CURSO SUPERIOR DE BACHARELADO EM SISTEMAS DE INFORMAÇÕES

INSTITUTO FEDERAL DO ESPÍRITO SANTO

JOSUÉ RAMOS e JOÃO VITOR DELLAQUA

INTRODUÇÂO À PROGRAMAÇÂO MULTITHREAD COM PTHREADS

SERRA-ES

2022

> Introdução

Nesse trabalho nós desenvolvemos um programa que busca por números primos em uma matriz de tamanho 10.000x10.000 aleatória, essa busca pode ser feita de modo paralelo usando um determinado número de threads. A matriz em que realizamos a busca é dividida em macroblocos (pedaços da matriz) de tamanho definido, cada thread lê um macrobloco e conta os números primos nele, por fim mostramos quantos números primos nós contamos.

Para a realização desse programa utilizamos das PThreads para paralelizar a busca pelos números primos, nesse relatório se encontram algumas observações e considerações que encontramos enquanto desenvolvíamos o trabalho, os valores de tempo e speedup ao utilizar diferentes números de threads e alguns gráficos para ilustrar o que encontramos.

> Resultados e Testes

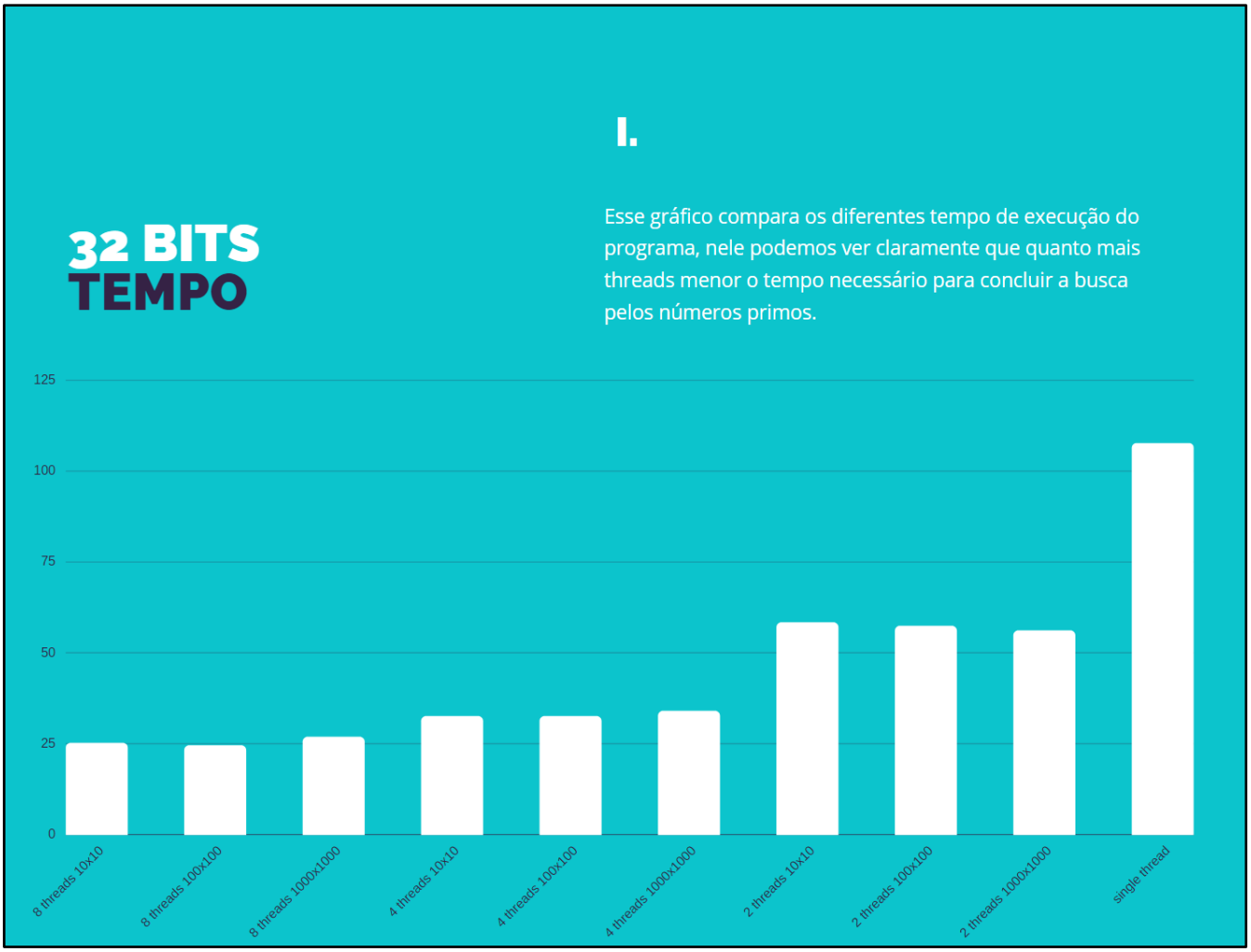
Com o programa funcionando começamos a testar qual era a diferença entre as velocidades de busca quando mudamos os números de threads usados pelo programa, nós testamos para 8 threads, 4 threads, 2 threads e single thread, por fim testamos usando 100 threads a fim de testar o overhead. Para os testes feitos em 8, 4 e 2 threads também testamos mudar o tamanho dos macroblocos para 10x10, 100x100 e 1000x1000.

