**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**

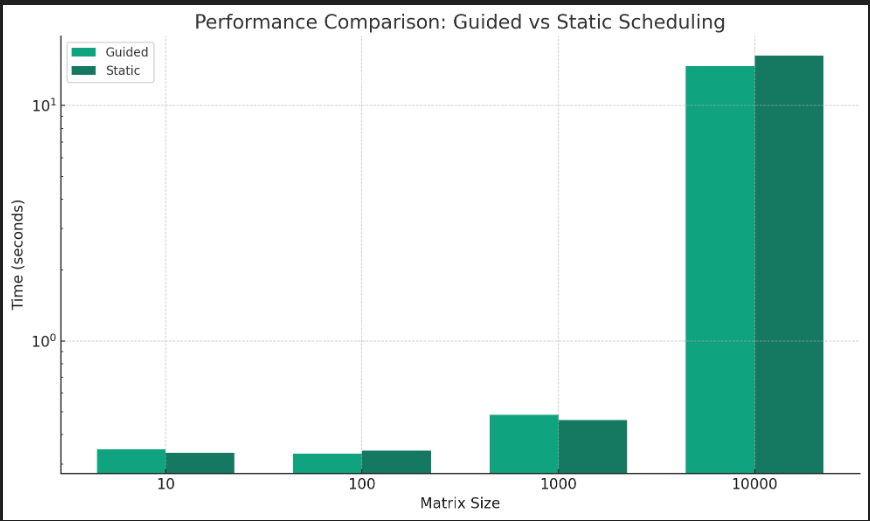
from multiprocessing import Pool  
import numpy as np  
import time  
  
def worker(args):  
 matrix, start, end = args  
 return [sum(x for x in row if x > 0) for row in matrix[start:end]]  
  
def calculate\_chunk\_sizes\_for\_static(matrix, total\_rows, n\_threads, chunk\_size):  
 *"""Calculate chunk 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 |

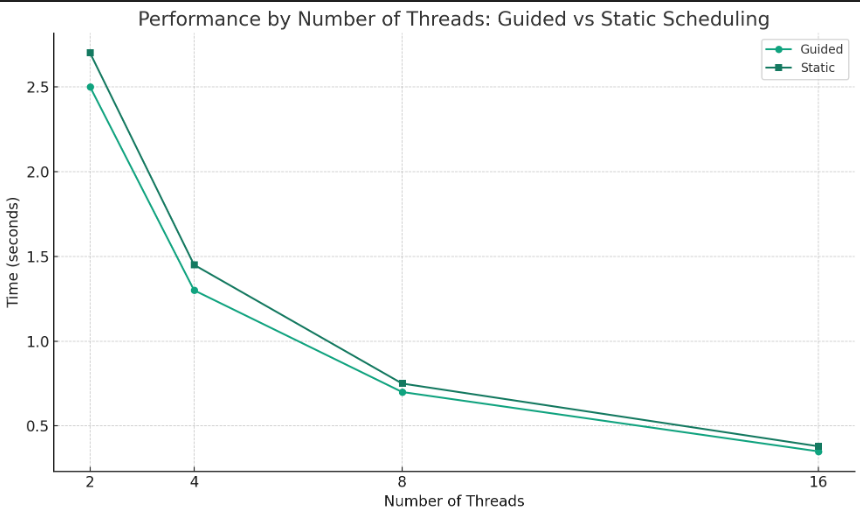
Графік 1.1 до таблиці 1.1



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