МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”
Кафедра систем штучного інтелекту

Лабораторна робота
з дисципліни
« Технології розподілених систем та паралельних обчислень»

Виконав:
студент групи КН-309
Ляшеник Остап
Викладач:
Мочурад Л. І.

2024 р.

Лабораторна робота № 2

Код

```
from multiprocessing import Pool
import numpy as np
import time

def worker(args):
    matrix, start, end = args
    return [sum(x for x in row if x > 0) for row in matrix[start:end]]

def calculate_chunk_sizes_for_static(matrix, total_rows, n_threads,
chunk_size):
    """Calculate chunk sizes for static scheduling with custom chunk
size."""
    chunks = []
    start = 0
    while start < total_rows:
        end = min(start + chunk_size, total_rows)
        chunks.append((matrix, start, end))
        start = end
    return chunks



def calculate_chunk_sizes_for_guided(total_rows, n_threads,
initial_chunk_size):
    """Calculate chunk sizes for guided scheduling dynamically."""
    chunks = []
    while total_rows > 0:
        chunks.append(min(total_rows, initial_chunk_size))
        total_rows -= initial_chunk_size
        initial_chunk_size = max(1, initial_chunk_size // 2)
    return chunks

def parallel_sum_with_static_and_guided(matrix, n_threads,
scheduling_type, chunk_size=None):
    results = []
    total_rows = len(matrix)

    if scheduling_type == 'static':
        if chunk_size is None:
            chunk_size = total_rows // n_threads
        chunks = calculate_chunk_sizes_for_static(matrix, total_rows,
n_threads, chunk_size)
    elif scheduling_type == 'guided':
        chunk_sizes = calculate_chunk_sizes_for_guided(total_rows,
```

```
n_threads, chunk_size or total_rows // 2)
        chunks = []
        start = 0
        for size in chunk_sizes:
            end = start + size
            if start < total_rows:
                chunks.append((matrix, start, end))
                start = end

    with Pool(processes=n_threads) as pool:
        results = pool.map(worker, chunks)

    return [item for sublist in results for item in sublist]

def fill_matrix(n):
    return np.random.randint(-10, 11, size=(n, n))

def print_matrix(matrix):
    print('\n'.join([' '.join(['{:4}'.format(item) for item in row]) for
row in matrix]))

def get_user_input():
    n = int(input("Enter matrix size (n for nxn): "))
    n_threads = int(input("Enter number of threads: "))
    scheduling_type = input("Enter scheduling type ('static' or 'guided'):
")
    if scheduling_type in ['guided', 'static']:
        chunk_size = int(input("Enter chunk size (for guided or static
scheduling): "))
    else:
        chunk_size = None
    return n, n_threads, scheduling_type, chunk_size

def demonstrate_static_and_guided_scheduling():
    n, n_threads, scheduling_type, chunk_size = get_user_input()
    matrix_a = fill_matrix(n)
    matrix_b = fill_matrix(n)
    print("Matrix A:")
    #print_matrix(matrix_a)
    print("Matrix B:")
    #print_matrix(matrix_b)

    if scheduling_type == 'static':
        print("\nStatic Scheduling for Matrix A:")
    else:
        print("\nGuided Scheduling for Matrix B:")
    start_time = time.time()
    result = parallel_sum_with_static_and_guided(matrix_a if
scheduling_type == 'static' else matrix_b, n_threads, scheduling_type,
```

