Relatório do EP1 de MAC0219

Renan Ryu Kajihara, NUSP: 14605762

18 de outubro de 2024

Resumo

O presente relatório descreve as atividades realizadas no primeiro Exercício-Programa da disciplina "Programação Concorrente e Paralela (MAC0219)", que consistiu na paralelização de um programa, que calcula o Conjunto de Mandelbrot, por dois métodos diferentes: por meio da biblioteca Pthreads e pelas diretivas de compilador fornecidas pelo OpenMP, verificando o tempo de execução da versão sequencial e das versões paralelas do código fornecido e observando o impacto de diferentes números de Threads e de diferentes tamanhos de entrada no tempo de execução.

Conteúdo

1 Introdução

Com o intuito de observar a diferença de tempo de execução entre versões sequenciais e versões paralelizadas de um mesmo programa, realizou-se a paralelização de um algoritmo que calculava o Conjunto de Maldelbrot. Além disso, efetuo-se uma série de experimentos para diferentes números de Threads, diferentes tamanhos de entrada e com a existência ou não de operações de entrada e saída e alocação de memória, observando o impacto dessas variáveis no tempo de execução do programa.

2 Objetivos

O exercício teve como objetivo a implementação de técnicas de paralelização de códigos na linguagem de programação C, bem como a comparação entre os dados coletados por meio do código sequencial e por meio dos códigos paralelizados.

3 Cronograma

O desenvolvimento deste Exercício Programa envolveu algumas etapas-chave, elencadas abaixo:

- (01/10/2024-03/10/2024) - Paralelização do código sequencial, utilizando a biblioteca Pthreads e as diretivas de compilação fornecidas pela interface OpenMP.
- (04/10/2024-10/10/2024) - Coleta dos dados referentes ao tempo de execução dos programas, com diferentes parâmetros.
- (10/10/2024-15/10/2024) - Análise de dados, implementação de gráficos que ajudassem a visualização das análises.
- (10/10/2024-17/10/2024) - Redação do relatório do Exercício Programa.

4 Métodos

Primeiramente, para paralelizar o código fonte original, que continha a versão sequencial do programa do cálculo do Conjunto de Mandelbrot, foi utilizado dois métodos, a biblioteca Pthreads e as diretivas de compilação do OpenMP. Em ambos os métodos, foi paralelizado o laço mais externo que calculava o Conjunto de Mandelbrot.

A coleta de dados foi feita a partir da execução das diferentes versões do programa utilizando o profiling com a ferramenta perf. Nesse sentido, cada versão do programa, com diferentes número de Threads, tamanhos de entrada, com ou sem operações de entrada e saída alocação de memória e diferentes regiões do Conjunto de Mandelbrot foram executadas 10 vezes, sendo observado o tempo de execução médio de cada versão e o intervalo de confiança. Tais dados foram registrados em um planilha, que seria útil para o tratamento desses dados.

Para analisar os dados de maneira eficiente, gerando gráficos que ajudariam na compreensão dos experimentos, utilizou-se as bibliotecas Numpy, Pandas e Matplotlib da linguagem de programação Python, que utilizariam os dados contidos na planilha para criar diferentes formas de visualização dos dados obtidos.

5 Análise dos resultados experimentais

A planilha contendo os dados obtidos pode ser acessada a partir desta referência.

5.1 Resultado dos experimentos para diferente número de Threads

A partir dos gráficos que comparam o tempo de execução no eixo y em função do número de Threads, observa-se que para entradas pequenas, o número de Threads não impacta profundamente no tempo de execução, podendo até ocorrer o caso em que as versões com menos Threads sejam realizadas mais rapidamente comparada às versões com mais Threads. Isso ocorre porque, nesses casos em que a entrada é pequena, a criação e gerenciamento das Threads demora mais do que a própria execução do programa, resultando em um tempo médio de execução maior às versões que utilizam mais Threads.

Além disso, evidencia-se que, para as entradas maiores, houve uma melhora signifativa no desempenho ao aumentar o número de Threads até 8. Entretanto, é notório que as versões com 16 e 32 Threads são apenas um pouco mais rápidas do que a versão com 8 Threads. Tal observação sucede do fato de que os testes foram realizados em um computador que possui 8 núcleos de processamento. Dessa forma, ao executar o programa com mais de 8 Threads, ocorrerá uma alternação da execução das threads entre os núcleos de processameto, ou seja, as Threads, na realidade, não serão executadas simultaneamente.

