INSTITUTO FEDERAL DO ESPÍRITO SANTO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO
SISTEMAS OPERACIONAIS

Trabalho 1 - PThreads

Introdução à Programação Multithread com PThreads

Alex Rossoni
João Pedro Pagotto
Sofia De Alcantara

Serra - ES
2024

Sumário

Introdução

Introduzidas pelo britânico Arthur Cayley em 1855, as matrizes são estruturas matemáticas famosas, amplamente utilizadas nas áreas das Ciências, Administração, Matemática e, claro, na Computação. As matrizes são especialmente úteis para representar e manipular dados de forma estruturada, permitindo um amplo conjunto de operações matemáticas e algoritmos complexos sejam implementados.

Neste contexto, vamos explorar no trabalho em questão um algoritmo que trabalha com matrizes de números naturais aleatórios, no intervalo de 0 (zero) a 31999 (trinta e um mil e novecentos e noventa e nove). O objetivo final do algoritmo é contabilizar quantos números primos existem na matriz, medindo o tempo necessário para essa contagem. Para isso, implementamos duas abordagens distintas:

- **Modo Serial**

Nesta abordagem, a contagem dos números primos será realizada de forma sequencial, verificando um número após o outro. Este método servirá como referência para comparação de desempenho.

- **Modo Paralelo**

Nesta abordagem, a matriz será subdividida em "macroblocos" (submatrizes) para distribuir a verificação de números primos entre múltiplos processos ou threads. A subdivisão será feita sem cópia de dados, utilizando apenas os índices para definir os limites de cada macrobloco. Essa técnica tem como objetivo acelerar o processo de contagem ao aproveitar a capacidade de processamento paralelo das CPUs modernas.

Demais orientações de implementação do algoritmo proposto podem ser lidas aqui, na especificação elaborada pelo professor Flávio Giraldeli. A seguir, temos a tabela com as configurações básicas das máquinas utilizadas nos testes descritos no atual relatório.

Computadores utilizados e suas configurações básicas

	PC Alex	PC João Pedro	PC Sofia
Sistema Operacional	Windows 10	Windows 11	Windows 11
CPU			
Modelo	AMD Ryzen 5 1500X	Intel Core i7 8550U	Intel Core i7 13700KF
Frequência Base	3,50 GHz	1,80 GHz	3,40 GHz
Frequência Boost	3,70 GHz	4 GHz	5,40 GHz
Técnica de Fabricação	14 nm	14 nm	10 nm
Quant. Núcleos Físicos	4	4	8P + 8E
Quant. Núcleos Lógicos	8	8	24
Cache L1 por núcleo (Dados)	32 KB	32 KB	8 x 48 KB + 8 x 32 KB
Cache L1 por núcleo (Instruções)	64 KB	32 KB	8 x 32 KB + 8 x 64 KB
Cache L2 por núcleo (ou unificada)	512 KB	256 KB	8 x 2 MB + 2 x 4 MB
Cache L3 unificada (total)	16 MB	8 MB	30 MB
Ano de Lançamento	2017	2017	2022
Memória Principal			
Tecnologia	DDR4	DDR4	DDR5
Frequência Aparente (MHz)	3200 MHz	2400 MHz	5600 Mhz
Capacidade Total	16 GB	8 GB	32 GB
Quant. de Módulos	2	2	4

A configuração básica das máquinas utilizadas para desenvolvimento e testes foi listada acima. O Intel Core i7 13700KF tende a ser significativamente superior em processamento, com 8 núcleos de performance (P) e 8 núcleos de eficiência (E), totalizando 24 thread: característica que define uma arquitetura heterogênea. A presença de mais núcleos e threads facilita a execução de múltiplos processos simultaneamente, o que é crucial para aplicações multithreads.

"A principal razão para se ter threads é que em muitas aplicações múltiplas atividades estão ocorrendo simultaneamente e algumas delas podem bloquear de tempos em tempos. Ao decompormos uma aplicação dessas em múltiplos threads sequenciais que são executados em quase paralelo, o modelo de programação torna-se mais simples." (TANENBAUM, 2016)

Embora o AMD Ryzen 5 1500X tenha uma alta frequência base e boost e o Intel Core i7 8550U tenha uma alta frequência boost, o Intel Core i7 13700KF apresenta uma frequência boost muito maior (5,40 GHz), o que melhora a performance em tarefas intensivas de processamento.

O Intel Core i7 13700KF utiliza memória DDR5 com uma frequência de 5600 MHz, que é superior à DDR4 utilizada pelos outros dois processadores. Memórias mais rápidas e com maior capacidade são benéficas para operações que envolvem grandes volumes de dados e exigem alta largura de banda.

Considerando os aspectos técnicos apresentados, o computador com processador Intel Core i7 13700KF destaca-se como o melhor dos três para programação multithreads. Suas 24 threads, alta frequência boost, grande cache L3 e uso de memória DDR5 de alta frequência oferecem um desempenho significativamente superior. Mas, quão superior é esse desempenho? E como se compara aos demais? Vamos explorar isso em detalhes na seção de análise de desempenho.

Análise de Desempenho

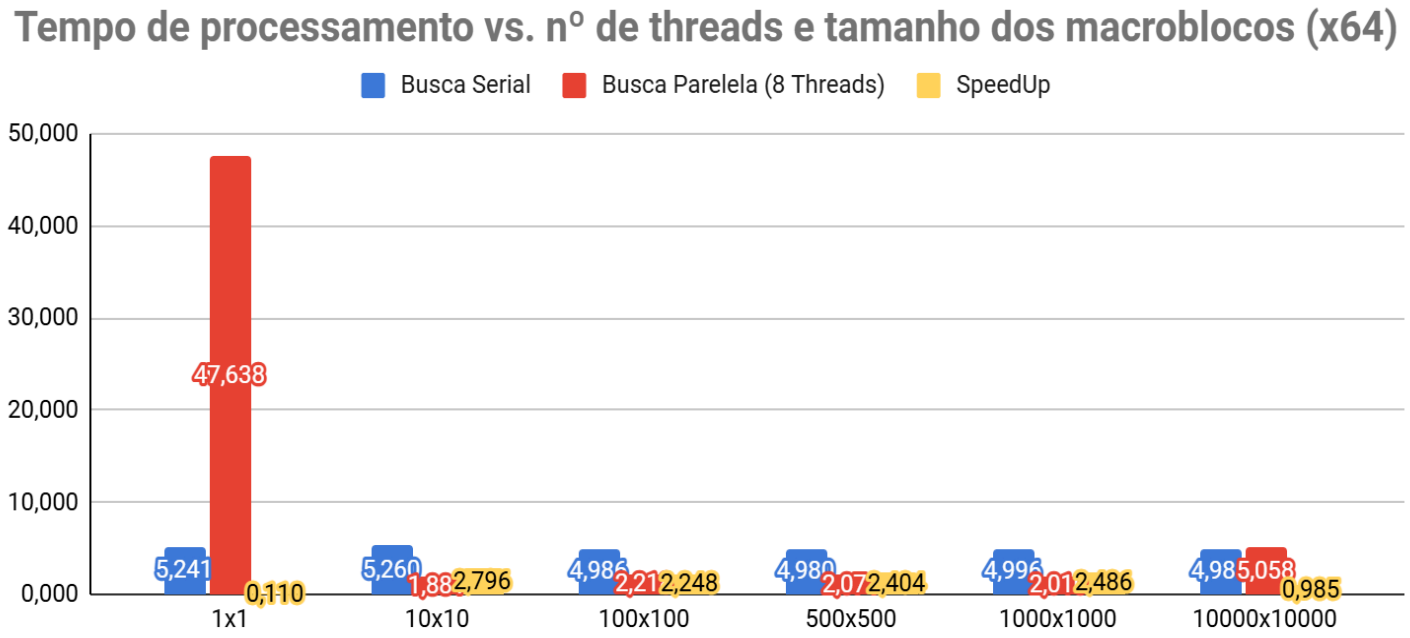
Nesta seção, apresentamos os resultados dos testes de desempenho realizados nas diferentes configurações de CPU e threads. Com o algoritmo solicitado na especificação desenvolvido, iniciamos os testes que visam avaliar como diferentes fatores, a exemplo do número de threads e o tamanho dos macroblocos, impactam o tempo de execução das tarefas. Abaixo estão descritos os cenários testados e seus resultados.

1. Número de threads fixo X Tempo de execução

Neste teste, o número de threads foi fixado de acordo com o número máximo de núcleos físicos de cada CPU. O objetivo foi comparar o tempo de execução em diferentes tamanhos de macroblocos, variando de muito pequenos (1x1) até muito grandes (10000x10000, tamanho total da matriz).

