PRUEBAS CON LA MEMORIA CACHÉ

Selene Barrios Cornejo

1 EJERCICIO 1

Implementar y comparar los 2-bucles anidados FOR presentados en el cap. 2 del libro, pag 22.

```
for (auto MAX : v){
 cin >> MAX ;
 t01 = clock();
 for(int i = 0; i < MAX; i++){
  for(int j = 0; j < MAX; j++){
   y[i] += A[i][j]*x[j];
  }
 }
t11 = clock();
double timeBUCLE1 = (double(t11-t01)/
    CLOCKS_PER_SEC);
}
```

Ambos algoritmos se ejecutan utilizando las herramientas *valgrind* y *kcachegrind* para obtener una evaluación mas precisa de su desempeño en términos de cache misses.

- *Barrios Cornejo Selene*
  sbarrios@unsa.edu.pe,
  Universidad Nacional de San Agustín.


```
==11874== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright inf
==11874== Command: ./prueba01
==11874==
--11874-- warning: L3 cache found, using its data for the LL simulation.
100
Tiempo de ejecucion BUCLE1: 0.000841
200
Tiempo de ejecucion BUCLE1: 0.000386
400
Tiempo de ejecucion BUCLE1: 0.002424
500
Tiempo de ejecucion BUCLE1: 0.002808
800
==11874==
==11874== Process terminating with default action of signal 11 (SIGSEGV)
==11874==  Access not within mapped region at address 0x1FFF001000
==11874==    at 0x109726: main (prueba01.cpp:18)
==11874==  If you believe this happened as a result of a stack
==11874==  overflow in your program's main thread (unlikely but
==11874==  possible), you can try to increase the size of the
==11874==  main thread stack using the --main-stacksize= flag.
==11874==  The main thread stack size used in this run was 8388608.
==11874==
==11874== I   refs:      2,816,025
==11874== I1  misses:        2,726
==11874== LLi misses:        2,229
==11874== I1  miss rate:      0.10%
==11874== LLi miss rate:      0.08%
==11874==
==11874== D   refs:      1,053,691  (831,119 rd   + 222,572 wr)
==11874== D1  misses:       16,218  ( 13,728 rd   +   2,490 wr)
==11874== LLd misses:        9,350  (  7,756 rd   +   1,594 wr)
==11874== D1  miss rate:       1.5% (    1.7%     +     1.1%  )
==11874== LLd miss rate:       0.9% (    0.9%     +     0.7%  )
==11874==
==11874== LL refs:          18,944  ( 16,454 rd   +   2,490 wr)
==11874== LL misses:        11,579  (  9,985 rd   +   1,594 wr)
==11874== LL miss rate:       0.3% (    0.3%     +     0.7%  )
Segmentation fault (core dumped)
pejelagarta@pejelagarta-FX503VD:~/Desktop/PARALELA/LAB01$
```

Figure 1. Resultados de pruebas con matrices cuadradas de 100, 200, 400, 500 y 800 con el primer par de bucles

```
for (auto MAX : v){
 cin >> MAX;
 t02 = clock();
 for(int j = 0; j < MAX; j++){
  for(int i = 0; i < MAX; i++){
   y[i] += A[i][j]*x[j];
  }
 }
 t12 = clock();
 double timeBUCLE2 = (double(t12-t02)/
     CLOCKS_PER_SEC);
 }
}
```

```
pejelagarta@pejelagarta-FX503VD:~/Desktop/PARALELA/LAB01$ g++ -g
pejelagarta@pejelagarta-FX503VD:~/Desktop/PARALELA/LAB01$ valgrin
==13060== Cachegrind, a cache and branch-prediction profiler
==13060== Copyright (C) 2002-2017, and GNU GPL'd, by Nicholas Net
==13060== Using Valgrind-3.18.1 and LibVEX; rerun with -h for cop
==13060== Command: ./prueba02
==13060==
--13060-- warning: L3 cache found, using its data for the LL simu
100
Tiempo de ejecucion BUCLE2: 0.001068
200
Tiempo de ejecucion BUCLE2: 1.8e-05
400
Tiempo de ejecucion BUCLE2: 1.6e-05
500
Tiempo de ejecucion BUCLE2: 1.6e-05
800
Tiempo de ejecucion BUCLE2: 1.7e-05
==13060==
==13060== I   refs:      2,422,882
==13060== I1  misses:        2,982
==13060== LLi misses:        2,364
==13060== I1  miss rate:      0.12%
==13060== LLi miss rate:      0.10%
==13060==
==13060== D   refs:        812,589  (602,423 rd   + 210,166 wr)
==13060== D1  misses:       16,327  ( 13,832 rd   +   2,495 wr)
==13060== LLd misses:        9,341  (  7,747 rd   +   1,594 wr)
==13060== D1  miss rate:       2.0% (    2.3%     +     1.2%  )
==13060== LLd miss rate:       1.1% (    1.3%     +     0.8%  )
==13060==
==13060== LL refs:          19,309  ( 16,814 rd   +   2,495 wr)
==13060== LL misses:        11,705  ( 10,111 rd   +   1,594 wr)
==13060== LL miss rate:       0.4% (    0.3%     +     0.8%  )
pejelagarta@pejelagarta-FX503VD:~/Desktop/PARALELA/LAB01$
```

