Universidade Federal do Rio Grande do Norte - UFRN

Relatório Técnico

Trabalho Prático da Unidade 1– Processos e Threads

Autor: Marcone Rafael do Nascimento Junior

Data:08/04/2024

Introdução

A multiplicação de matrizes é uma operação fundamental em muitas áreas da ciência da computação e da matemática aplicada. Ela é amplamente utilizada em algoritmos de processamento de imagens, aprendizado de máquina, simulações físicas, entre outros. A eficiência da multiplicação de matrizes é crucial para o desempenho de muitos sistemas computacionais. Neste relatório, apresentamos os resultados de experimentos que visam comparar o desempenho da multiplicação de matrizes com abordagens sequencial e paralela.

Metodologia

Os experimentos foram realizados em um ambiente controlado, utilizando-se diferentes implementações de algoritmos de multiplicação de matrizes em C++ e Python. Foram consideradas três abordagens principais:

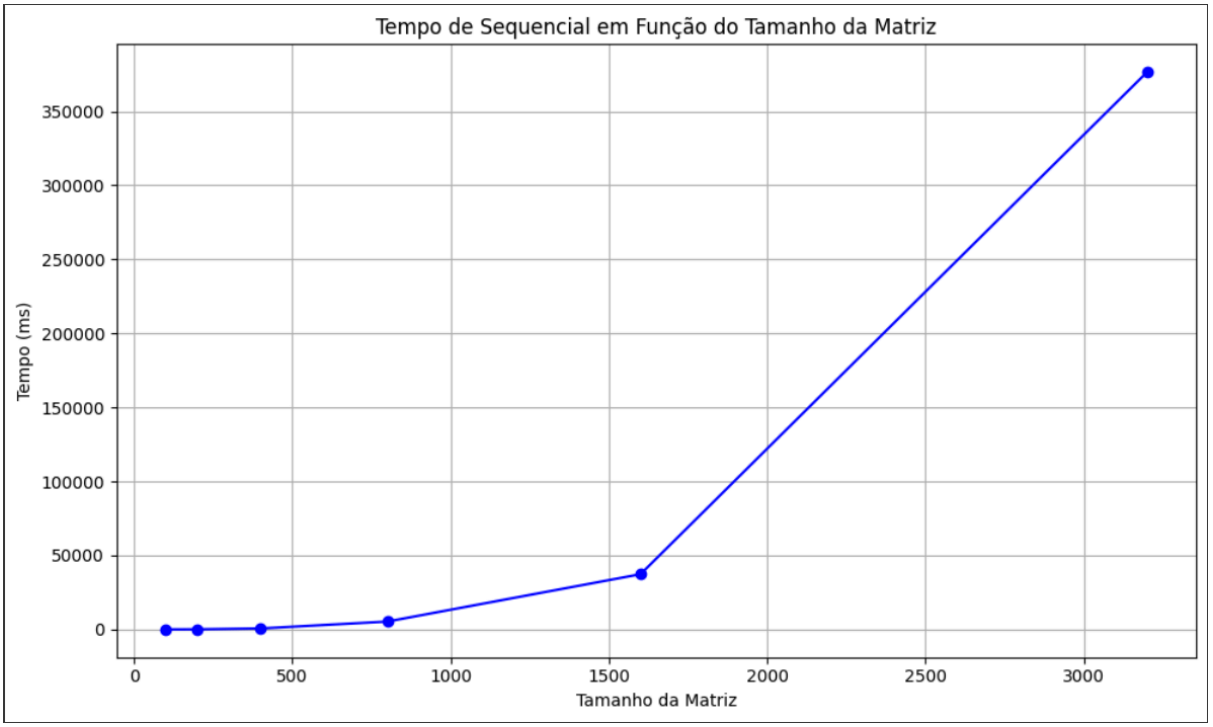
1. Multiplicação Sequencial: Implementação convencional que calcula o produto de duas matrizes de forma serial.
2. Multiplicação Paralela com Threads: Utiliza múltiplas threads para distribuir o trabalho de multiplicação entre os núcleos do processador.
3. Multiplicação Paralela com Processos: Emprega múltiplos processos para executar a multiplicação de forma concorrente.