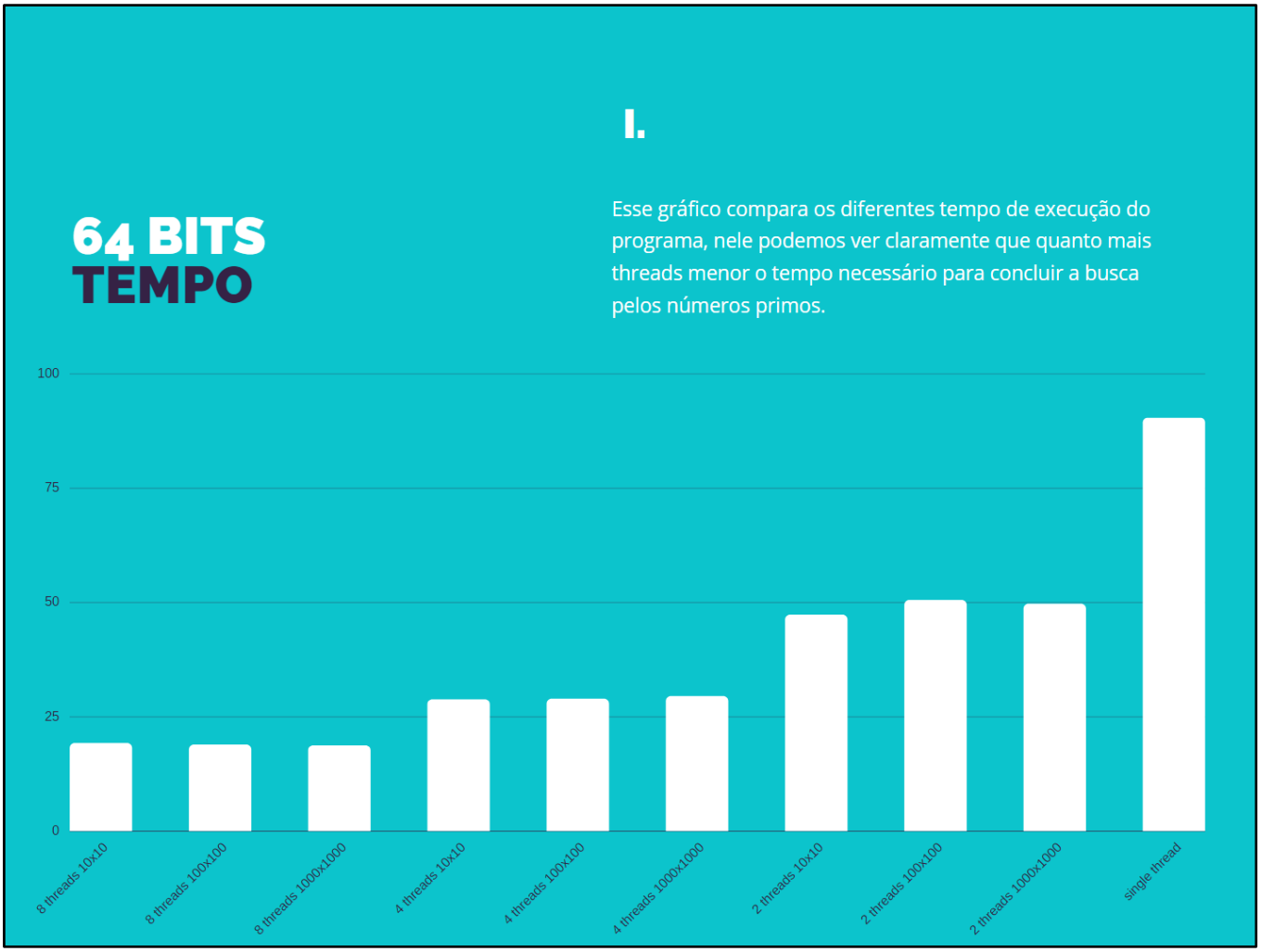
Esses testes foram realizados em 64 bits e em 32 bits, cada teste foi realizado algumas vezes a fim de ter o resultado mais fidedigno possível (o valor que escolhemos mostrar para cada teste é a mediana entre os resultados).

