Instituto Politécnico Nacional.
Escuela Superior de Cómputo
(ESCOM).

Desarrollo de Sistemas Distribuidos.

Tarea 2.
“Uso eficiente de la memoria cache”.

Profesor Carlos Pineda
Alumno: Sánchez Martínez Eli
Grupo: 4CM5

07/10/2020

Objetivo: Uso de la memoria cache a través de dos programas con diferente implementación, pero la misma salida, observando como hacer uso eficiente de la memoria cache.

Datos:

- Marca: AMD
- Modelo: E2-1800 APU with Radeon
- Tamaño de la cache: 32k
- Tamaño de la RAM: 4GB

Código:

MultiplicaMatriz.java

Esta operación funciona correctamente sin embargo es ineficiente ya que cada vez que se accede a un elemento de la matriz se necesita transferir una línea completa de cache.

```java
//MultiplicaMatriz 1
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");
  }
```

```
33. }
```

Multiplicación_2.java

Este programa es mas eficaz ya que los elementos de una de las matrices se puede leer de forma secuencial, lo cual nos ayuda a los datos temporales y las localidades para acceder a estos.

```
1.  //UpGraded Multiplicacion
2.  public class MultiplicaMatriz_2 {
3.    static int N = 1000;
4.    static int[][] A = new int[N][N];
5.    static int[][] B = new int[N][N];
6.    static int[][] C = new int[N][N];
7.
8.    public static void main(String[] args)
9.    {
10.     long t1 = System.currentTimeMillis();
11.
12.     // inicializa las matrices A y B
13.
14.     for (int i = 0; i < N; i++)
15.       for (int j = 0; j < N; j++)
16.       {
17.         A[i][j] = 2 * i - j;
18.         B[i][j] = i + 2 * j;
19.         C[i][j] = 0;
20.       }
21.
22.     // transpone la matriz B, la matriz traspuesta queda en B
23.
24.     for (int i = 0; i < N; i++)
25.       for (int j = 0; j < i; j++)
26.       {
27.         int x = B[i][j];
28.         B[i][j] = B[j][i];
29.         B[j][i] = x;
30.       }
31.
32.     // multiplica la matriz A y la matriz B, el resultado queda en la matriz C
33.     // notar que los indices de la matriz B se han intercambiado
34.
35.     for (int i = 0; i < N; i++)
36.       for (int j = 0; j < N; j++)
37.         for (int k = 0; k < N; k++)
38.            C[i][j] += A[i][k] * B[j][k];
39.
40.     long t2 = System.currentTimeMillis();
41.     System.out.println("Tiempo: " + (t2 - t1) + "ms");
42.   }
43. }
```

Comparación:

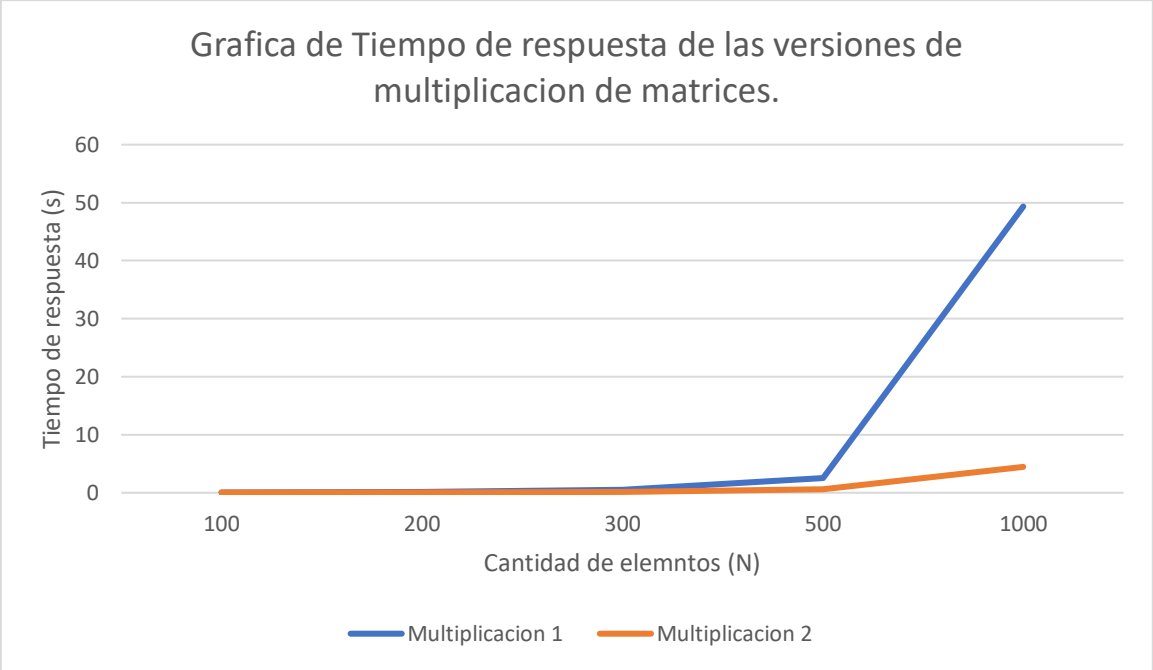
Para comparar entre el primer programa y el segundo, se le fueron dando valores destinos al tamaño de nuestras matrices:

N	Matriz Multiplica	Matriz Multiplica 2
100	42ms =0.042s	46ms = 0.046s
200	140ms=0.140s	75ms=0.075ms
300	552ms=0.552s	177ms=0.177s
500	2560ms=2.560s	583ms=0.583s
1000	49337ms=49.337s	4461ms=4.461s

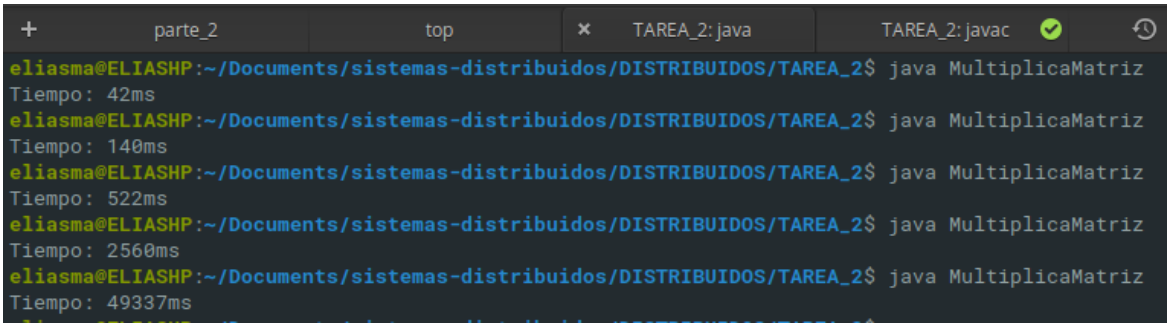
Podemos observar que el primer código tiene una complejidad en el peor de los casos de $O(n^3)$ mientras que el segundo programa se podría mantener con una complejidad de $O(n^2)$ pero en el peor de los casos puede ser de $O(n^3)$.

Esto se debe a como se accede a la memoria cache para sacar los datos, por ejemplo el primer programa va resolviendo la matriz renglón por renglón debido a esto hace ineficiente este programa, pero el segundo para darle solución a este problema es transponer la matriz y esto nos da una mayor flexibilidad tanto para realizar las operaciones como la memoria cache, para que tenga menor numero de operaciones a realizar.

