Proyecto 1

Introducción a la computación paralela

Primer Semestre 2023, Prof. Cecilia Hernández

Fecha Inicio: Jueves 13 de Abril 2023.

Fecha Entrega: Jueves 4 de Mayo 2023 (23:59 hrs).

El tarea consiste en resolver el problema de multiplicación de matrices usando los algoritmos tradicionales, los amigables con el cache, y el de Strassen en forma secuencial y paralela. Y realizar un análisis de tiempo de ejecución analítico y experimental.

1. [0.5 puntos] Analice e implemente los algoritmos de multiplicación de matrices tradicional secuencial y paralela.

2. [0.5 puntos] Analice e implemente los algoritmos de multiplicación de matrices secuencial amigables con el cache.

3. [1.0 puntos] El algoritmo de multiplicación de matrices de Strassen resuelve el problema de multiplicación de matrices usado la técnica de dividir y conquistar para reducir su complejidad. Para estudiar el problema de multiplicación de matrices considere las siguientes alternativas de estudio. Considere dos matrices a multiplicar A y B cuya matriz resultante es C. Se pide analice los siguientes desarrolle los siguientes puntos, proporcione los algoritmos y los implemente. Además realice un análisis experimental para los dos casos datos, mas los casos de multiplicación de matrices tradicional mas el caso donde utilice mejor uso del cache. Proporcione un análisis experimentar donde incluya tiempos de ejecución para matrices cuadradas de dimensión N, variando el N en potencias de 2 hasta $N = 8192$ elementos enteros.

   a) [1.0 puntos] Primero, particione las matrices A y B en bloques de la siguiente manera y analice la complejidad del cómputo de la multiplicación usando esta descomposición.

   $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$, $B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$, $C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$, donde

   $C_{11} = A_{11} \times B_{11} + A_{12} \times B_{21}$
   $C_{12} = A_{11} \times B_{12} + A_{12} \times B_{22}$
   $C_{21} = A_{21} \times B_{11} + A_{22} \times B_{21}$
   $C_{22} = A_{22} \times B_{12} + A_{22} \times B_{22}$

   b) [1.0 puntos] El algoritmo de Strassen consiste en definir un conjunto de operaciones en base a submatrices de manera de reducir el número de multiplicaciones requeridas y luego usarlas en el cómputo de C. Note que el costo de computar multiplicaciones es el que se quiere reducir dado que son mucho mas caras de computar que las sumas. De acuerdo a las operaciones definidas por Strassen dadas a continuación determine el la complejidad de su tiempo de ejecución:
   $M_1 = (A_{11} + A_{22})(B_{11} + B_{22}$
   $M_2 = (A_{21} + A_{22})B_{11}$

$$M_3 = A_{11}(B_{12} - B_{22})$$
$$M_4 = A_{22}(B_{21} - B_{11})$$
$$M_5 = (A_{11} + A_{12})B_{22}$$
$$M_6 = (A_{21} - A_{11})(B_{11} + B_{12}$$
$$M_7 = (A_{12} - A_{22})(B_{21} + B_{22}$$

$$C_{11} = M_1 + M_4 - M_5 + M_7$$
$$C_{12} = M_3 + M_5$$
$$C_{21} = M_2 + M_4$$
$$C_{22} = M_1 - M_2 + M_3 + M_6$$

4. [4.0 puntos] Diseñe, analice e implemente los algoritmos paralelos para los algoritmos secuenciales del item 3. Para esta sección puede usar teorema de Brent con análsis asintótico.

5. Las implementaciones paralelas debe incluir uso de hebras y vectorización. Opcionalmente puede agregar openMP.

6. Realice un análisis experimental de sus implementaciones. Para ello ejecute por lo menos 10 veces cada configuración y reporte promedios y desviación estándar. Puede considerar matrices con dimensiones en potencias de 2.

7. La entrega debe incluir un informe y los códigos fuentes de sus implementaciones, junto con un readme.txt que diga como compilar y explique la ejecución incluyendo parámetros de configuración de manera de probar distintos dimensiones para las matrices y nivel de recursión.

8. Medición de tiempos en C++.

```
#include <chrono>

using namespace std;

auto start = chrono::high_resolution_clock::now();
auto finish = chrono::high_resolution_clock::now();
auto d = chrono::duration_cast<chrono::nanoseconds> (finish - start).count();
cout <<"total time "<< duration << " [ns]" << " \n";

// nota, si estima necesario puede usar milliseconds
// en lugar de nanoseconds
```