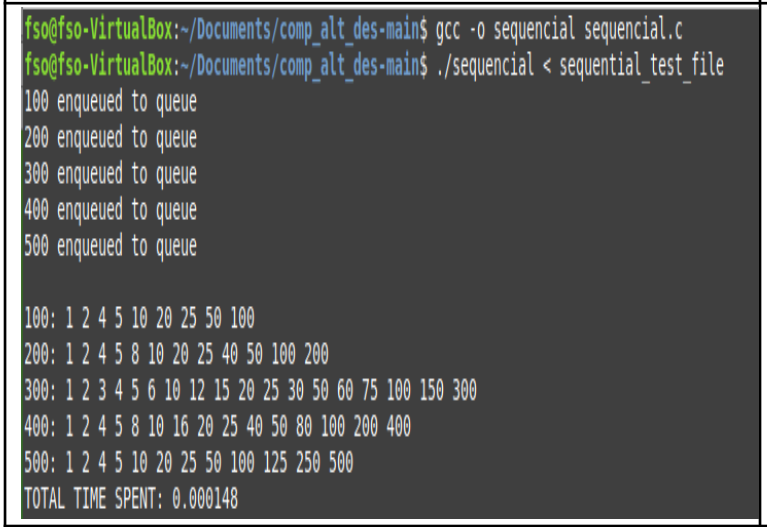
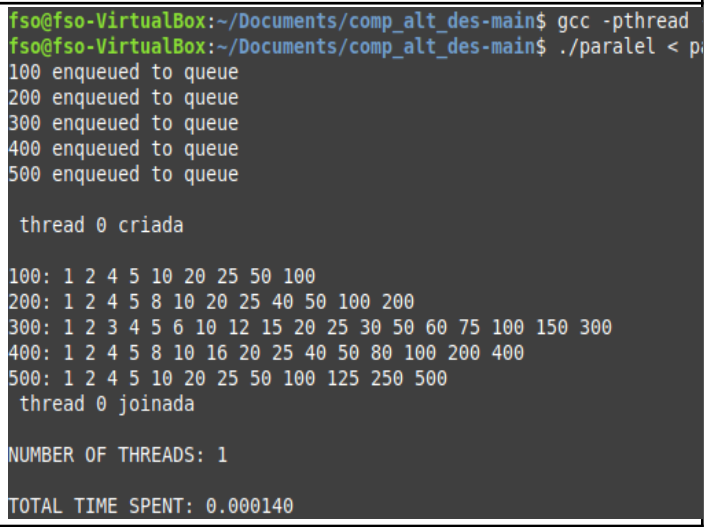
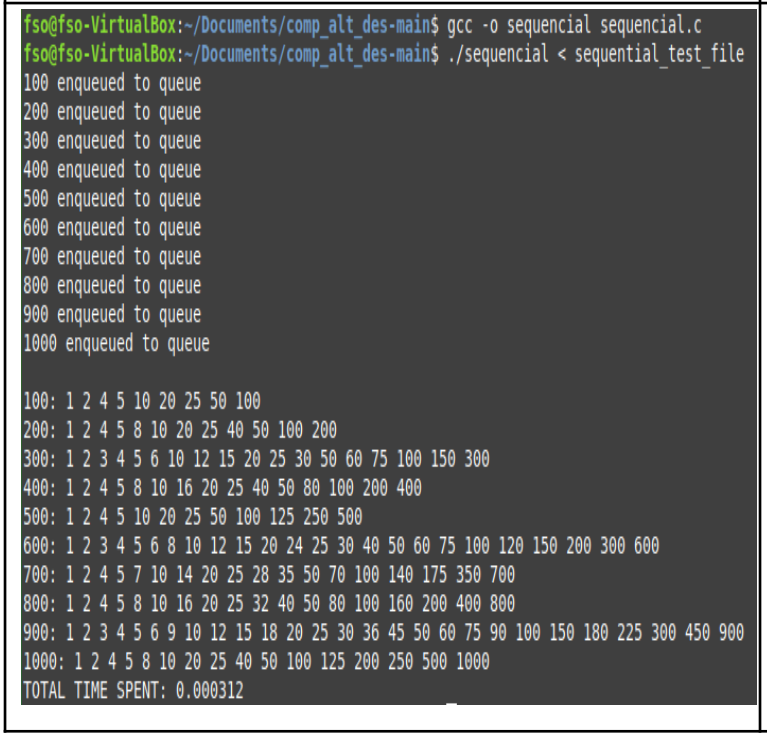
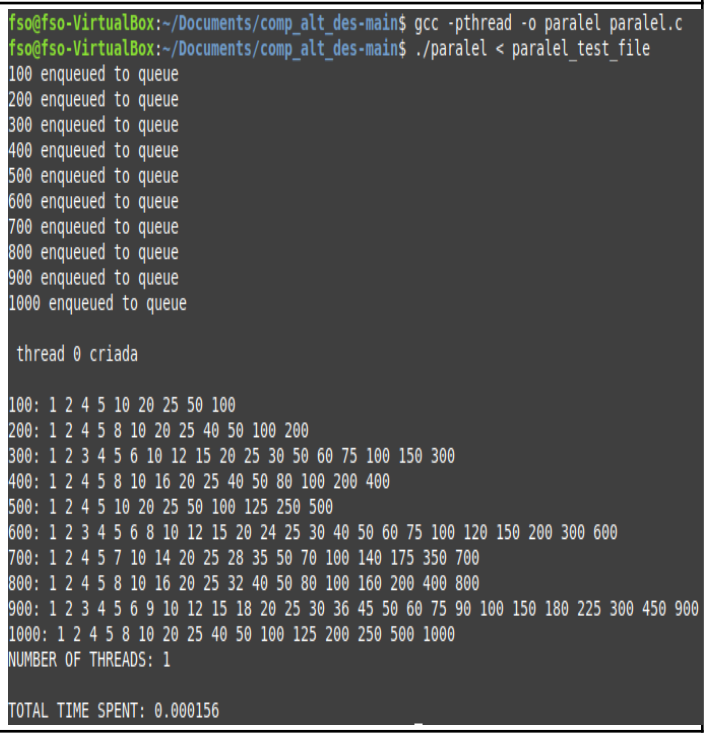
Trabalho GA - Computação de Alto Desempenho

