ALEJANDRO DE JESÚS ZEPEDA FLORES
DESARROLLO DE SISTEMAS DISTRIBUIDOS - 4CM3

Tarea 2. Uso eficiente de la memoria cache.

La localidad de las referencias, es un fenómeno según el cual, basándonos en el pasado reciente de un programa podemos predecir con una precisión razonable qué instrucciones y datos utilizará en el futuro.

- **Localidad Temporal:** si en un momento una posición de memoria particular es referenciada, entonces es muy probable que la misma ubicación vuelva a ser referenciada en un futuro cercano. En este caso es común almacenar una copia de los datos referenciados en caché para lograr un acceso más rápido a ellos.
- **Localidad Espacial:** si una localización de memoria es referenciada en un momento concreto, es probable que las localizaciones cercanas a ella sean también referenciadas pronto. En este caso es común estimar las posiciones cercanas para que estas tengan un acceso más rápido.

Desarrollo.

Iniciamos con la explicación del primer programa MultiplicaMatriz.java El programa comienza con la declaración de una variable “N” de tipo int que indica las dimensiones de las matrices que a continuación se declaran.

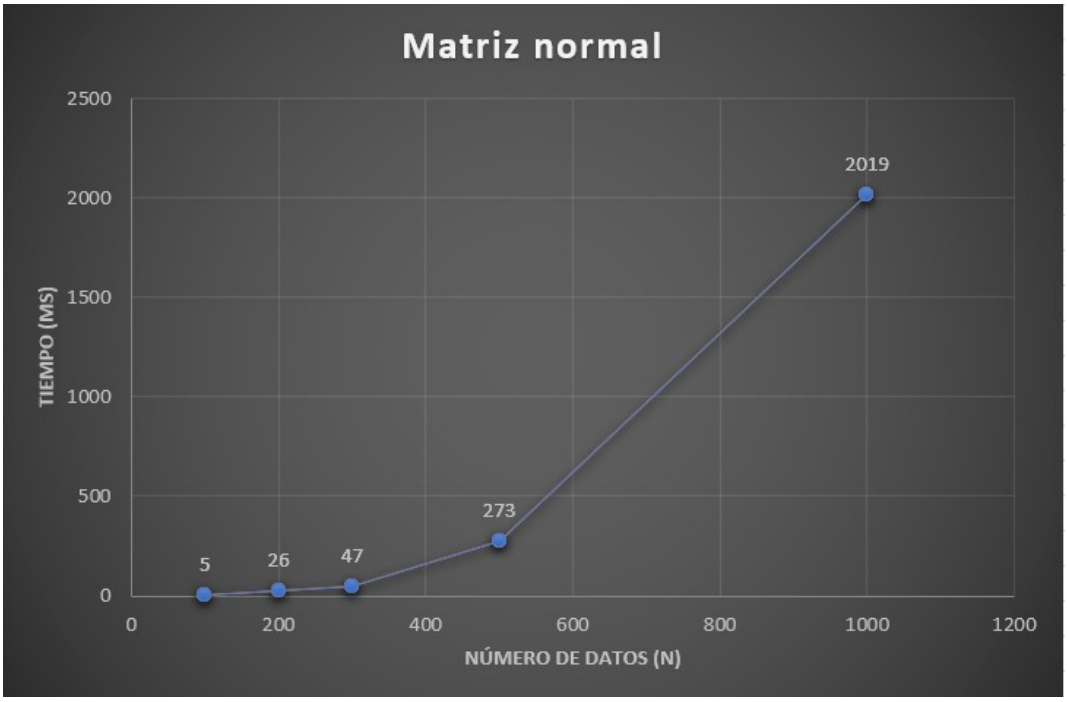
```
01. class MultiplicaMatriz{
02.     static int N = 1000;
03.     static int A[][] = new int[N][N];
04.     static int B[][] = new int[N][N];
05.     static int C[][] = new int[N][N];
06.
07.     public static void main(String args[]){
08.         long t1 = System.currentTimeMillis();
09.         for(int i = 0; i<N ;i++){
10.             for(int j = 0; j<N ;j++){
11.                 A[i][j] = 2*i-j;
12.                 B[i][j] = i+2*j;
13.                 C[i][j] = 0;
14.             }
15.         }
16.
17.         for(int i = 0; i<N ;i++)
18.             for(int j = 0; j<N ;j++)
19.                 for(int k = 0; k<N ;k++)
20.                     C[i][j] = A[i][k]*B[k][j];
21.
22.         long t2 = System.currentTimeMillis();
23.         System.out.println("Tiempo: "+(t2-t1)+"ms");
24.     }
25. }
```

