

Instituto Politécnico Nacional

Escuela Superior de Cómputo

*Asignatura*:

Desarrollo de sistemas distribuidos

*Grupo*:4CM3

Tarea 2. Memoria cache

*Alumno*: Garcia Garcia Rafael

*Profesor*: Carlos Pineda Guerrero



# Capturas de pantalla de la tarea:

Lo primero a realizar es ejecutar nuestro programa para obtener los tiempos utilizando memoria caché o sin usarlo en mi caso decidí modificar un poco los programas para que el método *main* donde se realiza la multiplicación de matrices se sustituya un método llamado *multiplicar* y así crear un nuevo método *main* en donde haya un Array con los valores de prueba. Posterior a toda esta edición de código procedemos a ejecutarlos y compilarlos para tener en la salida de terminal los siguientes valores:

|  |
| --- |
|  |
| Imagen 1. Tiempos de ejecución con memoria caché |
|  |
| Imagen 2. Tiempos de ejecución sin memoria caché |

Observando las terminales de Netbeans podemos ver una enorme reducción de tiempo en valores grandes ya que el acceso de memoria es mucho más rápido. Graficando estos valores en Exel podemos ver los siguientes resultados:

|  |
| --- |
|  |
| Imagen 3. Gráfica con la comparación de los tiempos de ejecución de los programas |

Como observamos en la gráfica el uso de la memoria caché es de vital importancia para el manejo de grandes cantidades de datos, además de como ingenieros poder entender entre los conceptos de localidad espacial y temporal ya que este tipo de acceso en la memoria puede ayudarnos a reducir tiempos de ejecución en nuestros programas.

Ahora expliquemos cada uno de los programas en las siguientes partes:

## Programa sin memoria caché:

1. **public** **class** Scache
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. **static** **int** multiplicar(**int** N){
9. **long** t1 = System.currentTimeMillis();
11. // inicializa las matrices A y B
13. **for** (**int** i = 0; i < N; i++)
14. **for** (**int** j = 0; j < N; j++)
15. {
16. A[i][j] = 2 \* i - j;
17. B[i][j] = i + 2 \* j;
18. C[i][j] = 0;
19. }
21. // multiplica la matriz A y la matriz B, el resultado queda en la matriz C
23. **for** (**int** i = 0; i < N; i++)
24. **for** (**int** j = 0; j < N; j++)
25. **for** (**int** k = 0; k < N; k++)
26. C[i][j] += A[i][k] \* B[k][j];
28. **long** t2 = System.currentTimeMillis();
29. System.out.println("Tiempo: " + (t2 - t1) + "ms");
30. **return** N;
31. }
33. **public** **static** **void** main(String[] args){
34. **int**[] x = **new** **int** []{100, 200, 300, 500,1000};
35. **for** (**int** i=0;i!=5;i++){
36. multiplicar(x[i]);
37. }
38. }
39. }

En la primera parte del programa inicializamos nuestras matrices en los atributos de la clase principal, además de que se sustituye el método ***main*** por el método ***multiplicar*** para que podamos ingresar los valores con un array y tener los resultados de los tiempos en una sola ejecución del programa, en el método multiplicar estamos multiplicando las matrices sin hacer uso de la memoria caché agarrando los elementos de la matriz de manera que no se utiliza localidad espacial.

## Memoria con caché:

1. /\*\*
2. \*
3. \* @author Ralph
4. \*/
5. **public** **class** Ccache {
7. **static** **int** N = 1000;
8. **static** **int**[][] A = **new** **int**[N][N];
9. **static** **int**[][] B = **new** **int**[N][N];
10. **static** **int**[][] C = **new** **int**[N][N];
12. **public** **static** **int** multiplicar(**int** N)
13. {
14. **long** t1 = System.currentTimeMillis();
16. // inicializa las matrices A y B
18. **for** (**int** i = 0; i < N; i++)
19. **for** (**int** j = 0; j < N; j++)
20. {
21. A[i][j] = 2 \* i - j;
22. B[i][j] = i + 2 \* j;
23. C[i][j] = 0;
24. }
26. // transpone la matriz B, la matriz traspuesta queda en B
28. **for** (**int** i = 0; i < N; i++)
29. **for** (**int** j = 0; j < i; j++)
30. {
31. **int** x = B[i][j];
32. B[i][j] = B[j][i];
33. B[j][i] = x;
34. }
36. // multiplica la matriz A y la matriz B, el resultado queda en la matriz C
37. // notar que los indices de la matriz B se han intercambiado
39. **for** (**int** i = 0; i < N; i++)
40. **for** (**int** j = 0; j < N; j++)
41. **for** (**int** k = 0; k < N; k++)
42. C[i][j] += A[i][k] \* B[j][k];
44. **long** t2 = System.currentTimeMillis();
45. System.out.println("Tiempo: " + (t2 - t1) + "ms");
46. **return** N;
47. }
49. **public** **static** **void** main(String[] args){
50. **int**[] x = **new** **int** []{100, 200, 300, 500,1000};
51. **for** (**int** i=0;i!=5;i++){
52. multiplicar(x[i]);
53. }
54. }
55. }

Como en el programa anterior sustituimos el método ***main*** por el método ***multiplicar*** para que podamos ingresar los valores a medir en un arreglo de enteros, ahora en el método multiplicar primero transponemos la matriz B para que los valores que existen tengan un acceso de memoria por localidad espacial, esto como podemos ver en las salidas de la terminal reduce bastante el tiempo de ejecución de nuestro programa.

## Valores del sistema de ejecución: