

**Instituto Politecnico Nacional**

**ESCOM “ESCUELA SUPERIOR DE CÓMPUTO”**

*DESARROLLO DE SISTEMAS DISTRIBUIDOS*

*TAREA 2: Uso eficiente de la memoria caché*

PROFE: PINEDA GUERRERO CARLOS

ALUMNO: Rojas Alvarado Luis Enrique

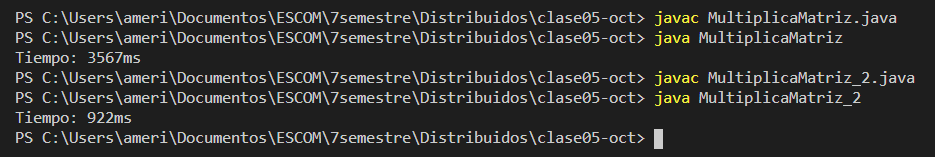
GRUPO: 4CM5

1.- Objetivo

Compilar y ejecutar los programas de MultiplicaMatriz.java y MultiplicaMatriz\_2.java proporcionados por el profesor para comparar la eficiencia (el tiempo en el que logra hacer la multiplicación de matrices) de estos, para un tamaño de matriz de 100, 200, 300, 500 y 1000.

2.- Pruebas

Se compila y se ejecutan ambos programas para comprobar su eficiencia.\*



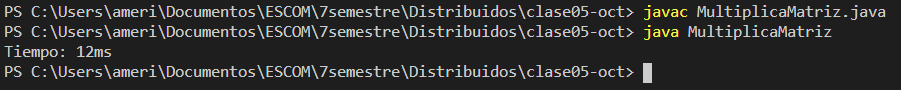
Debemos tomar en cuenta que java almacena las matrices en forma de renglones.

El programa MultiplicaMatriz.java, cada que se accede a la matriz B, se transfiere una línea completa de la RAM a la caché, lo que hace menos eficiente.  
El acceso a la matriz A es más eficiente debido a que sus elementos se leen secuencialmente (por renglones), como se encuentra almacenada en memoria.

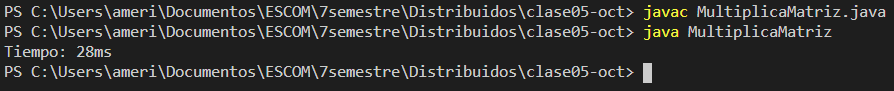
En el programa MultiplicaMatriz\_2.java se modifica de manera que el acceso a la matriz B sea por renglones y no por columnas, simplemente intercambiando los índices con los que se accede a la matriz B (se transpone la matriz). Lorando de esta manera un acceso a los elementos de la matriz B más eficiente, debido a que ahora se leen en forma de renglones y no por columnas (como está guardada en memoria).

2.1. Diferentes tamaños de matriz.

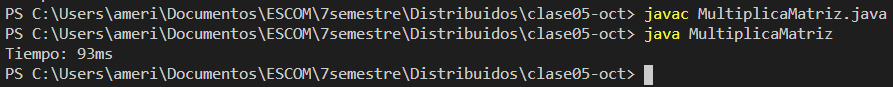
Se mide la eficiencia del programa MultiplicaMatriz.java con el tamaño de matriz de 100:



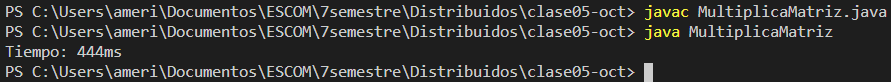
200:



300:



500:



1000:

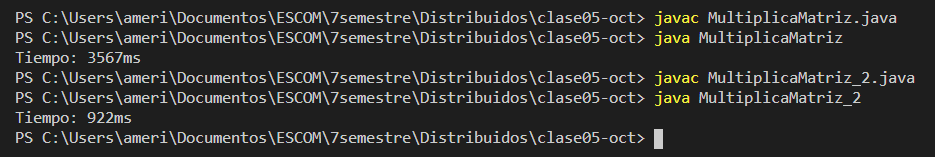
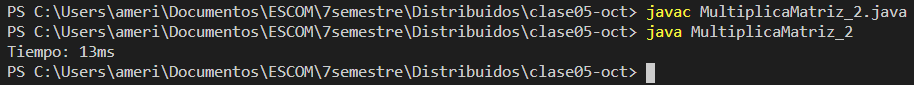


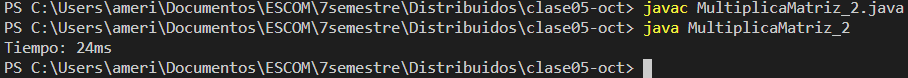
Tabla 1. Eficiencia MultiplicaMatriz.java

|  |  |
| --- | --- |
| TAMAÑO DE MATRIZ | TIEMPO MEDIDO |
| 100 | 12ms |
| 200 | 28ms |
| 300 | 93ms |
| 500 | 444ms |
| 1000 | 3567ms |

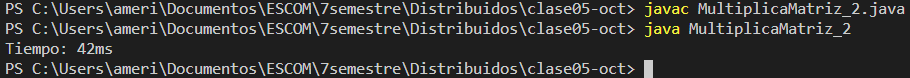
Se mide la eficiencia del programa MultiplicaMatriz\_2.java con el tamaño de matriz de 100:



