



Instituto Politécnico Nacional
Escuela Superior de Cómputo



Desarrollo de Sistemas Distribuidos

Tarea 2. Uso eficiente de la memoria cache

Nombre: Sampayo Hernández Mauro

Grupo: 4CV1

Profesor: Pineda Guerrero Carlos

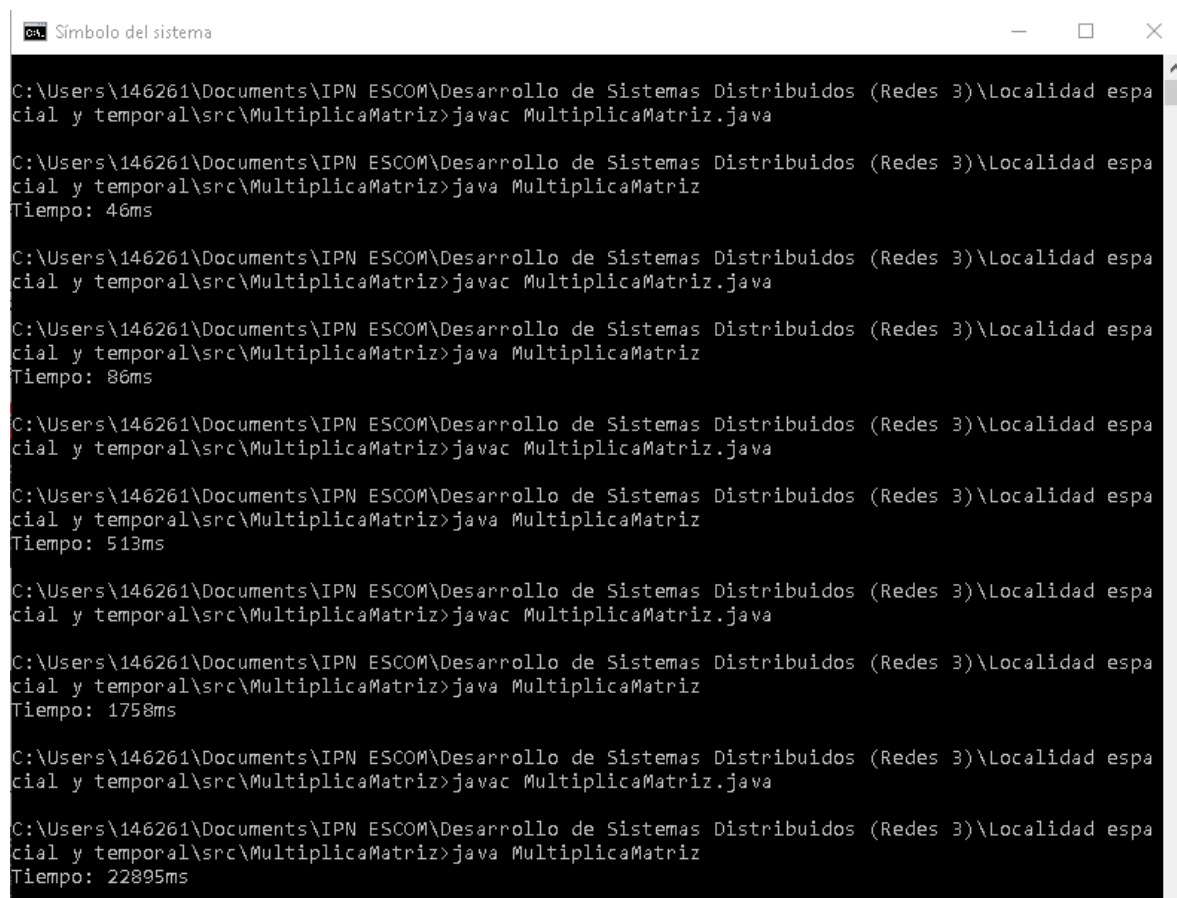
Introducción:

Para esta tarea se realizó la ejecución de los programas “MultiplicaMatriz.java” y “MultiplicaMatriz_2.java” los cuales realizan la multiplicación de dos matrices A y B, con los valores en N: 100, 200, 300, 500, 1000