Grafica:

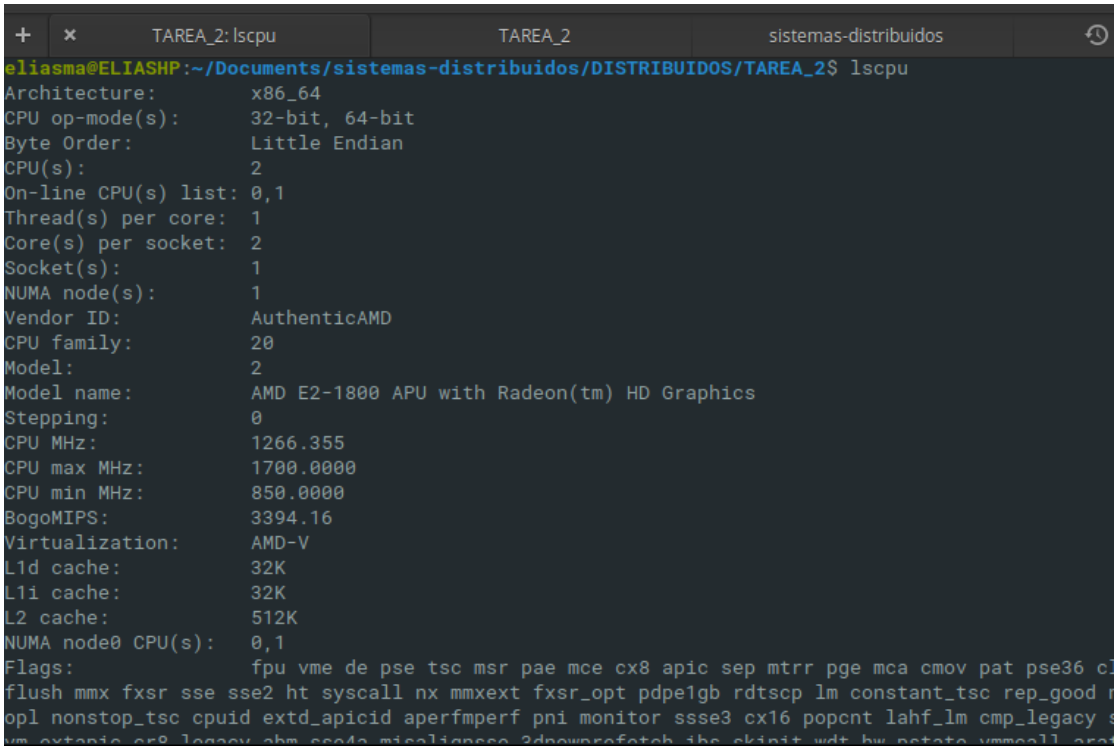


Pruebas:

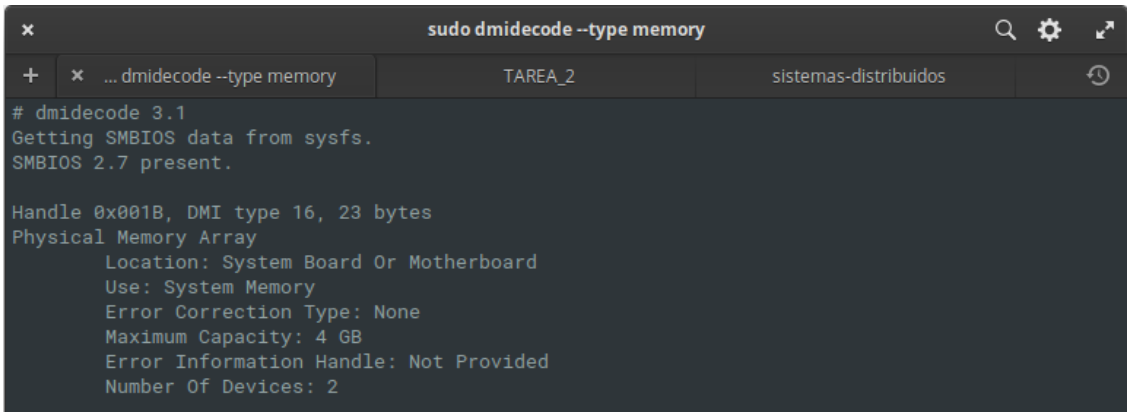


***1**. Tiempo de respuesta del primer programa con N de tamaño.*

***2**. Tiempo de respuesta del segundo programa con N tamaño.*



3.Información de la computador en la cual se realizó la prueba.



4. Información de la RAM