Lucas Schneider

Michel Martins

Comparação de execução sequencial e paralela:

Sequencial	Paralela
-	1 thread
5 números	5 números
	
Sequencial	Paralela
-	1 thread
10 números	10 números
	

Sequencial	Paralela
-	4 threads
15 números	15 números

```
fso@fso-VirtualBox:~/Documents/comp_alt_des-main$ ./sequencial < sequential_test_file
10 enqueued to queue
20 enqueued to queue
30 enqueued to queue
40 enqueued to queue
50 enqueued to queue
60 enqueued to queue
70 enqueued to queue
80 enqueued to queue
90 enqueued to queue
100 enqueued to queue
200 enqueued to queue
300 enqueued to queue
400 enqueued to queue
500 enqueued to queue
600 enqueued to queue

10: 1 2 5 10
20: 1 2 4 5 10 20
30: 1 2 3 5 6 10 15 30
40: 1 2 4 5 8 10 20 40
50: 1 2 5 10 25 50
60: 1 2 3 4 5 6 10 12 15 20 30 60
70: 1 2 5 7 10 14 35 70
80: 1 2 4 5 8 10 16 20 40 80
90: 1 2 3 5 6 9 10 15 18 30 45 90
100: 1 2 4 5 10 20 25 50 100
200: 1 2 4 5 8 10 20 25 40 50 100 200
300: 1 2 3 4 5 6 10 12 15 20 25 30 50 60 75 100 150 300
400: 1 2 4 5 8 10 16 20 25 40 50 80 100 200 400
500: 1 2 4 5 10 20 25 50 100 125 250 500
600: 1 2 3 4 5 6 8 10 12 15 20 24 25 30 40 50 60 75 100 120 150 200 300 600
TOTAL TIME SPENT: 0.000272
```

```
fso@fso-VirtualBox:~/Documents/comp_alt_des-main$ ./paralel < paralel_test_file
10 enqueued to queue
20 enqueued to queue
30 enqueued to queue
40 enqueued to queue
50 enqueued to queue
60 enqueued to queue
70 enqueued to queue
80 enqueued to queue
90 enqueued to queue
100 enqueued to queue
200 enqueued to queue
300 enqueued to queue
400 enqueued to queue
500 enqueued to queue
600 enqueued to queue

10: 1 2 5 10
20: 1 2 4 5 10 20
30: 1 2 3 5 6 10 15 30
40: 1 2 4 5 8 10 20 40
50: 1 2 5 10 25 50
60: 1 2 3 4 5 6 10 12 15 20 30 60
70: 1 2 5 7 10 14 35 70
80: 1 2 4 5 8 10 16 20 40 80
90: 1 2 3 5 6 9 10 15 18 30 45 90
100: 1 2 4 5 10 20 25 50 100
200: 1 2 4 5 8 10 20 25 40 50 100 200
400: 1 2 4 5 8 10 16 20 25 40 50 80 100 200 400
500: 1 2 4 5 10 20 25 50 100 125 250 500
600: 1 2 3 4 5 6 8 10 12 15 20 24 25 30 40 50 60 75 100 120 150 200 300 600
300: 1 2 3 4 5 6 10 12 15 20 25 30 50 60 75 100 150 300
NUMBER OF THREADS: 4

TOTAL TIME SPENT: 0.000253
```

Com os inputs providos pela professora (arquivo e1, o mais custoso computacionalmente):

Os resultados gerais da computação serão omitidos para facilitar a visualização.

Execução Sequencial:

Tempo de processamento total: 73.04 segundos.

Execução paralela:

Tempo de execução com 2 threads: 34.06 segundos

Tempo de execução com 4 threads: 20.04 segundos

Tempo de execução com 8 threads: 15.03 segundos

Tempo de execução com 16 threads: 13.070 segundos

Tempo de execução com 32 threads: 12.088 segundos

Conclusão

Como podemos notar acima, a execução sequencial, tende a ter em sua grande maioria, um tempo de execução maior que a execução em paralelo. Apesar de serem poucos dados, ainda é possível ver uma diferença de 100% do tempo na execução com 10 números e com apenas uma thread. Porém, essa diferença tende a ficar maior, de acordo com o número de dados a serem processados e o número de threads disponíveis para execução.

No caso da execução mais custosa (arquivo e1), é possível perceber que um platô se forma e o ganho de performance se torna mínimo quando o número de threads fica muito alto. A provável causa deste problema é o lock de leitura na fila de dados a serem processados, conforme o número de threads aumenta, a fila fica mais tempo bloqueada, se tornando um gargalo na execução.

Uma forma possível de combater isso seria dividir os inputs em diferentes filas e atribuir cada uma das filas a um número menor de threads e controlar os locks individualmente, diminuindo o tempo total que a fila fica bloqueada e, possivelmente, aumentando a performance geral.