La ejecución de los programas se realizó en una computadora de escritorio marca HP con las siguientes características:

- Procesador AMD A6-7310 APU with AMD Radeon R4 Graphics
- Tamaño de cache de 2MB
- Memoria RAM 8.00GB (6.93 GB utilizable)

A continuación, se muestran las ejecuciones del programa “MultiplicaMatriz.java” asignándole como valor a N 100, 200, 300, 500 y 1000 y sus respectivos tiempos de ejecución



```
C:\Users\146261\Documents\IPN ESCOM\Desarrollo de Sistemas Distribuidos (Redes 3)\Localidad espacial y temporal\src\MultiplicaMatriz>javac MultiplicaMatriz.java

C:\Users\146261\Documents\IPN ESCOM\Desarrollo de Sistemas Distribuidos (Redes 3)\Localidad espacial y temporal\src\MultiplicaMatriz>java MultiplicaMatriz
Tiempo: 46ms

C:\Users\146261\Documents\IPN ESCOM\Desarrollo de Sistemas Distribuidos (Redes 3)\Localidad espacial y temporal\src\MultiplicaMatriz>javac MultiplicaMatriz.java

C:\Users\146261\Documents\IPN ESCOM\Desarrollo de Sistemas Distribuidos (Redes 3)\Localidad espacial y temporal\src\MultiplicaMatriz>java MultiplicaMatriz
Tiempo: 86ms

C:\Users\146261\Documents\IPN ESCOM\Desarrollo de Sistemas Distribuidos (Redes 3)\Localidad espacial y temporal\src\MultiplicaMatriz>javac MultiplicaMatriz.java

C:\Users\146261\Documents\IPN ESCOM\Desarrollo de Sistemas Distribuidos (Redes 3)\Localidad espacial y temporal\src\MultiplicaMatriz>java MultiplicaMatriz
Tiempo: 513ms

C:\Users\146261\Documents\IPN ESCOM\Desarrollo de Sistemas Distribuidos (Redes 3)\Localidad espacial y temporal\src\MultiplicaMatriz>javac MultiplicaMatriz.java

C:\Users\146261\Documents\IPN ESCOM\Desarrollo de Sistemas Distribuidos (Redes 3)\Localidad espacial y temporal\src\MultiplicaMatriz>java MultiplicaMatriz
Tiempo: 1758ms

C:\Users\146261\Documents\IPN ESCOM\Desarrollo de Sistemas Distribuidos (Redes 3)\Localidad espacial y temporal\src\MultiplicaMatriz>javac MultiplicaMatriz.java

C:\Users\146261\Documents\IPN ESCOM\Desarrollo de Sistemas Distribuidos (Redes 3)\Localidad espacial y temporal\src\MultiplicaMatriz>java MultiplicaMatriz
Tiempo: 22895ms
```

