INFORME – TRABAJO DE LABORATORIO 1

ALGORITMOS PARALELOS

Alumno: Sebastian Wilde Alarcón Arenas

Algoritmos

Three nested loop

```
for (int i=0;i<MAX;i++)
    for(int j=0;j<MAX;j++)
        for(int k=0;k<MAX;k++)
            C.mat[i][j]+=A.mat[i][k]*B.mat[k][j];
```

Blocked versión with six nested loops

```
int bloque=MAX/10;
for(int i=0;i<MAX;i+=bloque)
    for(int j=0;j<MAX;j+=bloque)
        for(int k=0;k<MAX;k+=bloque)
            for(int x=i;x<i+bloque;x++)
                for(int y=j;y<j+bloque;y++)
                    for(int z=k;z<k+bloque;z++)
                        C.mat[x][y]+=A.mat[x][z]*B.mat[z][y];
```

Resultados al ejecutar el programa:

Utilizando la librería time de C++ se obtuvo los siguientes resultados:

Elementos	Three_nested_loop	Blocked_version_with_six_nested_loops
100	196.12.16g milisegundos	135.93.16g milisegundos
1000	383850.16g milisegundos	251590.16g milisegundos

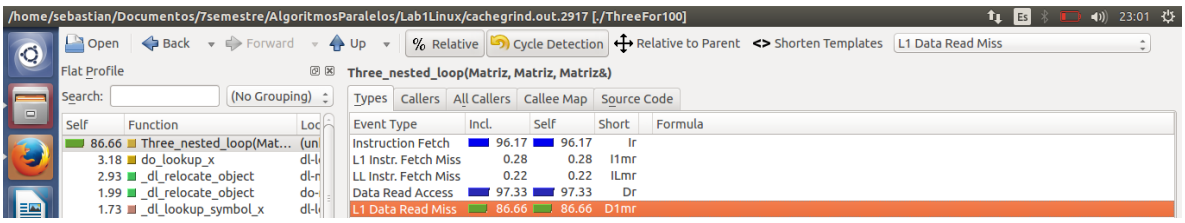
Siendo el segundo método más rápido a que en el segundo no sobreescribir tantas veces la caché

Usando Valgrind:

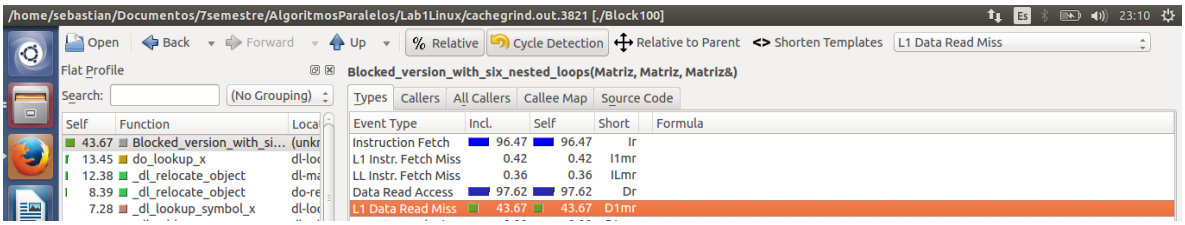
	I refs:	I1 misses	LLi misses
Three_nested_loop 100 elementos	**53,102,555**	**1,421**	**1,384**
Blocked_version_with_six_nested_loops 100 elementos	**57,585,744**	**1,424**	**1,386**
Three_nested_loop 1000 elementos	**51,063,082,530**	**1,441**	**1,425**
Blocked_version_with_six_nested_loops 1000 elementos	**54,197,489,402**	**1,442**	**1,426**

Usando kcachegrind:

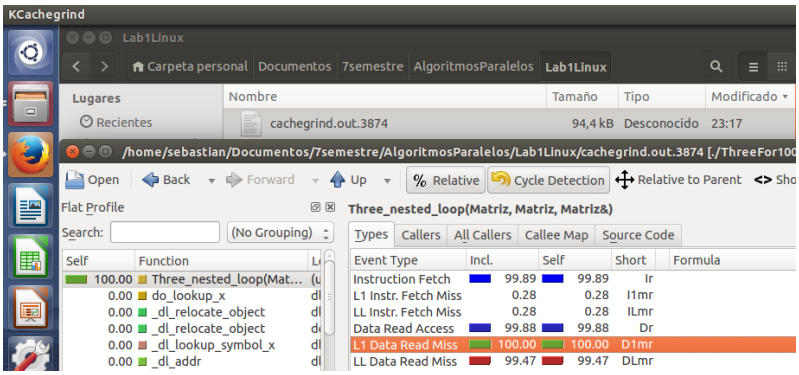
Multiplicación de matrices con el método normal con 100 elementos por matriz



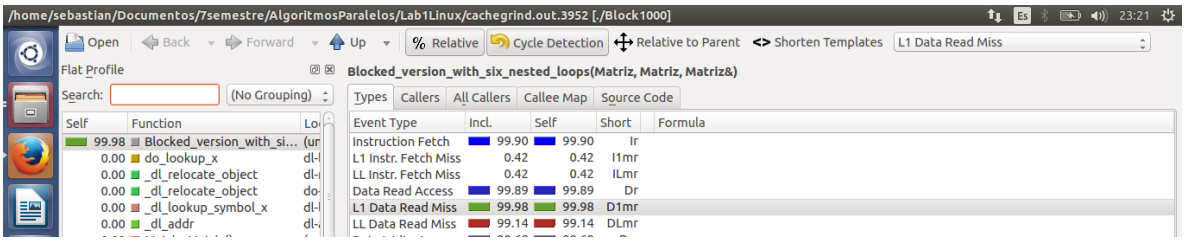
Multiplicación de matrices con el método de bloques con 100 elementos por matriz



Multiplicación de matrices con el método normal con 1000 elementos por matriz



Multiplicación de matrices con el método de bloques con 1000 elementos por matriz



El caché miss se refiere a un estado en el que los datos solicitados para su procesamiento por un componente o aplicación no se encuentra en la memoria caché. Esto causa demoras en la ejecución al requerir que el programa. Y como se puede apreciar en las imágenes, que son la captura de **kcachegrind** para visualizar mejor los datos obtenidos del **valgrind** (que se encuentra en la hoja anterior) podemos comprobar que el método por bloques es más rápido que el método de 3 fors, debido a los accesos a caché que como se muestra hay menos errores en la cache (donde no encuentra el elemento) en el método de 3 fors, lo cual lo hace más lento.