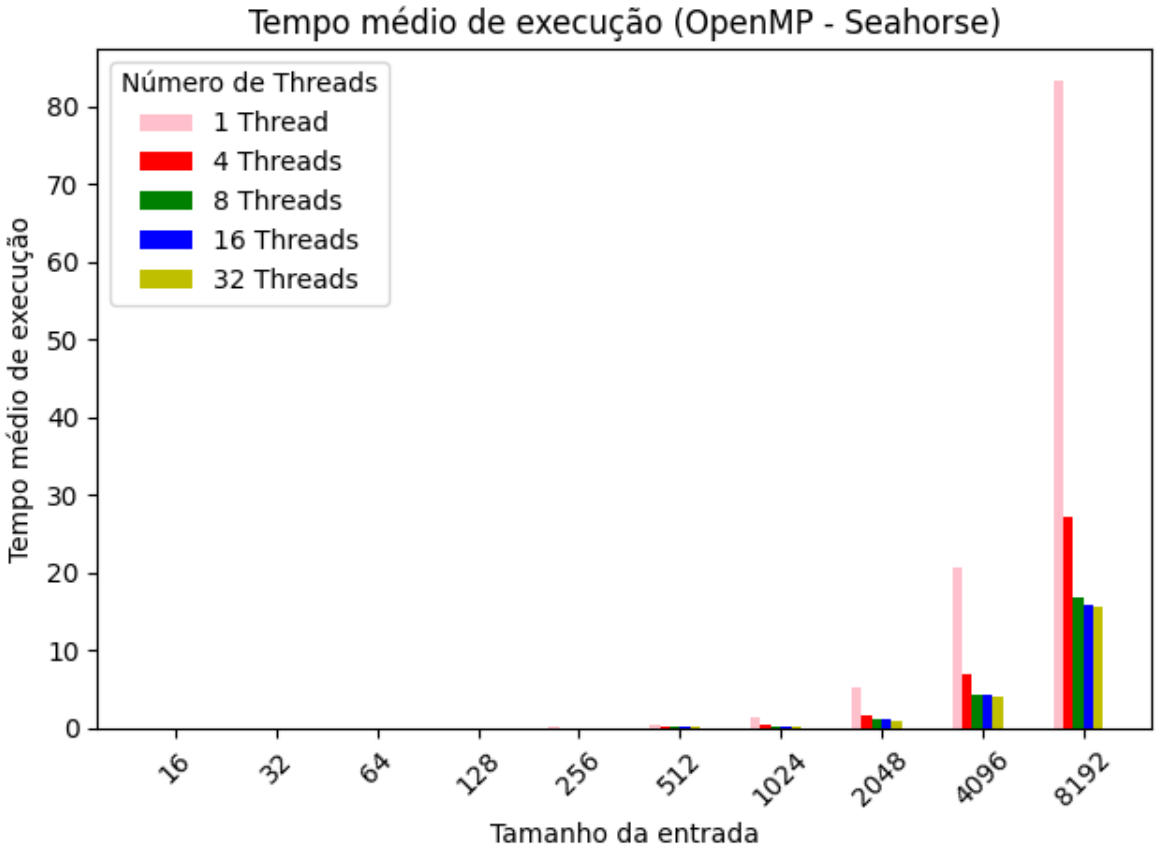

Figura 1: Gráfico do tempo médio de execução em relação ao tamanho da entrada, para diferentes número de Threads (OpenMP - Seahorse)

