COMPUTACIÓN PARALELA Y CÁLCULO DISTRIBUIDO
Entrega 2

Vojtěch Obhlídal

13 de junio de 2022

El objetivo de la **segunda entrega** era rehacer la multiplicación de dos matrices usándo partición en bloques. La primera tarea se concentraba a programar la multiplicación de matrices por columnas con todas posibles permutaciónes de los bucles (ijk,ikj,jik,jki,kij,kji). Luego evaluar el tiempo de computación y elegir la permutación que sale más eficiente. Como tarea principal, se programó la multiplicación de matrices con bloques z se evaluó el tiempo de computación, la escalabilidad fuerte y el speed up. Se comparó con la mutliplicación de matrices de la primera entrega. Por último, cabe mencionar que los cálculos se realizaron en C++ y para mayor estabilidad cada proceso se ejecutó 5 veces y los gráficos resultantes muestran la media y la desviación estándar.

1. Permutaciónes de bucles en multiplicación de matrices

En primer lugar, nos centramos en la comparación del tiempo de cálculo para diferentes permutaciones de bucles en la multiplicación de matrices. Debido al acceso a diferentes tipos de memoria, este tiempo puede variar de manera significativa.
Registro los tiempos resultantes y la varianza de las mediciones individuales en un Gráfico 1, que muestra que la disposición más adecuada de los índices es **kij** en este caso particular. La evaluación se realizó con los matrices cuadradas del tamaño 1000x1000 y solo un hilo.
Además, observamos que algunas permutaciones prácticamente no se desvían del tiempo medio cuando se vuelven a ejecutar, mientras que otras permutaciones fluctúan de forma muy significativa. La razón es, como ya hemos mencionado, la necesidad de acceder a diferentes tipos de memoria.

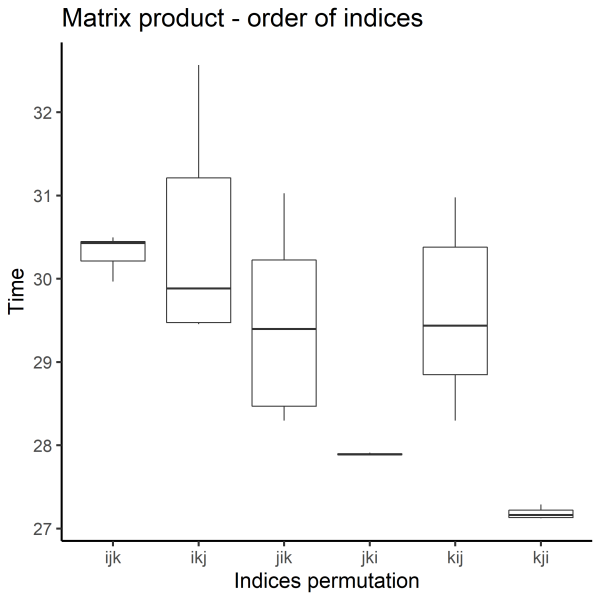

Figura 1: Boxplots de los permutaciones diferentes de bucles.

2. Algoritmo de multiplicación de matrices por bloques

En la segunda parte de este trabajo, implementamos el algoritmo de multiplicación de matrices por bloques. El algoritmo de multiplicación que utilizamos fue el de la mejor permutación, del que hablamos en la sección anterior.

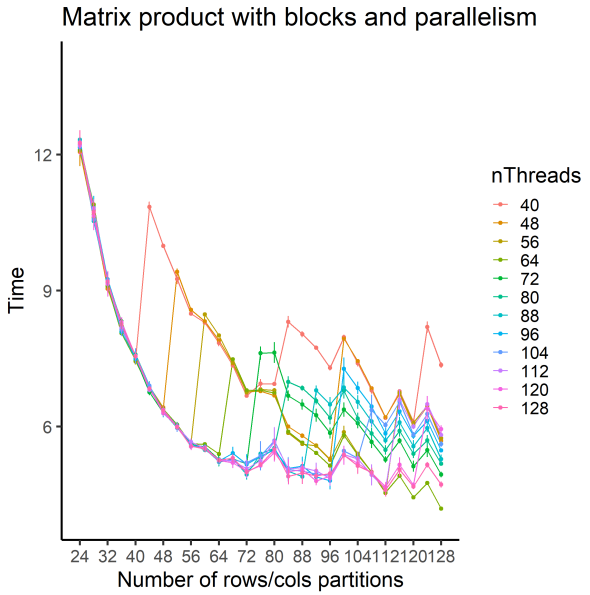

Figura 2: Evaluación de algoritmo con bloques con distintos numeros de particiónes y hilos. El número de bloques coresponde al número de particiónes al cudarado.

