempo debido al overhead de creación y manejo de los hilos.
    - Para matrices más grandes (desde 1600 en adelante), el uso de múltiples hilos (especialmente 4) muestra consistentemente mejores tiempos en comparación con 1 o 2 hilos, ya que el trabajo se distribuye más eficientemente.
3. **Comparación entre métodos (clásico vs transpuesta)**:
    - El método clásico parece ser más rápido para tamaños de matrices pequeños y medianos, especialmente con 1 hilo, probablemente debido a su simplicidad y menor overhead.
    - Para tamaños grandes (como 2000 y superiores), el método transpuesta con 4 hilos comienza a acercarse al rendimiento del clásico, aunque sigue siendo ligeramente más lento en general.
4. **Consistencia entre estudiantes**:
    - Aunque los tiempos varían ligeramente entre los estudiantes (posiblemente debido a diferencias en hardware), los patrones generales de los tiempos son consistentes, lo que refuerza las conclusiones generales.

Conclusiones

Este proyecto de multiplicación de matrices en paralelo utilizando hilos es una excelente herramienta para estudiar y comparar el rendimiento de diferentes algoritmos de multiplicación de matrices. Los resultados obtenidos pueden ser muy valiosos para comprender mejor el comportamiento de estos algoritmos y tomar decisiones informadas sobre cuál método utilizar en diferentes situaciones.

Al analizar los tiempos de ejecución y las métricas de rendimiento, se podrá determinar si el método de la transpuesta ofrece una ventaja significativa sobre el método clásico, y en qué condiciones se justifica su uso. Esto puede ser muy útil para mejorar el rendimiento de aplicaciones que requieren operaciones intensivas de multiplicación de matrices, como en el campo del procesamiento de señales, la computación científica y la inteligencia artificial.

En el caso de este proyecto, el análisis del rendimiento considerando el número de hilos sugiere que el programa con 4 hilos debería ser el más rápido para matrices grandes, siempre que el hardware tenga al menos 4 núcleos y la paralelización esté implementada de manera eficiente. Sin embargo, para matrices pequeñas, el overhead de manejo de hilos podría superar las ganancias, haciendo que el uso de 1 o 2 hilos sea más eficiente. Este comportamiento destaca la importancia de considerar el tamaño de las matrices y los recursos hardware disponibles al diseñar soluciones paralelas.

Además, al comparar los dos métodos de multiplicación (clásico y transpuesta), se observa que:

1. **El método clásico** es generalmente más rápido para tamaños pequeños y medianos (800 a 1400), especialmente con un solo hilo, debido a su simplicidad y menor overhead.
2. **El método transpuesta**, aunque tiene un rendimiento menor en general, puede ser competitivo para matrices grandes (1600 o más) cuando se emplean 4 hilos, aunque no supera al clásico en este análisis.

Finalmente, este proyecto puede servir como base para futuras investigaciones y optimizaciones de los algoritmos de multiplicación de matrices, explorando técnicas más avanzadas de paralelización, uso de diferentes estructuras de datos o incluso la integración con aceleradores hardware como GPUs.