Posteriormente, se declara una variable “t1” de tipo long para indicar el tiempo de inicio en la ejecución del programa. Después inicializamos las matrices con una serie de operaciones predeterminadas.

El siguiente paso es el importante, ya que vamos a recorrer las matrices, obtener su valor en cada posición, multiplicarlos entre si y almacenarlos en la matriz C.

En teoría, son operaciones simples que hemos realizado a lo largo de la carrera; sin embargo, lo interesante es observar que la matriz A se recorre de manera secuencial, pero la matriz B se recorre por columnas haciendo que se transfiere una línea de cache completa de la RAM a la cache.

A continuación, se muestra la gráfica 1 donde podemos ver como el tiempo va a aumentando a medida que los datos aumentan. A pesar de que la unidad de medida del tiempo es ms, su crecimiento es acelerado y tiende a tener un comportamiento exponencial.



Gráfica 1. Matriz normal.

Ahora, vamos a hacer un pequeño ajuste en la programación de las matriz para el archivo MultiplicaMatriz_2.java. La declaración e inicialización de la matriz y la variable de incremente es la misma.

```
01. class MultiplicaMatriz_2{
02.     static int N = 1000;
03.     static int A[][] = new int[N][N];
04.     static int B[][] = new int[N][N];
05.     static int C[][] = new int[N][N];
06.
07.     public static void main(String args[]){
08.         long t1 = System.currentTimeMillis();
09.         for(int i = 0; i<N ;i++){
10.             for(int j = 0; j<N ;j++){
11.                 A[i][j] = 2*i-j;
12.                 B[i][j] = i+2*j;
13.                 C[i][j] = 0;
14.             }
15.         }
16.
17.         for(int i = 0; i<N ;i++){
18.             for(int j = 0; j<N ;j++){
19.                 int x = B[i][j];
20.                 B[i][j] = B[j][i];
21.                 B[j][i] = x;
22.             }
23.         }
24.
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[j][k];
29.
30.         long t2 = System.currentTimeMillis();
31.         System.out.println("Tiempo: "+(t2-t1)+"ms");
32.     }
33. }
```

Sin embargo, hacemos un pequeño cambio antes de realizar la multiplicación de las mismas. Codificamos dos for's anidados para poder transponer la matriz.

Una vez realizado este procedimiento, procedemos a multiplicar las matrices y notaremos que los cambios en el tiempo son bastantes significativos.

Realmente, lo que sucedió fue que se accedió a los datos de la matriz B de manera secuencial, por lo que facilita el proceso de la misma.

La gráfica 2, muestra un cambio importante a relación de la gráfica 1. En cantidades pequeñas de datos observamos que el programa se comporta de manera similar y si existen cambios, son mínimos. Sin embargo, a medida que se van incrementando el número de datos a procesar, el cambio es radical.

El ejemplo más claro es el comportamiento cuando N = 1000, dónde en la primer gráfica tomo un tiempo 2019 ms, mientras que en la segunda solamente tomo un tiempo de 501 ms, esto quiere decir que disminuyó ¼ parte del tiempo original.

Gráfica 2. Matriz transpuesta.

Adicionalmente, se agregan las especificaciones técnicas de la computadora dónde se ejecutaron los programas.

Sistema

Fabricante:	HP
Modelo:	HP Laptop 15-dw0xxx
Procesador:	Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80 GHz
Memoria instalada (RAM):	8.00 GB (7.89 GB utilizable)
Tipo de sistema:	Sistema operativo de 64 bits, procesador x64
Lápiz y entrada táctil:	La entrada táctil o manuscrita no está disponible para esta pantalla

Caché	6 MB Intel® Smart Cache