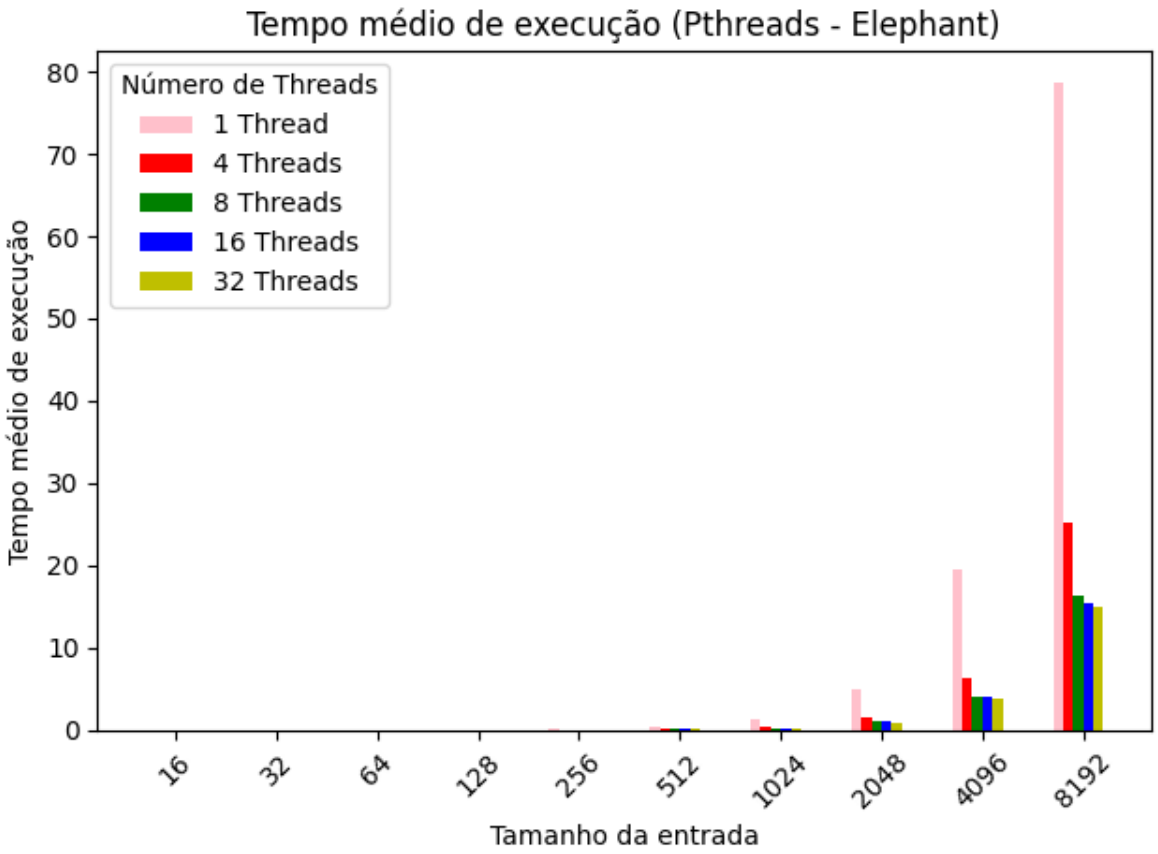

Figura 2: Gráfico do tempo médio de execução em relação ao tamanho da entrada, para diferentes número de Threads (Pthreads - Elephant)

5.2 Impacto das operações de I/O e alocação de memória

Após realizar os experimentos, notou-se que as operações de entrada e saída e alocação de memória eram as operações que mais demoravam para ser executadas, sendo o cálculo do Conjunto de Mandelbrot um método muito mais rápido que as outras operações, principalmente para entradas maiores, como 8192, mesmo na versão sequencial. Tal diferença é muito expressiva, fazendo com que, para algumas regiões do Conjunto de Mandelbrot, alguns tamanhos de entrada e algumas versões do arquivo fonte, a versão sem alocação de memória e operações de entrada e saída sejam mais de 10 vezes mais rápida do que a versão com essas operações.

