Tarea 2

Multiplicación de matrices

Con optimización a nivel caché

Miércoles, 7 de octubre 2020

Joel Harim Hernández Javier

Descripción

Compilar y ejecutar los programas MultiplicaMatriz.java y MultiplicaMatriz_2.java que vimos en clase, para los siguientes valores de N: 100, 200, 300, 500, 1000.

Utilizando Excel hacer una gráfica de dispersión (con líneas, sin marcadores) donde se muestre el tiempo de ejecución de ambos programas con respecto a N (N en el eje X y el tiempo en el eje Y).

Se deberá subir a la plataforma un documento PDF incluyendo el código fuente de los programas, la explicación de cada programa, la gráfica y los siguientes datos:

- ➔ Marca y modelo del CPU donde ejecutó los programas.
- ➔ Tamaño de la caché del CPU.
- ➔ Tamaño de la RAM.

Explicación de MatrixMultiplication.java

En este programa se implementa la multiplicación de matrices con el algoritmo clásico sin ningún tipo de optimización, por lo que las direcciones de memoria a las que se acceden en el momento en el que se acceden no tienen localidad espacial inmediata, por lo que el procesador se ve obligado a realizar más operaciones de escritura y de lectura al tener que pasar cierta información a la caché.

Se puede ver que los índices no son consecutivos respecto al arreglo lineal B:

```
for (int i = 0; i < N; ++i){
    for (int j = 0; j < N; ++j){
        for (int k = 0; k < N; ++k){
            C[i][j] += A[i][k] * B[k][i];
        }
    }
}
```

Explicación de MatrixMultiplication2.java

En este programa se implementa el algoritmo de multiplicación de matrices con una optimización, al sacar la traspuesta de B, los índices (y por lo tanto las regiones de memoria) se acceden de forma lineal y por lo tanto obtienen localidad espacial.

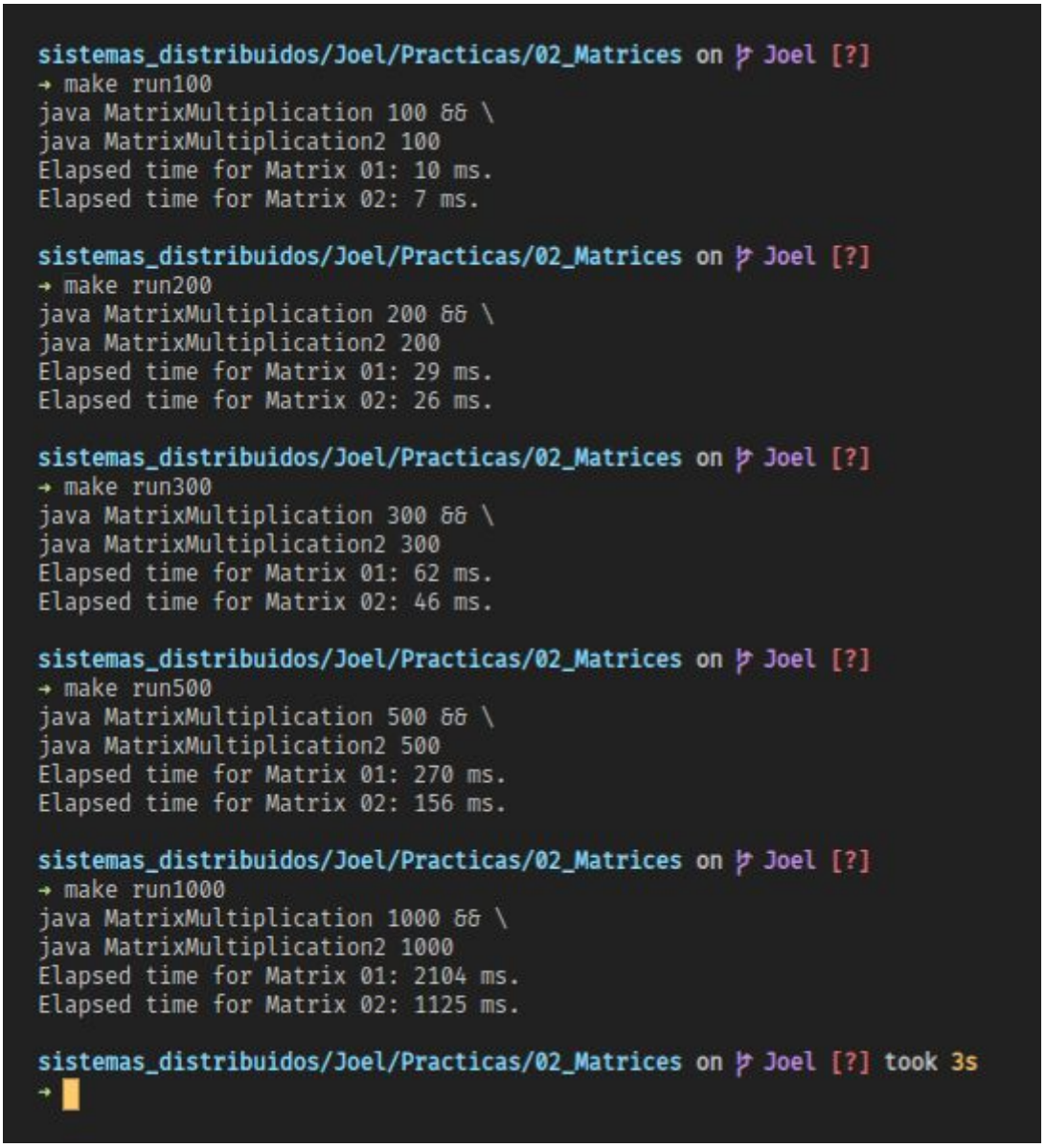
Se puede observar que los índices j, k apuntan a memoria que se accede de forma lineal, es decir, primero (0, 0), luego (0, 1) … hasta (1, 1), por lo que recorren el arreglo de principio a fin sin saltos:

```
// Transpose matrix B
for (int i = 0; i < N; ++i){
    for (int j = 0; j < N; ++j){
        int x = B[i][j];
        B[i][j] = B[j][i];
        B[j][i] = x;
     }
}

// Multiplication modified
for (int i = 0; i < N; ++i){
    for (int j = 0; j < N; ++j){
        for (int k = 0; k < N; ++k){
            C[i][j] += A[i][k] * B[j][k];
        }
    }
}
```

