



UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

INSTITUTO DE COMPUTAÇÃO

COMPUTAÇÃO CONCORRENTE

VICTOR DA SILVA MOURA

DRE: 120124419

27 DE ABRIL DE 2022, RIO DE JANEIRO, BRASIL.

Avaliação de desempenho do código de multiplicação de matrizes

Para a resolução desse laboratório, foi utilizado um computador com quatro processadores. Porém, o código foi rodado em uma máquina virtual, visto que ainda não tenho o sistema linux, porém, estou providenciando...

Com uma thread:

→ 500

```
victor@victor-Virtual-Machine:~/Documentos$ ./novo 500 1
SAO IGUAIS!, A RESOLUCAO FUNCIONA!
O tempo utilizando a forma sequencial, foi de : 0.366374
O tempo utilizando a forma concorrente, foi de : 0.425327
O desempenho ganho foi de : 0.861393
victor@victor-Virtual-Machine:~/Documentos$ ./novo 500 1
SAO IGUAIS!, A RESOLUCAO FUNCIONA!
O tempo utilizando a forma sequencial, foi de : 0.377899
O tempo utilizando a forma concorrente, foi de : 0.438702
O desempenho ganho foi de : 0.861404
victor@victor-Virtual-Machine:~/Documentos$ ./novo 500 1
SAO IGUAIS!, A RESOLUCAO FUNCIONA!
O tempo utilizando a forma sequencial, foi de : 0.462455
O tempo utilizando a forma concorrente, foi de : 0.440875
O desempenho ganho foi de : 1.048950
victor@victor-Virtual-Machine:~/Documentos$
```

O melhor tempo do modo concorrente foi o primeiro, em questão de desempenho em comparação com a função sequencial, foi o terceiro.

→ 1000

```
v Ficheros victor@victor-Virtual-Machine:~/Documentos$ ./novo 1000 1
SAO IGUAIS!, A RESOLUÇÃO FUNCIONA!
O tempo utilizando a forma sequencial, foi de : 3.725468
O tempo utilizando a forma concorrente, foi de : 4.467300
O desempenho ganho foi de : 0.833942
victor@victor-Virtual-Machine:~/Documentos$ ./novo 1000 1
SAO IGUAIS!, A RESOLUÇÃO FUNCIONA!
O tempo utilizando a forma sequencial, foi de : 3.491915
O tempo utilizando a forma concorrente, foi de : 4.111478
O desempenho ganho foi de : 0.849309
victor@victor-Virtual-Machine:~/Documentos$ ./novo 1000 1
SAO IGUAIS!, A RESOLUÇÃO FUNCIONA!
O tempo utilizando a forma sequencial, foi de : 3.641585
O tempo utilizando a forma concorrente, foi de : 4.468945
O desempenho ganho foi de : 0.814865
victor@victor-Virtual-Machine:~/Documentos$
```

O melhor tempo foi o segundo!

→ 2000

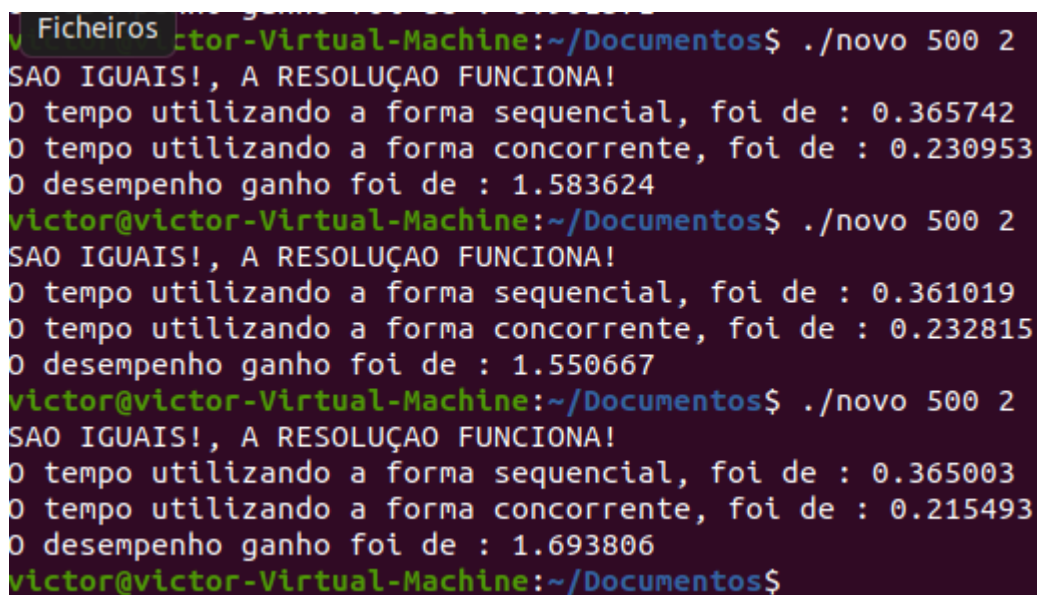
```
O desempenho ganho foi de : 0.814865
victor@victor-Virtual-Machine:~/Documentos$ ./novo 2000 1
SAO IGUAIS!, A RESOLUÇÃO FUNCIONA!
O tempo utilizando a forma sequencial, foi de : 68.706804
O tempo utilizando a forma concorrente, foi de : 74.089244
O desempenho ganho foi de : 0.927352
victor@victor-Virtual-Machine:~/Documentos$ ./novo 2000 1
SAO IGUAIS!, A RESOLUÇÃO FUNCIONA!
O tempo utilizando a forma sequencial, foi de : 71.060668
O tempo utilizando a forma concorrente, foi de : 74.759615
O desempenho ganho foi de : 0.950522
victor@victor-Virtual-Machine:~/Documentos$ ./novo 2000 1
SAO IGUAIS!, A RESOLUÇÃO FUNCIONA!
O tempo utilizando a forma sequencial, foi de : 73.091992
O tempo utilizando a forma concorrente, foi de : 76.013088
O desempenho ganho foi de : 0.961571
victor@victor-Virtual-Machine:~/Documentos$
```

O mais rápido foi o primeiro, com um melhor desempenho foi o último.

