

Análisis de algoritmos

Angela Keshia Talavera Ormeño

Abril 2018

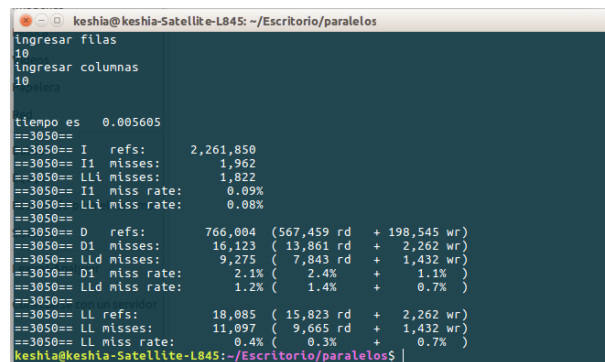
1. Introducción

En el presente se hace una comparación de algoritmos de multiplicación de matrices, matriz clásica y matriz con bloques. Dichas comparaciones se darán de acuerdo al tiempo que demoran en ser ejecutadas usando las herramientas Valgrind y Kcachegrind para obtener una evaluación más precisa de su desempeño en términos de Cache Misses.

1.1. Primera comparación

Esta primera comparación se dará con matrices pequeñas, para poder saber cual es la diferencia que existe entre cada una de ellas.

- Resultados multiplicación clásica, con matrices 10*10



```
keshia@keshia-Satellite-L845: ~/Escritorio/paralelos
Ingresar filas
10
Ingresar columnas
10
tiempo es 0.005605
==3050==
==3050== I refs: 2,261,850
==3050== I1 misses: 1,962
==3050== LL1 misses: 1,822
==3050== I1 miss rate: 0.09%
==3050== LL1 miss rate: 0.08%
==3050==
==3050== D refs: 766,084 (567,459 rd + 198,545 wr)
==3050== D1 misses: 16,123 (13,801 rd + 2,262 wr)
==3050== LLd misses: 9,275 (7,849 rd + 1,432 wr)
==3050== D1 miss rate: 2.1% ( 2.4% + 1.1% )
==3050== LLd miss rate: 1.2% ( 1.4% + 0.7% )
==3050==
==3050== LL refs: 18,085 (15,823 rd + 2,262 wr)
==3050== LL misses: 11,097 (9,665 rd + 1,432 wr)
==3050== LL miss rate: 0.4% ( 0.3% + 0.7% )
keshia@keshia-Satellite-L845:~/Escritorio/paralelos$
```

- Resultados multiplicación bloques, con matrices 10*10, bloque tamaño 3
- Resultados multiplicación bloques, con matrices 10*10, bloque tamaño 10

```

keshia@keshia-Satellite-L845: ~/Escritorio/paralelos
ingresar filas
10
ingresar columnas
10
tiempo 0.011004
==3073==
==3073== I refs:      2,309,003
==3073== I1 misses:    1,966
==3073== L1l misses:   1,821
==3073== I1 miss rate: 0.09%
==3073== L1l miss rate: 0.08%
==3073==
==3073== D refs:      794,258 (586,444 rd + 207,814 wr)
==3073== D1 misses:   16,122 (13,860 rd + 2,262 wr)
==3073== L1d misses:   9,275 (7,843 rd + 1,432 wr)
==3073== D1 miss rate: 2.0% (2.4% + 1.1%)
==3073== L1d miss rate: 1.2% (1.3% + 0.7%)
==3073==
==3073== LL refs:     18,088 (15,826 rd + 2,262 wr)
==3073== LL misses:   11,096 (9,664 rd + 1,432 wr)
==3073== LL miss rate: 0.4% (0.3% + 0.7%)
keshia@keshia-Satellite-L845:~/Escritorio/paralelos$

```

```

keshia@keshia-Satellite-L845: ~/Escritorio/paralelos
ingresar filas
10
ingresar columnas
10
tiempo 0.010535
==3101==
==3101== I refs:      2,284,424
==3101== I1 misses:    1,966
==3101== L1l misses:   1,821
==3101== I1 miss rate: 0.09%
==3101== L1l miss rate: 0.08%
==3101==
==3101== D refs:      779,885 (576,454 rd + 203,431 wr)
==3101== D1 misses:   16,122 (13,860 rd + 2,262 wr)
==3101== L1d misses:   9,275 (7,843 rd + 1,432 wr)
==3101== D1 miss rate: 2.1% (2.4% + 1.1%)
==3101== L1d miss rate: 1.2% (1.4% + 0.7%)
==3101==
==3101== LL refs:     18,088 (15,826 rd + 2,262 wr)
==3101== LL misses:   11,096 (9,664 rd + 1,432 wr)
==3101== LL miss rate: 0.4% (0.3% + 0.7%)
keshia@keshia-Satellite-L845:~/Escritorio/paralelos$

```

1.2. Segunda comparación

Aquí haremos comparaciones con matrices grandes, y en el caso de bloques, los bloques que tomaremos será uno grande otro pequeño.

