COMPUTACIÓN PARALELA Y CÁLCULO DISTRIBUIDO
Entrega 1

Vojtěch Obhlídal

20 de mayo de 2022

El objetivo de la **primera entrega** era probar los cálculos paralelos en problemas estándar, que eran la suma y la multiplicación de dos matrices, donde ambas operaciónes se puede paralelizar fácilmente. La primera tarea se concentraba a programar estas operaciones. Por motivos de paralelización, fue necesario utilizar la indexación lineal y almacenar las matrices en forma de vector. Como tarea principal, se comparó el tiempo de cálculo de cada operación utilizando el multiprocesamiento frente al procesamiento secuencial, debido al mayor tamaño de las matrices. La comparación también se ha centrado en las diferentes formas de almacenar la matriz mediante un vector, que puede hacerse tanto con filas como con columnas. Por último, cabe mencionar que los cálculos se realizaron en C++ y para mayor estabilidad cada proceso se ejecutó 10 veces y los gráficos resultantes muestran la media y la desviación estándar.

1. Suma de matrices

En primer lugar, vamos a centrarnos en la operación más sencilla de las dos mencionadas anteriormente, la suma. Esta es la operación con complejidad $\mathcal{O}(n^2)$ para matrices cuadradas, con las que trabajaremos la mayor parte del tiempo. Al almacenar las matrices como un vector, esta operación puede realizarse en un solo ciclo, eliminando así la discusión de almacenar los datos mediante filas o columnas.
Para los cálculos de la suma de matrices, trabajaremos con las dimensiones 1000x1000, 10000x10000 y 20000x20000 para probar la escalabilidad débil y el speed up. En la Figura 1 vemos la comparación de la escalabilidad débil para matrices con las dimensiones mencionadas. Se puede observar que la velocidad de cálculo disminuye significativamente con un mayor número de hilos, sin embargo la tendencia a la baja se mitiga a partir de 8-10 hilos y la reducción de tiempo posterior es bastante pequeña. Por el contrario, a medida que aumenta el número de hilos, se produce un efecto negativo en este sencillo problema y el tiempo de cálculo aumenta al utilizar más hilos en comparación con el caso de menos hilos.

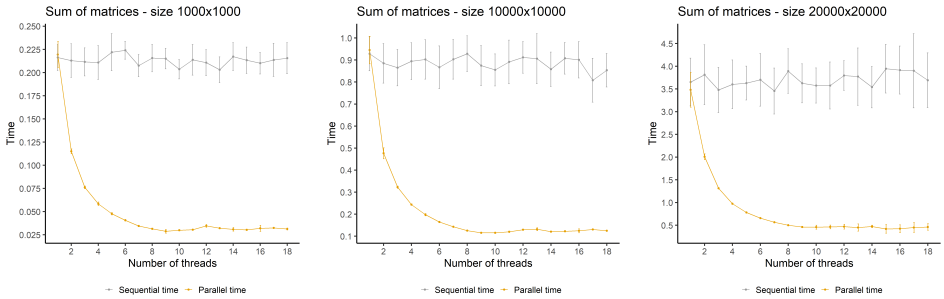

Figura 1: Escalabilidad débil de la suma de las matrices.

Podemos confirmar la afirmación anterior en la Figura 2, donde podemos ver el speed up.

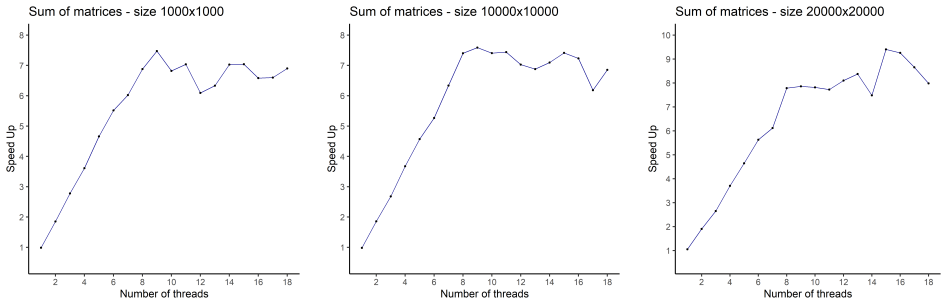

Figura 2: Speed Up de la suma de las matrices.

El último gráfico de la Figura 3 para la suma muestra la escalabilidad fuerte, donde aumentamos proporcionalmente el tamaño de las matrices con el número de hilos, de modo que cada hilo tiene la misma carga como en caso cuando se utiliza un solo hilo. El tamaño inicial de las matrices era de 2000x2000, y luego se aumentó la segunda dimensión de la matriz proporcionalmente.
Podemos ver que para más de 8 hilos, el tiempo de cálculo por hilo aumenta y tenemos que considerar si usar más hilos sigue siendo beneficioso desde el punto de vista computacional.

