Analisis

Sergio Rolando Valdivia Santisteban

March 2017

1 Introducción

Se comporaran los resultados de algoritmos de multiplicación de matrices para ver cual de ellos presenta menos cache misses y tarda menos.

2 Comparación

```
sergio@sergio-Lenovo-Y50-70:~/Algoritmos-Paralelos/MultiplicacionMatrices$ valgrind --tool=cachegrind ./three
==4728== Cachegrind, a cache and branch-prediction profiler
==4728== Copyright (C) 2002-2015, and GNU GPL'd, by Nicholas Nethercote et al.
==4728== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==4728== Command: ./three
==4728==
--4728-- warning: L3 cache found, using its data for the LL simulation.
==4728==
==4728== I   refs:      2,158,679
==4728== I1  misses:        1,512
==4728== LLi misses:        1,445
==4728== I1  miss rate:      0.07%
==4728== LLi miss rate:      0.07%
==4728==
==4728== D   refs:        722,808  (531,428 rd   + 191,380 wr)
==4728== D1  misses:       15,678  ( 13,487 rd   +   2,191 wr)
==4728== LLd misses:        9,111  (  7,723 rd   +   1,388 wr)
==4728== D1  miss rate:       2.2% (    2.5%     +     1.1%  )
==4728== LLd miss rate:       1.3% (    1.5%     +     0.7%  )
==4728==
==4728== LL refs:          17,190  ( 14,999 rd   +   2,191 wr)
==4728== LL misses:        10,556  (  9,168 rd   +   1,388 wr)
==4728== LL miss rate:        0.4% (    0.3%     +     0.7%  )
sergio@sergio-Lenovo-Y50-70:~/Algoritmos-Paralelos/MultiplicacionMatrices$ time ./three

real    0m0.001s
user    0m0.000s
sys     0m0.000s
sergio@sergio-Lenovo-Y50-70:~/Algoritmos-Paralelos/MultiplicacionMatrices$
```

Figure 1: Multipliacion simple de 3x3

```
sergio@sergio-Lenovo-Y50-70:~/Algoritmos-Paralelos/MultiplicacionMatrices$ valgrind --tool=cachegrind ./six
==4704== Cachegrind, a cache and branch-prediction profiler
==4704== Copyright (C) 2002-2015, and GNU GPL'd, by Nicholas Nethercote et al.
==4704== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==4704== Command: ./six
==4704==
--4704-- warning: L3 cache found, using its data for the LL simulation.
------------------------------
==4704==
==4704== I   refs:      2,174,735
==4704== I1  misses:        1,642
==4704== LLi misses:        1,547
==4704== I1  miss rate:      0.08%
==4704== LLi miss rate:      0.07%
==4704==
==4704== D   refs:        728,805  (535,476 rd   + 193,329 wr)
==4704== D1  misses:       15,830  ( 13,629 rd   +   2,201 wr)
==4704== LLd misses:        9,152  (  7,763 rd   +   1,389 wr)
==4704== D1  miss rate:       2.2% (    2.5%     +     1.1%  )
==4704== LLd miss rate:       1.3% (    1.4%     +     0.7%  )
==4704==
==4704== LL refs:          17,472  ( 15,271 rd   +   2,201 wr)
==4704== LL misses:        10,699  (  9,310 rd   +   1,389 wr)
==4704== LL miss rate:        0.4% (    0.3%     +     0.7%  )
sergio@sergio-Lenovo-Y50-70:~/Algoritmos-Paralelos/MultiplicacionMatrices$ time ./six
------------------------------

real    0m0.002s
user    0m0.000s
sys     0m0.000s
sergio@sergio-Lenovo-Y50-70:~/Algoritmos-Paralelos/MultiplicacionMatrices$
```

