Práctico 1 del curso GPGPU:
Impacto del acceso a los datos en CPU

Introducción

Como se vio en la Clase 2 Computación Paralela, en el desempeño computacional de un algoritmo implementado en CPU, además de la complejidad propia del algoritmo, existe otro factor que tiene una gran incidencia en el tiempo de ejecución, el acceso a los datos. Este aspecto resultará crucial en las implementaciones en GPU.

En este práctico veremos experimentalmente el efecto del acceso a los datos en varios algoritmos implementados en CPU.

Ejercicio 1: Recorridas lineales

a) Suma de los elementos de un vector: Implemente a partir del archivo `Ej1.c` la función `int suma_vector(int *a)` que suma todos los elementos de un vector.

b) Suma de los elementos de la diagonal de una matriz: Implemente a partir del archivo `Ej1.c` la función `int suma_matriz(int **m)` que suma todos los elementos que están almacenados en la diagonal de una matriz. Note que una matriz de $N \times N$ elementos tiene N elementos en la diagonal.

c) Realice una evaluación experimental considerando los tiempos de ejecución de ambas funciones para los siguientes valores de N: 32, 64, 128, 256, 512, 1024, 2048, 4096 y 8192.

d) Analice las diferencias en los tiempos de ejecución. ¿A qué se deben?

Ejercicio 2: Recorridas en matrices

a) Suma de todos los elementos de una matriz por filas: Implemente a partir del archivo `Ej2.c` la función `int suma_porfilas(int **m)` que suma todos los elementos de una matriz por filas.

b) Suma de todos los elementos de una matriz por columnas: Implemente a partir del archivo `Ej2.c` la función `int suma_porcolumnas(int **m)` que suma todos los elementos de una matriz por columnas.

c) Realice una evaluación experimental considerando los tiempos de ejecución de ambas funciones para matrices de dimensiones $N \times N$ para los siguientes valores de N: 32, 64, 128, 256, 512, 1024, 2048, 4096 y 8192.

d) Analice las diferencias en los tiempos de ejecución. ¿A qué se deben?

Ejercicio 3: Multiplicación de matrices

La multiplicación de matrices es una de las operaciones básicas de Álgebra Lineal Numérica más utilizadas. Si $A \in \mathbb{R}^{m\times p}$ y $B \in \mathbb{R}^{p\times n}$, cada elemento c_{ij} de $C = A \times B$ puede computarse como el producto escalar de la fila i de A con la columna j de B según la expresión

$$c_{ij} = \sum_{k=1}^{p} a_{ik} \times b_{kj}. \tag{1}$$

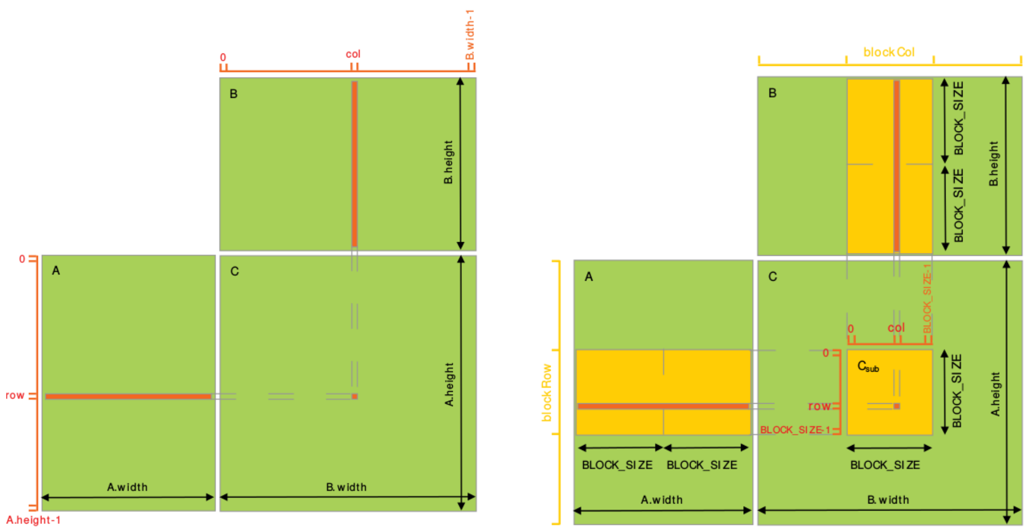

Figura 1: Multiplicación de matrices estándar y por bloques

Sin embargo, el orden en que se realizan las operaciones necesarias para computar el producto afecta el patrón de acceso a la memoria principal y a la caché.

Para maximizar la reutilización de los datos alojados en la caché, podemos dividir las matrices en *tiles* o bloques de tamaño $nb \times nb$ para cierto nb y realizar la multiplicación "por bloques". La multiplicación por bloques consiste en tomar todos los pares posibles de bloques (uno tomado de la matriz A y el otro de B), y multiplicarlos acumulando el resultado parcial en el lugar de la matriz C que corresponda. El proceso se describe en la Figura 1.

- a) Construya la versión de la multiplicación de matrices con el patrón de acceso usual, es decir, computando todas las operaciones correspondientes a una entrada c_{ij} antes de avanzar a la siguente. Las matrices no tienen por qué ser cuadradas. Registre los tiempos de ejecución para los siguientes tamaños:
    - A (2048 × 1024) y B (1024 × 2048)
    - A (1024 × 2048) y B (2048 × 1024)
    - A (2048 × 2048) y B (2048 × 2048)
- b) Construya otra versión que acceda por fila tanto a la matriz A como a la matriz B. Registre los tiempos de ejecución para los siguientes tamaños:
    - A (2048 × 2048) y B (2048 × 2048)
    - A (2048 × 1024) y B (1024 × 2048)

- A (1024 × 2048) y B (2048 × 1024)

- c) Construya otra versión que multiplique ambas matrices "por bloques". Registre los tiempos de ejecución para los siguientes tamaños de matriz y los siguientes tamaños de bloque:
  - A (2048 × 2048) y B (2048 × 2048)
  - A (2048 × 1024) y B (1024 × 2048)
  - A (1024 × 2048) y B (2048 × 1024)
  - $nb = 64, 128, 256, 512$

- d) Analice los resultados en términos de la reutilización de memoria. Puede utilizar herramientas de profiling (por ejemplo `perf` o `valgrind`) para analizar las métricas relacionadas con las memorias caché.

Entregar

1. Todos los archivos fuente que compongan la solución de los problemas planteados.
2. Makefile (preferentemente) o make.bat.
3. Un informe que contenga los resultados de la evaluación experimental, incluyendo una discusión sobre los resultados obtenidos.