

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.

1. //MultiplicaMatriz 1
2. **class** MultiplicaMatriz
3. {
4. **static** **int** N = 1000;
5. **static** **int**[][] A = **new** **int**[N][N];
6. **static** **int**[][] B = **new** **int**[N][N];
7. **static** **int**[][] C = **new** **int**[N][N];
9. **public** **static** **void** main(String[] args)
10. {
11. **long** t1 = System.currentTimeMillis();
13. // inicializa las matrices A y B
15. **for** (**int** i = 0; i < N; i++)
16. **for** (**int** j = 0; j < N; j++)
17. {
18. A[i][j] = 2 \* i - j;
19. B[i][j] = i + 2 \* j;
20. C[i][j] = 0;
21. }
23. // multiplica la matriz A y la matriz B, el resultado queda en la matriz C
25. **for** (**int** i = 0; i < N; i++)
26. **for** (**int** j = 0; j < N; j++)
27. **for** (**int** k = 0; k < N; k++)
28. C[i][j] += A[i][k] \* B[k][j];
30. **long** t2 = System.currentTimeMillis();
31. System.out.println("Tiempo: " + (t2 - t1) + "ms");
32. }
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];
8. **public** **static** **void** main(String[] args)
9. {
10. **long** t1 = System.currentTimeMillis();
12. // inicializa las matrices A y B
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. }
22. // transpone la matriz B, la matriz traspuesta queda en B
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. }
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
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];
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:

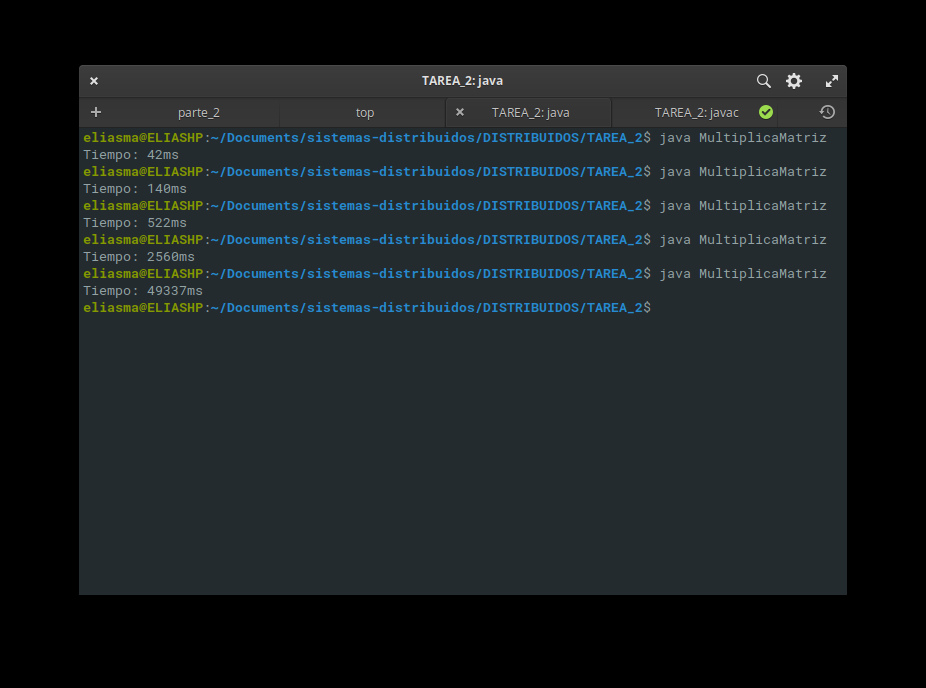
|  |  |  |
| --- | --- | --- |
| N | Matriz Multiplica | Matriz Multiplica 2 |
| 100 | 42ms =0.042s | 46ms = 0.046s |
| 200 | 140ms=0.140s | 75ms=0.075ms |
| 300 | 552ms=0.552s | 177ms=0.177s |
| 500 | 2560ms=2.560s | 583ms=0.583s |
| 1000 | 49337ms=49.337s | 4461ms=4.461s |

Podemos observar que el primer código tiene una complejidad en el peor de los casos de mientras que el segundo programa se podría mantener con una complejidad de pero en el peor de los casos puede ser de .

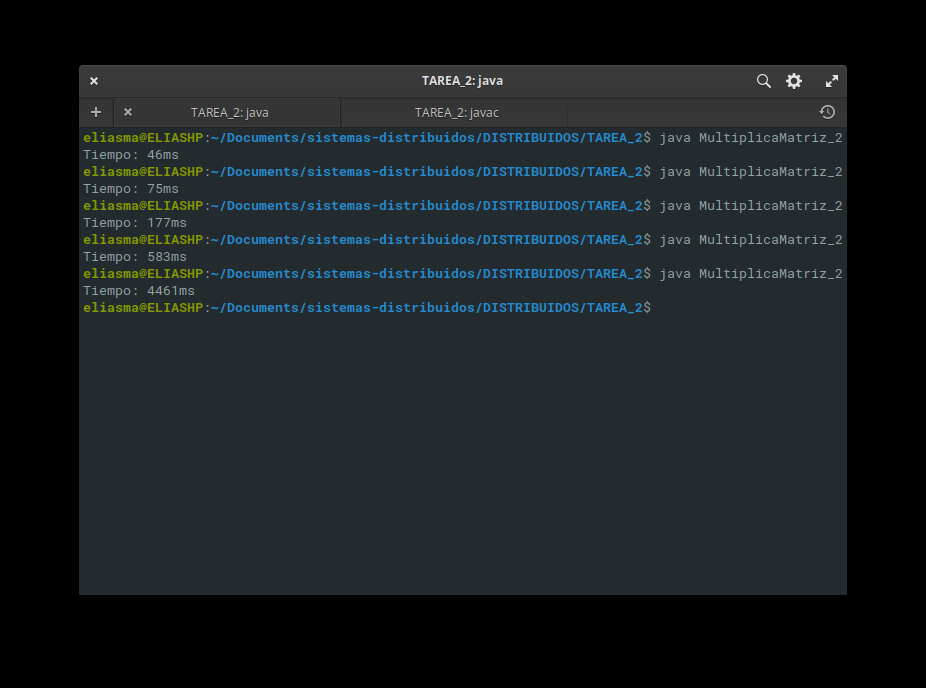
Esto se debe a como se accede a la memoria cache para sacar los datos, por ejemplo el primer programa va resolviendo la matriz renglón por renglón debido a esto hace ineficiente este programa, pero el segundo para darle solución a este problema es transponer la matriz y esto nos da una mayor flexibilidad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

