repo:

artículo:
http://www.drdobbs.com/parallel/cache-friendly-code-solving-manycores-ne/240012736

Tamaño de caché usando (lscpu | grep "cache")
L1d cache: 32K
L1i cache: 32K
L2 cache: 256K
L3 cache: 3072K

Código:

```
1    static volatile int array[Size];
2    static void test_function(void)
3    {
4        for (int i = 0; i < Iterations; i++)
5            for (int x = 0; x < Size; x++)
6              array[x]++;
7    }
```

La memoria de 256K puede tener 65536 enteros (4 bytes). Cuando el tamaño es mayor demora 0.178486s y cuando es menor demora 0.170997s

Bucles Ineficientes
Es importante definir el orden de los bucles anidados, porque pueden producir un desperdicio de memoria caché y hacer más lenta la ejecución
Código 1:

```
1    double array[SIZE][SIZE];
2    for (int col = 0; col < SIZE; col++)
3        for (int row = 0; row < SIZE; row++)
4            array[row][col] = f(row, col);
```

Código 2:

```
1    double array[SIZE][SIZE];
2        for (int row = 0; row < SIZE; row++)
3             for (int col = 0; col < SIZE; col++)
4                 array[row][col] = f(row, col);
```

El código 1 demora 0.0234266 segundos, mientras el código 2 demora 0.0054767. Esto se debe a que en cada iteración del bucle interno del código 1 se trae a la caché todo el sub array, pero solo se lee un elemento, desperdiciando memoria y tiempo. En el código 2, se trae todo el sub array y se lee todos los elementos.

Orden de asignaciones

También es importante ordenar adecuadamente las asignaciones. La mayoría de procesadores modernos asignan espacios de memoria que sean múltiplos del tamaño del elemento, desperdiciando espacio si es que no se da un orden correcto a las asignaciones.

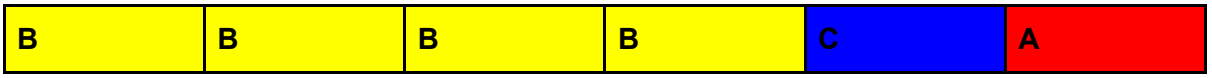
Código 1:

```
1    struct record {
2        char a;
3        int b;
4        char c;
5    };
```

En el código 1 se usan en total 9 bytes, pero si se mueve el valor ‘b’ antes que el ‘a’ se puede reducir el espacio utilizado.

Código 2:

```
1    struct record {
2        int b;
3        char a;
4        char c;
5    };
```



En este caso sólo se usan 6 bytes.