- Resultados multiplicación clásica, con matrices 500*500

```

==3555== Cachegrind, a cache and branch-prediction profiler
==3555== Copyright (C) 2002-2015, and GNU GPL'd, by Nicholas Nethercote et al.
==3555== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info.
==3555== Command: ./mg
==3555==
..3555.. warning: L3 cache found, using its data for the LL simulation.
ingresar filas
500
ingresar columnas
500
tiempo 55.8448
==3555==
==3555== I refs:      9,425,272,235
==3555== I1 misses:    1,994
==3555== L1l misses:   1,882
==3555== I1 miss rate: 0.00%
==3555== L1l miss rate: 0.00%
==3555==
==3555== D refs:      4,658,005,471 (3,907,630,108 rd + 750,375,363 wr)
==3555== D1 misses:   2,908,725 (2,855,493 rd + 53,232 wr)
==3555== L1d misses:   57,348 (5,443 rd + 48,905 wr)
==3555== D1 miss rate: 0.1% (0.1% + 0.0%)
==3555== L1d miss rate: 0.0% (0.0% + 0.0%)
==3555==
==3555== LL refs:      2,908,719 (2,857,487 rd + 51,232 wr)
==3555== LL misses:    59,230 (10,325 rd + 48,905 wr)
==3555== LL miss rate: 0.0% (0.0% + 0.0%)
keshia@keshia-Satellite-L845:~/Escritorio/paralelos$

```

- Resultados multiplicación bloques, con matrices 500*500, bloque tamaño 10

```

==3743== Cachegrind, a cache and branch-prediction profiler
==3743== Copyright (C) 2002-2015, and GNU GPL'd, by Nicholas Nethercote et al.
==3743== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==3743== Command: ./n6
==3743==
..3743.. warning: L3 cache found, using its data for the LL simulation.
Ingresar filas
500
Ingresar columnas
500

tiempo 57.9978
==3743==
==3743== I refs:      8,775,363,787
==3743== I1 misses:    1,997
==3743== L1L misses:    1,885
==3743== I1 miss rate:  0.00%
==3743== L1L miss rate: 0.00%
==3743==
==3743== O refs:      4,278,743,726 (3,645,955,262 rd + 632,788,464 wr)
==3743== O1 misses:    8,370,172 ( 8,318,948 rd +   51,224 wr)
==3743== L1d misses:    57,142 ( 8,437 rd + 48,905 wr)
==3743== O1 miss rate:  0.2% ( 0.2% + 0.0% )
==3743== L1d miss rate: 0.0% ( 0.0% + 0.0% )
==3743==
==3743== LL refs:      8,372,169 ( 8,320,937 rd +   51,232 wr)
==3743== LL misses:    59,227 ( 10,322 rd + 48,905 wr)
==3743== LL miss rate:  0.0% ( 0.0% + 0.0% )
keshla@keshla-Satellite-L845:~/Escritorio/paralelos$

```

- Resultados multiplicación bloques, con matrices 500*500, bloque tamaño 200

```

==3619== Cachegrind, a cache and branch-prediction profiler
==3619== Copyright (C) 2002-2015, and GNU GPL'd, by Nicholas Nethercote et al.
==3619== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==3619== Command: ./n3
==3619==
..3619.. warning: L3 cache found, using its data for the LL simulation.
Ingresar filas
500
Ingresar columnas
500

tiempo es 37.0707
==3619==
==3619== I refs:      6,390,549,268
==3619== I1 misses:    1,989
==3619== L1L misses:    1,882
==3619== I1 miss rate:  0.00%
==3619== L1L miss rate: 0.00%
==3619==
==3619== O refs:      2,882,612,625 (2,756,364,815 rd + 126,247,810 wr)
==3619== O1 misses:    137,411,683 (137,360,457 rd +   51,226 wr)
==3619== L1d misses:    56,544 ( 8,046 rd + 48,898 wr)
==3619== O1 miss rate:  4.8% ( 5.0% + 0.0% )
==3619== L1d miss rate: 0.0% ( 0.0% + 0.0% )
==3619==
==3619== LL refs:      137,413,672 (137,362,446 rd +   51,226 wr)
==3619== LL misses:    58,826 ( 9,928 rd + 48,898 wr)
==3619== LL miss rate:  0.0% ( 0.0% + 0.0% )
keshla@keshla-Satellite-L845:~/Escritorio/paralelos$

```

2. Conclusiones

2.0.1. Conclusiones primera comparación

Como se puede observar en las pruebas el tiempo que demora en ejecutar es más rápido en la multiplicación de matrices normal ya que por ser pequeñas no se necesitan los bloques, lo que lo hace un poco mas lento. Cuando trabajamos con matrices pequeñas resulta mas rápido trabajar con la multiplicación clásica.

2.0.2. Conclusiones segunda comparación

Aquí, podemos ver que existe una mejora significativa de acuerdo a la matriz clásica con la matriz en bloques, siempre y cuando el bloque sea grande, la memoria caché permite optimizar el tiempo de ejecución.

Cuando utiliza Callgrind para perfilar una aplicación, su aplicación se transforma en un lenguaje intermedio y luego se ejecuta en un procesador virtual emulado por valgrind. Esto tiene una sobrecarga de tiempo de ejecución enorme, pero la precisión es realmente buena y sus datos de perfil están completos.

Una aplicación que se ejecuta en Callgrind puede ser de 10 a 50 veces más lenta de lo normal.