Introducción a la computación paralela
Tarea 1

Martín Garcés Lucas Escalona

24 Abril 2025

1 Algoritmo naive y algoritmo caché friendly

La implementación del algoritmo naive es una multiplicación simple de los valores sin ninguna simplificación.

En cambio el segundo algoritmo es más amigable con caché ya, primero utiliza un arreglo unidimensional en vez de un arreglo de dos dimensiones, lo que mejora la localidad espacial y reduce accesos indirectos. Segundo ordena los bucles de tal forma que los accesos son secuenciales, lo que aprovecha mejor los accesos a caché disminuyendo la cantidad de datos a traer.

1.1 Experimentos y resultados

Ancho y largo de la matriz	Naive	caché friendly
250	5.702	5.73
500	19.85	23.57
1000	136075	166.61

Table 1: Tiempo de ejecución en segundos

Ancho y largo de la matriz	Naive	caché friendly
250	19,69%	20,40%
500	23,9%	21,32 %
1000	46,51%	46,14 %

Table 2: caché misses porcentaje

2 Blocks

2.1 Descripción

Este algoritmo realiza la multiplicación de dos matrices dividiendo cada matriz en cuatro bloques de igual tamaño, multiplica y suma los bloques correspondientes de forma recursiva hasta llegar a un tamaño mínimo (min_Size), donde se usa la multiplicación estándar. Luego, combina los resultados en una única matriz resultado usando la función merge.

2.1.1 Merge

La función merge recibe 4 matrices $n \times n$ y los combina en una $2n \times 2n$, permitiendo construir la matriz final a partir de las resultantes de los pasos anteriores.

2.2 Análisis secuencial

En cada nivel de la recurrencia el algoritmo divide en 4 los cálculos de matrices $n \times n$, donde cada uno consiste de 2 multiplicaciones de matrices de $n/2 \times n/2$ más la suma de esta. Luego, el costo de dividir y unir las matrices es $O(n^2)$ cada uno.

Por lo tanto la recurrencia está dada por:

$$\begin{aligned} T(n) &= 4(2T(n/2) + O(n^2)) + 2O(n^2) \\ &= 8T(n/2) + 6O(n^2) \\ &= 8T(n/2) + O(n^2) \end{aligned}$$

Luego, siendo $a = 8$, $b = 2$ y $f(n) = n^2$, por primer caso del teorema maestro, dado que $n^2 \in O(n^{3-e})$ para un $e > 0$, entonces $T(n) = \Theta(n^3)$

2.3 Análisis paralelo

El algoritmo paralelo igualmente divide las matrices originales en cuatro sub-bloques y aplica la multiplicación de manera recursiva sobre ellos. Para hacerlo ejecuta en paralelo las operaciones correspondientes a los bloques parciales.

En el caso base (cuando el tamaño del subproblema alcanza el umbral min_size), las multiplicaciones y sumas de sub-bloques necesarias para calcular los bloques $C1$, $C2$, $C3$ Y $C4$ se realizan de forma simultánea. De esta manera, se aprovechan las oportunidades de paralelismo presentes en la estructura del algoritmo para reducir el tiempo total de ejecución.

En el caso recursivo, las llamadas a la función para calcular los bloques parciales también se ejecutan en paralelo. Esto permite que distintas partes de la matriz resultado se construyan de manera concurrente.

Finalmente, los resultados parciales se combinan para formar la matriz final, tal como en la versión secuencial.

2.4 Estimación complejidad

Para estimar la complejidad paralela, por el análisis anterior tenemos que el trabajo está dado por $W = 8(n/2) + O(n^2)$. Por otro lado, dividir n por dos en cada nivel hasta min_size ($log(n/min\_size)$), luego hacer la multiplicación y suma ($O(min\_size^2)$) y por último el merge ($log(n/min\_size)$), Luego es span está dado por $S = 2log(n/min\_size) + O(1)$.