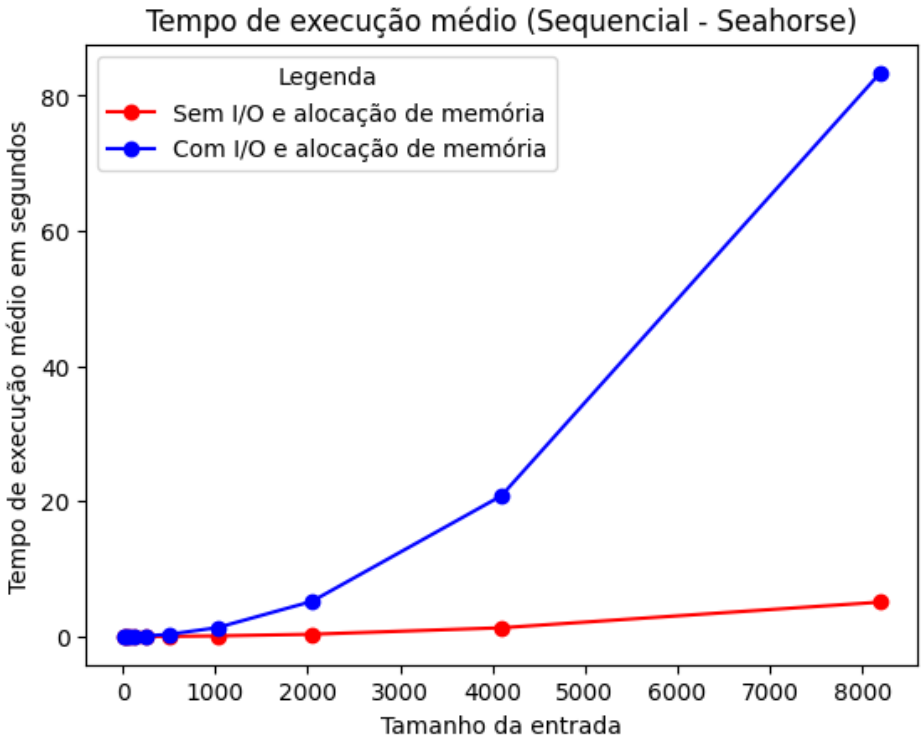

Figura 3: Gráfico do tempo médio de execução em relação ao tamanho da entrada, comparando o impacto das operações I/O e alocação de memória (Sequencial - Seahorse)

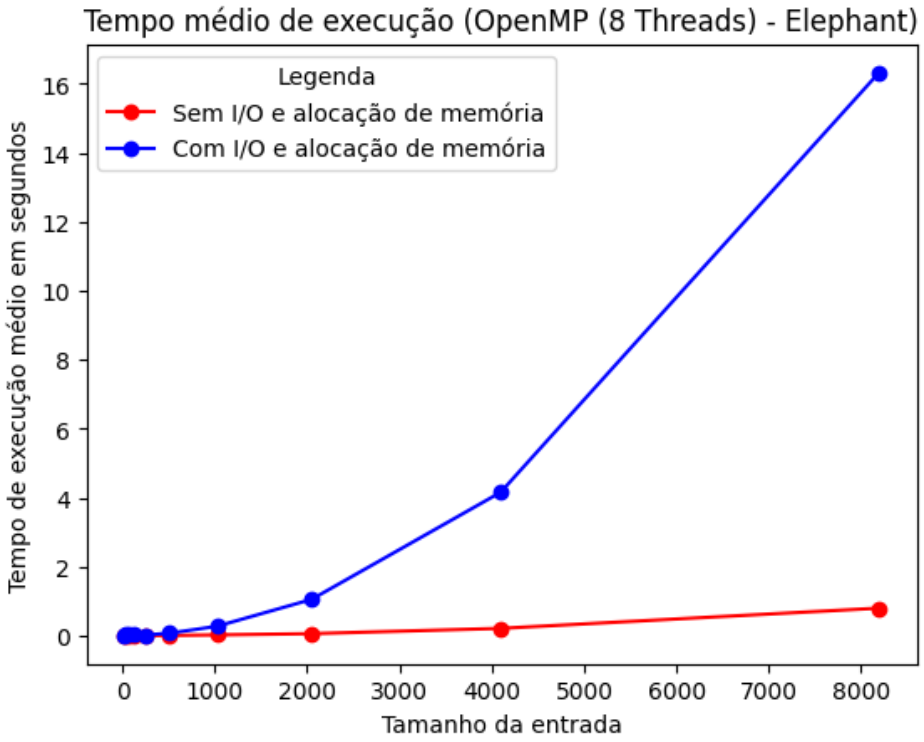

Figura 4: Gráfico do tempo médio de execução em relação ao tamanho da entrada, comparando o impacto das operações I/O e alocação de memória (OpenMP(8 Threads) - Elephant)

5.3 Diferença do tempo de execução para diferentes regiões do Conjunto de Mandelbrot

Analisando os dados obtidos, é perceptível que todas as regiões do Conjunto de Mandelbrot possuem tempo de execuções parecidos, exceto a imagem "Full"que possui o tempo de execução muito mais rápido do que as demais regiões. Isso ocorre porque muitos pontos próximos à margem da imagem estão fora do Conjunto de Mandelbrot, fazendo com que tais pontos façam poucas iterações no último laço encaixado da função que calcula o Conjunto de Mandelbrot.