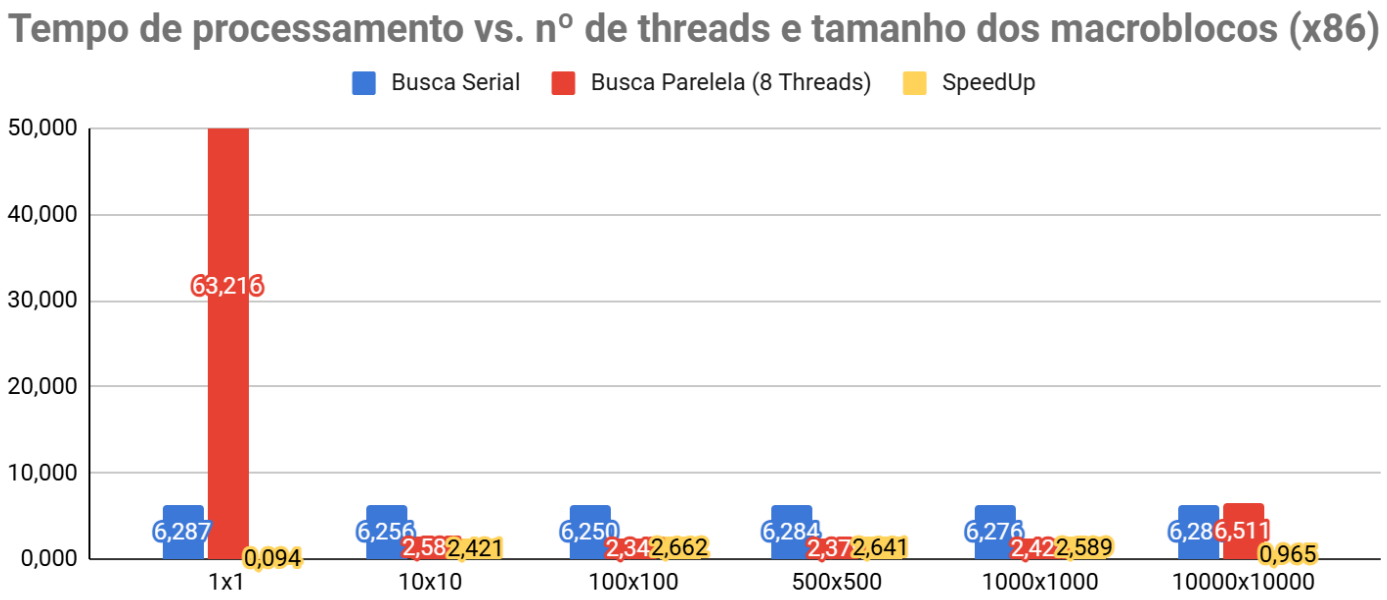
- **PC Sofia**

Intel Core i7 13700KF	**x64**					
Dimensões da matriz	10000x10000					
Tamanho dos macroblocos	1x1	10x10	100x100	500x500	1000x1000	10000x10000
Busca Serial	5,241	5,260	4,986	4,980	4,996	4,986
Busca Paralela (8 Threads)	47,638	1,881	2,218	2,071	2,010	5,058
SpeedUp	*0,110*	*2,796*	*2,248*	*2,404*	*2,486*	*0,985*



Código compilado em 64 bits

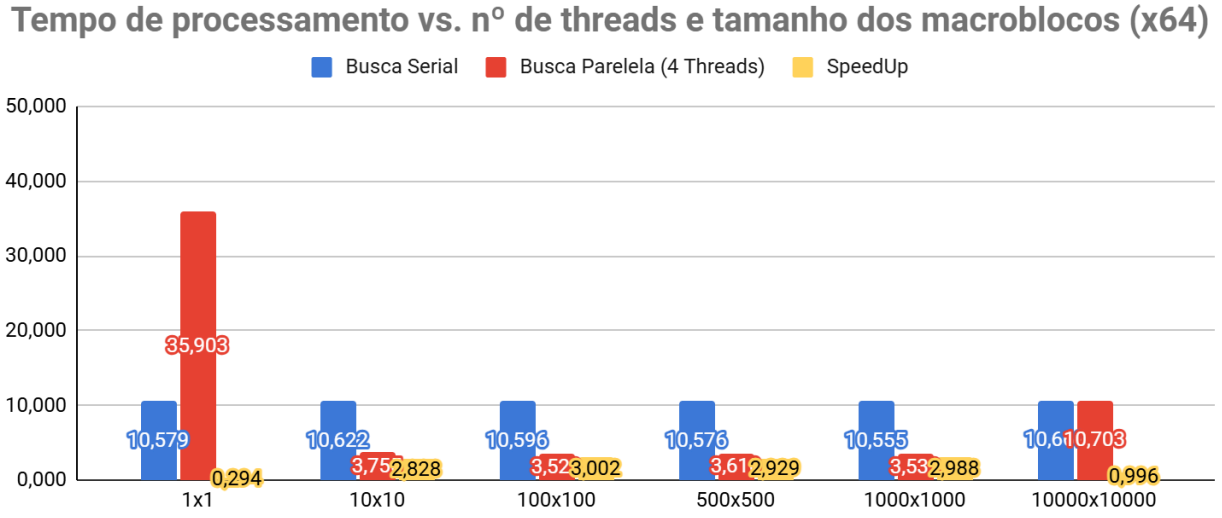
Intel Core i7 13700KF	**x86**					
Dimensões da matriz	10000x10000					
Tamanho dos macroblocos	1x1	10x10	100x100	500x500	1000x1000	10000x10000
Busca Serial	6,287	6,256	6,250	6,284	6,276	6,284
Busca Paralela (8 Threads)	63,216	2,587	2,347	2,379	2,424	6,511
SpeedUp	*0,094*	*2,421*	*2,662*	*2,641*	*2,589*	*0,965*



Código compilado em 32 bits

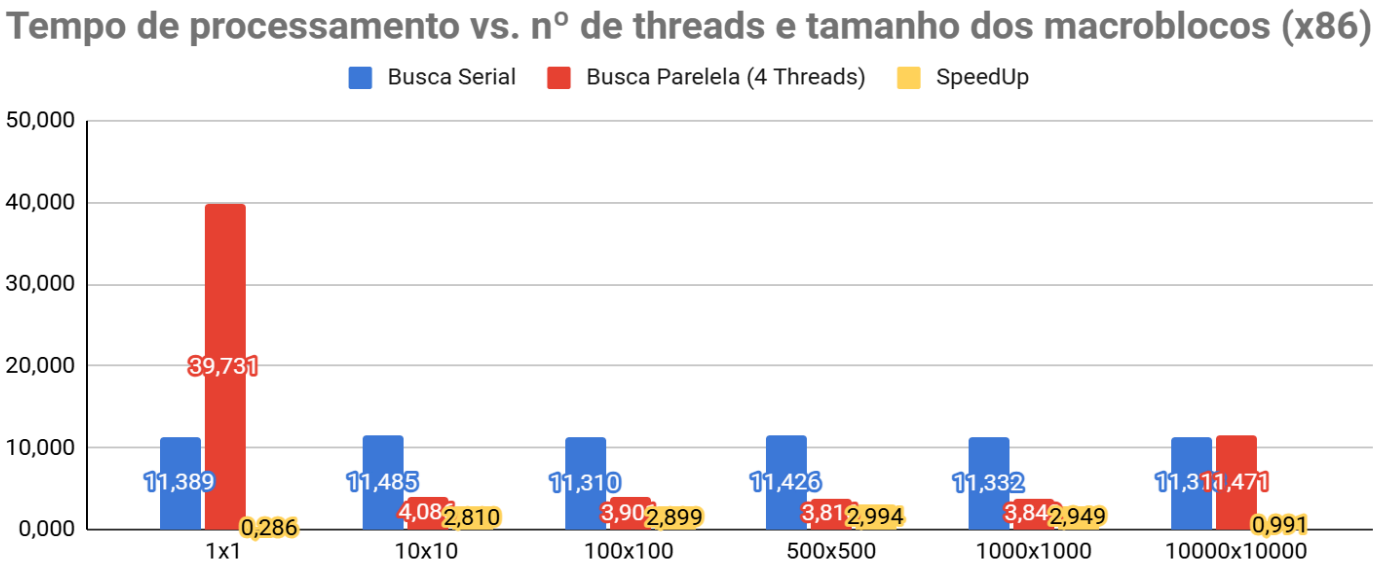
- **PC Alex**

AMD Ryzen 5 1500X	**x64**					
Dimensões da matriz	10000x10000					
Tamanho dos macroblocos	1x1	10x10	100x100	500x500	1000x1000	10000x10000
Busca Serial	10,579	10,622	10,596	10,576	10,555	10,669
Busca Paralela (4 Threads)	35,903	3,755	3,529	3,610	3,532	10,703
SpeedUp	*0,294*	*2,828*	*3,002*	*2,929*	*2,988*	*0,996*



