Algorítmos Paralelos
Three Nested Loop

Vanessa Santillana

23 Marzo 2017

1 Programa

Primeramente nuestro programa tiene por objetivo de multiplicar 2 matrices, sin embargo existen dos métodos de hacer tal multiplicación. También utilizaremos números randomicos al llenar las matrices.

El primer método serian la multiplicación de filas por columnas de matrices de tamaños $n * m$.

1.1 Código

```
1 void m_filas(int n, int m, int o,  int a[n][o], int b[n][m], int c[
      m][o]){
2     int i, j, k ;
3
4     for (i=0;i<n; i++)
5         for (j = 0; j<o; j++)
6           for (k=0;k<m; k++)
7               a[i][j] += b[i][k]* c[k][j] ;
8 }
```

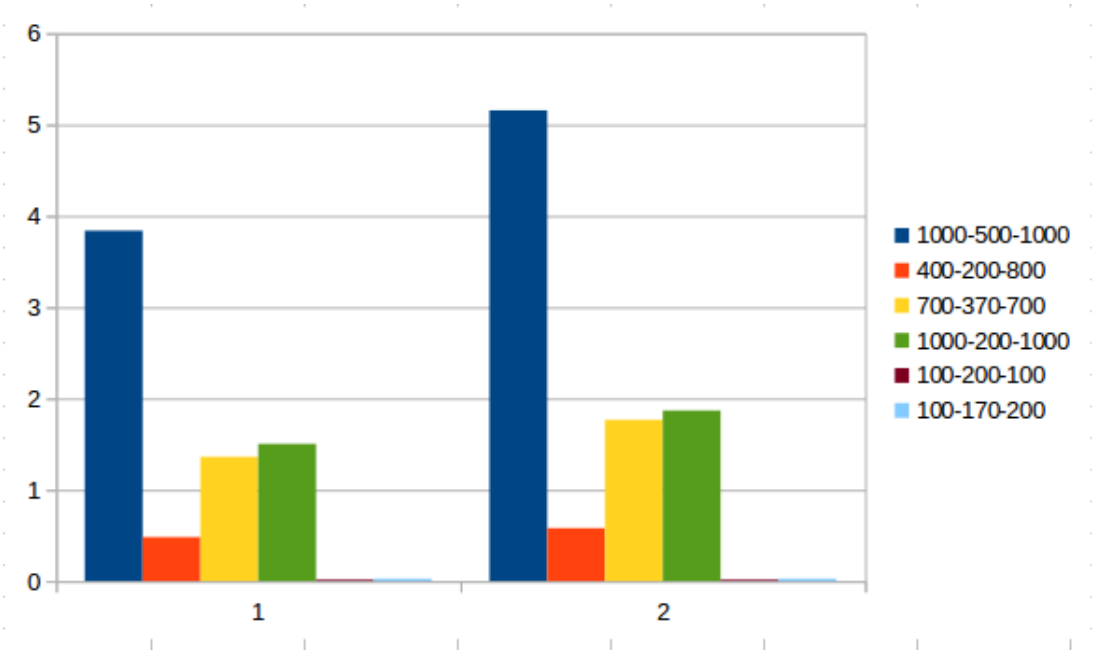
1.2 Código

El segundo método seria la multiplicación por columnas, y este quiere decir que la columna de la matriz A se multiplicara por el primer valor de la fila de B, y se suma a una matriz inicialmente vacía C que tendrá nuestro resultado.

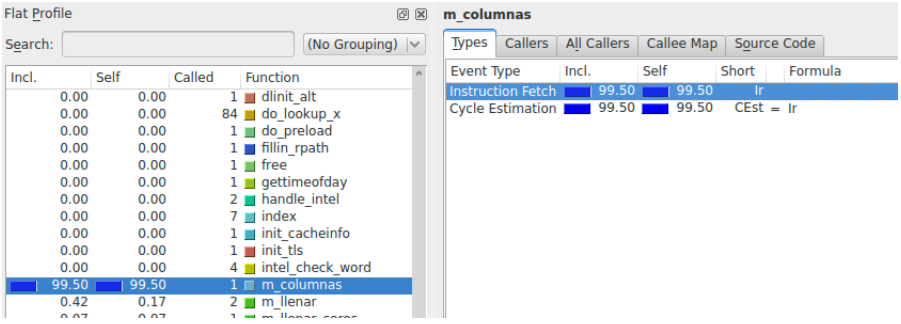
```
1 void m_columnas(int n, int m, int o, int a[n][o], int b[n][m], int
      c[m][o]){
2     int i, j, k ;
3
4     for (i=0;i<n; i++)
5         for (k=0;k<m; k++)
6           for (j = 0; j<o; j++)
7               a[i][j] += b[i][k]* c[k][j];
8 }
```