Figure 2: Multipliacion por bloques de 3x3

Para matrices pequeñas el algoritmo de la multiplicacion simple termina en menos tiempo que el de la multiplicacion por bloques y parece ser que el primero tiene un mejor manejo del cache porque presenta menos cache misses. Al aumentar el tamañp del problema:

```
sergio@sergio-Lenovo-Y50-70:~/Algoritmos-Paralelos/MultiplicacionMatrices$ valgrind --tool=cachegrind ./three
==4867== Cachegrind, a cache and branch-prediction profiler
==4867== Copyright (C) 2002-2015, and GNU GPL'd, by Nicholas Nethercote et al.
==4867== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==4867== Command: ./three
==4867==
--4867-- warning: L3 cache found, using its data for the LL simulation.
==4867==
==4867== I   refs:      53,974,798
==4867== I1  misses:         1,511
==4867== LLi misses:         1,444
==4867== I1  miss rate:       0.00%
==4867== LLi miss rate:       0.00%
==4867==
==4867== D   refs:      24,226,087  (22,874,975 rd   + 1,351,112 wr)
==4867== D1  misses:        83,461  (    79,670 rd   +     3,791 wr)
==4867== LLd misses:        11,093  (     8,273 rd   +     2,820 wr)
==4867== D1  miss rate:        0.3% (       0.3%     +       0.3%  )
==4867== LLd miss rate:        0.0% (       0.0%     +       0.2%  )
==4867==
==4867== LL refs:           84,972  (    81,181 rd   +     3,791 wr)
==4867== LL misses:         12,537  (     9,717 rd   +     2,820 wr)
==4867== LL miss rate:         0.0% (       0.0%     +       0.2%  )
sergio@sergio-Lenovo-Y50-70:~/Algoritmos-Paralelos/MultiplicacionMatrices$ time ./three

real    0m0.013s
user    0m0.012s
sys     0m0.000s
sergio@sergio-Lenovo-Y50-70:~/Algoritmos-Paralelos/MultiplicacionMatrices$
```

Figure 3: Multipliacion simple de 100x100

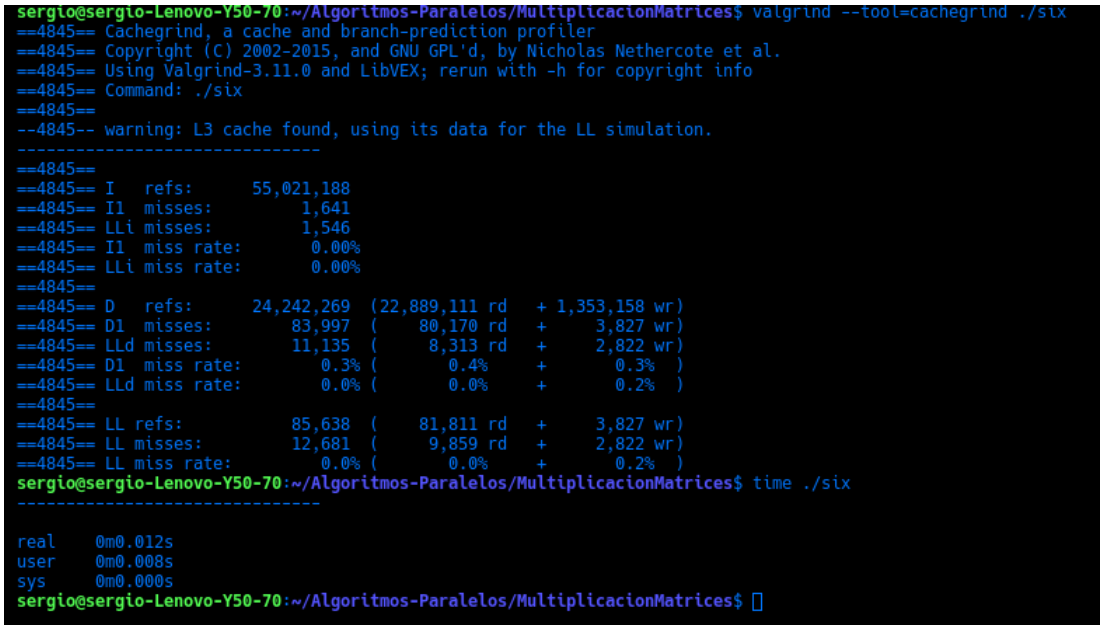

Figure 4: Multipliacion simple de 100x100

Al aumentar el tamaño del problema el algoritmo de la multiplicacion simple resulta ser mas lento que el de la multiplicacion por bloques. a la vez el primero presenta un mayor numero de cahce misses mientras que el segundo menos, esto ocurre porque el algoritmo de la multiplicaion por bloques carga en cache porciones de memoria que lego seran usadas asiendo que el numero de cache misses sea mas bajo que el primero.

3 Conclusiones

- Tener en cuenta el manejo de la memoria cache puede ayudar a mejorar bastante el tiempo de un programa
- La multiplicación de matrices por bloques es mejor cuando se habla de varios elementos en la matris debido a que hace un mejor uso de la memoria cache.