Código compilado em 64 bits

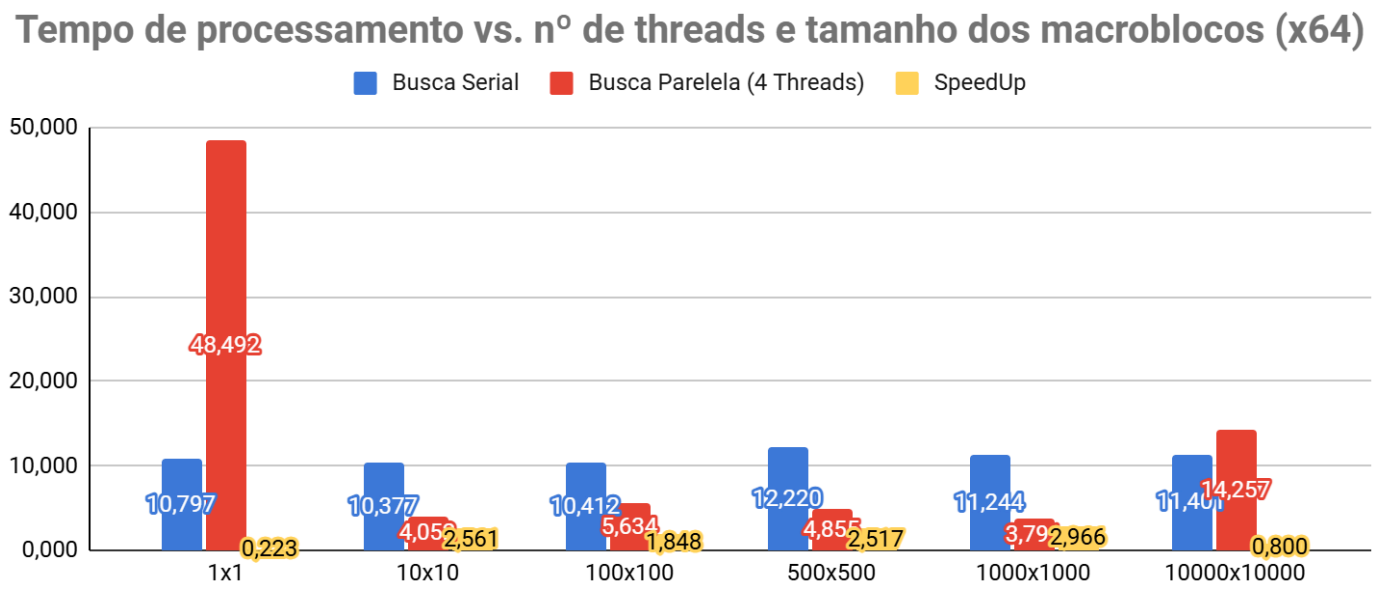
AMD Ryzen 5 1500X	**x86**					
Dimensões da matriz	10000x10000					
Tamanho dos macroblocos	1x1	10x10	100x100	500x500	1000x1000	10000x10000
Busca Serial	11,389	11,485	11,310	11,426	11,332	11,378
Busca Paralela (4 Threads)	39,731	4,086	3,901	3,816	3,842	11,471
SpeedUp	*0,286*	*2,810*	*2,899*	*2,994*	*2,949*	*0,991*



Código compilado em 32 bits

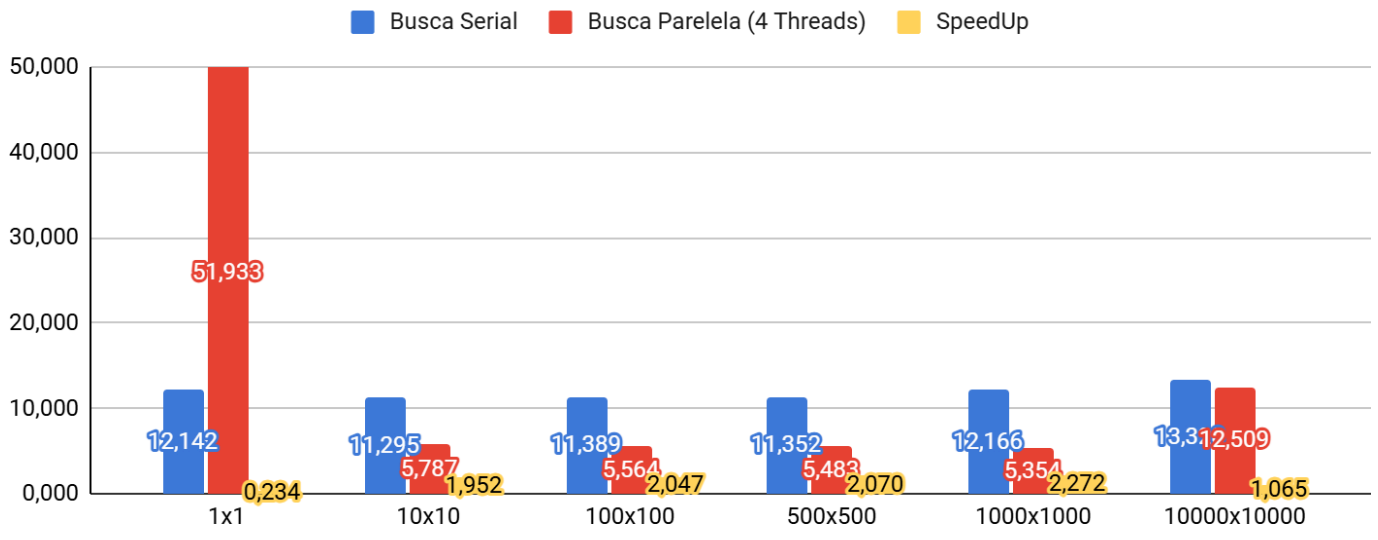
- **PC João Pedro**

Intel Core i7 8550U	**x64**					
Dimensões da matriz	10000x10000					
Tamanho dos macroblocos	1x1	10x10	100x100	500x500	1000x1000	10000x10000
Busca Serial	10,797	10,377	10,412	12,220	11,244	11,401
Busca Paralela (4 Threads)	48,492	4,052	5,634	4,855	3,791	14,257
SpeedUp	*0,223*	*2,561*	*1,848*	*2,517*	*2,966*	*0,800*



Código compilado em 64 bits

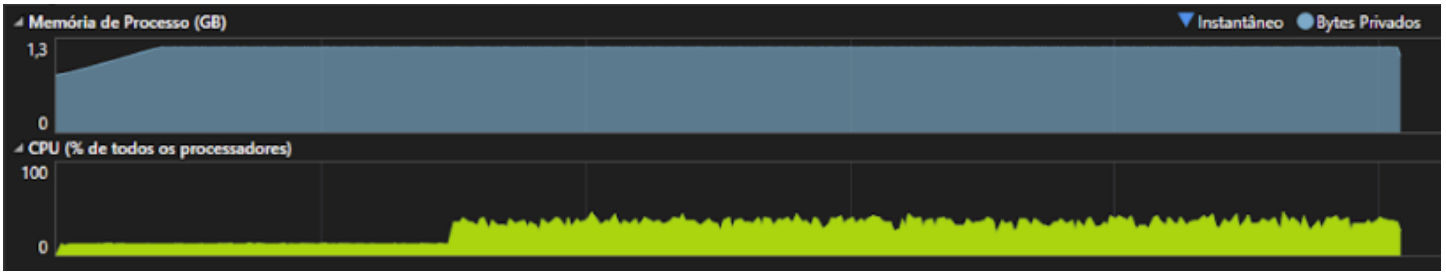
Intel Core i7 8550U	**x86**					
Dimensões da matriz	10000x10000					
Tamanho dos macroblocos	1x1	10x10	100x100	500x500	1000x1000	10000x10000
Busca Serial	12,142	11,295	11,389	11,352	12,166	13,328
Busca Paralela (4 Threads)	51,933	5,787	5,564	5,483	5,354	12,509
SpeedUp	*0,234*	*1,952*	*2,047*	*2,070*	*2,272*	*1,065*



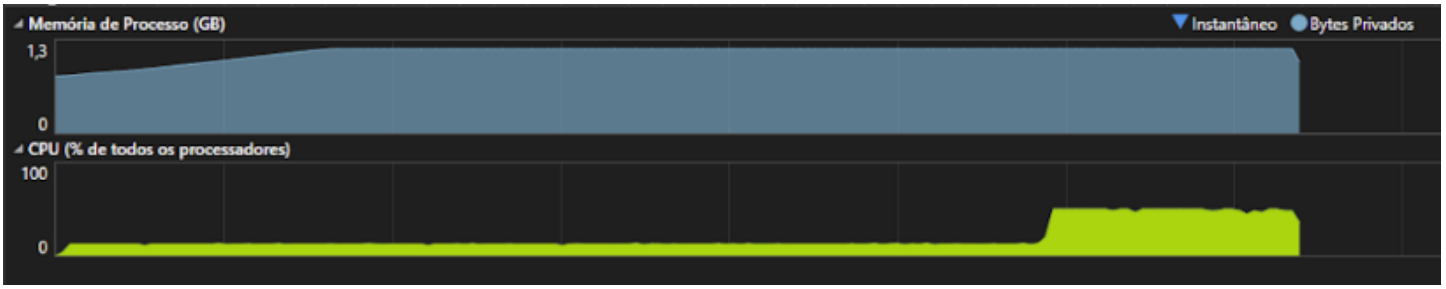
Código compilado em 32 bits

Ao analisar os resultados dos testes de desempenho nos três computadores, percebemos um padrão interessante: há uma faixa de tamanhos de macroblocos onde o tempo de execução é muito semelhante, independentemente do computador. Essa faixa vai de macroblocos de dimensões 10x10 até 1000x1000.

