Laboratório 2 de Computação Concorrente - Relatório de Desempenho

Miguel Santos Uchôa da Fonseca - DRE 120036412

O código utilizado está disponível no Github e se chama *matrix_mult.c*. O algoritmo implementa primeiro uma solução sequencial de multiplicação de matrizes, e depois uma solução concorrente, exibindo os seus respectivos tempos de execução. 5 testes foram realizados para cada uma das dimensões 500, 1000 e 2000, para cada um dos números de thread 1, 2 e 4.

Os testes foram realizados em uma máquina Windows, no WSL, com um processador AMD Ryzen 5 3600, que possui 6 cores e 12 threads. O ganho esperado era de 1 para 1 thread, 2 para 2 threads e 4 para 4 threads - e as expectativas foram confirmadas na prática.

Matrizes 500x500:

- 1 processador:
  - Tempo sequencial: $3,96 * 10^{-1}$ segundos.
  - Tempo concorrente: $3,43 * 10^{-1}$ segundos.
  - Ganho de desempenho: $1,15$.

- 2 processadores:
  - Tempo sequencial: $3,94 * 10^{-1}$ segundos.
  - Tempo concorrente: $1,77 * 10^{-1}$ segundos.
  - Ganho de desempenho: $2,22$.

- 4 processadores:
  - Tempo sequencial: $3,95 * 10^{-1}$ segundos.
  - Tempo concorrente: $9,27 * 10^{-2}$ segundos.
  - Ganho de desempenho: $4,26$.

Matrizes 1000x1000:

- 1 processador:
  - Tempo sequencial: $3,17$ segundos.
  - Tempo concorrente: $2,72$ segundos.
  - Ganho de desempenho: $1,17$.

- 2 processadores:
  - Tempo sequencial: $3,17$ segundos.
  - Tempo concorrente: $1,42$ segundos.
  - Ganho de desempenho: $2,23$.

- 4 processadores:
  - Tempo sequencial: $3,17$ segundos.
  - Tempo concorrente: $7,4 * 10^{-1}$ segundos.
  - Ganho de desempenho: $4,28$.

Matrizes 2000x2000:

- 1 processador:
  - Tempo sequencial: $37,07$ segundos.
  - Tempo concorrente: $33,62$ segundos.
  - Ganho de desempenho: $1,10$.

- 2 processadores:
  - Tempo sequencial: $38,04$ segundos.
  - Tempo concorrente: $18,15$ segundos.
  - Ganho de desempenho: $2,10$.

- 4 processadores:
  - Tempo sequencial: $38,26$ segundos.
  - Tempo concorrente: $10,22$ segundos.
  - Ganho de desempenho: $3,74$.