Computação Paralela – 2025/1
Aluno: Gilberto De Melo Júnior
RA: 10419275
Turma: 05P

Relatório

Compilação sem otimizações (Github Codespaces)

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS                                                    bash

@gilbertodemelo ➜ /workspaces/comppar-05p-threads-cpp-gilbertodemelo (main) $ g++ -std=c++11 -O0 -fno-tree-vectorize -fno-inline -o row matrix_mult_row.cpp && ./row 100
Tempo de execução (sequencial): 11618 microsegundos
@gilbertodemelo ➜ /workspaces/comppar-05p-threads-cpp-gilbertodemelo (main) $ g++ -std=c++11 -O0 -fno-tree-vectorize -fno-inline -o row matrix_mult_col.cpp && ./col 100
Tempo de execução (sequencial): 12838 microsegundos
@gilbertodemelo ➜ /workspaces/comppar-05p-threads-cpp-gilbertodemelo (main) $ g++ -std=c++11 -O0 -fno-tree-vectorize -fno-inline -o row matrix_mult_block.cpp && ./block 100 16
Tempo de execução (blocagem): 2074 microsegundos
@gilbertodemelo ➜ /workspaces/comppar-05p-threads-cpp-gilbertodemelo (main) $
```

```
@gilbertodemelo ➜ /workspaces/comppar-05p-threads-cpp-gilbertodemelo (main) $ valgrind --tool=cachegrind ./row 100
==10259== Cachegrind, a cache and branch-prediction profiler
==10259== Copyright (C) 2002-2017, and GNU GPL'd, by Nicholas Nethercote et al.
==10259== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==10259== Command: ./row 100
==10259==
--10259-- warning: L3 cache found, using its data for the LL simulation.
Uso: ./row <tamanho da matriz N>
==10259==
==10259== I   refs:      2,439,783
==10259== I1  misses:        1,786
==10259== LLi misses:        1,714
==10259== I1  miss rate:      0.07%
==10259== LLi miss rate:      0.07%
==10259==
==10259== D   refs:        749,627  (555,306 rd   + 194,321 wr)
==10259== D1  misses:       17,085  ( 14,604 rd   +   2,481 wr)
==10259== LLd misses:        9,963  (  8,387 rd   +   1,576 wr)
==10259== D1  miss rate:       2.3% (    2.6%     +     1.3%  )
==10259== LLd miss rate:       1.3% (    1.5%     +     0.8%  )
==10259==
==10259== LL refs:          18,871  ( 16,390 rd   +   2,481 wr)
==10259== LL misses:        11,677  ( 10,101 rd   +   1,576 wr)
==10259== LL miss rate:       0.4% (    0.3%     +     0.8%  )
```

```
@gilbertodemelo ➜ /workspaces/comppar-05p-threads-cpp-gilbertodemelo (main) $ valgrind --tool=cachegrind ./col 100
==10614== Cachegrind, a cache and branch-prediction profiler
==10614== Copyright (C) 2002-2017, and GNU GPL'd, by Nicholas Nethercote et al.
==10614== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==10614== Command: ./col 100
==10614==
--10614-- warning: L3 cache found, using its data for the LL simulation.
Tempo de execução (sequencial): 398109 microsegundos
==10614==
==10614== I   refs:      132,753,762
==10614== I1  misses:          2,121
==10614== LLi misses:          1,983
==10614== I1  miss rate:        0.00%
==10614== LLi miss rate:        0.00%
==10614==
==10614== D   refs:       70,846,279  (44,544,127 rd   + 26,302,152 wr)
==10614== D1  misses:        104,952  (    88,826 rd   +     16,126 wr)
==10614== LLd misses:         12,048  (     8,397 rd   +      3,651 wr)
==10614== D1  miss rate:         0.1% (       0.2%     +        0.1%  )
==10614== LLd miss rate:         0.0% (       0.0%     +        0.0%  )
==10614==
==10614== LL refs:           107,073  (    90,947 rd   +     16,126 wr)
==10614== LL misses:          14,031  (    10,380 rd   +      3,651 wr)
==10614== LL miss rate:          0.0% (       0.0%     +        0.0%  )
```