Finalmente, se muestran las ejecuciones del programa “MultiplicaMatriz_2.java” asignándole como valor a N 100, 200, 300, 500 y 1000 y sus respectivos tiempos de ejecución

```
Simbolo del sistema
C:\Users\146261\Documents\IPN ESCOM\Desarrollo de Sistemas Distribuidos (Redes 3)\Localidad espacial y temporal\src\MultiplicaMatriz_2>javac MultiplicaMatriz_2.java
C:\Users\146261\Documents\IPN ESCOM\Desarrollo de Sistemas Distribuidos (Redes 3)\Localidad espacial y temporal\src\MultiplicaMatriz_2>java MultiplicaMatriz_2
Tiempo: 36ms
C:\Users\146261\Documents\IPN ESCOM\Desarrollo de Sistemas Distribuidos (Redes 3)\Localidad espacial y temporal\src\MultiplicaMatriz_2>javac MultiplicaMatriz_2.java
C:\Users\146261\Documents\IPN ESCOM\Desarrollo de Sistemas Distribuidos (Redes 3)\Localidad espacial y temporal\src\MultiplicaMatriz_2>java MultiplicaMatriz_2
Tiempo: 56ms
C:\Users\146261\Documents\IPN ESCOM\Desarrollo de Sistemas Distribuidos (Redes 3)\Localidad espacial y temporal\src\MultiplicaMatriz_2>javac MultiplicaMatriz_2.java
C:\Users\146261\Documents\IPN ESCOM\Desarrollo de Sistemas Distribuidos (Redes 3)\Localidad espacial y temporal\src\MultiplicaMatriz_2>java MultiplicaMatriz_2
Tiempo: 164ms
C:\Users\146261\Documents\IPN ESCOM\Desarrollo de Sistemas Distribuidos (Redes 3)\Localidad espacial y temporal\src\MultiplicaMatriz_2>javac MultiplicaMatriz_2.java
C:\Users\146261\Documents\IPN ESCOM\Desarrollo de Sistemas Distribuidos (Redes 3)\Localidad espacial y temporal\src\MultiplicaMatriz_2>java MultiplicaMatriz_2
Tiempo: 298ms
C:\Users\146261\Documents\IPN ESCOM\Desarrollo de Sistemas Distribuidos (Redes 3)\Localidad espacial y temporal\src\MultiplicaMatriz_2>javac MultiplicaMatriz_2.java
C:\Users\146261\Documents\IPN ESCOM\Desarrollo de Sistemas Distribuidos (Redes 3)\Localidad espacial y temporal\src\MultiplicaMatriz_2>java MultiplicaMatriz_2
Tiempo: 1807ms
C:\Users\146261\Documents\IPN ESCOM\Desarrollo de Sistemas Distribuidos (Redes 3)\Localidad espacial y temporal\src\MultiplicaMatriz_2>
```

MultiplicarMatriz.java:

- **Código Fuente**

```
public class MultiplicaMatriz {
    static int N = 1000;
    static int[][] A= new int[N][N];
    static int[][] B= new int[N][N];
    static int[][] C= new int[N][N];

    public static void main(String[] args) {
        long t1 = System.currentTimeMillis();

        //inicializa las matrices A y B
        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;
            }

        //Multiplica la matriz A y la matriz B, el resultado queda en la
        matriz C
        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][j];

        long t2 = System.currentTimeMillis();
        System.out.println("Tiempo: "+(t2-t1)+"ms");
    }
}

```

- **Explicación del Código**

El siguiente programa multiplica dos matrices cuadradas A y B utilizando el algoritmo estándar (renglón por columna), en el código las matrices tienen un tamaño de 1000x1000, mas sin embargo el valor de esta variable fue cambiado conforme los requerimientos de la tarea.

En el programa se realiza la inicialización de 3 arreglos de dos dimensiones A, B y C, las cuales son inicializadas por medio de un ciclo for. Posteriormente se realiza la multiplicación de la matriz A y la matriz B, cuyo resultado se guarda en la matriz C

Cabe destacar que este algoritmo resulta ineficiente, pues si bien, el acceso a la matriz A es muy eficiente debido a que los elementos de la matriz A se leen secuencialmente, es decir, el acceso es por renglón; el acceso a la matriz B resulta ser muy ineficiente al ser que Java almacena las matrices en la memoria como renglones, por lo que acceder por columnas para el caso de esta matriz resulta bastante tardado, ya que cada vez que se accede un elemento de la matriz B, se transfiere una línea de cache completa de la RAM a la cache.

MultiplicarMatriz 2.java:

- **Código Fuente**

```

public class MultiplicaMatriz_2 {

    static int N = 1000;

    static int[][] A= new int[N][N];

    static int[][] B= new int[N][N];

    static int[][] C= new int[N][N];

    public static void main(String[] args) {

        long t1 = System.currentTimeMillis();

        //inicializa las matrices A y B
    }
}

```

```

    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;
        }

    //Transpone la matriz B, la matriz transpuesta queda en 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;
    }

    //Multiplica la matriz A y la matriz B, el resultado queda en la
matriz C
    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];

    long t2 = System.currentTimeMillis();
    System.out.println("Tiempo: "+(t2-t1)+"ms");
}
}

```

- **Explicación del Código**

El siguiente programa, al igual que el anterior multiplica dos matrices cuadradas A y B utilizando el algoritmo estándar (renglón por columna), sin embargo este cuenta con un pequeño cambio al momento de acceder a los elementos de la matriz B intercambiando los índices usados para acceder los elementos de la matriz B, por medio de su transpuesta; resultando así en un acceso más eficiente