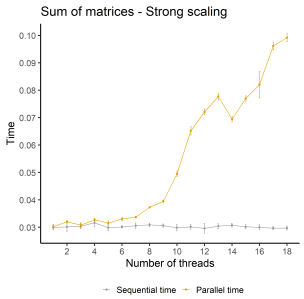

Figura 3: Escalabilidad fuerte de la suma de las matrices. Se han usado unas matrices de tamaño $2000K$ x 2000, donde el número K representa el número de threads.

2. Producto de matrices

El producto de matrices es una operación más exigente desde el punto de vista computacional con una complejidad $\mathcal{O}(n^3)$. Además, presentaremos todos los gráficos en versiones de almacenamiento de filas y columnas. Esto no se puede eludir de la misma manera que para la suma. Nos centraremos en matrices cuadradas de dimensiones 500x500, 1000x1000 y 1500x1500 para la mayoría de los cálculos, y luego utilizaremos matrices no cuadradas cuando la escalabilidad sea fuerte.
En los gráficos de la Figura 4 y la Figura 5 podemos ver el débil escalamiento para los tamaños de matriz mencionados. El primer gráfico es para guardar por filas, el segundo por columnas. En los gráficos podemos ver que el tiempo de cálculo disminuye rápidamente y empieza a estancarse en torno a los 30-40 hilos, que es un número significativamente mayor en comparación con la suma. La cuestión es que este cálculo ya requiere 3 bucles y se puede paralelizar mucho más.

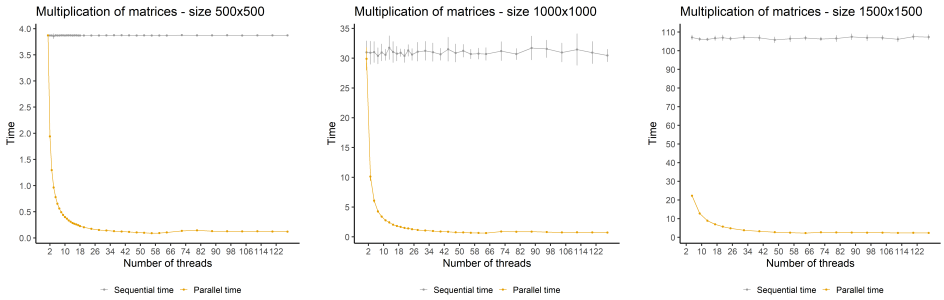

Figura 4: Escalabilidad débil del producto por filas de las matrices.

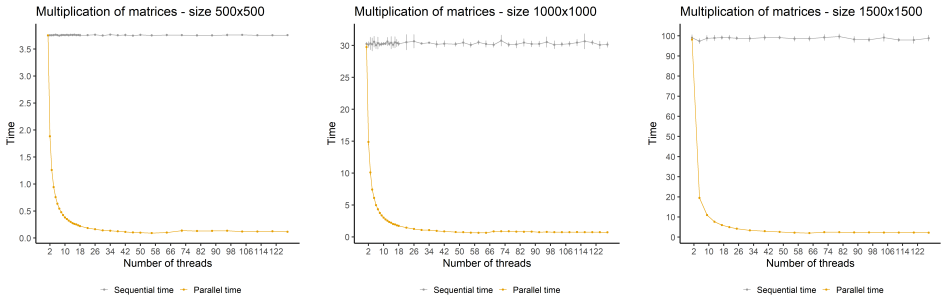

Figura 5: Escalabilidad débil del producto por columnas de las matrices.

En los gráficos de la Figura 6 y la Figura 7 observamos speed up, que confirma de nuevo los gráficos anteriores, y concluimos que el umbral a partir del cual no hay más mejora con hilos adicionales está entre 50 y 70 hilos, dependiendo del tamaño de las matrices multiplicadas.

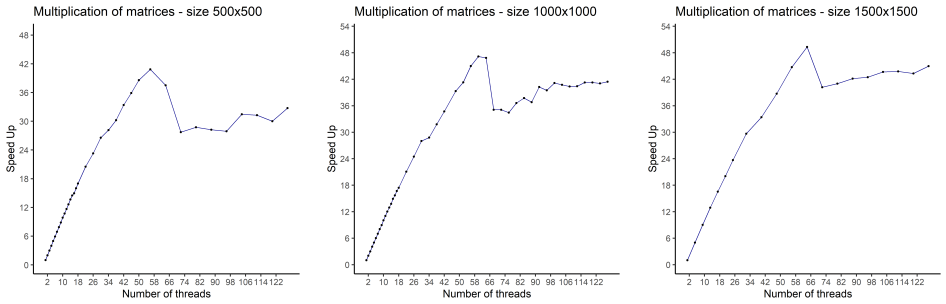

Figura 6: Speed Up del producto por columnas de las matrices.

