Computação Paralela - Lab: Threads em C++

Matheus de Andrade Lourenço - 10419691

- **Multiplicação de matrizes: Lógica utilizada**

Dada uma matriz MxN, a lógica paralela utilizada consta na criação de M threads (para o percurso em linha), em cada thread se responsabiliza pelo cálculo de uma das linhas na matriz resultante. No caso do percurso em coluna, são criadas N threads.

- **Execução - Multiplicação por linha**

```
@MATHEUS-DE-ANDRADE-LOURENCO →/workspaces/comppar-05p-threads-cpp-MATHEUS-DE-ANDRADE-LOURENCO (main) $ ./linha
Tempo de execução: 0.374623 segundos
```

- **Execução - Multiplicação por coluna**

```
@MATHEUS-DE-ANDRADE-LOURENCO →/workspaces/comppar-05p-threads-cpp-MATHEUS-DE-ANDRADE-LOURENCO (main) $ ./col
Tempo de execução: 0.385873 segundos
```

- **Valgrind - Análise**

```
@MATHEUS-DE-ANDRADE-LOURENCO →/workspaces/comppar-05p-threads-cpp-MATHEUS-DE-ANDRADE-LOURENCO (main) $ valgrind --tool=cachegrind ./col
==51040== Cachegrind, a cache and branch-prediction profiler
==51040== Copyright (C) 2002-2017, and GNU GPL'd, by Nicholas Nethercote et al.
==51040== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==51040== Command: ./col
==51040==
--51040-- warning: L3 cache found, using its data for the LL simulation.
Tempo de execução: 15.4488 segundos
==51040==
==51040== I   refs:      4,852,547,622
==51040== I1  misses:            6,732
==51040== LLi misses:            2,502
==51040== I1  miss rate:          0.00%
==51040== LLi miss rate:          0.00%
==51040==
==51040== D   refs:      2,310,329,935  (2,164,448,683 rd   + 145,881,252 wr)
==51040== D1  misses:       64,014,953  (   63,221,499 rd   +     793,454 wr)
==51040== LLd misses:           71,525  (       10,147 rd   +      61,378 wr)
==51040== D1  miss rate:           2.8% (          2.9%     +         0.5%  )
==51040== LLd miss rate:           0.0% (          0.0%     +         0.0%  )
==51040==
==51040== LL refs:          64,021,685  (   63,228,231 rd   +     793,454 wr)
==51040== LL misses:            74,027  (       12,649 rd   +      61,378 wr)
==51040== LL miss rate:            0.0% (          0.0%     +         0.0%  )
```

Execução por coluna

```
@MATHEUS-DE-ANDRADE-LOURENCO →/workspaces/comppar-05p-threads-cpp-MATHEUS-DE-ANDRADE-LOURENCO (main) $ valgrind --tool=cachegrind ./linha
==51909== Cachegrind, a cache and branch-prediction profiler
==51909== Copyright (C) 2002-2017, and GNU GPL'd, by Nicholas Nethercote et al.
==51909== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==51909== Command: ./linha
==51909==
--51909-- warning: L3 cache found, using its data for the LL simulation.
Tempo de execução: 15.3865 segundos
==51909==
==51909== I   refs:      4,852,546,457
==51909== I1  misses:            6,826
==51909== LLi misses:            2,504
==51909== I1  miss rate:          0.00%
==51909== LLi miss rate:          0.00%
==51909==
==51909== D   refs:      2,310,329,456  (2,164,448,434 rd   + 145,881,022 wr)
==51909== D1  misses:       59,716,729  (   59,626,706 rd   +      90,023 wr)
==51909== LLd misses:           71,249  (        9,883 rd   +      61,366 wr)
==51909== D1  miss rate:           2.6% (          2.8%     +         0.1%  )
==51909== LLd miss rate:           0.0% (          0.0%     +         0.0%  )
==51909==
==51909== LL refs:          59,723,555  (   59,633,532 rd   +      90,023 wr)
==51909== LL misses:            73,753  (       12,387 rd   +      61,366 wr)
==51909== LL miss rate:            0.0% (          0.0%     +         0.0%  )
```

Execução por linha

Através do Valgrind, é possível perceber que existem algumas diferenças gritantes nos dois modos de se resolver o problema no que tange ao acesso da memória cache. Em uma matriz 500x500, ao percorrer por coluna, é possível notar um número maior de miss no geral, sendo uma diferença maior na quantidade de D1 misses que diz respeito à busca dos dados dentro do nível L1 do cache. Isso ocorre pois a arquitetura em que o código foi executado é baseada em linha, o que faz com que os blocos de memórias que são extraídos da memória principal e guardados no cache para um futuro acesso são linhas. Quando se percorre a matriz por coluna, os dados que serão guardados no cache serão menos utilizados pelo algoritmo pois os dados que precisam ser acessados nas iterações não estão nos blocos já guardados no cache, o que implica em misses e mais buscas na memória principal.