2 Comparación con un tamaño de datos especifico

Métodos	1000-500-1000	400-200-800	700-370-700	1000-200-1000	100-200-100	100-170-200
Filas	3.837597	0.481088	1.360849	1.501527	0.015253	0.025742
Columnas	5.155229	0.578829	1.766762	1.868259	0.015364	0.026034



3 Usando kcachegrind



4 Pruebas con Valgrind

Utilizando la herramienta Valgrind con el método 2 y con una matriz A de $1000*500$ y una matriz B con $500*1000$, obteniendo una matriz como resultado de $1000*1000$, detecta lo siguiente:

```
==22298== Memcheck, a memory error detector
==22298== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==22298== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==22298== Command: ./metodo1
==22298==
==22298== Warning: client switching stacks?  SP change: 0xffefffcb0 --> 0xffee17820
==22298==          to suppress, use: --max-stackframe=2000016 or greater
==22298== Warning: client switching stacks?  SP change: 0xffee17820 --> 0xffec2f390
==22298==          to suppress, use: --max-stackframe=2000016 or greater
==22298== Warning: client switching stacks?  SP change: 0xffec2f390 --> 0xffe85ea80
==22298== HEAP SUMMARY:
==22298==     in use at exit: 0 bytes in 0 blocks
==22298==   total heap usage: 0 allocs, 0 frees, 0 bytes allocated
==22298==
==22298== All heap blocks were freed -- no leaks are possible
==22298==
==22298== For counts of detected and suppressed errors, rerun with: -v
==22298== ERROR SUMMARY: 10000000 errors from 9 contexts (suppressed: 0 from 0)
```

Utilizando la herramienta Valgrind con el método 2 y con una matriz A de $1000*500$ y una matriz B con $500*1000$, obteniendo una matriz como resultado de $1000*1000$, detecta lo siguiente:

```
==22436== Memcheck, a memory error detector
==22436== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==22436== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==22436== Command: ./metodo2
==22436==
==22436== Warning: client switching stacks?  SP change: 0xffefffcb0 --> 0xffee17820
==22436==          to suppress, use: --max-stackframe=2000016 or greater
==22436== Warning: client switching stacks?  SP change: 0xffee17820 --> 0xffec2f390
==22436==          to suppress, use: --max-stackframe=2000016 or greater
==22436== Warning: client switching stacks?  SP change: 0xffec2f390 --> 0xffe85ea80
==22436==          to suppress, use: --max-stackframe=4000016 or greater
==22436==          further instances of this message will not be shown.
==22436== HEAP SUMMARY:
==22436==     in use at exit: 0 bytes in 0 blocks
==22436==   total heap usage: 0 allocs, 0 frees, 0 bytes allocated
==22436==
==22436== All heap blocks were freed -- no leaks are possible
==22436==
```

```
==22436== For counts of detected and suppressed errors, rerun with: -v
==22436== ERROR SUMMARY: 10000000 errors from 9 contexts (suppressed: 0 from 0)
```

5 Resultados

Si usamos el segundo método, podemos observar que el recorrido es por columna en la matriz A, entonces cuando quiere obtener el siguiente dígito de la columna tendrá que buscar directamente en memoria, ya que los datos no se encuentran en cache por ser una variable temporal, finalmente este recorrido no seria el mas efectivo.
Usando el primer método, nos damos cuenta que al hacer un recorrido por fila en la matriz A, se convierte en una variable espacial, la cual carga toda la fila y al hacer la operación de multiplicación no necesitara ir hasta memoria, ya que los datos se encuentra en la cache, de tal manera que es más efectivo.

6 Conclusión

Según las pruebas hechas, la multiplicación de matriz por filas, es decir el primer método es la mejor opción con el Three Nested Loop, ya que el tiempo que necesita para obtener los datos es menor gracias a la memoria cache.