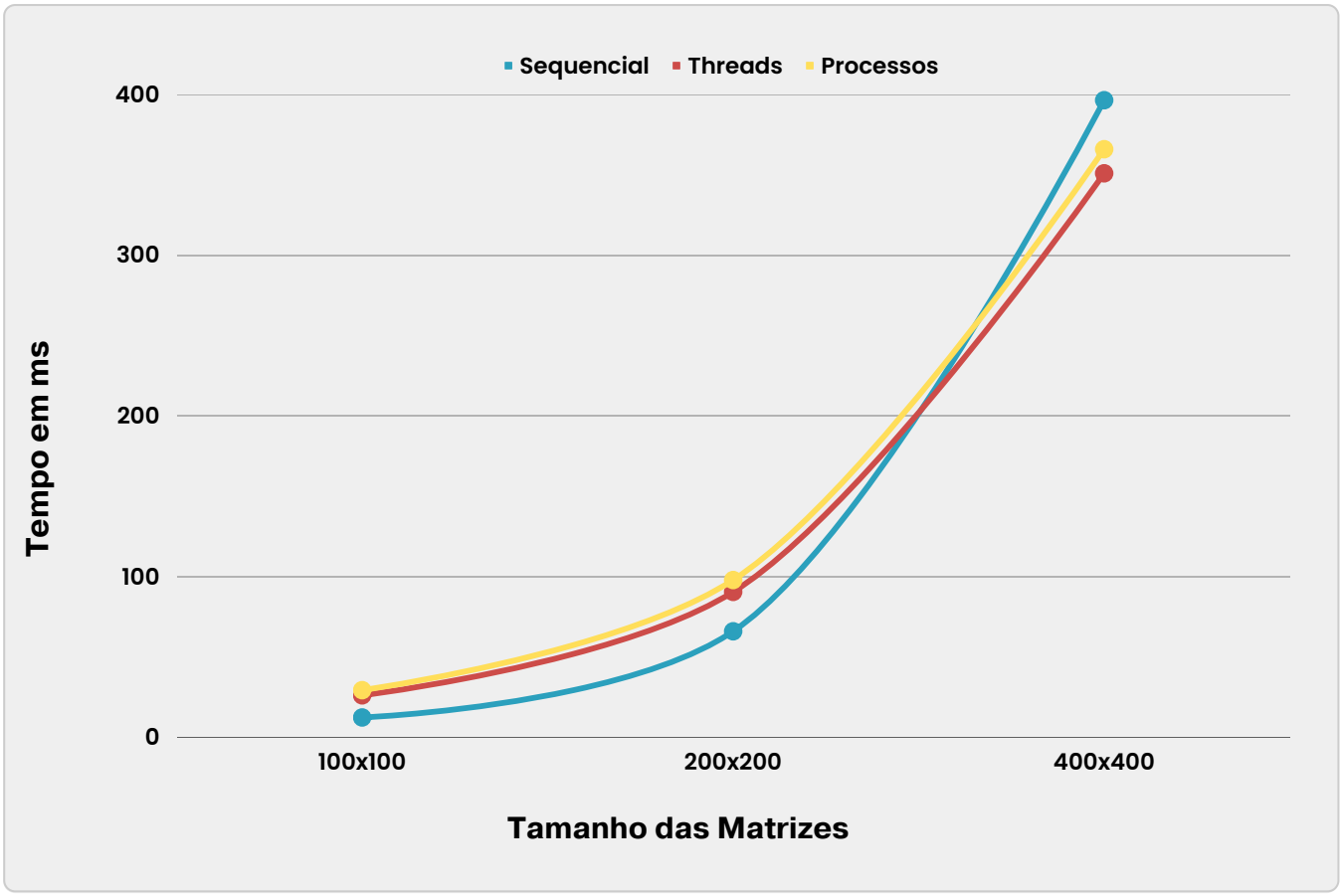