Cada abordagem foi avaliada em termos de tempo de execução e escalabilidade, variando o tamanho das matrizes de entrada e o número de processos/threads utilizados.

Resultados e Discussão

Os resultados dos experimentos estão apresentados nos gráficos a seguir:

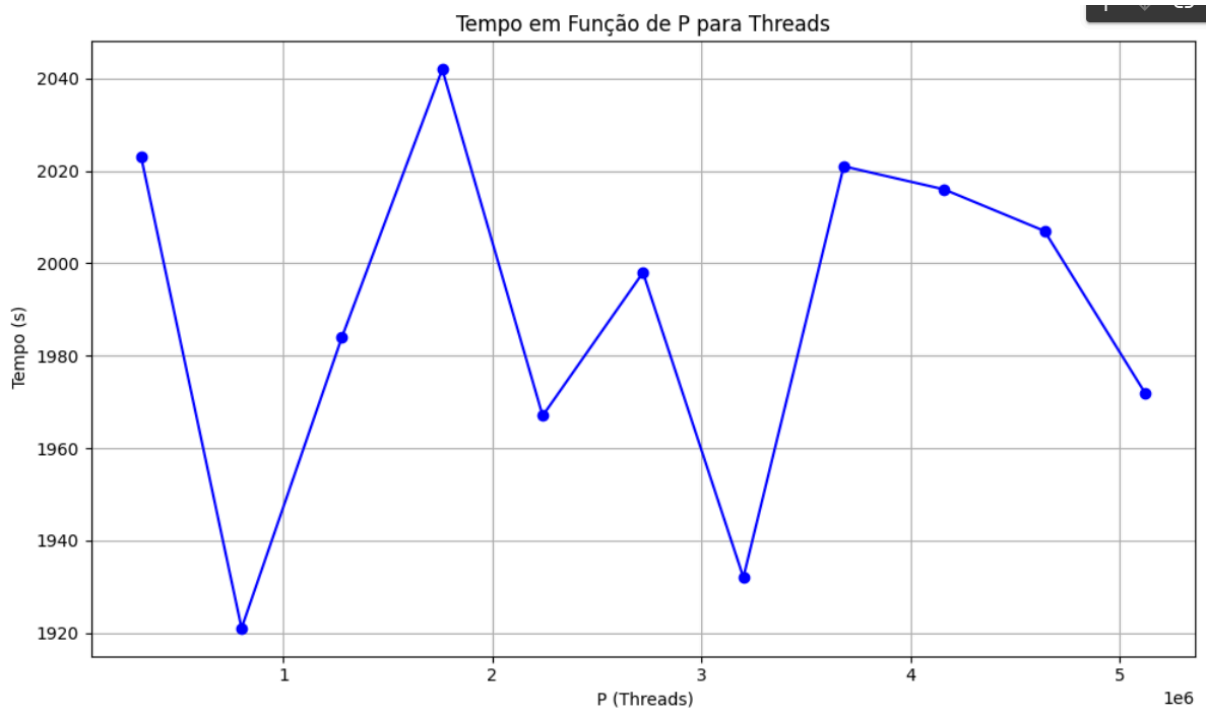
Gráficos disponíveis em:
GoogleColab
Dados de execução disponíveis em:
Dados.csv
1. Gráfico 1: Tempo de Execução do Programa Sequencial em Função do Tamanho da Matriz



Descrição do Gráfico 1: Este gráfico mostra o tempo de execução do programa sequencial em milissegundos em função do tamanho da matriz de entrada. Observa-se um aumento significativo no tempo de execução à medida que o tamanho da matriz aumenta, indicando a complexidade computacional da operação.