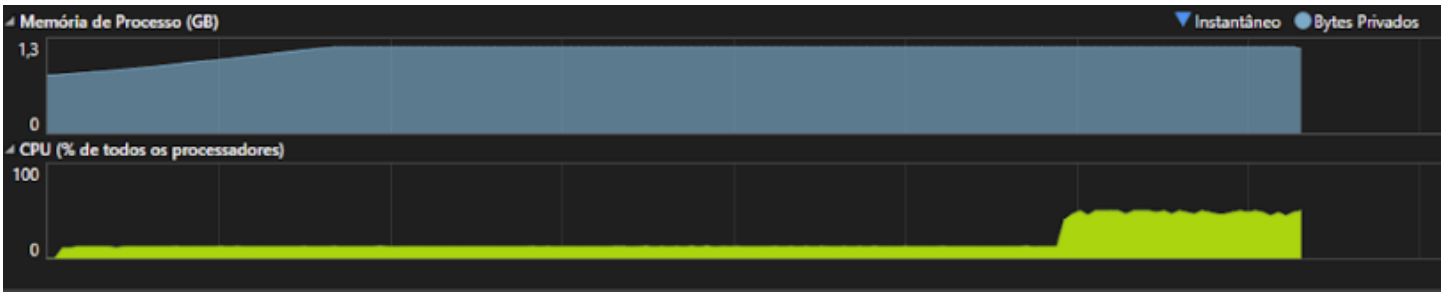
Pode-se observar também um padrão no comportamento do uso de memória e CPU nos casos em que o tempo de execução é semelhante (macroblocos de 10x10 até 1000x1000). Os gráficos a seguir, obtidos por meio das Ferramentas de Diagnóstico do Visual Studio, nos testes feitos com o PC do Alex (AMD Ryzen 5 1500X), ilustram as diferenças de comportamento entre esses casos e os casos de macroblocos de dimensões 1x1 e 10000x10000. No teste com macroblocos de 1x1, o uso da CPU chega a quase 50%, com considerável variação. Já no teste com macroblocos de 10000x10000, o gráfico é praticamente constante, sem variações bruscas, mantendo-se em aproximadamente 10% do uso geral da CPU. Também é interessante observar nos gráficos os pontos em que se iniciam a busca paralela, momento onde o uso da CPU aumenta consideravelmente em relação à constância anterior dos gráficos.



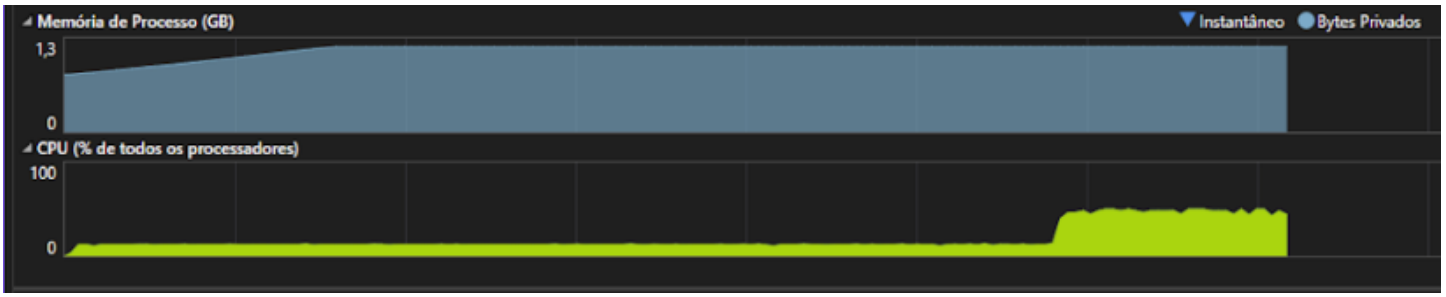
Macrobloco 1x1 - 4 Threads - x64 - Computador de processador AMD Ryzen 5 1500X



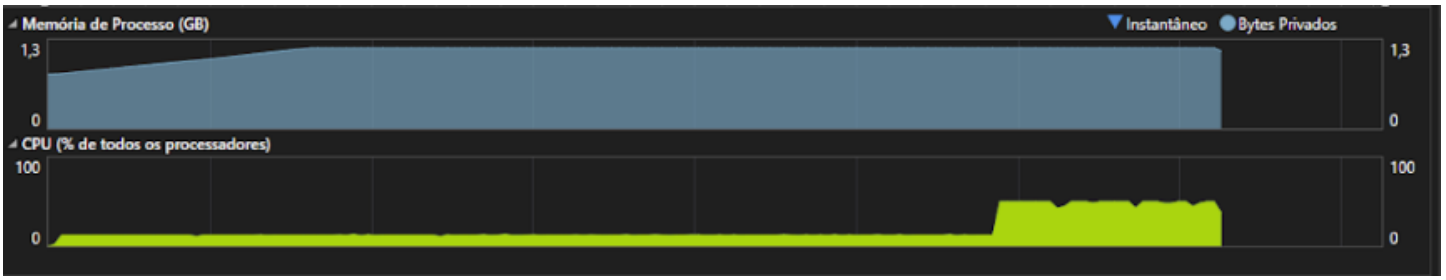
Macrobloco 10x10 - 4 Threads - x64 - Computador de processador AMD Ryzen 5 1500X



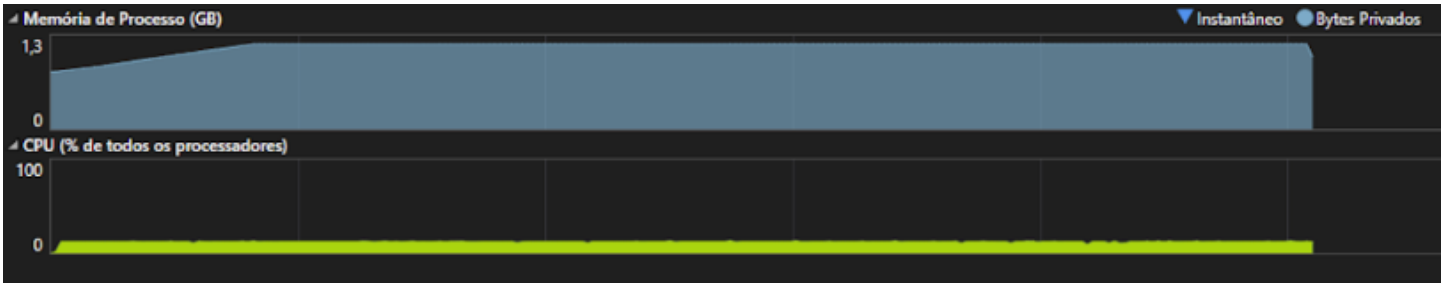
Macrobloco 100x100 - 4 Threads - x64 - Computador de processador AMD Ryzen 5 1500X



Macrobloco 500x500 - 4 Threads - x64 - Computador de processador AMD Ryzen 5 1500X



Macrobloco 1000x1000 - 4 Threads - x64 - Computador de processador AMD Ryzen 5 1500X



Macrobloco 10000x10000 - 4 Threads - x64 - Computador de processador AMD Ryzen 5 1500X

Observamos também que, quando usamos um único macrobloco para cobrir toda a matriz (dimensão 10000x10000), o tempo de execução da busca paralela é muito semelhante ao tempo de execução gasto na busca serial em todos os testes: isso ocorre pois uma única thread é atribuída para realizar a leitura de toda a matriz, o equivalente ao que ocorre na busca serial.

No entanto, quando os macroblocos têm dimensão mínima (1x1), há um aumento significativo no tempo de execução da busca paralela. Isso ocorre porque este tamanho de macrobloco implica a leitura de 100 milhões de blocos na matriz, resultando em uma sobrecarga gerencial considerável para as threads, enquanto a alta frequência de bloqueio e desbloqueio dos mutexes contribui significativamente para esse aumento no tempo de execução.

Também é possível observar uma clara diferença de desempenho do computador com CPU i7 13700KF em comparação aos outros: tanto na busca serial quanto na busca paralela o processador apresenta menores tempos de execução nos testes, com desempenho até duas vezes superior aos demais PC's. Essa superioridade de performance se deve as configurações mais avançadas e recentes desse computador, incluindo um maior número de núcleos, memória RAM mais rápida e com maior capacidade, uma frequência de boost mais alta e um cache L3 de maior tamanho, e também devido a presença de núcleos de eficiência.