Grafica:

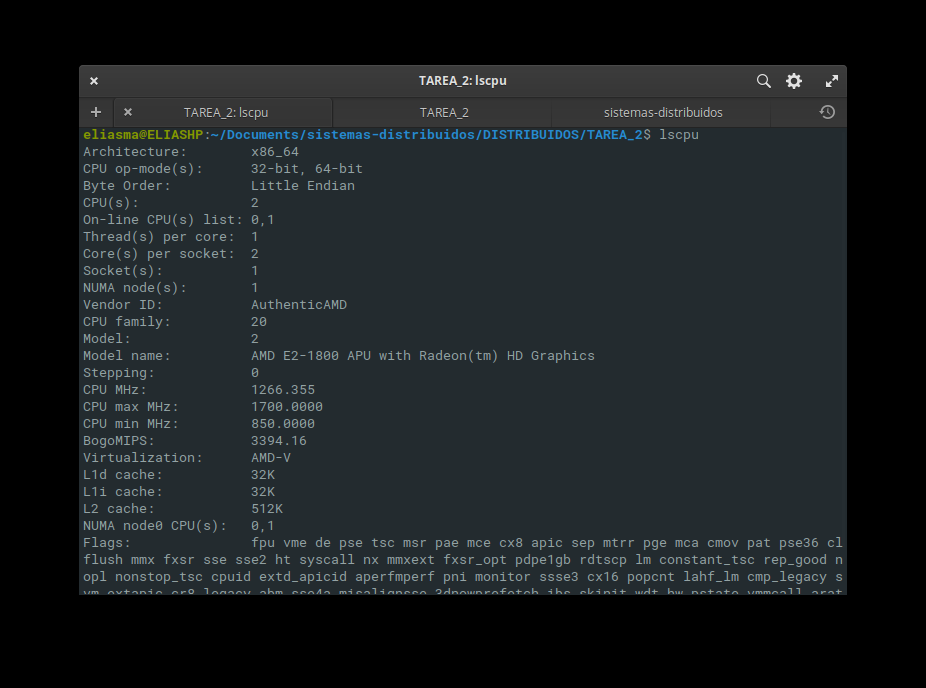
**Pruebas:**



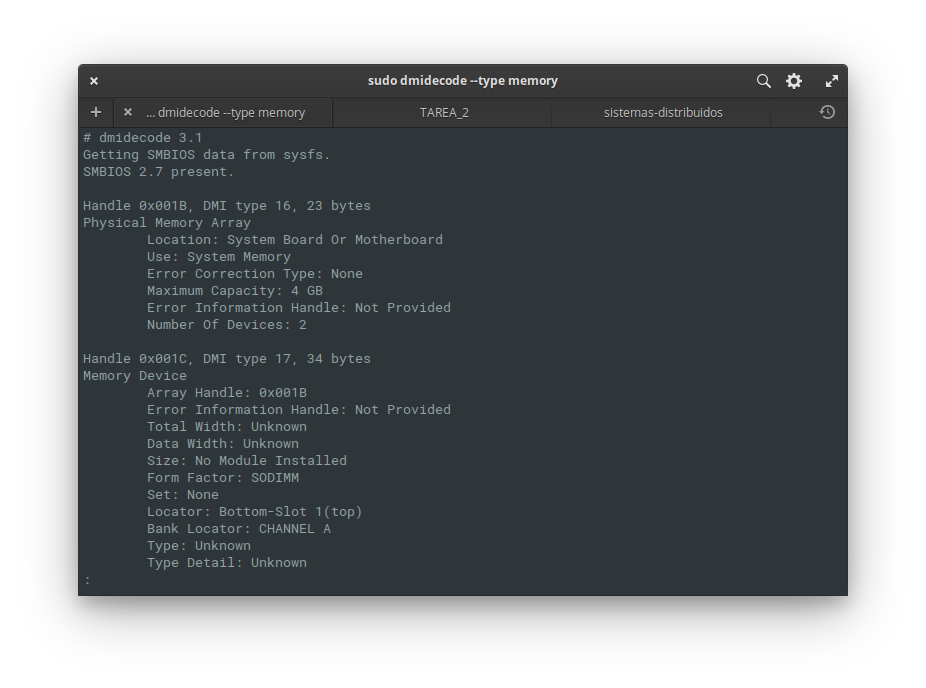
**1**. Tiempo de respuesta del primer programa con N de tamaño.



**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