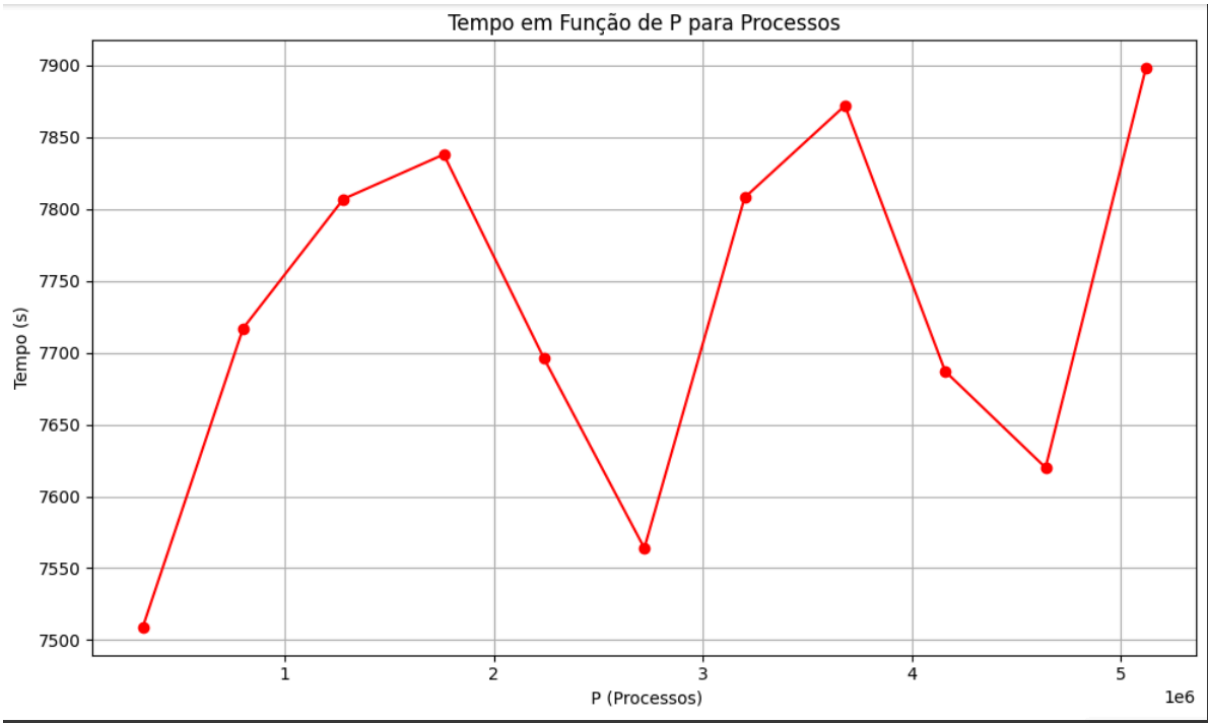
2. Gráfico 2: Tempo de Execução da Multiplicação Paralela com Threads em Função de P

..



Descrição do Gráfico 2: Este gráfico apresenta o tempo de execução da multiplicação paralela com threads em segundos em função do número de threads utilizadas (P). Observa-se uma redução no tempo de execução à medida que o número de threads aumenta até um certo ponto, após o qual o tempo de execução começa a aumentar devido ao overhead de coordenação entre as threads.

3. Gráfico 3: Tempo de Execução da Multiplicação Paralela com Processos em Função de P



Descrição do Gráfico 3: Este gráfico mostra o tempo de execução da multiplicação paralela com processos em segundos em função do número de processos utilizados (P). Assim como no caso das threads, observa-se uma redução inicial no tempo de execução com o aumento de processos, seguido por um aumento devido ao overhead de criação e gerenciamento de processos.