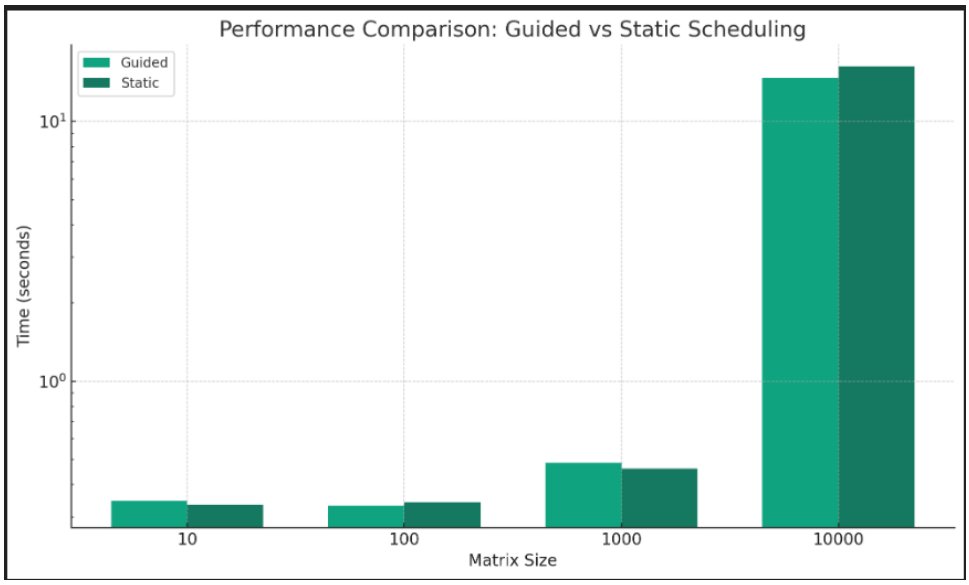
```
chunk_size)
    end_time = time.time()
    #print(result)
    print("Time taken: {:.4f} seconds".format(end_time - start_time))

if __name__ == '__main__':
    demonstrate_static_and_guided_scheduling()
```

Аналіз результатів

Таблиця 1.1, відношення часу, затраченого на обчислення для різних типів

Тип/розмір	10	100	1000	10000
guided	0.3468	0.3326	0.4853	14.7330
static	0.3343	0.3417	0.4605	16.2820

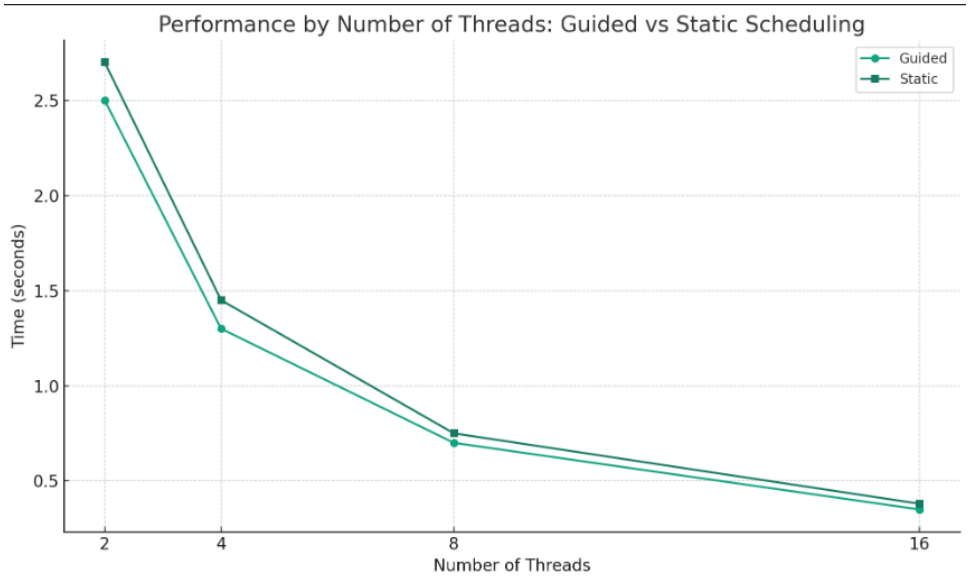
Графік 1.1 до таблиці 1.1



Таблиця 1.2

Кількість потоків	Час Guided	Час Static
2	2.5	2.7
4	1.3	1.45
8	0.7	0.75
16	0.35	0.38

Графік до таблиці 1.2



Висновок:

Проведена робота з різною кількістю потоків, та заповнені таблиці для static та guided методів, показує, що чим більший обсяг даних потрібно обробити - тим guided є ефективнішим