```
@gilbertodemelo ➜ /workspaces/comppar-05p-threads-cpp-gilbertodemelo (main) $ valgrind --tool=cachegrind ./block 100 16
==10935== Cachegrind, a cache and branch-prediction profiler
==10935== Copyright (C) 2002-2017, and GNU GPL'd, by Nicholas Nethercote et al.
==10935== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==10935== Command: ./block 100 16
==10935==
--10935-- warning: L3 cache found, using its data for the LL simulation.
Tempo de execução (blocagem): 22041 microsegundos
==10935==
==10935== I   refs:      14,058,868
==10935== I1  misses:         2,121
==10935== LLi misses:         1,985
==10935== I1  miss rate:       0.02%
==10935== LLi miss rate:       0.01%
==10935==
==10935== D   refs:       7,235,531  (4,678,972 rd   + 2,556,559 wr)
==10935== D1  misses:        23,333  (   17,305 rd   +     6,028 wr)
==10935== LLd misses:        12,048  (    8,398 rd   +     3,650 wr)
==10935== D1  miss rate:        0.3% (      0.4%     +       0.2%  )
==10935== LLd miss rate:        0.2% (      0.2%     +       0.1%  )
==10935==
==10935== LL refs:           25,454  (   19,426 rd   +     6,028 wr)
==10935== LL misses:         14,033  (   10,383 rd   +     3,650 wr)
==10935== LL miss rate:         0.1% (      0.1%     +       0.1%  )
```

Compilação com otimizações

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS

@gilbertodemelo ➜ /workspaces/comppar-05p-threads-cpp-gilbertodemelo (main) $ g++ -std=c++11 -O3 -o row matrix_mult_row.cpp && ./row 1000
Tempo de execução (sequencial): 1193215 microsegundos
@gilbertodemelo ➜ /workspaces/comppar-05p-threads-cpp-gilbertodemelo (main) $ g++ -std=c++11 -O3 -o col matrix_mult_col.cpp && ./col 1000
Tempo de execução (sequencial): 1184298 microsegundos
@gilbertodemelo ➜ /workspaces/comppar-05p-threads-cpp-gilbertodemelo (main) $ g++ -std=c++11 -O3 -o block matrix_mult_block.cpp && ./block 1000 16
Tempo de execução (blocagem): 36155 microsegundos
@gilbertodemelo ➜ /workspaces/comppar-05p-threads-cpp-gilbertodemelo (main) $
```

```
@gilbertodemelo ➜ /workspaces/comppar-05p-threads-cpp-gilbertodemelo (main) $ valgrind --tool=cachegrind ./row 1000
==15648== Cachegrind, a cache and branch-prediction profiler
==15648== Copyright (C) 2002-2017, and GNU GPL'd, by Nicholas Nethercote et al.
==15648== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==15648== Command: ./row 1000
==15648==
--15648-- warning: L3 cache found, using its data for the LL simulation.
==15648== brk segment overflow in thread #1: can't grow to 0x484f000
==15648== (see section Limitations in user manual)
==15648== NOTE: further instances of this message will not be shown
Tempo de execução (sequencial): 38758762 microsegundos
==15648==
==15648== I   refs:      9,196,269,802
==15648== I1  misses:            1,980
==15648== LLi misses:            1,872
==15648== I1  miss rate:          0.00%
==15648== LLi miss rate:          0.00%
==15648==
==15648== D   refs:      4,067,109,871  (3,041,307,334 rd   + 1,025,802,537 wr)
==15648== D1  misses:    1,439,651,465  (1,439,270,906 rd   +       380,559 wr)
==15648== LLd misses:          199,411  (        8,387 rd   +       191,024 wr)
==15648== D1  miss rate:          35.4% (         47.3%     +           0.0%  )
==15648== LLd miss rate:           0.0% (          0.0%     +           0.0%  )
==15648==
==15648== LL refs:       1,439,653,445  (1,439,272,886 rd   +       380,559 wr)
==15648== LL misses:           201,283  (       10,259 rd   +       191,024 wr)
==15648== LL miss rate:            0.0% (          0.0%     +           0.0%  )
```

```
@gilbertodemelo ➜ /workspaces/comppar-05p-threads-cpp-gilbertodemelo (main) $ valgrind --tool=cachegrind ./col 1000
==16708== Cachegrind, a cache and branch-prediction profiler
==16708== Copyright (C) 2002-2017, and GNU GPL'd, by Nicholas Nethercote et al.
==16708== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==16708== Command: ./col 1000
==16708==
--16708-- warning: L3 cache found, using its data for the LL simulation.
==16708== brk segment overflow in thread #1: can't grow to 0x484f000
==16708== (see section Limitations in user manual)
==16708== NOTE: further instances of this message will not be shown
Tempo de execução (sequencial): 40262743 microsegundos
==16708==
==16708== I   refs:      9,198,265,932
==16708== I1  misses:            1,980
==16708== LLi misses:            1,872
==16708== I1  miss rate:          0.00%
==16708== LLi miss rate:          0.00%
==16708==
==16708== D   refs:      4,069,107,871  (3,043,305,334 rd   + 1,025,802,537 wr)
==16708== D1  misses:    1,444,943,370  (1,441,751,060 rd   +     3,192,310 wr)
==16708== LLd misses:          199,411  (        8,387 rd   +       191,024 wr)
==16708== D1  miss rate:          35.5% (         47.4%     +           0.3%  )
==16708== LLd miss rate:           0.0% (          0.0%     +           0.0%  )
==16708==
==16708== LL refs:       1,444,945,350  (1,441,753,040 rd   +     3,192,310 wr)
==16708== LL misses:           201,283  (       10,259 rd   +       191,024 wr)
==16708== LL miss rate:            0.0% (          0.0%     +           0.0%  )
```

```
@gilbertodemelo ➜ /workspaces/comppar-05p-threads-cpp-gilbertodemelo (main) $ valgrind --tool=cachegrind ./block 1000 16
==17282== Cachegrind, a cache and branch-prediction profiler
==17282== Copyright (C) 2002-2017, and GNU GPL'd, by Nicholas Nethercote et al.
==17282== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==17282== Command: ./block 1000 16
==17282==
--17282-- warning: L3 cache found, using its data for the LL simulation.
==17282== brk segment overflow in thread #1: can't grow to 0x484f000
==17282== (see section Limitations in user manual)
==17282== NOTE: further instances of this message will not be shown
Tempo de execução (blocagem): 1269300 microsegundos
==17282==
==17282== I   refs:      513,022,568
==17282== I1  misses:          1,980
==17282== LLi misses:          1,872
==17282== I1  miss rate:        0.00%
==17282== LLi miss rate:        0.00%
==17282==
==17282== D   refs:      196,172,816  (138,901,336 rd   + 57,271,480 wr)
==17282== D1  misses:     41,071,373  ( 40,753,560 rd   +    317,813 wr)
==17282== LLd misses:        199,414  (      8,389 rd   +    191,025 wr)
==17282== D1  miss rate:        20.9% (       29.3%     +        0.6%  )
==17282== LLd miss rate:         0.1% (        0.0%     +        0.3%  )
==17282==
==17282== LL refs:        41,073,353  ( 40,755,540 rd   +    317,813 wr)
==17282== LL misses:         201,286  (     10,261 rd   +    191,025 wr)
==17282== LL miss rate:          0.0% (        0.0%     +        0.3%  )
```

