Caches and programs: An example

Fabrizio Miguel Mattos Cahui,
Ciencia de la Computación
Universidad Nacional de San Agustín de Arequipa
Arequipa, Perú

I. Implementación

Para la realización de este análisis se implementó código en C++, de acuerdo al libro de referencia, en el que se compara el rendimiento de dos bucles, que realizan el mismo procedimiento de llenar un array de elementos de acuerdo a los valores de una matriz y otro array, pero en cada bucle la forma de acceder a la memoria de la matriz sigue un orden diferente.

- Primer bucle: En el primer caso el bucle anidado accede a la memoria de la matriz siguiendo un orden horizontal, es decir accede a cada fila, pero no continua a la siguiente hasta terminar con todas las columnas, fila por columna.

```
for (int i = 0; i < MAX; i++)
    for (int j = 0; j < MAX; j++)
        y[i] += A[i][j] * x[j];
```

- Segundo bucle: El segundo bucle anidado, a diferencia del primer caso, accede a la memoria de la matriz siguiendo el orden de columna por fila, de forma vertical, que como se verá en los resultados resulta ser más lento en ejecución que en el primer caso.

```
for (int j = 0; j < MAX; j++)
    for (int i = 0; i < MAX; i++)
        y[i] += A[i][j]*x[j];
```

- Inicialización de datos: Para realizar la prueba, se siguió las indicaciones del libro de referencia, en donde se inicializaba la matriz A y el array x con valores, mientras el array Y se llenaba de 0, este proceso se realizó de la siguiente manera.

```
double A[MAX][MAX], x[MAX], y[MAX];
for(int i=0;i<MAX;i++)
{
    for(int j=0;j<MAX;j++)
    {
        A[i][j] = j*2;
    }
}
for (int i=0;i<MAX;i++)
{
    x[i] = i+1;
}
memset(y, 0, sizeof(y));
```

II. Resultados

Para comparar el rendimiento entre el primer bucle anidado y el segundo bucle anidado, se realizaron 4 pruebas, partiendo con 500 elementos, 1000 elementos, 2000 elementos y finalmente con 3000 elementos. En las pruebas se midió el tiempo de ejecución de cada bucle.

- Primer prueba (500 elementos):

```
Time taken by first pair of loops: 642 microseconds
Time taken by second pair of loops: 725 microseconds
```

- Segunda prueba (1000 elementos):

```
Time taken by first pair of loops: 2592 microseconds
Time taken by second pair of loops: 3208 microseconds
```

- Tercera prueba (2000 elementos):

```
Time taken by first pair of loops: 10403 microseconds
Time taken by second pair of loops: 19356 microseconds
```

- Cuarta prueba (3000):

```
Time taken by first pair of loops: 23577 microseconds
Time taken by second pair of loops: 57073 microseconds
```

Como se puede observar el primer bucle anidado tiene un mejor rendimiento en todos los casos el cuál se hace más notorio conforme aumenta la cantidad de elementos de la matriz, llegando a ser hasta 2.5 veces más rápido que el segundo bucle anidado. Este resultado se debe a la forma en como se acceden a los elementos de la matriz, en el primer caso los elementos son accedidos de forma continua, lo que hace que haya menos cache misses, cuando empieza el bucle y la cache no contiene ningún dato de la matriz, esta memoria se carga con los datos de manera continua hasta llegar al límite de la memoria, los accesos serán de la forma [0][0], [0][1], [0][2], etc, los cuales en su mayoría dependiendo del límite ya están en la memoria cache, reduciendo así el número de misses y evicts durante toda la ejecución del bucle. Por otro

lado el segundo bucle itera columnas por filas lo que resulta en un mayor número de misses y evicts, por lo que aumenta el número de operaciones que se harían en la ejecución del bucle, esto se debe a que, siguiendo la misma lógica que en el primer caso, al inicio del bucle, los datos de la matriz se cargaran en cache hasta el límite permitido, pero como en este caso la iteración se da de la forma [0][0],[1][0],[2][0], etc, después de leer el primer dato se debe buscar el dato [1][0], el cuál es posible que no se encuentre en memoria después de cargar datos en el inicio del bucle, y más aún si se continua iterando la columna, resultando en un mayor número de misses y evicts en la memoria cache.

III. Conclusión

En conclusión, es importante tener en cuenta los limites que tiene la memoria cache y la forma en la que se guardan los datos, para que de esta manera se pueda crear programas más eficientes en el manejo de este tipo de memoria que suele tener poco espacio.