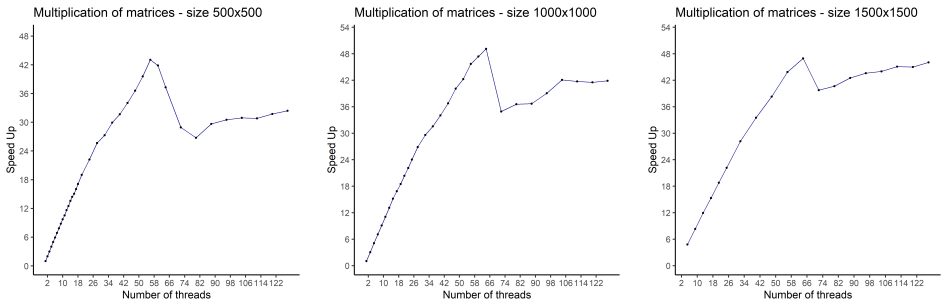

Figura 7: Speed Up del producto por filas de las matrices.

Los dos gráficos siguientes, Figura 8 y Figura 6, muestran la escalabilidad fuerte de ambas opciones de almacenamiento de datos. La carga por hilo aumenta ligeramente con el número de hilos, y luego la curva se rompe hasta un aumento brusco, lo que indica que ya no hay un aumento significativo de la velocidad de cálculo para un mayor número de hilos. El umbral de ambos gráficos es de unos 60 hilos.

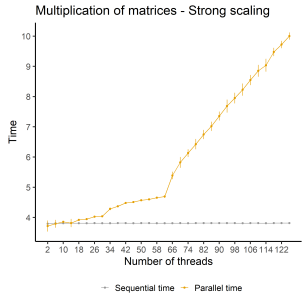

Figura 8: Escalabilidad fuerte del producto por las filas de las matrices. Se han usado unas matrices de tamaño $500K$ x 500, donde el número K representa el número de threads.

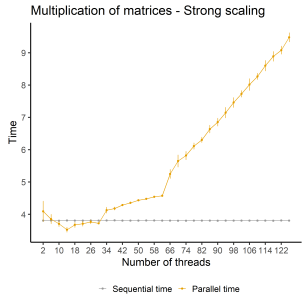

Figura 9: Escalabilidad fuerte del producto por las columnas de las matrices. Se han usado unas matrices de tamaño $500K$ x 500, donde el número K representa el número de threads.

La última serie de gráficos Figura 10, Figura 11 y Figura 12 muestran la comparación de la velocidad de cálculo cuando se almacena la matriz por filas o columnas. Para matrices de menor tamaño, la diferencia no puede discernirse gráficamente, pero para matrices de 1500x1500 está claro que el almacenamiento por columnas es más ventajoso debido al tiempo requerido.

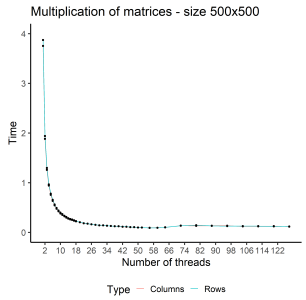

Figura 10: Comparacion de la escalabilidad débil del producto de las matrices, usándo el producto por filas y por columnas. El tamaño de matrices se ha usado 500x500.

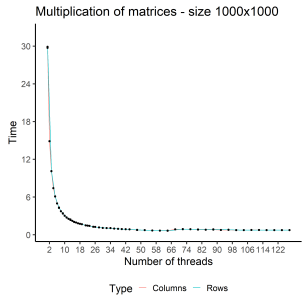

Figura 11: Comparacion de la escalabilidad débil del producto de las matrices, usándo el producto por filas y por columnas. El tamaño de matrices se ha usado 1000x1000.

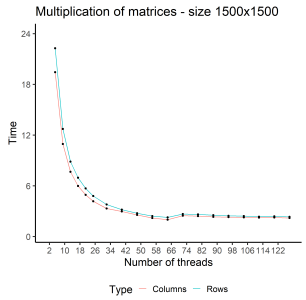

Figura 12: Comparacion de la escalabilidad débil del producto de las matrices, usándo el producto por filas y por columnas. El tamaño de matrices se ha usado 1500x1500.