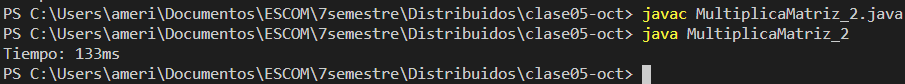
200:



300:



500:



1000:

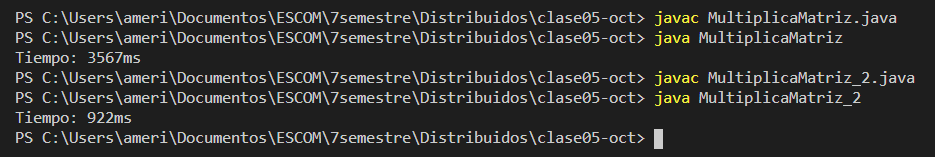
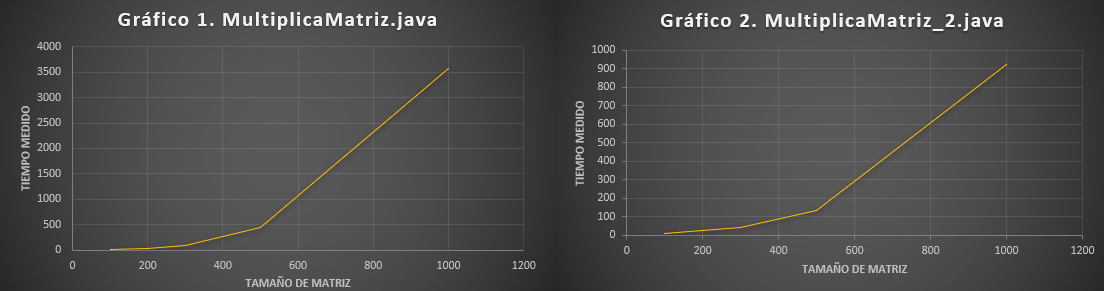


Tabla 2. Eficiencia MultiplicaMatriz\_2.java

|  |  |
| --- | --- |
| TAMAÑO DE MATRIZ | TIEMPO MEDIDO |
| 100 | 13ms |
| 200 | 24ms |
| 300 | 42ms |
| 500 | 133 |
| 1000 | 922ms |

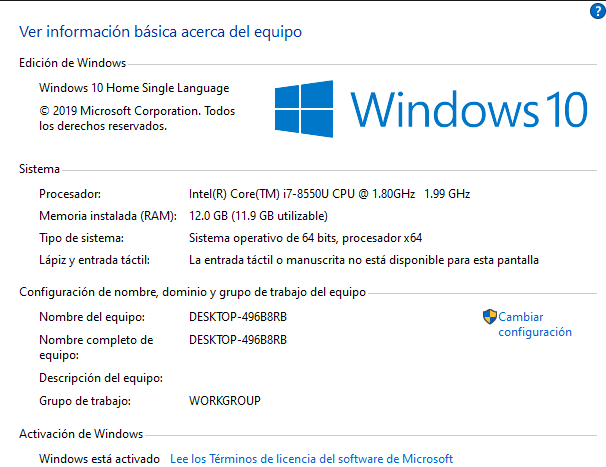
Forma 1. Comparativo de ambos programas:



3.- ESPECIFICACIONES TÉCNICAS DE EQUIPO\*

El equipo que fue utilizado para esta cuenta con las siguientes especificaciones:

* Procesador intel core i7-8550U 1.8GHz hasta 4.0 GHz
* Memoria caché 8 M
* Memoria RAM de 12 GB DDR4



ANEXO1

MultiplicaMatriz.java

1. **import** java.io.\*;
3. **class** MultiplicaMatriz{
4. **static** **int** N = 300;
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. }

ANEXO 2

MultiplicaMatriz\_2.java

1. **import** java.io.\*;
3. **class** MultiplicaMatriz\_2
4. {
5. **static** **int** N = 500;
6. **static** **int**[][] A = **new** **int**[N][N];
7. **static** **int**[][] B = **new** **int**[N][N];
8. **static** **int**[][] C = **new** **int**[N][N];
10. **public** **static** **void** main(String[] args)
11. {
12. **long** t1 = System.currentTimeMillis();
14. // inicializa las matrices A y B
16. **for** (**int** i = 0; i < N; i++)
17. **for** (**int** j = 0; j < N; j++)
18. {
19. A[i][j] = 2 \* i - j;
20. B[i][j] = i + 2 \* j;
21. C[i][j] = 0;
22. }
24. // transpone la matriz B, la matriz traspuesta queda en B
26. **for** (**int** i = 0; i < N; i++)
27. **for** (**int** j = 0; j < i; j++)
28. {
29. **int** x = B[i][j];
30. B[i][j] = B[j][i];
31. B[j][i] = x;
32. }
34. // multiplica la matriz A y la matriz B, el resultado queda en la matriz C
35. // notar que los indices de la matriz B se han intercambiado
37. **for** (**int** i = 0; i < N; i++)
38. **for** (**int** j = 0; j < N; j++)
39. **for** (**int** k = 0; k < N; k++)
40. C[i][j] += A[i][k] \* B[j][k];
42. **long** t2 = System.currentTimeMillis();
43. System.out.println("Tiempo: " + (t2 - t1) + "ms");
44. }
45. }