Por lo tanto, por teorema de Brent la complejidad paralela está dada por $Tp <= \frac{\Theta(n^3)}{p} + log(n/min\_size)$

3 Strassen

3.1 Análisis secuencial

Esta versión del algoritmo de Strassen realiza una sola capa de partición de matrices, sin aplicar recursión. Se divide cada matriz $n \times n$ en submatrices de tamaño $\frac{n}{2} \times \frac{n}{2}$ y se realizan:

- 7 multiplicaciones clásicas entre submatrices.
- 15 a 20 operaciones de suma o resta de matrices.
- Combinación de los bloques resultantes en la matriz final.

3.1.1 División de matrices

Dividir una matriz $n \times n$ en 4 submatrices $n/2 \times n/2$ requiere copiar elementos:

$$T_{\text{división}}(n) = \Theta(n^2)$$

3.1.2 Multiplicaciones de submatrices

Cada una de las 7 multiplicaciones se realiza entre matrices de tamaño $n/2 \times n/2$ utilizando el algoritmo clásico:

$$T_{\text{mult}}(n) = 7 \cdot \left(\frac{n}{2}\right)^3 = \frac{7}{8}n^3 = \Theta(n^3)$$

3.1.3 Suma y resta de matrices

Cada operación de suma o resta entre dos matrices de tamaño $n/2 \times n/2$ tiene complejidad $\Theta(n^2)$. Se realizan aproximadamente 15–20 de estas operaciones:

$$T_{\text{suma/resta}}(n) = \Theta(n^2)$$

3.1.4 Unificación de la matriz final

Unir las submatrices $C_{11}, C_{12}, C_{21}, C_{22}$ en la matriz resultado toma:

$$T_{\text{unificación}}(n) = \Theta(n^2)$$

3.1.5 Resultado total

Sumando todos los componentes:

$$T(n) = \Theta(n^2) + \Theta(n^3) + \Theta(n^2) + \Theta(n^2) = \Theta(n^3)$$

3.2 Análisis recursivo

Este algoritmo no incluye ninguna recursivo por lo cual no se puede hacer análisis recursivo.
Para este algoritmo se puede reducir mucho la complejidad, generando una recursividad al realizar las multiplicaciones internas en el teorema mediante la utilización del mismo algoritmo de Strassen para estas, pero encontramos la idea muy tarde por lo cual no la implementamos, utilizando para el proyecto la versión secuencial que solo mejora el algoritmo a $o(n^{2.8})$.

3.3 Teorema de Brent

Por teorema de Brent al no realizar las multiplicaciones internas con el mismo algoritmo Strassen de forma recursiva, no mejoraría demasiado la paralización en casos donde $p \to \infty$ tal que consideramos:

$$t_\infty(n) = o(n^3)$$

Este representa el trabajo del algoritmo para el caso del camino critico el cual son las multiplicaciones internas

$$t_\infty(n) = o(n^3)$$

tal que por teorema de Brent la complejidad seguiría siendo de

$$o(n^3)$$

Nota: importante notar que esto es para el caso del algoritmo implementado, no si se tomara en cuenta las multiplicaciones internas utilizando Strassen.

4 Experimento y resultados

Tamaño de la matriz	Block	Block paralelizado	Strassen	Strassen paralelizado
32	0.347401	0.129025	0.197329	0.216528
64	1.56616	0.567053	1.18215	1.20384
128	7.60515	3.1937	8.25542	6.55335
256	59.3988	19.2546	48.6667	48.1188
512	428.99	131.988	374.416	171.959

Table 3: Tiempo de ejecución en ms

Tamaño de la matriz	Block	Block paralelizado	Strassen	Strassen paralelizado
32	0.057087	0.030453	0.0298025	0.0360361
64	0.0351772	0.128139	0.0615483	0.0614349
128	0.344313	0.195606	0.0931397	0.212866
256	4.58139	2.01565	0.155518	2.06291
512	1.44039	11.6549	5.037	6.13557

Table 4: Desviación estándar

Código disponible en: https://github.com/mgm1221/tar_1_paralela.git