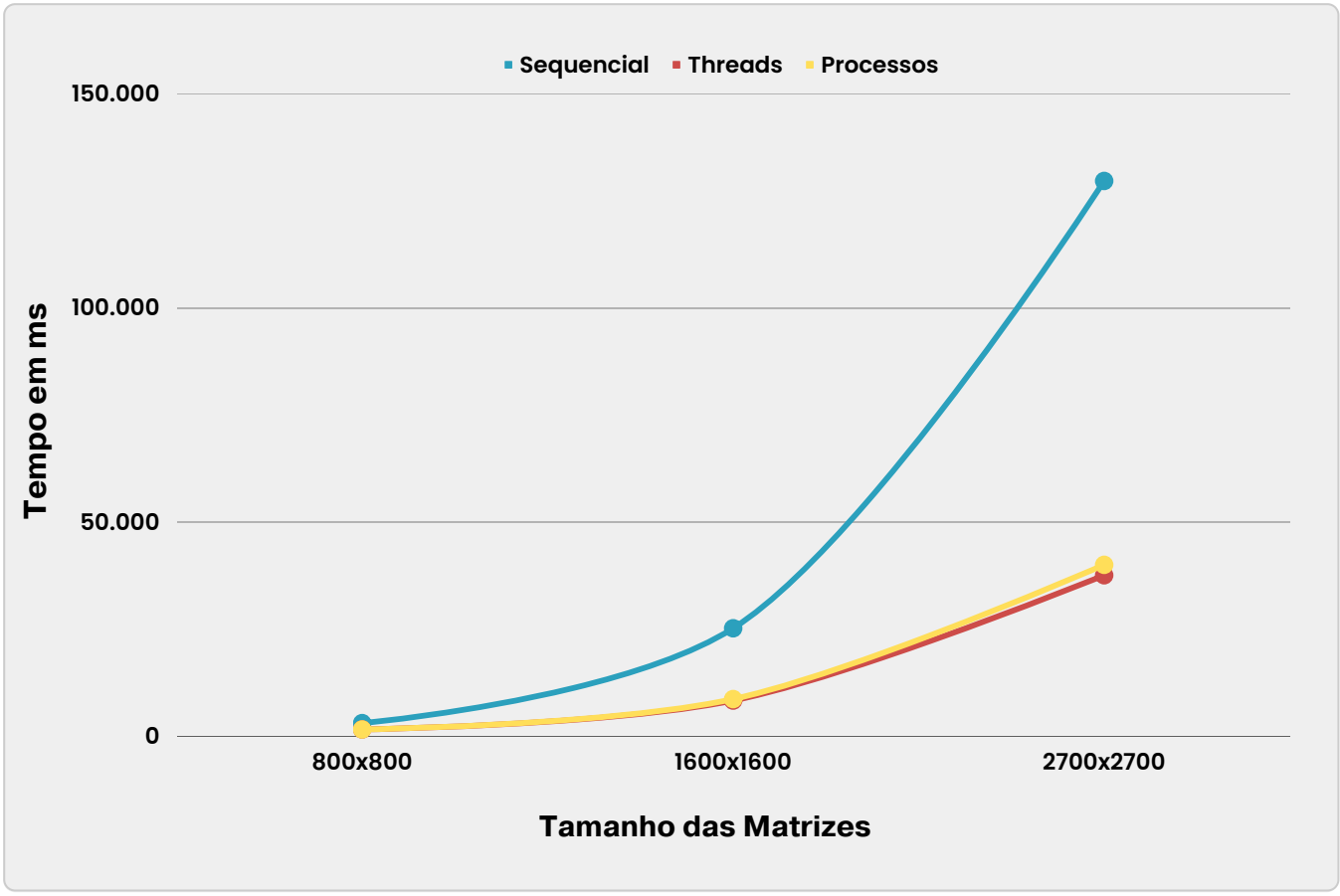
