Resultados do Laboratorio 2

Leonardo Santiago (120036072)

1 Métodos

O computador utilizado para cálcular tem as seguintes especificações:

```
Model name:           Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz
    CPU family:         6
    Model:              158
    Thread(s) per core: 2
    Core(s) per socket: 4
    Socket(s):          1
    Stepping:           10
    CPU max MHz:        4100,0000
    CPU min MHz:        800,0000
```

O programa em C `lab2.c` foi compilado utilizando

```
gcc lab2.c -o lab2 -Wall -lpthread -O3
```

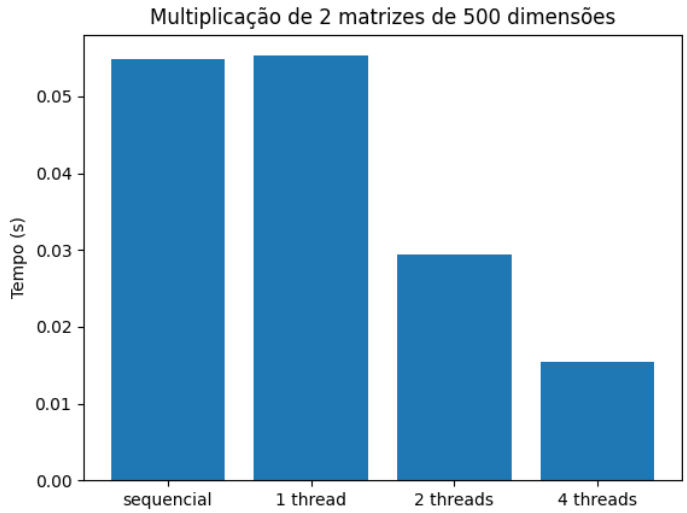
Utilizei o seguinte script em python para medir e plotar os gráficos:

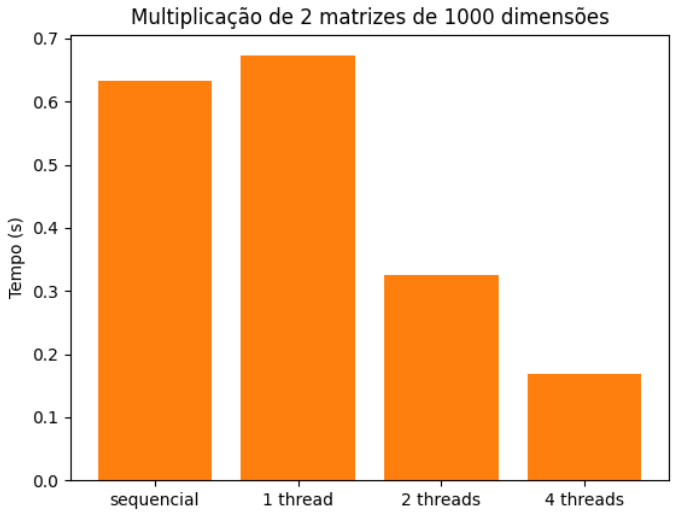
```python
from matplotlib import pyplot as plt
from matplotlib import cm
import subprocess
import re
import numpy as np

labels = ["sequencial", "1 thread", "2 threads", "4 threads"]
x_pos = np.arange(len(labels))

dimensoes = [500, 1000, 2000]
nthreads = [1, 2, 4]

melhorias = []
```

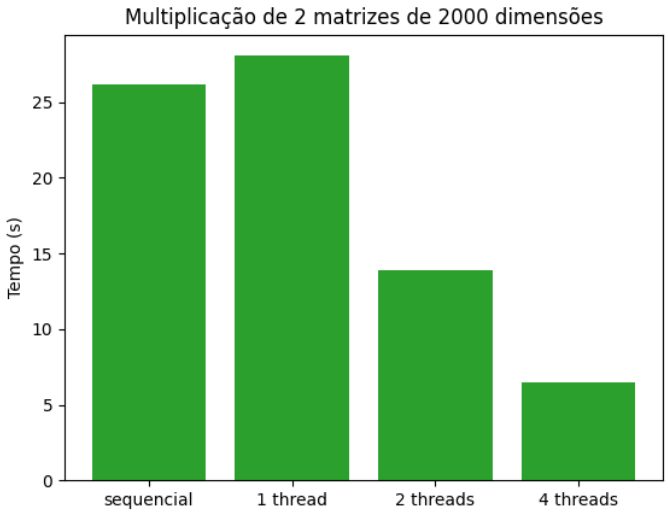
```
for dim in dimensoes:
    melhor_s = 1000
    tempos = []
    for thread in nthreads:
        melhor_c = 1000
        for i in range(5):
            output = subprocess.check_output(["./lab2", str(dim),
             ↪  str(thread)]).decode("utf-8")
            conc = re.search("CONCORRENTE: A solução concorrente levou
             ↪  (\d+\.\d+)s", output).group(1)
            seq = re.search("SEQUENCIAL: A solução sequencial levou
             ↪  (\d+\.\d+)s", output).group(1)
            if float(conc) < melhor_c:
                melhor_c = float(conc)
            if float(seq) < melhor_s:
                melhor_s = float(seq)
        tempos.append(melhor_c)
    [melhorias.append((thread, melhor_s/tempo)) for thread, tempo in
     ↪  zip(nthreads, tempos)]
    tempos.insert(0, melhor_s)
    plt.bar(x_pos, tempos)
    plt.xticks(x_pos, labels)
    plt.ylabel("Tempo (s)")
    plt.title(f"Multiplicação de 2 matrizes de {dim} dimensões")
    plt.savefig(f"dim{dim}thread{thread}.png")

fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(1, 1, 1)

x = np.linspace(1, 5, 1000)
ys = [(slope * x, nthread) for nthread, slope in melhorias]

for y, nthread in ys:
    ax.plot(x, y, c=cm.Dark2(nthread/4))

ax.plot(x, x, "r--", label="sequencial")
ax.legend([f'{n} threads' for n in nthreads], loc="upper left")

fig.savefig("performance.png", dpi=fig.dpi)
```

2 Resultados

Podemos ver claramente comparando os tempos que a multiplicação implementada concorrentemente foi mais rápida. Podemos ver isso nas imagens 1, 2 e 3, que demonstram claramente a relação:







Não somente foi mais rápido, como a relação entre o número de threads e o tempo levado parece ser linear. No gráfico 4, visualizamos cada uma das retas como o coeficiente de $T_{sequencial}/T_{concorrente}$ e relacionamos ao número de threads: é explicito que a perfomance está diretamente relacionada ao número de threads, como esperado.