Relatório do EP1 de MAC0219 (2023)

Eduardo Sandalo Porto (NUSP 11796510) Maximilian Cabrajac Göritz (NUSP 11795418)

Introdução

Este texto contêm o relatório de execução dos exercícios propostos no EP1 da disciplina *Programação Concorrente e Paralela* (MAC0219) do Instituto de Matemática e Estatística da Universidade de São Paulo (IME-USP), no oferecimento de 2023.

Todos os experimentos foram realizados em uma máquina com um processador *11th Gen Intel i5-11320H*, e apresentam a média do tempo de execução de cada implementação variando o tamanho da grade do algoritmo e a quantidade de *threads* usadas, com uma barra que representa o intervalo de confiança com nível de 95%.

Implementação sequencial

Os resultados dos experimentos podem ser observados no seguinte gráfico:

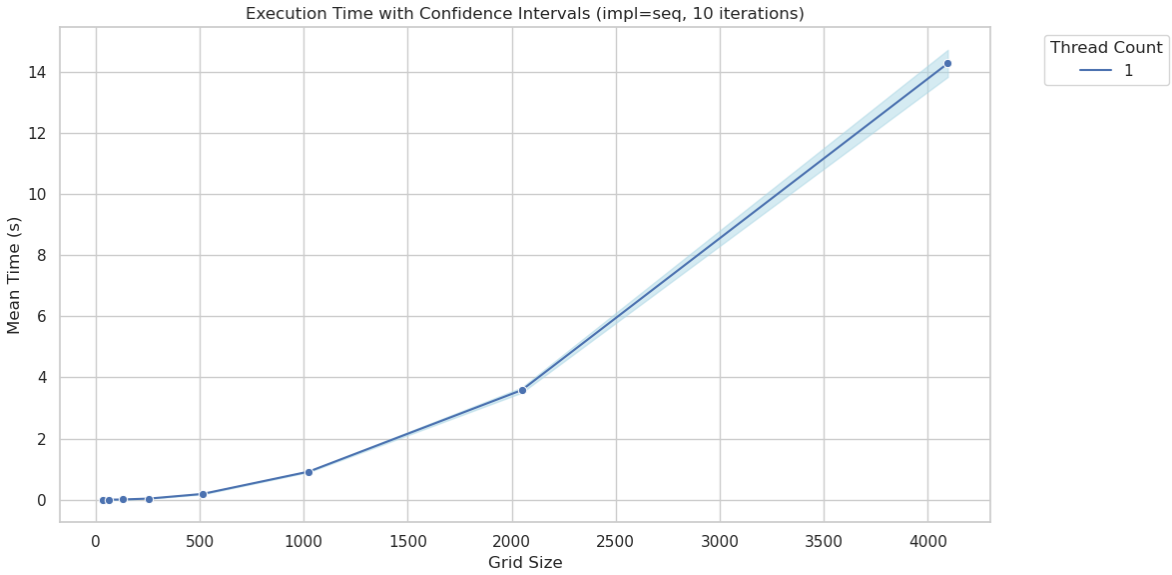

Figure 1: Implementação sequencial

Estes resultados serão considerados *base*, isto é, que serão a comparação para as outras implementações. Como esperado, o tempo de execução cresce de forma quadrática conforme o tamanho da grade aumenta.

Implementação com *POSIX threads*

Os resultados dos experimentos podem ser observados no seguinte gráfico:

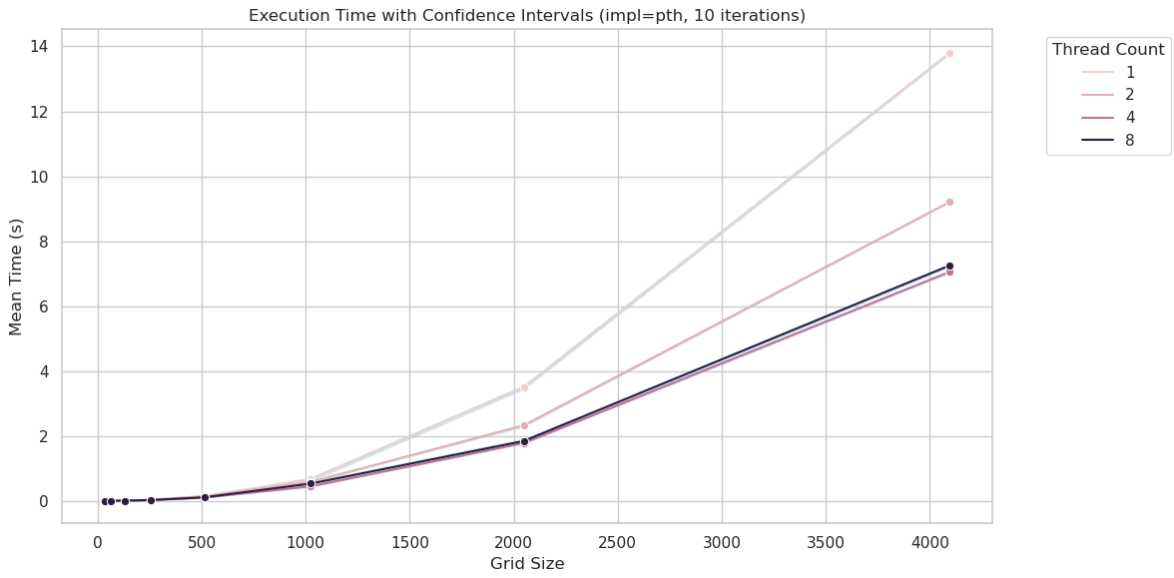

Figure 2: Implementação *POSIX* threads

Podemos ver que o tempo de execução caiu conforme aumentamos a quantidade de threads, estagnando com 4, já que o processador utilizado só possui 4 núcleos. É interessante notar que a queda em tempo de execução não seguiu uma proporção 1:1 com a quantidade de threads, isto é, aumentar as threads em 2x não fez o tempo cair em 2x. Isto pode ser explicado pelo overhead de paralelização e pelas partes sequenciais do programa que não se beneficiam da paralelização.

Implementação com *OpenMP*

Os resultados dos experimentos podem ser observados no seguinte gráfico:

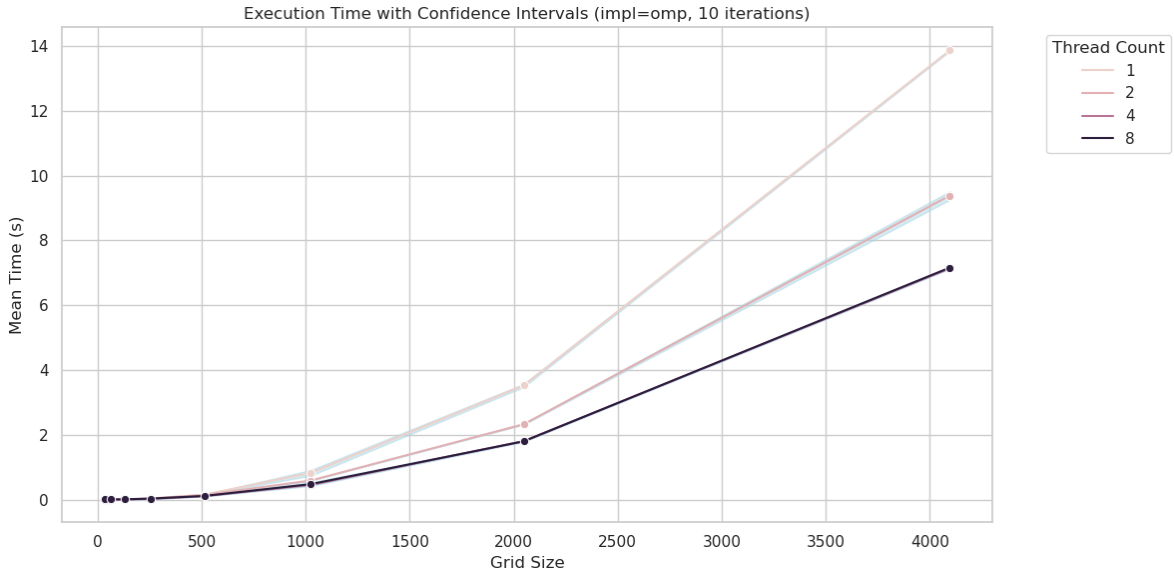

Figure 3: Implementação *OpenMP*

Os resultados destes experimentos são muito semelhantes à implementação com *POSIX threads*.

Anexo A: Script de experimentos

```python
#!/usr/python3

import ast
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from os import popen as shell
from itertools import product

# Parallelization implementation
IMPL = ["seq", "pth", "omp"][2]

# Number of iterations for each test
ITERATIONS = 10

# Output csv file
OUTPUT_FILE = f"stats/data_{IMPL}_{ITERATIONS}.csv"


def main():
    try:
        df = pd.read_csv(OUTPUT_FILE)
        plot(df)
    except IOError:
        make()
        df = execute()
        plot(df)


def make():
    print("================= Making =================")
    print(shell(f"make clean").read())
    print(shell(f"make all").read())


def execute():
    print()
    print("================ Executing ===============")
    data = []
    threads_exps = range(0, 3+1)
    array_exps = range(5, 12+1)
    for (threads_exp, array_exp) in product(threads_exps, array_exps):
        num_threads = 2 ** threads_exp
        grid_size = 2 ** array_exp

        cmd = f"./time_test --grid_size {grid_size} --num_threads {num_threads} --impl {IMPL}"
        info = f"{num_threads}, 2^{array_exp}: "

        print(info, end="")

        times = []
        for i in range(ITERATIONS):
            times.append(float(shell(cmd).read().strip().replace("s", "")))
        time = sum(times) / ITERATIONS

        print(f"{time:.5f}s")
```

```
            data.append((times, num_threads, grid_size))

    df = pd.DataFrame(data, columns=['Timings', 'Thread Count', 'Grid Size'])
    df.to_csv(OUTPUT_FILE)
    return df


def plot(df):
    print()
    print("================ Plotting ================")
    try:
        df['Mean'] = df['Timings'].apply(lambda x: sum(x) / len(x))
    except:
        df['Timings'] = df['Timings'].apply(ast.literal_eval)
        df['Mean'] = df['Timings'].apply(lambda x: sum(x) / len(x))

    df['CI'] = df['Timings'] \
        .apply(lambda x: 1.96 * np.std(x) / np.sqrt(len(x)))  # 95% confidence interval
    df['Low CI'] = df['Mean'] - df['CI']
    df['High CI'] = df['Mean'] + df['CI']

    # df.to_csv("data_with_ci.csv")

    sns.set(style='whitegrid')
    sns.color_palette("hls", 8)
    plt.figure(figsize=(12, 6))
    sns.lineplot(data=df, x='Grid Size', y='Mean',
                 hue='Thread Count', marker='o')

    for (_thread_count, df_tmp) in df.groupby("Thread Count"):
        plt.fill_between(df_tmp['Grid Size'],
                         df_tmp['Low CI'], df_tmp['High CI'],
                         alpha=0.5, color='#ADD8E6')

    plt.xlabel('Grid Size')
    plt.ylabel('Mean Time (s)')
    plt.title(
        f'Execution Time with Confidence Intervals (impl={IMPL}, {ITERATIONS} iterations)')
    plt.legend(
        title='Thread Count',
        bbox_to_anchor=(1.05, 1),
        loc='upper left')
    plt.tight_layout()
    plt.show()


if __name__ == "__main__":
    main()
```