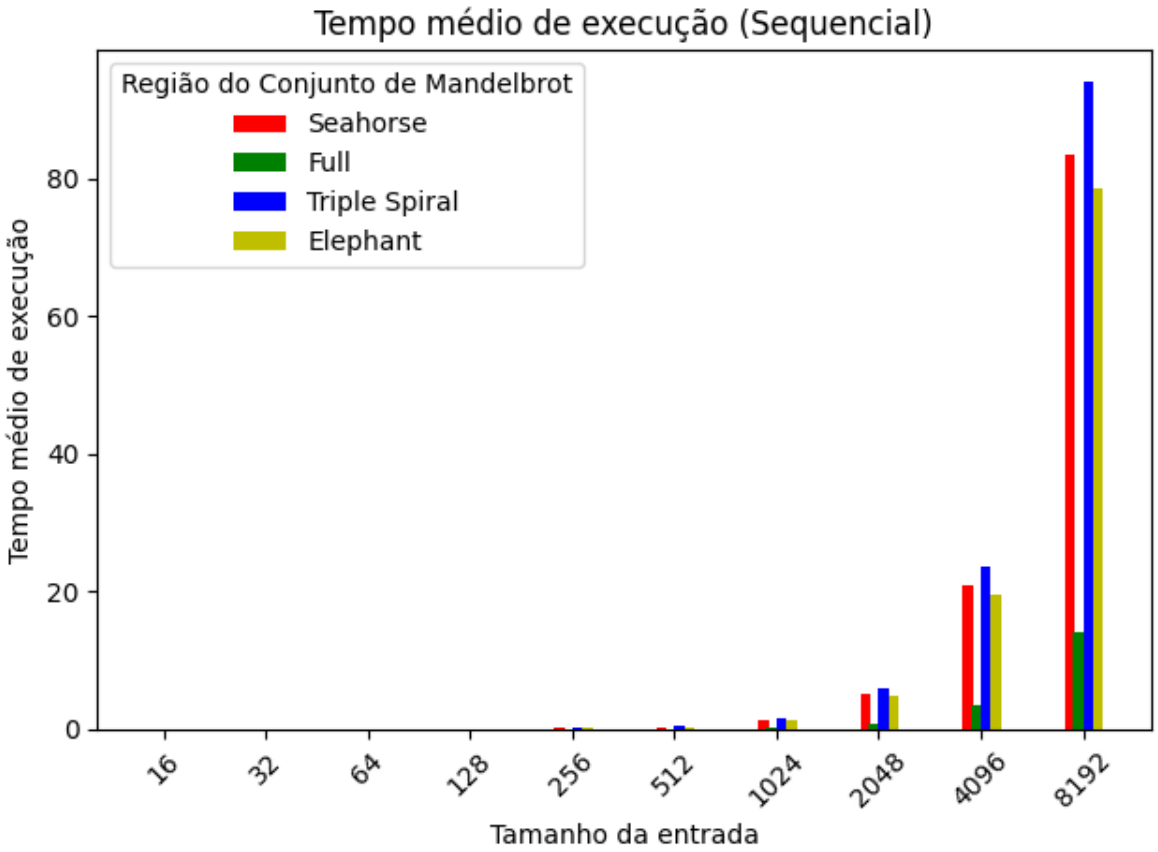

Figura 5: Gráfico do tempo médio de execução em relação ao tamanho da entrada, comparando as diferentes regões do Conjunto de Mandelbrot (Sequencial)

6 Conclusão

Primeiramente, é notório que a programação paralela é muito útil para a programação de alta performance, uma vez que ela permite uma melhor distribuição das tarefas entre os núcleos de processamento da máquina, diminuindo, asssim, o tempo de execução de um programa, como ocorreu no cálculo do Conjunto de Mandelbrot. Entretanto, é perceptível que para tarefas pequenas, como o cálculo do Conjunto de Mandelbrot para tamanhos de entrada pequeno, pode ser mais interessante utilizar a programação sequencial, uma vez que a criação e gerenciamento das Threads gasta um tempo que pode ser maior que o tempo de execução da tarefa. Nesse contexto, a biblioteca Pthreads e as diretivas de compilação do OpenMP podem ser muito úteis para a paralelização de programas, uma vez que ambas são fáceis de utilizar e possuem praticamente o mesmo desempenho.

Além disso, evidenciou-se que operações de entrada e saída e alocação de memória podem ser muito custosas dependendo do tamanho da memória alocada e do arquivo de saída que deve ser gerado pelo programa.