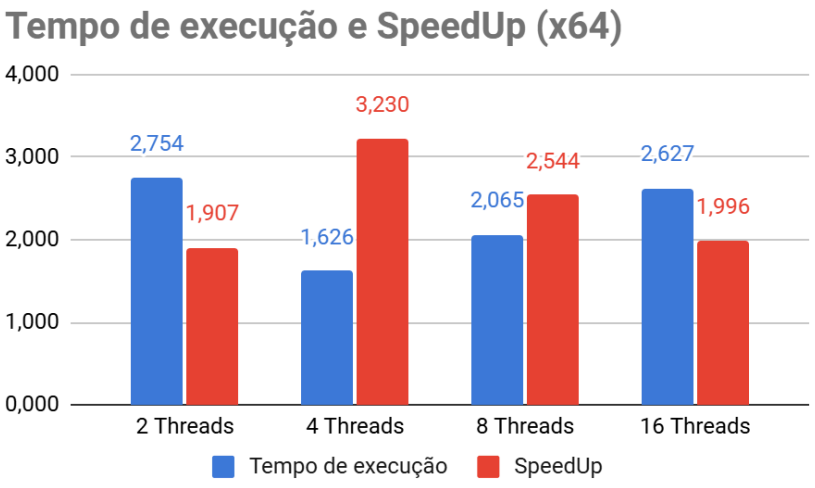
Ainda é possível visualizar nos resultados que a arquitetura de 64 bits oferece um desempenho significativamente melhor para o algoritmo de busca paralela em comparação com a arquitetura de 32 bits. Esta diferença é atribuída às vantagens técnicas da arquitetura de 64 bits, incluindo registradores maiores e melhor gerenciamento de memória. Por exemplo, considerando o caso do Intel Core i7 13700KF: em macroblocos 1x1, a Busca Paralela durou 47,638s em x64, enquanto em x86 o mesmo caso teve tempo de execução de 63,216s - um aumento de 32,72%.

2. Tamanho de macrobloco fixo X Tempo de execução

Para este teste, o tamanho dos macroblocos foi fixado, e o tempo de execução foi comparado com diferentes números de threads. Essa análise permitiu observar a relação entre a quantidade de threads e o desempenho na execução.

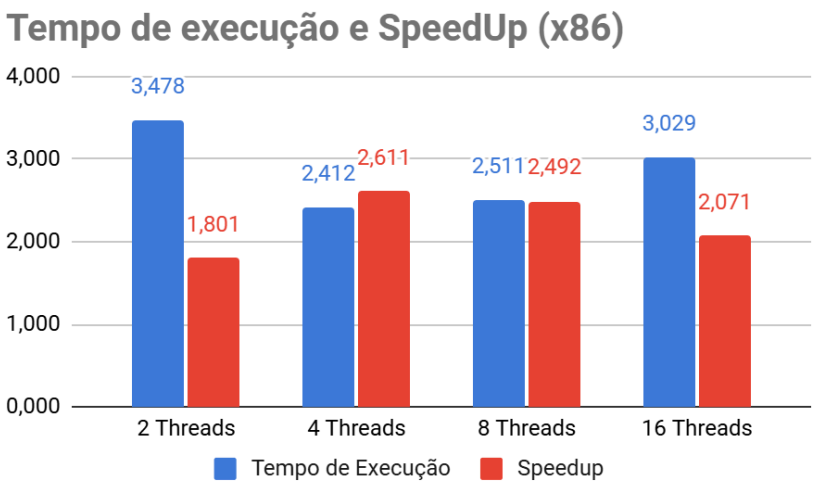
- **PC Sofia**

Intel Core i7 13700KF	**x64**	
Dimensões da matriz	10000x10000	
Dimensões do macrobloco	500x500	
	Tempo de execução	**SpeedUp**
2 Threads	2,754	1,907
4 Threads	1,626	3,230
8 Threads	2,065	2,544
16 Threads	2,627	1,996



Código compilado em 64 bits

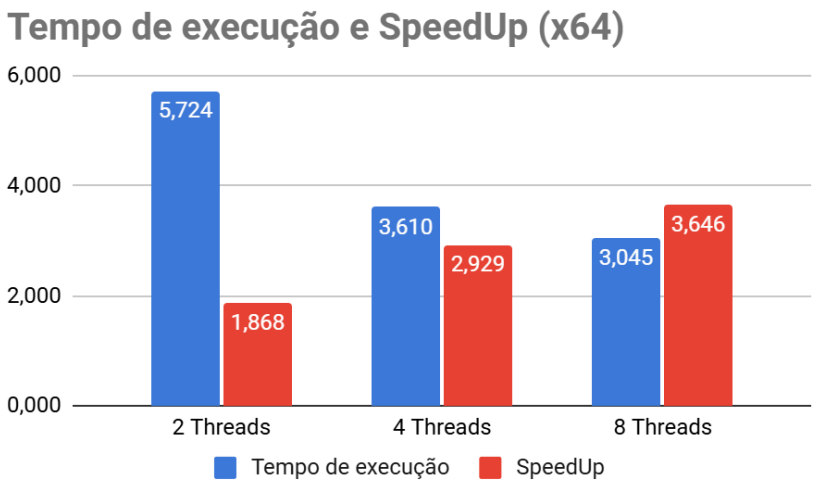
Intel Core i7 13700KF	**x86**	
Dimensões da matriz	10000x10000	
Dimensões do macrobloco	500x500	
	Tempo de execução	**SpeedUp**
2 Threads	3,478	1,801
4 Threads	2,412	2,611
8 Threads	2,511	2,492
16 Threads	3,029	2,071



Código compilado em 32 bits

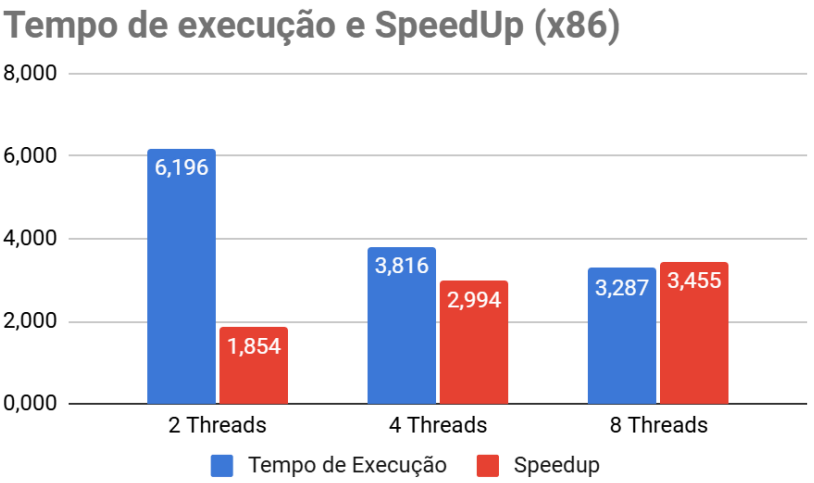
- **PC Alex**

AMD Ryzen 5 1500X	x64	
Dimensões da matriz	10000x10000	
Dimensões do macrobloco	500x500	
	Tempo de execução	**SpeedUp**
2 Threads	5,724	1,868
4 Threads	3,610	2,929
8 Threads	3,045	3,646



Código compilado em 64 bits

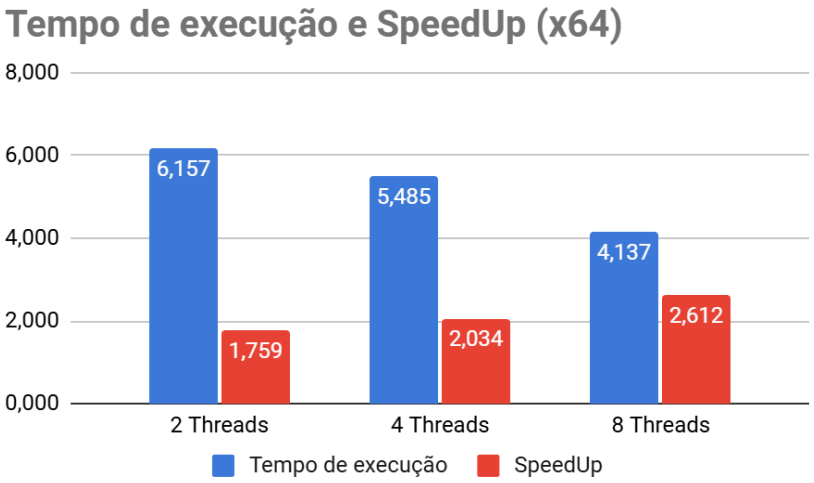
AMD Ryzen 5 1500X	x86	
Dimensões da matriz	10000x10000	
Dimensões do macrobloco	500x500	
	Tempo de execução	**SpeedUp**
2 Threads	6,196	1,854
4 Threads	3,816	2,994
8 Threads	3,287	3,455



Código compilado em 32 bits

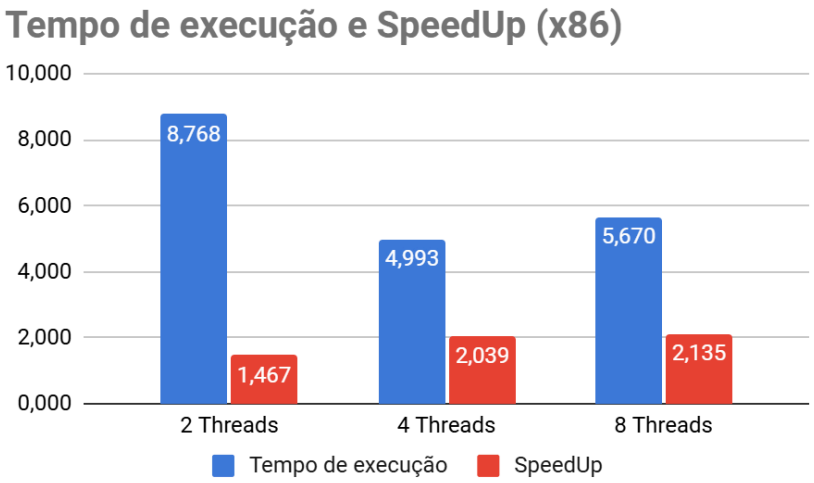
- **PC João Pedro**

Intel Core i7 8550U	x64	
Dimensões da matriz	10000x10000	
Dimensões do macrobloco	500x500	
	Tempo de execução	**SpeedUp**
2 Threads	6,157	1,759
4 Threads	5,485	2,034
8 Threads	4,137	2,612



Código compilado em 64 bits

Intel Core i7 8550U	x86	
Dimensões da matriz	10000x10000	
Dimensões do macrobloco	500x500	
	Tempo de execução	**SpeedUp**
2 Threads	8,768	1,467
4 Threads	4,993	2,039
8 Threads	5,670	2,135



Código compilado em 32 bits

Analisando os resultados utilizando diferentes números de threads, observamos que, apesar de pequenas variações normais, **especificamente nestes casos de testes**, o aumento no número de threads resulta em uma melhora no tempo de execução. Este comportamento está em conformidade com a Lei de Amdahl, que prevê um aumento no speedup de acordo com a quantidade de núcleos disponíveis. Além disso, há um ganho de desempenho significativo quando as CPUs passam a utilizar seus núcleos lógicos, evidenciando a eficiência em aproveitar a capacidade total de processamento do hardware.

