# USO EFICIENTE DE LA MEMORIA CACHE

Tarea 2

Raúl Sanchez Rico
Desarrollo de sistemas distribuidos
4CV2
ESCOM - IPN

## Desarrollo

Como se ve en las clases, la cache juega un papel importante al momento de optimizar, sin embargo, cuando uno programa casi nunca toma en cuenta esta situación, el claro ejemplo es la multiplicación de matrices, en donde con un simple cambio hace una gran diferencia en tiempo de ejecución. El primer programa es *MultiplicacionMatriz,* el cual es el método de siempre, donde se multiplica renglón por columna como se han enseñado en las clases de algebra, teniendo en cuenta que Java almacena las matrices como renglones, haciendo que al momento de obtener el acceso de las columnas de la otra matriz sea ineficiente, por que se tiene que transferir toda una línea de cache de la RAM a la cache.

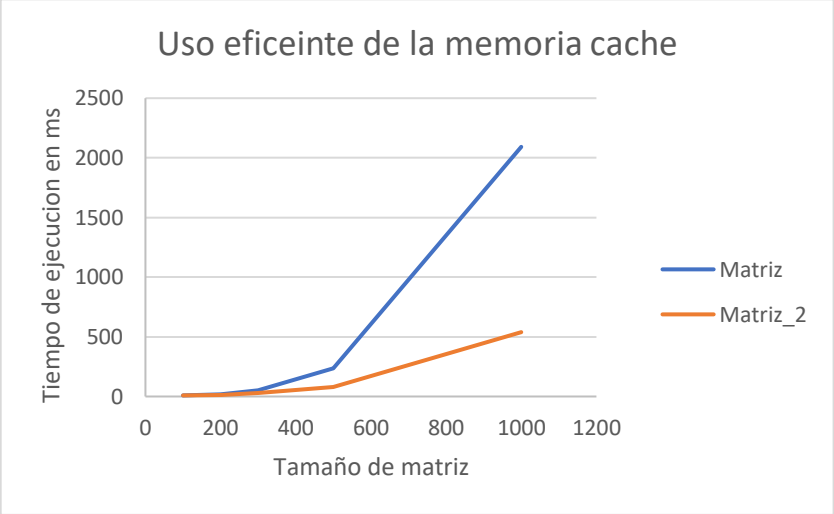
Por otro lado, el programa *MultiplicacionMatriz_2* accede por renglones en la segunda matriz, esto es muy simple ya que solo se tienen que intercambiar los índices cuando se hacen las operaciones haciendo un proceso mucho mejor.

### Grafica

*Tabla 1 Tiempos de ejecución*

| | 1000 | 500 | 300 | 200 | 100 |
|---|---|---|---|---|---|
| Matriz | 2092 | 237 | 54 | 19 | 9 |
| Matriz_2 | 539 | 81 | 31 | 15 | 9 |

*Tabla 2 Tiempos de ejecución*




### Datos

- Marca CPU: Intel
- Modelo CPU: Quad-Core Intel Core i7,
- Cache CPU: Cache Nivel 2 : 256 KB , Cache Nivel 3 : 6 MB
- RAM: 16 GB, DDR3, 1600 MHz

## Código

### MultiplicacionMatriz

```java
class MultiplicaMatriz
{
  static int N = 10000;
  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");
  }
}
```

### MultiplicacionMatriz_2

```java
class MultiplicaMatriz_2
{
  static int N = 10000;
  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++)
      {
```

```
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. }
```