Gráfica

Captura



Datos del equipo

Marca	AMD
Modelo	Ryzen 5 3600
Tamaño de la caché	Caché L1: 384 KB Caché L2: 3MB Caché L3: 32 MB
Tamaño de la RAM	32 GB - T-FORCE @ 3000 Mhz (4x8GB, Quad Channel)

Código

MatrixMultiplication.java

```
public class MatrixMultiplication {
   static int N;
   static int[][] A, B, C;

   public static void main(String [] args) {
       if (args.length != 1) {
           System.err.println(ConsoleColors.ERR + " Number N not provided. Usage:
");
           System.err.println("java " + MatrixMultiplication.class.getName() + "
<N>");
           System.exit(1);
       }

       try {
           N = Integer.valueOf(args[0]);
           A = new int[N][N];
           B = new int[N][N];
           C = new int[N][N];
       } catch (NumberFormatException e) {
           System.err.println(ConsoleColors.ERR + " Wrong format, node must be a
number. Usage: ");
           System.err.println("java " + MatrixMultiplication.class.getName() + "
<node>");
           System.exit(1);
       }

       long time = System.currentTimeMillis();

       for (int i = 0; i < N; ++i) {
           for (int j = 0; j < N; ++j) {
               A[i][j] = 2 * i - j;
               B[i][j] = i + 2 * j;
               C[i][j] = 0;
           }
       }

       for (int i = 0; i < N; ++i){
           for (int j = 0; j < N; ++j){
               for (int k = 0; k < N; ++k){
                   C[i][j] += A[i][k] * B[k][i];
               }
           }
       }
```

```
        System.out.println("Elapsed time for Matrix 01: " +
(System.currentTimeMillis() - time) + " ms.");
    }
}
```

MatrixMultiplication2.java

```
public class MatrixMultiplication2 {
    static int N;
    static int[][] A, B, C;

    public static void main(String [] args) {
        if (args.length != 1) {
            System.err.println(ConsoleColors.ERR + " Number N not provided. Usage:
");
            System.err.println("java " + MatrixMultiplication2.class.getName() + "
<N>");
            System.exit(1);
        }

        try {
            N = Integer.valueOf(args[0]);
            A = new int[N][N];
            B = new int[N][N];
            C = new int[N][N];
        } catch (NumberFormatException e) {
            System.err.println(ConsoleColors.ERR + " Wrong format, node must be a
number. Usage: ");
            System.err.println("java " + MatrixMultiplication2.class.getName() + "
<node>");
            System.exit(1);
        }

        long time = System.currentTimeMillis();

        for (int i = 0; i < N; ++i) {
            for (int j = 0; j < N; ++j) {
                A[i][j] = 2 * i - j;
                B[i][j] = i + 2 * j;
                C[i][j] = 0;
            }
        }

        // Transpose matrix B
        for (int i = 0; i < N; ++i){
            for (int j = 0; j < N; ++j){
                int x = B[i][j];
```

```
                B[i][j] = B[j][i];
                B[j][i] = x;
            }
        }

        // Multiplication modified
        for (int i = 0; i < N; ++i){
            for (int j = 0; j < N; ++j){
                for (int k = 0; k < N; ++k){
                    C[i][j] += A[i][k] * B[j][k];
                }
            }
        }

        System.out.println("Elapsed time for Matrix 02: " +
(System.currentTimeMillis() - time) + " ms.");
    }
}
```