Vale ressaltar que o processador Intel Core i7 13700KF possui, no total, 16 núcleos e 24 threads. Acontece que sua construção possui abordagem da arquitetura heterogênea, como citado

anteriormente. Dessa forma, os oito núcleos de alta eficiência (E-Cores) não foram considerados no teste, uma vez que não é possível aplicar a Lei de Amdhal em "*cores*" desse tipo.

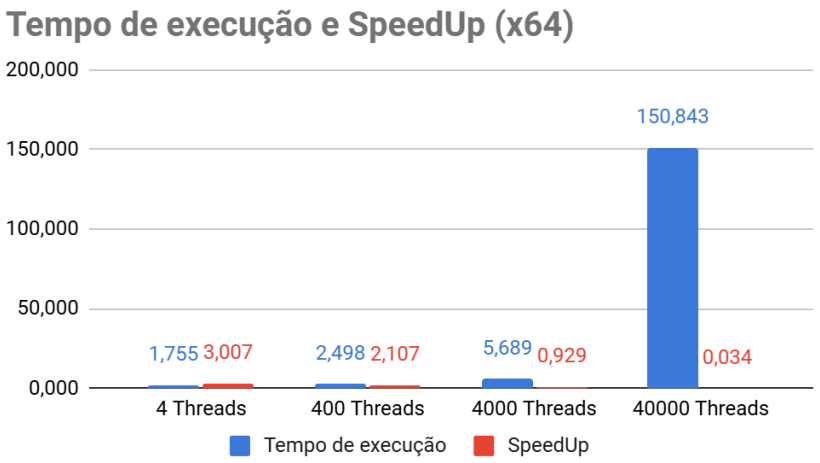
Resumidamente, a Lei de Amdahl assume que todos os N núcleos têm desempenho idêntico. Entretanto, em uma arquitetura heterogênea, os núcleos de alta eficiência são projetados para menor consumo de energia, não para o máximo desempenho. Portanto, o tempo de execução das partes paralelas do programa pode variar significativamente entre os núcleos de alta performance (P-cores) e os E-cores. Além disso, uma das limitações da Lei de Amdahl é desconsiderar o overhead no gerenciamento multicore, tornando ainda mais inviável a utilização dos núcleos E para análise no teste em questão. Portanto, apenas os oito núcleos P, de alta performance foram considerados para o teste.

3. Tamanho de macrobloco fixo X Alto número de threads

Aqui, fixamos o tamanho dos macroblocos e comparamos o tempo de execução utilizando um alto número de threads. O objetivo foi identificar como um grande número de threads afeta o tempo de execução e identificar possíveis pontos onde o desempenho não melhora ou até piora devido ao *overhead* crítico.

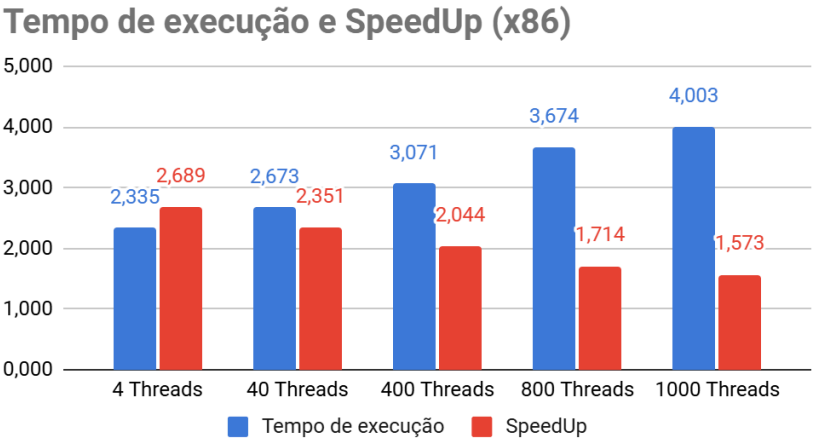
- **PC Sofia**

Intel Core i7 13700KF	**x64**	
Dimensões da matriz	10000x10000	
Dimensões do macrobloco	500x500	
	Tempo de execução	**SpeedUp**
4 Threads	1,755	3,007
400 Threads	2,498	2,107
4000 Threads	5,689	0,929
40000 Threads	150,843	0,034



Código compilado em 64 bits

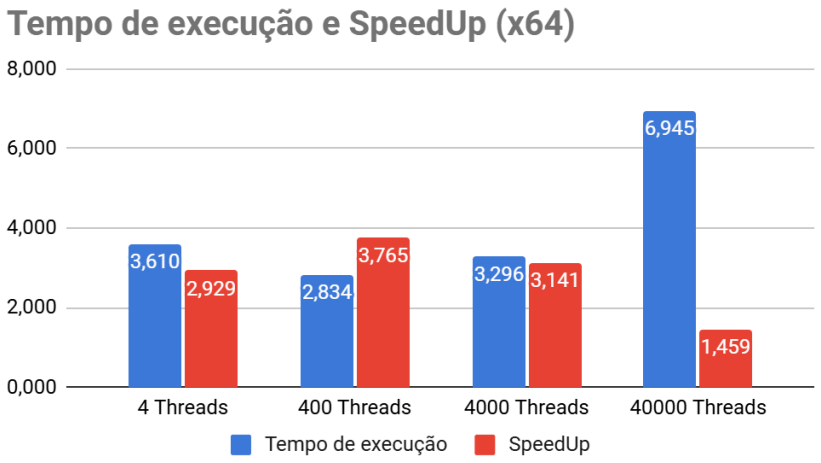
Intel Core i7 13700KF	**x86**	
Dimensões da matriz	10000x10000	
Dimensões do macrobloco	500x500	
	Tempo de execução	**SpeedUp**
4 Threads	2,335	2,689
40 Threads	2,673	2,351
400 Threads	3,071	2,044
800 Threads	3,674	1,714
1000 Threads	4,003	1,573



Código compilado em 32 bits

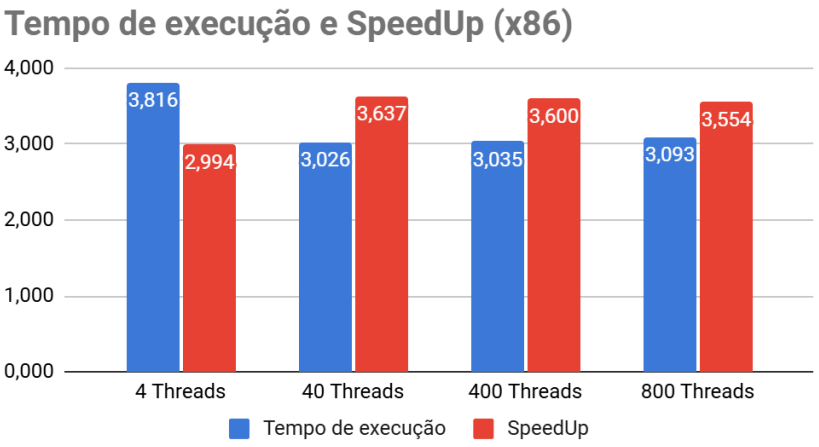
- **PC Alex**

AMD Ryzen 5 1500X	x64	
Dimensões da matriz	10000x10000	
Dimensões do macrobloco	500x500	
	Tempo de execução	**SpeedUp**
4 Threads	3,610	2,929
400 Threads	2,834	3,765
4000 Threads	3,296	3,141
40000 Threads	6,945	1,459