Quando utilizamos apenas uma thread, não temos nenhum ganho no desempenho do código. Visto que, quando utilizamos as threads, temos o foco em dividir a tarefa na quantidade de threads, então, quando só temos uma thread é como se ficasse como a função sequencial. Ademais, a elaboração exige um pouco mais de aperfeiçoamento do que a função sequencial (porém, isso se torna nulo quando utilizamos mais threads). Temos a utilização de ponteiros, outras funções e etc. Então, quando comparada com a sequencial, ela acaba perdendo no desempenho.(com uma thread)

Agora vamos ver com duas threads:

500:



```
Ficheiros victor@Victor-Virtual-Machine:~/Documentos$ ./novo 500 2
SAO IGUAIS!, A RESOLUÇÃO FUNCIONA!
O tempo utilizando a forma sequencial, foi de : 0.365742
O tempo utilizando a forma concorrente, foi de : 0.230953
O desempenho ganho foi de : 1.583624
victor@Victor-Virtual-Machine:~/Documentos$ ./novo 500 2
SAO IGUAIS!, A RESOLUÇÃO FUNCIONA!
O tempo utilizando a forma sequencial, foi de : 0.361019
O tempo utilizando a forma concorrente, foi de : 0.232815
O desempenho ganho foi de : 1.550667
victor@Victor-Virtual-Machine:~/Documentos$ ./novo 500 2
SAO IGUAIS!, A RESOLUÇÃO FUNCIONA!
O tempo utilizando a forma sequencial, foi de : 0.365003
O tempo utilizando a forma concorrente, foi de : 0.215493
O desempenho ganho foi de : 1.693806
victor@Victor-Virtual-Machine:~/Documentos$
```

O melhor desempenho foi o terceiro.

1000:

```
victor@victor-Virtual-Machine:~/Documentos$ ./novo 1000 2
SAO IGUAIS!, A RESOLUCAO FUNCIONA!
O tempo utilizando a forma sequencial, foi de : 3.579969
O tempo utilizando a forma concorrente, foi de : 2.045482
O desempenho ganho foi de : 1.750183
victor@victor-Virtual-Machine:~/Documentos$ ./novo 1000 2
SAO IGUAIS!, A RESOLUCAO FUNCIONA!
O tempo utilizando a forma sequencial, foi de : 3.686034
O tempo utilizando a forma concorrente, foi de : 2.060884
O desempenho ganho foi de : 1.788570
victor@victor-Virtual-Machine:~/Documentos$ ./novo 1000 2
SAO IGUAIS!, A RESOLUCAO FUNCIONA!
O tempo utilizando a forma sequencial, foi de : 3.566081
O tempo utilizando a forma concorrente, foi de : 2.120648
O desempenho ganho foi de : 1.681600
```

O mais rápido foi o primeiro e com o melhor desempenho o segundo.

2000:

```
victor@victor-Virtual-Machine:~/Documentos$ ./novo 2000 2
SAO IGUAIS!, A RESOLUCAO FUNCIONA!
O tempo utilizando a forma sequencial, foi de : 68.389600
O tempo utilizando a forma concorrente, foi de : 36.210680
O desempenho ganho foi de : 1.888658
victor@victor-Virtual-Machine:~/Documentos$ ./novo 2000 2
SAO IGUAIS!, A RESOLUCAO FUNCIONA!
O tempo utilizando a forma sequencial, foi de : 74.728933
O tempo utilizando a forma concorrente, foi de : 39.647229
O desempenho ganho foi de : 1.884846
victor@victor-Virtual-Machine:~/Documentos$ ./novo 2000 2
SAO IGUAIS!, A RESOLUCAO FUNCIONA!
O tempo utilizando a forma sequencial, foi de : 69.160850
O tempo utilizando a forma concorrente, foi de : 36.457891
O desempenho ganho foi de : 1.897006
```

O mais rápido foi o primeiro, em questão de desempenho foi o último.

Como podemos observar, os resultados estão como o esperado. Quando aumentamos os números de threads, conseguimos fazer com que as tarefas sejam executadas ao mesmo tempo, assim, ocasionando um melhor desempenho. Quanto mais aumentamos a dimensão, maior é o ganho de desempenho, isso ocorre pois na função sequencial vai ficando mais complicado, enquanto na concorrente as threads conseguem manter um ótimo desempenho, dividindo as tarefas.

CONCLUSÃO: A utilização de threads é essencial para melhorar o desempenho em algum código, foi notório a melhora ao utilizar duas threads. Pois ao dividirmos uma função para ser feita ao mesmo tempo, em vez de fazer uma por vez, temos um grande ganho no desempenho. Porém, a resolução tem que ser sempre muito bem pensada, para não acabarmos atrapalhando o código, do que melhorando-o.