Um ponto importante que deve ser dito é que não realizamos a comparação entre dois computadores diferentes, porque a configuração dos computadores dos integrantes do trabalho era exatamente igual (i7-7700HQ, 16GB RAM e GTX 1050), nós até testamos o código nos dois computadores, porém como os resultados encontrados eram praticamente iguais decidimos não incluir essa parte.

nós até procuramos alguém com um computador diferente que pudesse realizar os testes, mas não conseguimos a tempo

Os gráficos abaixo mostram todos os resultados encontrados menos o resultado para os testes com 100 threads, esse resultado está separado mais para frente do trabalho.



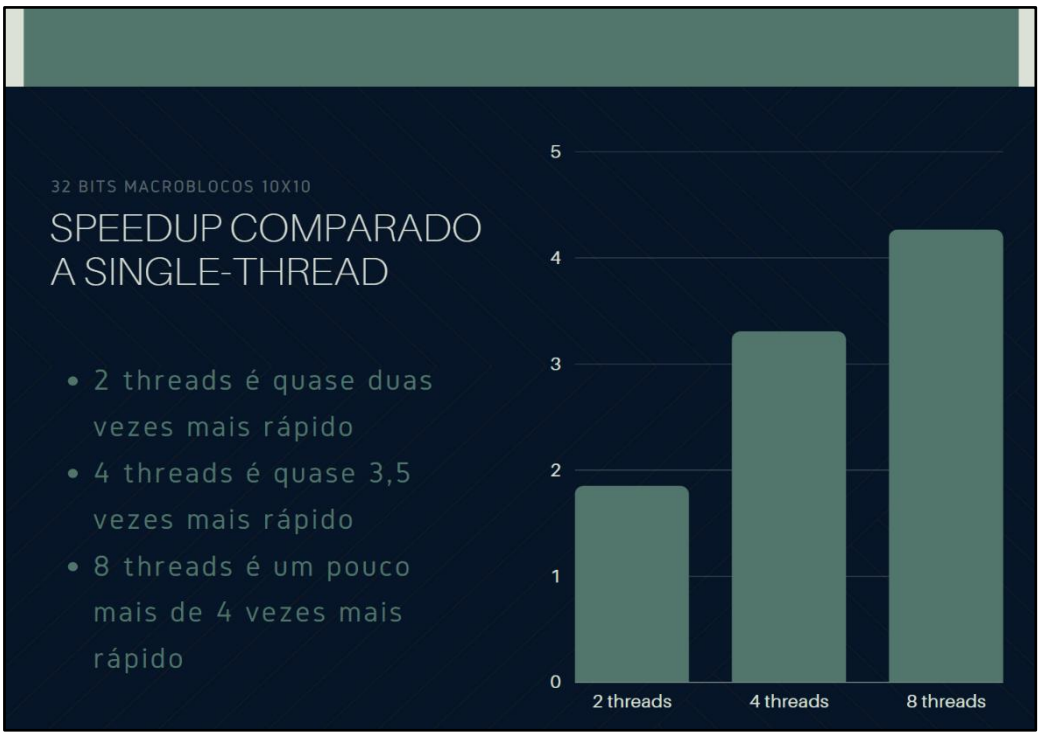


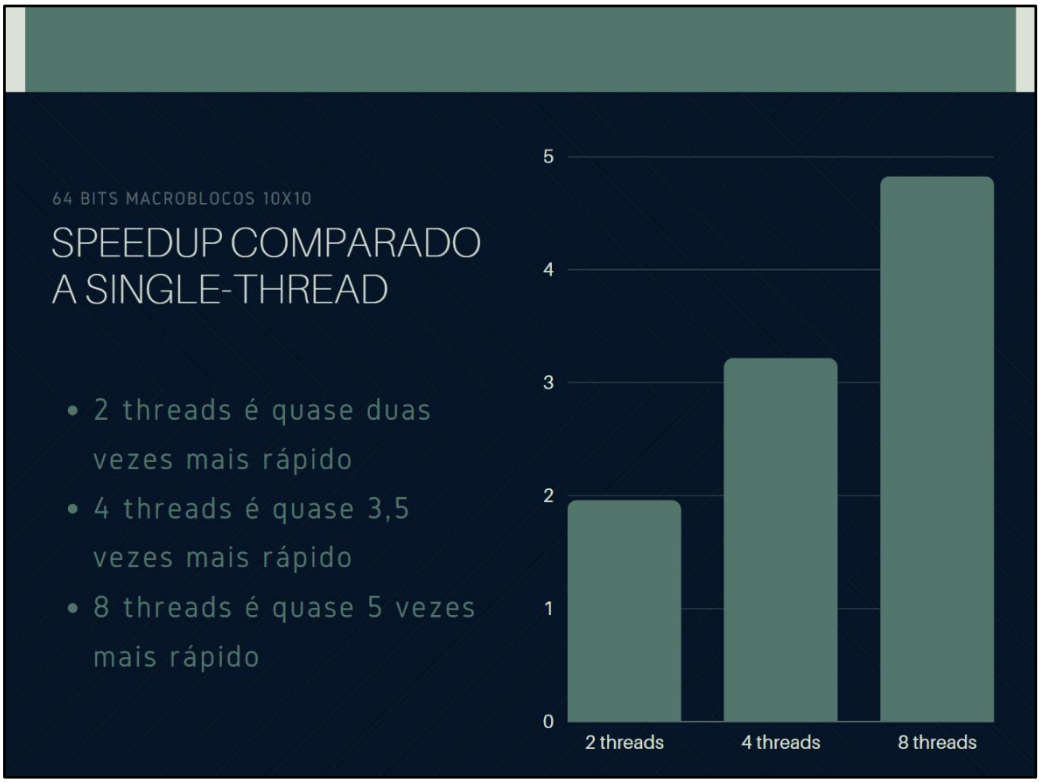
Nos gráficos nos podemos ver claramente que quanto mais threads menor o tempo necessário para concluir a busca, tanto em 32 bits quanto em 64 bits, também podemos notar que em 64 bits o tempo total é menor em todos os casos. Por fim notamos que o tamanho do macrobloco não foi tão impactante quanto esperávamos, a maior diferença que encontramos foi entre os resultados de 8 threads em 32 bits com macroblocos 100x100 e com macrobocos 1000x1000, utilizar de macroblocos 1000x1000 nesse caso teve um tempo de execução 9,48% maior (26,8719 segundo contra 24,5450 segundos).

	8 threads	4 threads	2 threads	Single Thread
32 bits 10x10	25,2395	32,584	58,3125	107,6233
32 bits 100x100	24,545	32,5656	57,3462	
32 bits 1000x1000	26,8719	24,0102	56,1309	
64 bits 10x10	19,1401	28,7265	47,2183	92,3052
64 bits 100x100	18,8166	28,8488	50,4388	
64 bits 1000x1000	18,6447	29,4053	49,604	