Figure 2. Resultados de pruebas con matrices cuadradas de 100, 200, 400, 500 y 800 con el segundo par de bucles

Para una mejor interpretación de los resultados se realizo la siguiente gráfica en Python:

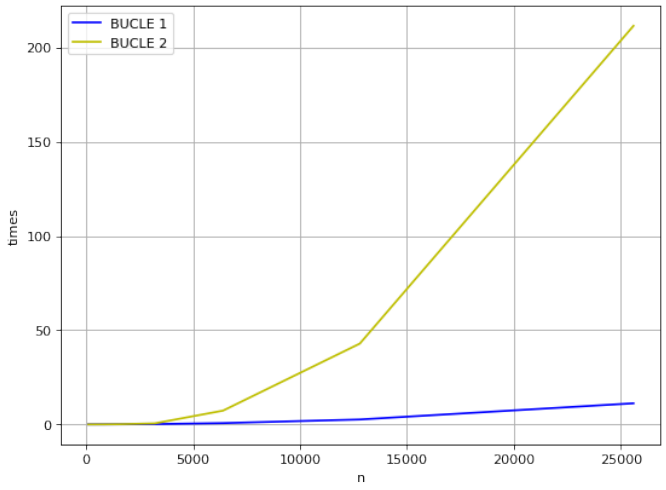

Figure 3. Se observa conforme se incrementa la matriz, la ineficiencia del segundo bucle respecto al primero se hace más patente.

2 EJERCICIO 2

Implementar en C/C++ la multiplicación de matrices clásica, la versión de tres bucles anidados y evaluar su desempeño considerando diferentes tamaños de matriz.

```
void multiplication(double A[500][500],
    int rowsA,int columnsA,double B
    [500][500],int rowsB,int columnsB,
    double C[500][500] ){
     if(columnsA==rowsB){
          for(int i=0;i<rowsA;i++){
               for(int j=0;j<columnsB;j++){
                    C[i][j]=0;
                    for(int k=0;k<columnsA;k
                        ++){
                         C[i][j]=C[i][j]+A[i][
                             k]*B[k][j];
                    }
               }
          }
     }
}
```

El producto de matrices se obtiene desarrollando el algoritmo con ciclos for y almacenamos el resultado en una nueva matriz, validando las filas y columnas.

```
pejelagarta@pejelagarta-FX503VD:~/Desktop/PARALELA/LAB01$ g++ -g Ejercicio2.cpp -o Ejercicio2
pejelagarta@pejelagarta-FX503VD:~/Desktop/PARALELA/LAB01$ valgrind --tool=cachegrind ./Ejercicio2
==17939== Cachegrind, a cache and branch-prediction profiler
==17939== Copyright (C) 2002-2017, and GNU GPL'd, by Nicholas Nethercote et al.
==17939== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==17939== Command: ./Ejercicio2
==17939==
--17939-- warning: L3 cache found, using its data for the LL simulation.
A:
Ingrese FILAS de la Matriz A: 100
Ingrese COLUMNAS de la Matriz A: 100
B:
ingrese COLUMNAS de la Matriz B: 100
tiempo: 0.160285
ingrese COLUMNAS de la Matriz B: 200
tiempo: 0.312361
ingrese COLUMNAS de la Matriz B: 300
tiempo: 0.487685
B:
ingrese COLUMNAS de la Matriz B: 400
tiempo: 0.621993
ingrese COLUMNAS de la Matriz B: 500
tiempo: 0.786284
ingrese COLUMNAS de la Matriz B: 600
tiempo: 0.921891
B:
ingrese COLUMNAS de la Matriz B: 700
tiempo: 1.07702
ingrese COLUMNAS de la Matriz B: 800
tiempo: 1.23154
ingrese COLUMNAS de la Matriz B: 900
tiempo: 1.37306
Ingrese COLUMNAS de la Matriz A: 1000
B:
ingrese COLUMNAS de la Matriz B: 1000
tiempo: 33.5252
ingrese COLUMNAS de la Matriz B: 1100
tiempo: 37.1574
```