Foram analisadas três formas de realizar a multiplicação de matrizes: uma versão que percorre os elementos por linha, outra que acessa por coluna e uma terceira utilizando blocagem para otimizar o uso da cache.

Os resultados obtidos com o Valgrind (Cachegrind) mostram que a ordem de acesso aos dados tem um impacto significativo no desempenho. A versão que percorre a matriz linha por linha apresentou um tempo de execução menor do que a versão que percorre coluna por coluna. Isso acontece porque, na maioria dos sistemas, a memória é organizada de forma que os elementos de uma mesma linha estão mais próximos uns dos outros, permitindo que sejam carregados para o cache de forma eficiente. Quando o acesso ocorre por colunas, cada nova leitura pode exigir um carregamento de dados da memória principal, resultando em mais cache misses e aumentando o tempo de execução.

A versão com blocagem foi disparadamente a mais rápida, com um tempo de execução 30 vezes menor do que as versões convencionais. Isso acontece porque, em vez de acessar os elementos de forma contínua por linha ou coluna, essa abordagem divide a matriz em pequenos blocos que cabem na cache. Dessa forma, os dados são reutilizados localmente antes de serem descartados, reduzindo drasticamente o número de acessos à memória RAM.

Os dados do Valgrind reforçam essas observações. Na multiplicação por colunas, a taxa de falhas no cache (D1 misses) foi mais alta, indicando que muitos acessos estavam indo direto para a RAM em vez de serem atendidos pela cache. Já na versão com blocagem, esse número caiu drasticamente, provando que a reutilização dos dados dentro do cache foi muito mais eficiente. Além disso, o número de instruções executadas foi muito menor, mostrando que a estratégia de

blocagem não apenas reduziu acessos desnecessários à memória, mas também tornou o código mais eficiente no uso da CPU.

No final das contas, os testes mostram que o jeito que acessamos os dados importa tanto quanto os cálculos que fazemos. Técnicas como a blocagem são fundamentais para otimizar o uso do hardware, garantindo um melhor aproveitamento do cache e melhorando significativamente o desempenho do programa. Essa abordagem é especialmente útil em aplicações que lidam com grandes volumes de dados, onde a eficiência da memória pode ser um fator determinante para a performance geral do sistema.