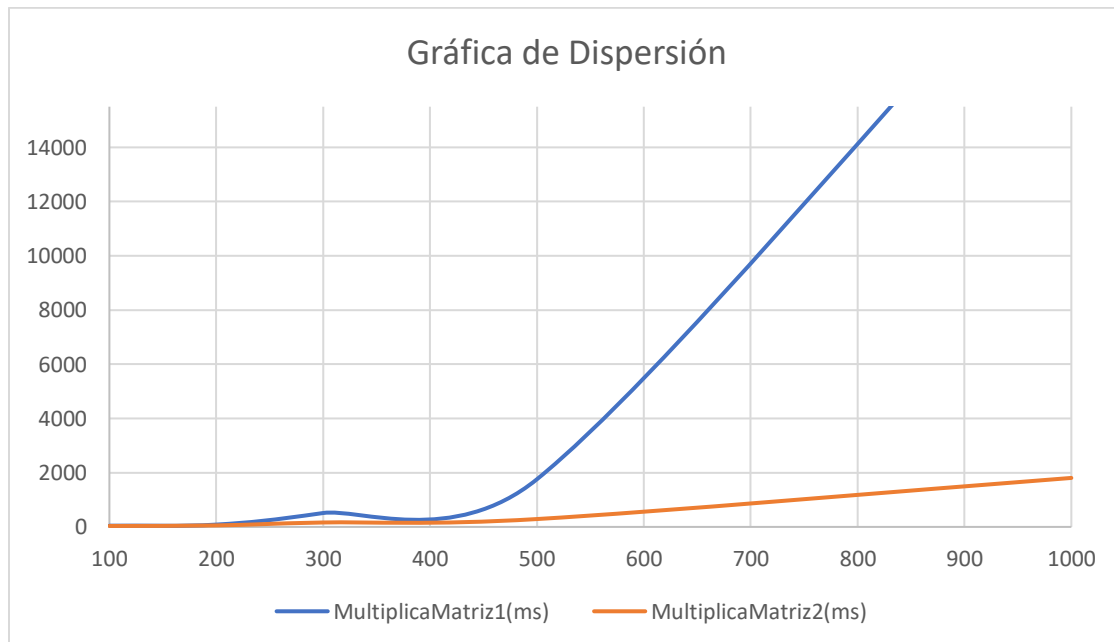
a los elementos de la matriz B, debido a que ahora se leen los elementos de B en forma secuencial, aumentando la localidad espacial y temporal de los datos.

En el programa se realiza la inicialización de 3 arreglos de dos dimensiones A, B y C, las cuales son inicializadas por medio de un ciclo for. Posteriormente se transpone la matriz B, sobrescribiendo así los datos de la matriz B con la traspuesta de esta misma. Finalmente se realiza la multiplicación de la matriz A y la matriz B, cuyo resultado se guarda en la matriz C

Grafica:

Al ejecutar los programas “MultiplicaMatriz.java” y “MultiplicaMatriz_2.java” con los valores de N es 100, 200, 300, 500 y 1000 se obtuvieron los siguientes tiempos (expresados en ms) los cuáles se enlistan en la siguiente tabla y se grafican a continuación:

N	MultiplicaMatriz1(ms)	MultiplicaMatriz2(ms)
100	46	36
200	86	56
300	513	164
500	1758	290
1000	22895	1807



Conclusión:

La ejecución de el programa "MultiplicaMatriz_2.java" al tener un acceso más eficiente a los datos de la matriz B al estos ser leídos también de manera secuencial siempre será más rápido que la ejecución del programa "MultiplicaMatriz.java" en el cuál el acceso de los datos de la matriz B se realiza por columna, notándose esta diferencia de rapidez entre más grande sea el tamaño de las matrices que se van a multiplicar; haciendo de estos programas un perfecto ejemplo de la importancia de plantear siempre un algoritmo tomando en consideración la localidad espacial y temporal de los datos para así aumentar la eficiencia de éste último al máximo.