Figure 4. Resultados de pruebas con matrices cuadradas de 100 a 1100 elementos

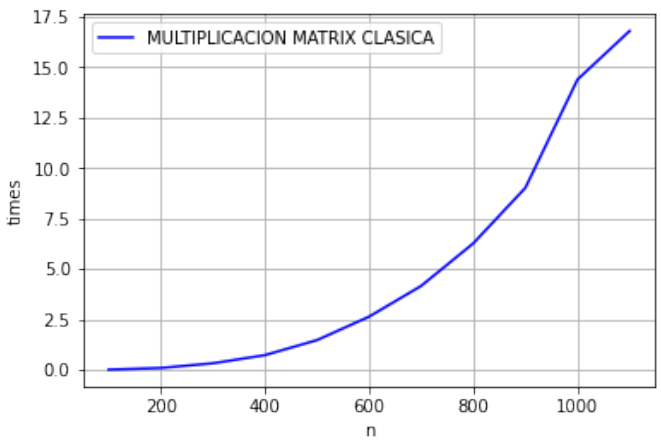

Figure 5. Desempeño del algoritmo considerando diferentes tamaños de matriz

Se observa que conforme los tamaños de matriz aumenta, su tiempo de ejecución incrementa.
En el orden *i,j* y *k*, se accede a los elementos de las matrices a y c en filas y a los elementos de la matriz b en columnas. Debido a que los elementos de la misma fila son adyacentes en el almacenamiento, y los elementos de la misma columna están separados en el almacenamiento, cuando la matriz es tan grande que no se pueden almacenar tres matrices en la caché L2 al mismo tiempo, el acceso a la matriz b puede causar muchos incidentes de fallas de caché secundario.

3 Ejercicio 3

Implementar la versión por bloques (investigar en internet), seis bloques anidados, evaluar su desempeño y compararlo con la multiplicación de matrices clásica.

```
void blockMultiplication(double A
    [500][500],double B[500][500],double C
    [500][500], int MAX ){
int N = MAX/10;
for(int i1=0; i1<MAX; i1+=N){
 for(int i2=0; i2<MAX; i2+=N){
  for(int i3=0; i3<MAX; i3+=N){
   int I1 =  min(i1 + N, MAX);
    for (int i = i1; i < I1; ++i){
     int I2 = min(i2 + N, MAX);
      for (int j = i2; j < I2; ++j){
       int I3 =min(i3 + N, MAX);
        for (int k = i3; k < I3; ++k){
         C[i][j] += A[i][k] * B[k][j];
```

```
            }
          }
        }
      }
    }
  }
}
```

```
pejelagarta@pejelagarta-FX503VD:~/Desktop/PARALELA/LAB01$ g++ -g Ejercicio3.cpp -o Ejercicio3
pejelagarta@pejelagarta-FX503VD:~/Desktop/PARALELA/LAB01$ valgrind --tool=cachegrind ./Ejercici
==20919== Cachegrind, a cache and branch-prediction profiler
==20919== Copyright (C) 2002-2017, and GNU GPL'd, by Nicholas Nethercote et al.
==20919== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==20919== Command: ./Ejercicio3
==20919==
--20919-- warning: L3 cache found, using its data for the LL simulation.
100
tiempo: 0.333044
200
tiempo: 2.03844
300
tiempo: 6.65431
==20919==
==20919== I   refs:      1,445,632,302
==20919== I1  misses:            2,814
==20919== LLi misses:            2,350
==20919== I1  miss rate:          0.00%
==20919== LLi miss rate:          0.00%
==20919==
==20919== D   refs:        727,400,104  (678,711,005 rd   + 48,689,099 wr)
==20919== D1  misses:          563,872  (    525,013 rd   +     38,859 wr)
==20919== LLd misses:           44,973  (     20,592 rd   +     24,381 wr)
==20919== D1  miss rate:           0.1% (        0.1%     +        0.1%  )
==20919== LLd miss rate:           0.0% (        0.0%     +        0.1%  )
==20919==
==20919== LL refs:             566,686  (    527,827 rd   +     38,859 wr)
==20919== LL misses:            47,323  (     22,942 rd   +     24,381 wr)
==20919== LL miss rate:            0.0% (        0.0%     +        0.1%  )
pejelagarta@pejelagarta-FX503VD:~/Desktop/PARALELA/LAB01$
```

Figure 6. Resultados de pruebas con matrices cuadradas de 100 a 300 elementos

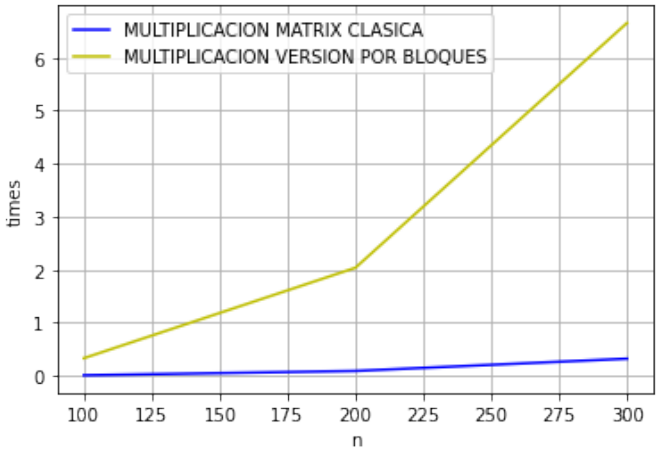

Figure 7. Desempeño de ambos algoritmos considerando diferentes tamaños de matriz

4 Ejercicio 4

Ejecutar ambos algoritmos paso a paso, y analizar el movimiento de datos entre la memoria principal y la memoria cache. Hacer una evaluación de acuerdo a la complejidad algorítmica.

5 Repositorio

El código se encuentra disponible en el siguiente repositorio de Github.