Código compilado em 64 bits

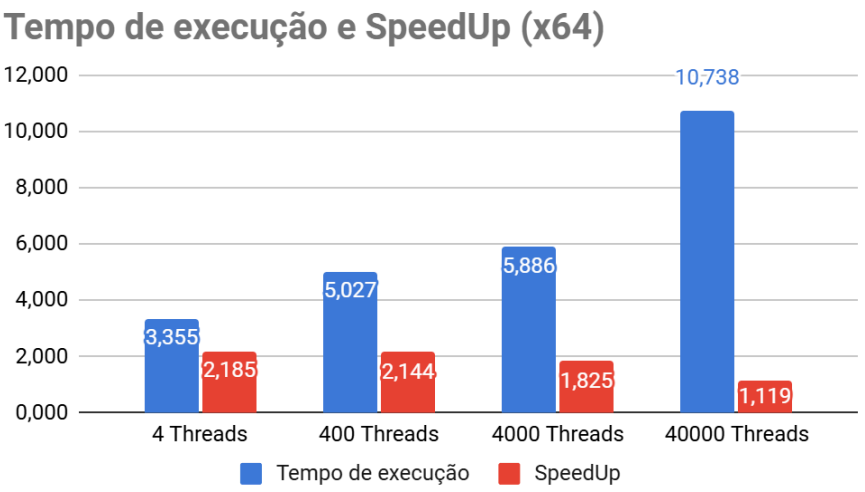
AMD Ryzen 5 1500X	x86	
Dimensões da matriz	10000x10000	
Dimensões do macrobloco	500x500	
	Tempo de execução	**SpeedUp**
4 Threads	3,816	2,994
40 Threads	3,026	3,637
400 Threads	3,035	3,600
800 Threads	3,093	3,554



Código compilado em 32 bits

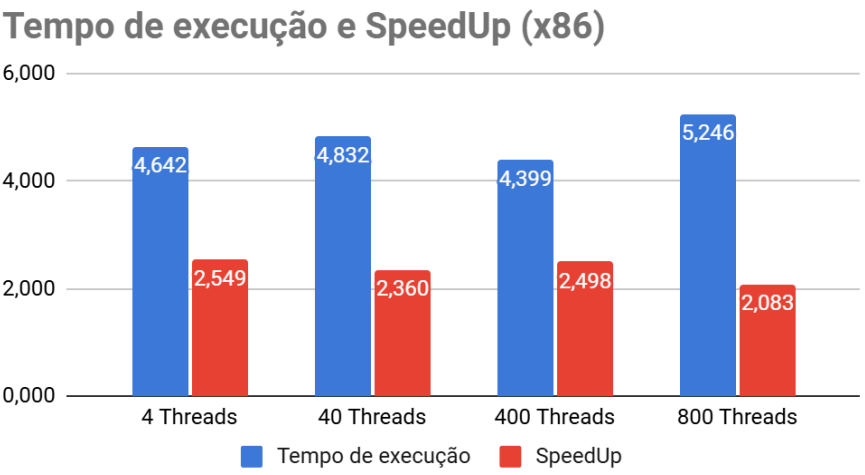
- **PC João Pedro**

Intel Core i7 8550U	x64	
Dimensões da matriz	10000x10000	
Dimensões do macrobloco	500x500	
	Tempo de execução	SpeedUp
4 Threads	3,355	2,185
400 Threads	5,027	2,144
4000 Threads	5,886	1,825
40000 Threads	10,738	1,119



Código compilado em 64 bits

Intel Core i7 8550U	x86	
Dimensões da matriz	10000x10000	
Dimensões do macrobloco	500x500	
	Tempo de execução	SpeedUp
4 Threads	4,642	2,549
40 Threads	4,832	2,360
400 Threads	4,399	2,498
800 Threads	5,246	2,083



Código compilado em 32 bits

Diferentemente dos resultados do teste anterior, **nesta análise**, ao aumentar consideravelmente o número de threads, não observamos um ganho significativo de desempenho no tempo de execução. Pelo contrário, em alguns casos, houve uma piora considerável no tempo de execução e no speedup em comparação com os casos com menos threads: vejamos os casos de 4 threads e 40000 threads em arquitetura x64 do Intel Core i7 13700KF: no primeiro caso, o tempo de execução do algoritmo se deu em 1,755s, enquanto no segundo caso o programa demorou 150,843s para "rodar". Isso nos permite concluir que aumentar o número de threads não necessariamente aumenta a eficiência da CPU. O aumento no número de threads também aumenta o overhead, impactando negativamente o desempenho a partir de um certo ponto.

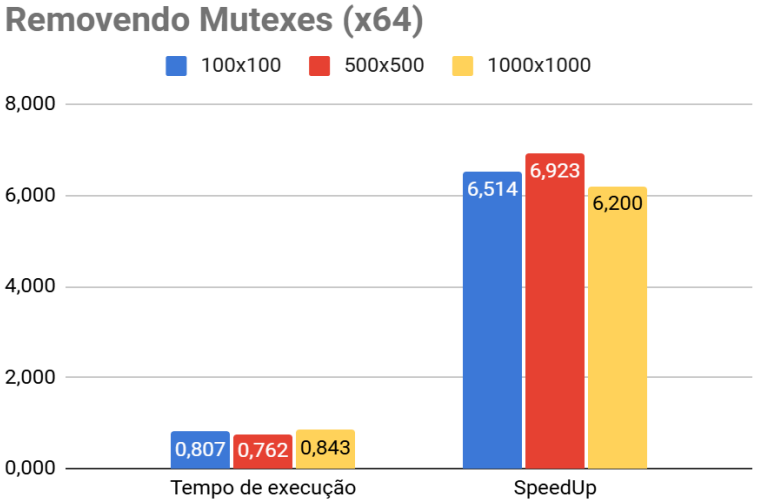
4. Removendo mutexes

Por fim, neste teste fixamos o número de threads de acordo com o número de núcleos lógicos de cada CPU e comparamos o tempo de execução para diferentes tamanhos de macroblocos (100x100, 500x500 e 1000x1000) sem utilizar controle de mutexes sobre as regiões críticas.

Antes de analisar os resultados obtidos nos testes, vale ressaltar do que se tratam as regiões críticas. *"Durante parte do tempo, um processo está ocupado realizando computações internas e outras coisas que não levam a condições de corrida. No entanto, às vezes um processo tem de acessar uma memória compartilhada ou arquivos, ou realizar outras tarefas críticas que podem levar a corridas. Essa parte do programa onde a memória compartilhada é acessada é chamada de região crítica ou seção crítica."* (TANENBAUM, 2016)

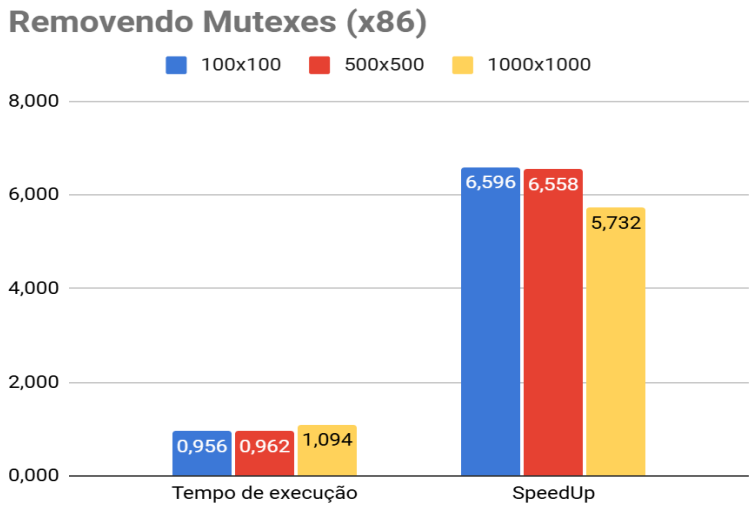
- **PC Sofia**

Intel Core i7 13700KF	**x64**		
Dimensões da matriz	10000x10000		
Núcleos Lógicos	8 threads		
Tamanho dos macroblocos	100x100	500x500	1000x1000
Tempo de execução	0,807	0,762	0,843
SpeedUp	6,514	6,923	6,200



Código compilado em 64 bits

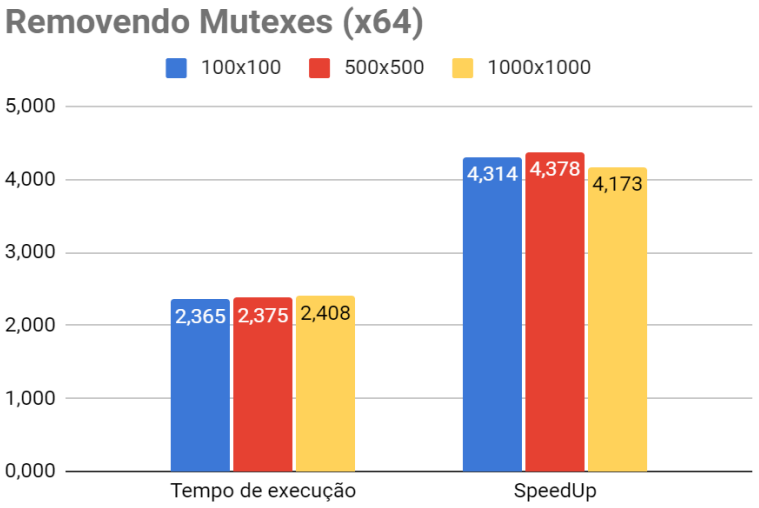
Intel Core i7 13700KF	**x86**		
Dimensões da matriz	10000x10000		
Núcleos Lógicos	8 threads		
Tamanho dos macroblocos	100x100	500x500	1000x1000
Tempo de execução	0,956	0,962	1,094
SpeedUp	6,596	6,558	5,732