A continuación, evaluamos este algoritmo para diferentes números de bloques e hilos. El gráfico 2, que muestra estos valores, nos da algunas ideas interesantes.

- Entre número de hilos y número de particiónes hay una correlación significante. Para un número determinado de hilos, parece mejor utilizar un número de particiones igual a un múltiplo entero del número de hilos. Si a continuación se aumenta el número de particiones con una sola división más, el tiempo de cálculo aumentará drásticamente y luego comenzará a disminuir de nuevo con cada división adicional.
- De nuevo, vemos que la disminución del tiempo de cálculo se está estabilizando. Sin embargo, no encontramos un límite específico para la cuadrícula de parámetros que hemos elegido.

- Probablemente la mejor opción sería elegir k^2 hilos para un número determinado de particiones k, es decir, k^2 bloques. Esto significaría que cada bloque correspondería a un hilo.

Evaluamos el tiempo de cálculo para diferentes valores de particiones en matrices de 5000x5000 utilizando 64 hilos. Elegimos el método de representación gráfica de speed up y escalabilidad débil, donde comparamos los resultados con el algoritmo de Entrega 1 para calcular la multiplicación de matrices utilizando un algoritmo estándar paralelizado también entre 64 hilos.
En los gráfico 3 y 4 vemos la ya mencionada comparación de los dos tiempos de cálculo. Como tenemos 64 hilos, podemos ver que la velocidad aumenta en función del número de particiones hasta 64 divisiones, durante las cuales supera al algoritmo estándar. Luego viene una fuerte caída seguida de otro aumento hasta un máximo local para 128 particiones, al que probablemente seguiría otra caída.

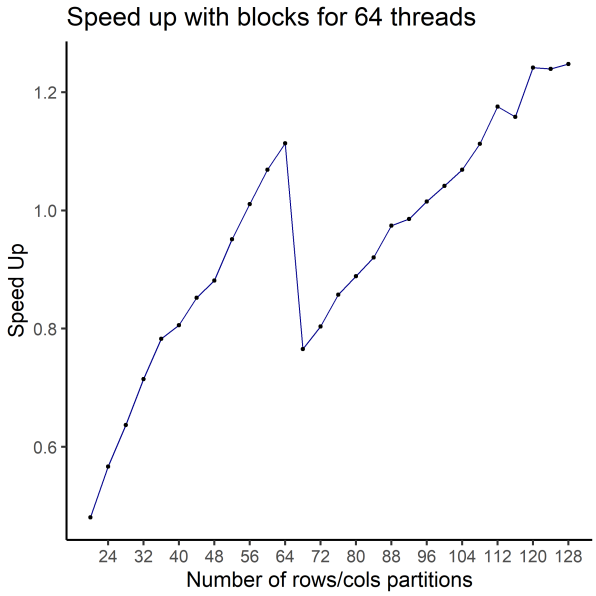

Figura 3: Speed up de la multiplicación por bloques con la referencia al agoritmo estandár.

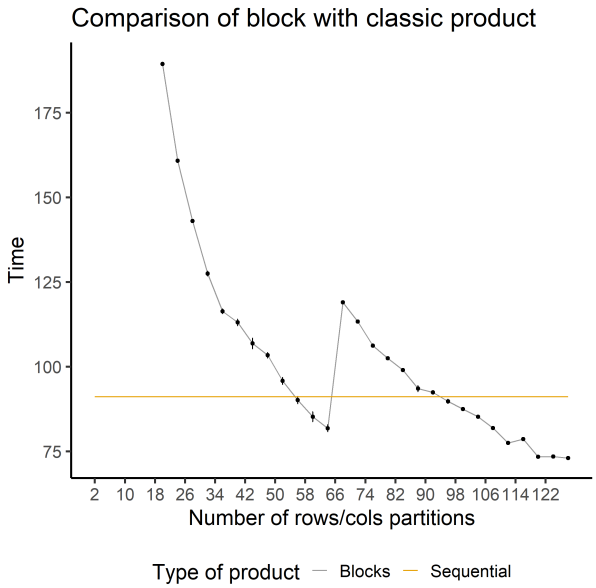

Figura 4: Tiempo de cálculo de la multiplicación por bloques comparado con el agoritmo estandár.