Laboratório 3

Carlos Eduardo de Schuller Banjar - 122056361

Especificações da máquina usada nos experimentos

Chip: Apple M2
Número Total de Núcleos: 8 (4 desempenho e 4 eficiência)
Memória: 16 GB

Experimentos

Todos os números gerados estão disponíveis e podem ser acessados em <https://docs.google.com/spreadsheets/d/1vYV25JmFz-hx-7qliR-K3-BdpYScFLNNyUqnvD-mSQM/edit?usp=sharing>.

Aceleração

> A aceleração é uma medida de quanto mais rápido um algoritmo concorrente executa em comparação com a sua versão sequencial.

Matrizes 500x500:

- **1 thread**: A aceleração é muito baixa (0,0008326), o que indica que a versão sequencial é mais rápida do que a concorrente. Isso pode ser devido ao overhead de criação e gerenciamento das threads para um problema pequeno.
- **2 threads**: A aceleração aumenta significativamente para 0,2521984, mas ainda assim, o ganho é pequeno.
- **4 threads**: A aceleração diminui para 0,1399722, o que indica que indica que overhead da concorrência está prejudicando a performance.
- **8 threads**: A aceleração continua a diminuir, chegando a 0,0970104, reforçando a ideia de que a paralelização não está trazendo benefícios

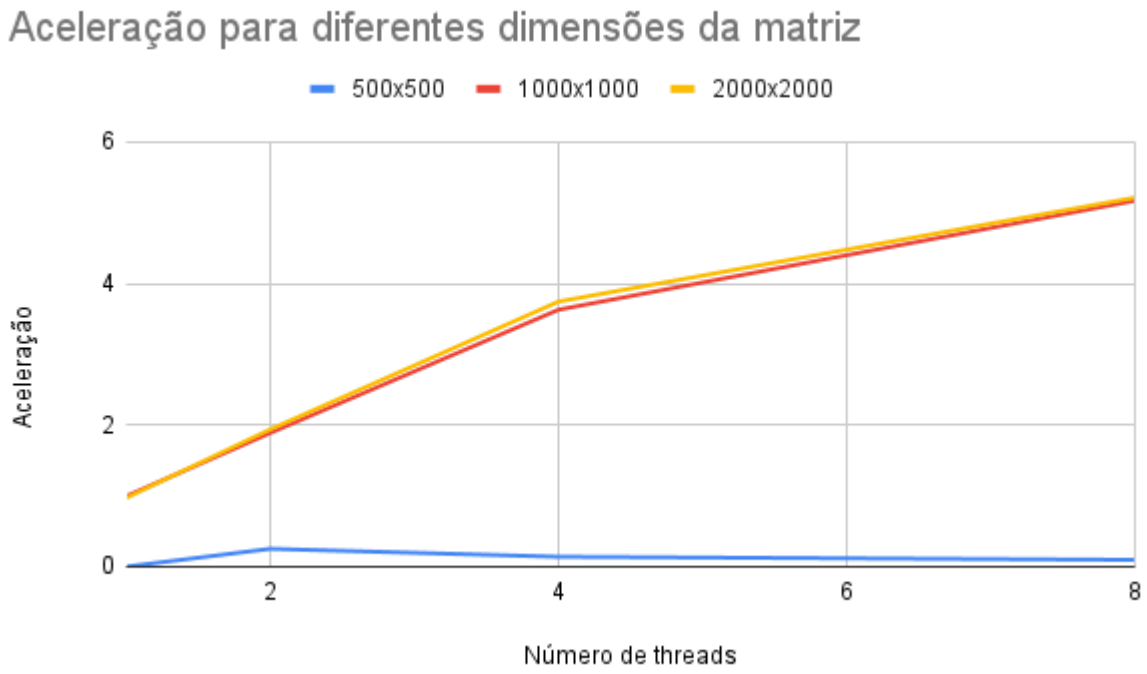
devido ao tamanho da matriz ser pequeno.

Matrizes 1000x1000

- **1 thread**: A aceleração está próxima de 1 (0,9950639714), indicando que a versão concorrente começa a igualar o desempenho da versão sequencial.
- **2 threads**: A aceleração dobra (1,888442302), mostrando que a paralelização começa a ser vantajosa à medida que o tamanho da matriz aumenta.
- **4 threads**: A aceleração aumenta significativamente para 3,628465962, demonstrando uma boa utilização das threads.
- **8 threads**: A aceleração continua a crescer (5,168155414), mas a taxa de crescimento está desacelerando.

Matrizes 2000x2000

- **1 thread**: A aceleração ainda está próxima de 1 (0,9713317414), similar ao caso anterior.
- **2 threads**: A aceleração quase dobra (1,939240736), mostrando um bom uso das threads.
- **4 threads**: A aceleração sobe para 3,743825908.
- **8 threads**: A aceleração aumenta para 5,211623336, mas o ganho adicional é menor em comparação ao salto anterior.