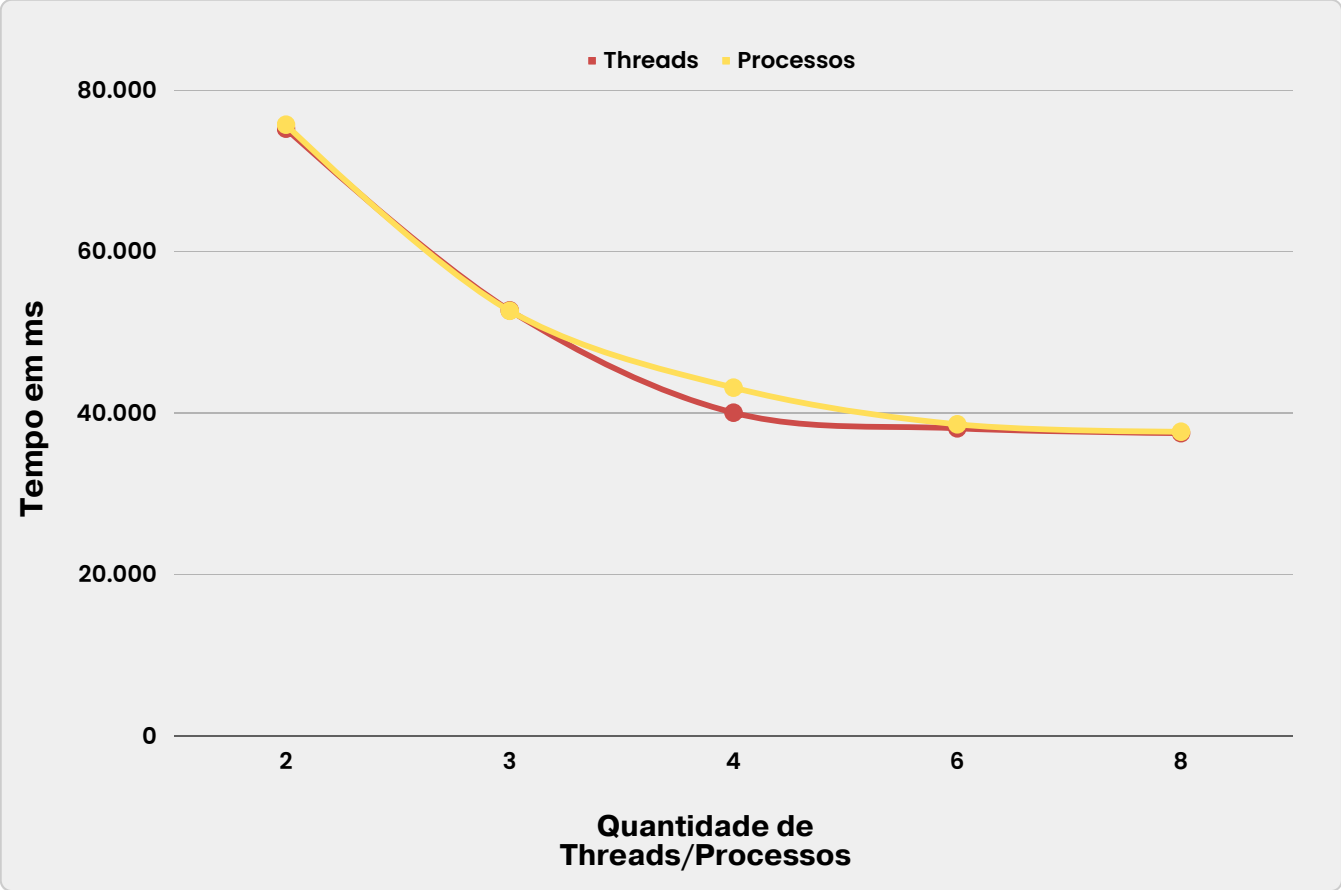
E1