Código compilado em 32 bits

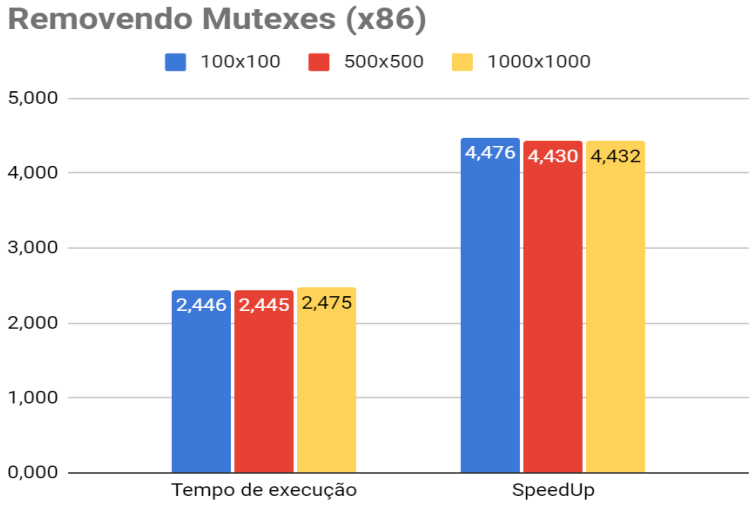
- **PC Alex**

AMD Ryzen 5 1500X	x64		
Dimensões da matriz	10000x10000		
Núcleos Lógicos	8 threads		
Tamanho dos macroblocos	100x100	500x500	1000x1000
Tempo de execução	2,365	2,375	2,408
SpeedUp	4,314	4,378	4,173



Código compilado em 64 bits

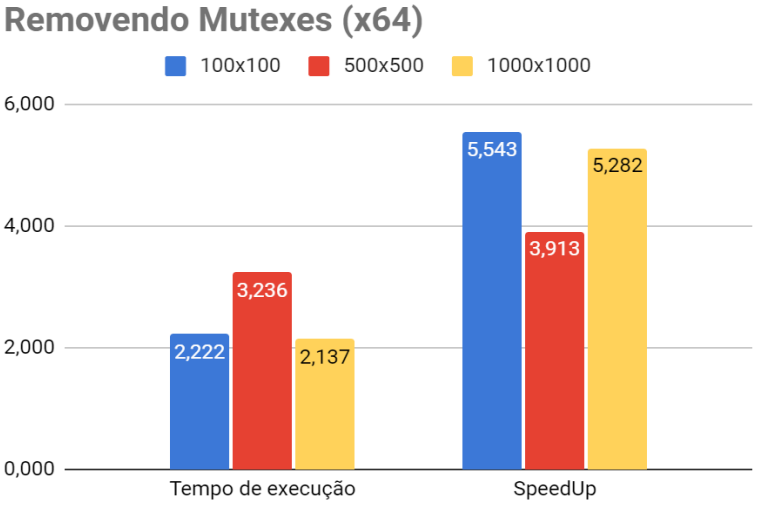
AMD Ryzen 5 1500X	x86		
Dimensões da matriz	10000x10000		
Núcleos Lógicos	8 threads		
Tamanho dos macroblocos	100x100	500x500	1000x1000
Tempo de execução	2,446	2,445	2,475
SpeedUp	4,476	4,430	4,432



Código compilado em 32 bits

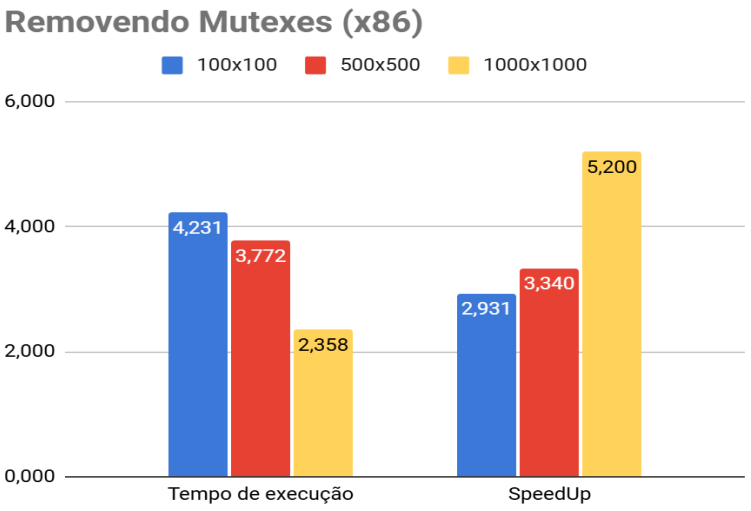
- **PC João Pedro**

Intel Core i7 8550U	x64		
Dimensões da matriz	10000x10000		
Núcleos Lógicos	8 threads		
Tamanho dos macroblocos	100x100	500x500	1000x1000
Tempo de execução	2,222	3,236	2,137
SpeedUp	5,543	3,913	5,282



Código compilado em 64 bits

Intel Core i7 8550U	x86		
Dimensões da matriz	10000x10000		
Núcleos Lógicos	8 threads		
Tamanho dos macroblocos	100x100	500x500	1000x1000
Tempo de execução	4,231	3,772	2,358
SpeedUp	2,931	3,340	5,200



Código compilado em 32 bits

Os resultados obtidos mostraram que, em todos os casos, o tempo de execução foi significativamente menor sem o uso de mutexes. No entanto, houve inconsistência na quantidade de números primos encontrados na matriz.

Os exemplos a seguir foram retirados dos testes feitos no PC da Sofia (Intel Core i7 13700KF) em condições iguais de número de threads (8 Threads) e tamanho de macroblocos (500x500) e 64 bits.

```
#       Executando Busca Serial     #

Busca Serial:
Quantidade de primos: 10883419
Tempo decorrido: 4.980000 segundos

#     Executando busca paralela       #

Busca Paralela:
Quantidade de threads: 8
Numero de blocos na matriz: 400
Quantidade de primos: 10883419
Tempo decorrido: 2.071000 segundos

Speedup: 2.404635
```

Busca paralela com mutexes

```
#       Executando Busca Serial     #

Busca Serial:
Quantidade de primos: 10883419
Tempo decorrido: 5.276000 segundos

#     Executando busca paralela       #

Busca Paralela:
Quantidade de threads: 8
Numero de blocos na matriz: 400
Quantidade de primos: 10486213
Tempo decorrido: 0.762000 segundos

Speedup: 6.923885
```

Busca paralela sem mutexes

Esta análise evidencia que as regiões críticas (RCs) causam um alto impacto no desempenho de um sistema. O uso de mutexes, embora necessário para garantir a correção e a consistência dos dados, introduz um overhead considerável devido ao bloqueio e desbloqueio frequente dos mutexes. Este overhead é eliminado quando os mutexes não são utilizados, resultando em um tempo de

execução menor. No entanto, a ausência de controle nas regiões críticas pode levar a resultados incorretos ou inconsistentes, como observado na contagem de números primos.

Portanto, este teste destaca a importância de um equilíbrio entre desempenho e consistência. Embora a eliminação de mutexes possa melhorar o tempo de execução, ela compromete a integridade dos dados, o que é inaceitável para muitas aplicações críticas.

Conclusão

O trabalho em questão permitiu um maior aprofundamento nos conceitos trabalhados em sala por meio da observação prática desses conteúdos, através dos diversos testes realizados ao decorrer do trabalho. Não somente assuntos relativos à disciplina de Sistemas Operacionais foram melhor estudados: conteúdos trabalhados em Arquitetura de Computadores também tiveram destaque durante a atividade, a exemplo dos conceitos de arquitetura heterogênea, Lei de Amdhal, arquiteturas em si (x64 e x86) e técnicas como Simultaneous multithreading (SMT) e conceitos de paralelismo. A retomada de tópicos apresentados na disciplina anterior, Arquitetura de Computadores, reforça a importância do real aprendizado do que é estudado em sala.

Além disso, o entendimento dos novos tópicos abordados, agora em Sistemas Operacionais, também se mostraram importantes na viabilização da execução do trabalho. Considerando os capítulos 3 (Processo), 4 (Threads), 6 (Sincronismo de Processos e Threads) e 7 (Deadlock) introduzem conceitos importantes como o de mutexes, elemento ímpar na elaboração do algoritmo proposto. Apesar de sentirmos dificuldades em compreender plenamente a utilidade de semáforos binários em sala, durante o trabalho pudemos verificar a importância de seu uso efetivo.

Ademais, o real entendimento dos conteúdos das disciplinas anteriormente citadas, auxiliaram a entender e solucionar com menos estresse *bugs* e desafios na compreensão teórica, sobretudo nas etapas de análises de resultados de desempenho. Por exemplo, no teste 3, onde pudemos compreender o real objetivo da análise.

Referências

- Slides e vídeo-aulas disponibilizadas, pelo Prof. Flávio Giraldeli.
- TANENBAUM, A. S. Sistemas operacionais modernos (4a. ed.). São Paulo: Pearson Educación, 2016.