Tabela com os resultados dos testes

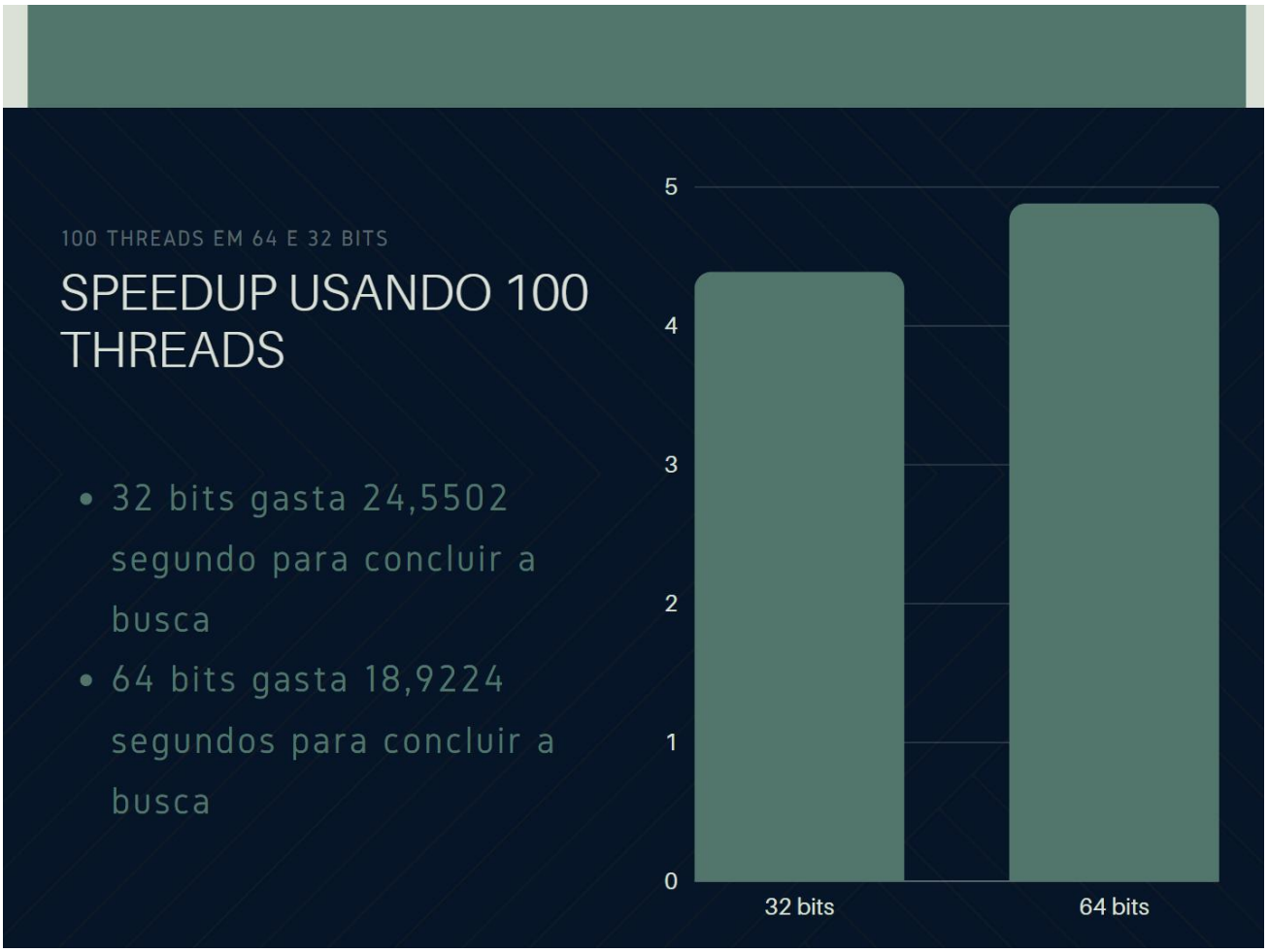
Com os tempos anotados o próximo passo foi comparar os speedups, para isso comparamos o tempo de execução single-thread com o tempo de execução dos diferentes multi-threads (nesses testes utilizamos macroblocos de tamanho 10x10). Novamente as comparações foram feitas em 32 bits e em 64 bits.





Podemos ver nos resultados de speedup que o tempo de execução não diminui de forma linear (por exemplo, ter 4 threads não quer dizer que o programa executa em 4x menos tempo), o “impacto” de ter mais threads diminui a cada thread (matéria de AOC ai). Além disso, é notável a diferença do speedup em 64 bits para 32 bits, em 64 bits o speedup é consideravelmente maior por exemplo nas 8 threads o speedup em 64 bits é de 4,8226x, já em 32 bits é de 4,2640x oque resulta em uma diferença de 0,5586x.

Por último ficaram as análises de quando utilizamos 100 threads (para esses testes usamos os macroblocos de tamanho 100x100)



Quando utilizamos um número muito alto de threads o tempo de execução do programa aumentou um pouco (comparando com o tempo de 8 threads) quando compilamos para 64 bits (o tempo foi de 18,8166 para 18,9224), já no programa compilado para 32 bits foi de 26,8719 para 24,5502, sendo assim o programa teve um aumento na velocidade de execução em 32 bits porém diminui seu tempo de execução em 64 bits **sinceramente não sei porque isso aconteceu**.

*** ESPAÇO PARA OS MUTEX ***

> Conclusão

Nesse trabalho podemos ver claramente as vantagens da programação multithread, ao utilizar esses recursos vimos um aumento considerável na velocidade de execução do nosso código e também notamos que aumentar as threads no código além da quantidade de threads que o processador possui fez pouco diferença no tempo de execução, podendo ter um tempo até pior (overhead).

Além disso, durante o desenvolvimento do programa utilizando os mutex podemos ver onde os mutex devem ser utilizados e o que acontece se não utilizarmos eles (valor de soma diferente). Nesse trabalho podemos aprender um pouco sobre programação multithread, sobre os cuidados de como utilizar os mutex e ainda por cima revisamos alguns conceitos que aprendemos durante a disciplina de AOC.