Sequencial
Threads
Processos
400
300
200
100
0
Tempo em ms
100x100
200x200
400x400
Tamanho das Matrizes


Sequencial
Threads
Processos
150.000
100.000
50.000
0
Tempo em ms
800x800
1600x1600
2700x2700
Tamanho das Matrizes

E2



Continua na próxima página.

Dividi o gráfico do E1 em dois para facilitar a visualização, pois o tempo escala bastante em função do tamanho das matrizes, assim o gráfico primeiro só varia em ms de 0 a 400, e não de 0 a 120.000 (que são 2 minutos como pedido na atividade).

Vou deixar aqui, e na descrição dessa atividade no Sigaa, o link para o vídeo demonstrando e explicando a implementação: https://www.youtube.com/watch?v=twqN85X0oI8

a) Quais o motivo dos resultados obtidos no experimento E1? O Quê pode ter causado o comportamento observado?

No primeiro experimento, o observado foi que quanto maior as matrizes mais tempo de cálculo. Nota-se também que os algoritmos utilizando processos / threads foram mais eficientes, o de threads tendo um desempenho ligeiramente superior em resultado, acredito que pelo fato de os processos requererem alocação de recursos separados, como espaço de endereçamento e tabelas de controle de processos. Essa maior eficiência deve-se ao uso dos vários núcleos disponíveis no processador, e não apenas a um como no código sequencial.

Portanto a quebra do código em partes para que elas façam uso dos vários núcleos do processador se mostra uma forma eficiente de operar algoritmos.

b) Qual o motivo dos resultados obtidos no experimento E2? O quê pode ter causado o comportamento observado?

No segundo gráfico, do experimento E2, observamos que quanto mais threads ou processos há também um ganho em velocidade. É importante ressaltar que fiz testes com mais do que 8 threads ou processos, porém me retornaram resultados superiores ou iguais, em tempo de processamento, aos utilizando 8, acredito que tal fato se deva pelo meu processador (M1 8 núcleos arquitetura ARM da Apple) possuir exatamente 8 núcleos, mas ainda assim de 4 para 8 processos / threads o ganho foi pouco. Vasculhando o site da Apple me deparei com essa informação acerca do meu processador:

"**O melhor desempenho por watt de CPU do mundo**

O M1 inclui uma CPU de 8 núcleos que consiste em quatro núcleos de alto desempenho e quatro núcleos de alta eficiência. Cada um dos núcleos de alto desempenho proporciona um desempenho líder de mercado para tarefas de um thread, mantendo a maior eficiência de funcionamento possível. Esses são os núcleos de CPU em silício de baixo consumo mais rápidos do mundo, permitindo que os fotógrafos editem fotografias de alta resolução rapidamente e os desenvolvedores criem apps em uma velocidade quase 3 vezes mais rápida do que anteriormente. E todos os quatro podem ser usados juntos, para um grande aumento de desempenho multithread.

Os quatro núcleos de alta eficiência proporcionam um desempenho impressionante usando apenas um décimo da potência. Por si só, esses quatro núcleos proporcionam um desempenho semelhante ao do MacBook Air de 2 núcleos da geração atual, utilizando muito menos potência. Eles são a maneira mais eficiente de executar tarefas rápidas do dia a dia, como verificar o e-mail ou navegar na internet, além de poupar autonomia da bateria como nunca até agora. E os oito núcleos podem trabalhar juntos para proporcionar uma potência de computação incrível para as tarefas mais exigentes e proporcionar o melhor desempenho de CPU por watt do mundo."

Assim, o desempenho maior é observado exatamente até o uso de 4 núcleos (relativo aos núcleos de desempenho), já no uso de 8 threads/processos o ganho não é tão expressivo pois há o uso dos 4 núcleos de eficiência, que consomem menos, porém não possuem tanto desempenho quantos os 4 primeiros supracitados.

c) Qual é o valor de P ideal para a multiplicação das matrizes M1 e M2? Justifique sua resposta através dos experimentos realizados.

Como já citado na questão anterior, o P ideal seria de 8(ou de Linhas1xColunas2/8, usando P como o valor trazido na questão). A justificativa reside exatamente no fato de meu processador possuir 8 núcleos de processamento, apesar de nem todos possuirem a mesma capacidade, ainda se torna mais rápido usar os 8 núcleos. Em matrizes maiores (testei com matrizes de 4500x4500, apesar do meu gráfico não exemplificá-las) o ganho em se usar 8 ao invés de 4 foi mais substancial, cerca de 10 a 15 segundos, mas o tempo total sendo 4 minutos, o quê mostra que percentualmente (4%~7%) o ganho ainda é baixo.

**Ps: optei por usar, tanto aqui quanto no algoritmo, como P sendo o número de threads ou processos, facilita nosso entendimento, afinal a divisão de partes da matrizes dada por n1*m2/N vai me gerar o uso precisamente de N threads ou N processos.