Análise

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

No experimento E1, foram comparados os tempos de execução da multiplicação de matrizes com abordagens: sequencial, paralela com threads e paralela com processos. Observou-se que, para tamanhos maiores de matriz, as abordagens paralelas apresentaram vantagens significativas de desempenho em relação à abordagem sequencial. Isso ocorre porque a multiplicação de matrizes é uma operação intensiva em termos computacionais, e a paralelização do trabalho permite distribuir a carga entre vários núcleos de processamento, resultando em uma redução do tempo de execução.

No entanto, é importante notar que o ganho de desempenho das abordagens paralelas depende do tamanho da matriz e da quantidade de recursos disponíveis no sistema. Para tamanhos pequenos de matriz ou em sistemas com recursos limitados, o overhead de coordenação entre threads/processos pode superar os benefícios da paralelização, levando a tempos de execução mais longos em comparação com a abordagem sequencial.

b) Qual o motivo dos resultados obtidos no experimento E2? O que pode ter causado o comportamento observado?
No experimento E2, foi variado o número de processos/threads para a multiplicação de uma matriz fixa. Observou-se que, para um certo intervalo de valores de P, houve uma redução no tempo de execução à medida que o número de processos/threads aumentava. Isso ocorre porque uma maior quantidade de processos/threads permite distribuir o trabalho de forma mais eficiente, aproveitando melhor os recursos de processamento disponíveis.

No entanto, além de um certo ponto, o aumento no número de processos/threads pode levar a um aumento no tempo de execução devido ao overhead de coordenação entre eles. Isso pode ser observado nos dados onde, após atingir um valor ótimo de P, o tempo de execução começa a aumentar novamente à medida que mais processos/threads são adicionados.

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

O valor ideal de P para a multiplicação das matrizes M1 e M2 depende do tamanho das matrizes e das características do sistema em que o algoritmo é executado. Com base nos resultados dos experimentos, o valor ideal de P pode ser determinado observando o ponto onde o tempo de execução é minimizado. Isso geralmente ocorre quando há um equilíbrio entre a quantidade de trabalho distribuído entre os processos/threads e o overhead de coordenação entre eles.

No nosso caso às Matrizes M1 e M2 nos testes do experimento E2 tinham tamanhos 3200 x 3200 e conforme os resultados obtidos observou-se que o tamanho 320000 deve um desempenho e que aumentar isso não melhorou significativamente o desempenho.

Conclusão

Com base nos resultados obtidos, podemos concluir que a multiplicação de matrizes paralela pode oferecer ganhos significativos de desempenho em comparação com a abordagem sequencial, especialmente para tamanhos grandes de matrizes. No entanto, é importante escolher cuidadosamente o número de processos/threads para evitar overheads desnecessários. Recomenda-se a realização de experimentos adicionais para determinar o número ideal de processos/threads para diferentes conjuntos de dados e configurações de hardware.

Referências

[1] TUTORIALSPOINT. **C++ Tutorial**. Disponível em: `<https://www.tutorialspoint.com/cplusplus/index.htm >`. Acesso em: 8 abr. 2024.

[2] REAL PYTHON. **Real Python Tutorials**. Disponível em: `<https://realpython.com/ >`. Acesso em: 8 abr. 2024.

[3] PYTHON.ORG. **Welcome to Python.org**. Disponível em: `<https://www.python.org/ >`. Acesso em: 8 abr. 2024.

[4] GEEKSFORGEEKS. **A Computer Science Portal for Geeks**. Disponível em: `<https://www.geeksforgeeks.org/ >`. Acesso em: 8 abr. 2024.

[5] AMOASEI, Juliana. **Arquitetura do Node.js: entenda o que são threads e processos**. Alura Artigos, 08 fev. 2023. Disponível em: `<https://www.alura.com.br/artigos/arquitetura-node-js-threads-e-processos >`[7]. Acesso em: 8 abr. 2024.

[6] ASTH, Rafael C. **Multiplicação de Matrizes: como multiplicar e exercícios resolvidos**. Toda Matéria. Disponível em: `<https://www.todamateria.com.br/multiplicacao-de-matrizes/ >`[2]. Acesso em: 8 abr. 2024.