Eficiência

> A eficiência é uma medida da eficácia com que as threads são utilizadas em um algoritmo paralelo.

Matrizes 500x500

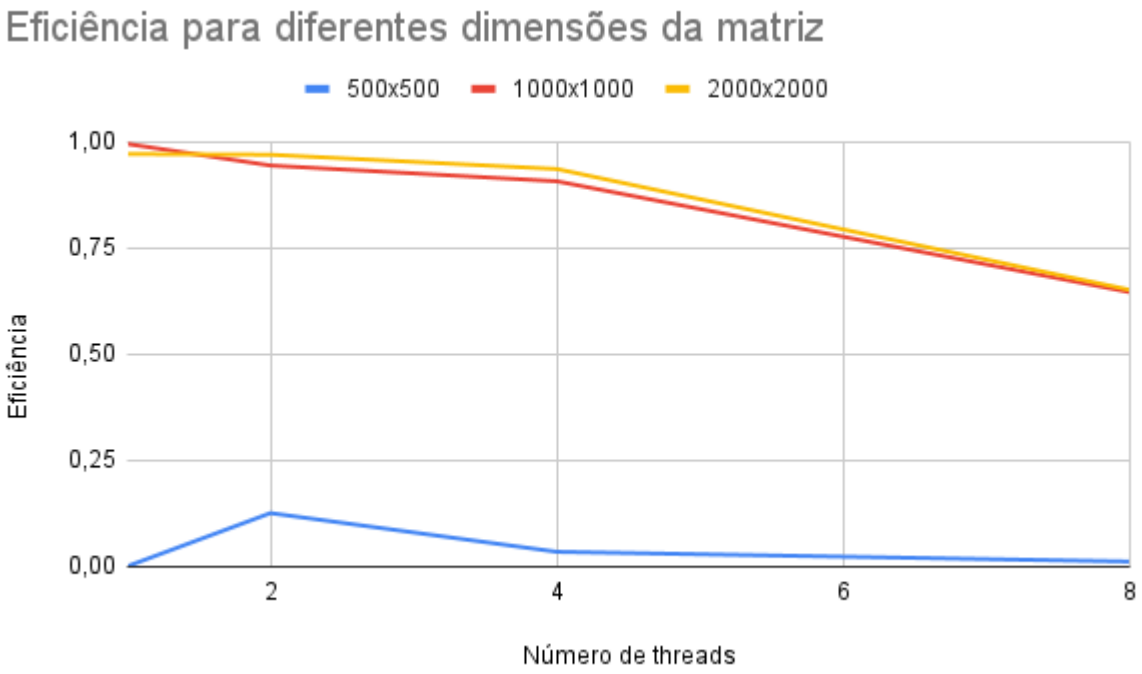
- **1 thread**: A eficiência é extremamente baixa (0,0008326), refletindo o fato de que a versão concorrente é menos eficiente do que a versão sequencial.
- **2 threads**: A eficiência melhora (0,1260992), mas ainda está muito longe de 1, o que indica que a paralelização não está sendo eficaz para este tamanho de matriz.
- **4 threads**: A eficiência cai drasticamente para 0,03499305, sugerindo que o aumento do número de threads está gerando mais overhead do que ganhos de desempenho.
- **8 threads**: A eficiência continua a cair à medida que mais threads são adicionadas (agora para 0,0121263), mostrando que o paralelismo não é vantajoso para matrizes desse tamanho.

Matrizes 1000x1000

- **1 thread**: A eficiência é quase perfeita (próxima de 1), indicando que a execução concorrente é praticamente tão eficiente quanto a sequencial.
- **2 threads**: A eficiência diminui ligeiramente para 0,9442211511, mas ainda é alta, o que mostra que a paralelização está valendo a pena.
- **4 threads**: A eficiência continua alta, acima de 0,9, mostrando uma boa utilização dos recursos de paralelismo.
- **8 threads**: A eficiência cai mais significativamente para 0,6460194268, indicando que o aumento do número de threads começa a encontrar limites.

Matrizes 2000x2000

- **1 thread**: Semelhante ao caso de 1000×1000, a eficiência inicial é alta (0,9713317414).
- **2 threads**: A eficiência permanece muito alta, quase 1, mostrando que o uso de 2 threads é quase ideal para essa configuração.
- **4 threads**: A eficiência continua elevada embora tenha decaído (0,9359564769) indicando boa utilização das threads.
- **8 threads**: A eficiência cai mais uma vez, agora de forma acentuada (0,651452917). Isso sugere overhead crescente. O paralelismo ainda oferece benefícios, embora a eficiência não seja ideal.



Com os resultados, vemos que o paralelismo não oferece benefícios em problemas pequenos (500×500) e, de fato, prejudica a performance devido ao overhead associado ao gerenciamento de threads. O paralelismo começa a mostrar ganhos significativos em problemas maiores, ainda que o aumento de threads não possa se dar de forma indiscriminada.