Relatório da multiplicação de uma matriz 1000 por 1000 com 18 treadinds

Código Fonte:

```
import numpy as np
import pandas as pd
from multiprocessing import Pool
import os
import time

# Função para calcular uma linha de C
def calc_row(i, A, B):
    return np.dot(A[i], B)

# Função para multiplicar as matrizes em paralelo
def parallel_matrix_multiply(A, B, num_processes):
    pool = Pool(processes=num_processes)
    results = pool.starmap(calc_row, [(i, A, B) for i in
range(A.shape[0])])
    pool.close()
    pool.join()
    return np.array(results)

# Função para multiplicar as matrizes de forma serial (1 processador)
def serial_matrix_multiply(A, B):
    return np.dot(A, B)

# Proteção para Windows
if __name__ == '__main__':
    # Verificar diretório atual
    print("Diretório atual:", os.getcwd())

    # Caminho da Matrix A
    A_path = os.path.join(os.getcwd(),'data','matrix_A_1000.csv')

    # Caminho da Matrix B
    B_path = os.path.join(os.getcwd(),'data','matrix_B_1000.csv')

    # Carregar as matrizes de arquivos .csv
    A = pd.read_csv(A_path, header=None).values  # Matriz A
    B = pd.read_csv(B_path, header=None).values  # Matriz B
```

```
    # Multiplicação serial (1 processador)
    start_time = time.time()
    C_serial = serial_matrix_multiply(A, B)
    time_serial = time.time() - start_time
    print(f"Tempo de execução serial: {time_serial:.4f} segundos")

    # Salvar o resultado da multiplicação com 1 processador em um
arquivo CSV
    result_path_serial = 'resultado_multiplicacao_serial.csv'
    pd.DataFrame(C_serial).to_csv(result_path_serial, index=False,
header=False)
    print(f"Resultado da multiplicação com 1 processador salvo em
'{result_path_serial}'")

    # Multiplicação com 1 até 20 processadores
    for num_processors in range(1, 21):  # De 1 a 20 processadores
        start_time = time.time()
        C_parallel = parallel_matrix_multiply(A, B, num_processors)
        time_parallel = time.time() - start_time
        print(f"Tempo de execução com {num_processors} processadores:
{time_parallel:.4f} segundos")

        # Salvar o resultado da multiplicação em um arquivo CSV
        result_path_parallel =
f'resultado_multiplicacao_{num_processors}_processadores.csv'
        pd.DataFrame(C_parallel).to_csv(result_path_parallel,
index=False, header=False)
        print(f"Resultado da multiplicação com {num_processors}
processadores salvo em '{result_path_parallel}'")
```

INFORMAÇÕES:

Tempo de execução serial: 0.4562 segundos
Resultado da multiplicação com 1 processador salvo em 'resultado_multiplicacao_serial.csv'
Tempo de execução com 1 processadores: 0.9593 segundos
Resultado da multiplicação com 1 processadores salvo em 'resultado_multiplicacao_1_processadores.csv'
Tempo de execução com 2 processadores: 0.8770 segundos
Resultado da multiplicação com 2 processadores salvo em 'resultado_multiplicacao_2_processadores.csv'

Tempo de execução com 3 processadores: 0.9441 segundos
Resultado da multiplicação com 3 processadores salvo em 'resultado_multiplicacao_3_processadores.csv'
Tempo de execução com 4 processadores: 1.0365 segundos
Resultado da multiplicação com 4 processadores salvo em 'resultado_multiplicacao_4_processadores.csv'
Tempo de execução com 5 processadores: 1.2385 segundos
Resultado da multiplicação com 5 processadores salvo em 'resultado_multiplicacao_5_processadores.csv'
Tempo de execução com 6 processadores: 1.3364 segundos
Resultado da multiplicação com 6 processadores salvo em 'resultado_multiplicacao_6_processadores.csv'
Tempo de execução com 7 processadores: 1.4946 segundos
Resultado da multiplicação com 7 processadores salvo em 'resultado_multiplicacao_7_processadores.csv'
Tempo de execução com 8 processadores: 1.6498 segundos
Resultado da multiplicação com 8 processadores salvo em 'resultado_multiplicacao_8_processadores.csv'
Tempo de execução com 9 processadores: 1.8208 segundos
Resultado da multiplicação com 9 processadores salvo em 'resultado_multiplicacao_9_processadores.csv'
Tempo de execução com 10 processadores: 2.0048 segundos
Resultado da multiplicação com 10 processadores salvo em 'resultado_multiplicacao_10_processadores.csv'
Tempo de execução com 11 processadores: 2.2377 segundos
Resultado da multiplicação com 11 processadores salvo em 'resultado_multiplicacao_11_processadores.csv'
Tempo de execução com 12 processadores: 2.3116 segundos
Resultado da multiplicação com 12 processadores salvo em 'resultado_multiplicacao_12_processadores.csv'
Tempo de execução com 13 processadores: 2.4260 segundos
Resultado da multiplicação com 13 processadores salvo em 'resultado_multiplicacao_13_processadores.csv'
Tempo de execução com 14 processadores: 2.6502 segundos
Resultado da multiplicação com 14 processadores salvo em 'resultado_multiplicacao_14_processadores.csv'
Tempo de execução com 15 processadores: 2.8684 segundos
Resultado da multiplicação com 15 processadores salvo em 'resultado_multiplicacao_15_processadores.csv'
Tempo de execução com 16 processadores: 3.0175 segundos
Resultado da multiplicação com 16 processadores salvo em 'resultado_multiplicacao_16_processadores.csv'
Tempo de execução com 17 processadores: 3.2549 segundos

Resultado da multiplicação com 17 processadores salvo em 'resultado_multiplicacao_17_processadores.csv'
Tempo de execução com 18 processadores: 3.5612 segundos
Resultado da multiplicação com 18 processadores salvo em 'resultado_multiplicacao_18_processadores.csv'
Tempo de execução com 19 processadores: 3.6792 segundos
Resultado da multiplicação com 19 processadores salvo em 'resultado_multiplicacao_19_processadores.csv'
Tempo de execução com 20 processadores: 3.9441 segundos
Resultado da multiplicação com 20 processadores salvo em 'resultado_multiplicacao_20_processadores.csv'