**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. І. І. СІКОРСЬКОГО»**

**ННК «ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ»**

**Кафедра системного проектування**

**Розрахункова-графічна робота**

**з дисципліни**

**Розподілені обчислювання та інтернет-технології**

Виконала: студентка групи ДА-11мп

Молчанова В.С.

Київ – 2021

## Завдання

Задача: множення матриці на матрицю

1. Описати алгоритм рішення задачі.
2. Написати та виконати програму.
3. Дослідити ефективність в залежності від розмірності задачі

## Алгоритм рішення задачі

Перша матриця розбивається на P однакових частин (Р – кількість процесорів), кожна з яких містить декілька рядків цієї матриці. За допомогою функції розсилки кожен з процесів отримує свій набор рядків першої матриці, та за допомогою функції широкомовного обміну кожен з них отримує другу матрицю.

Далі кожен процесор множить кожен з отриманих рядків першої матриці на кожен стовпець другої матриці, таким чином отримуючи декілька рядків результуючої матриці. Ці результуючі рядки надсилаються головному процесу за допомогою функції збору, а він об’єднує їх в єдину матрицю.

## Програма для рішення задачі

Програмний код:

from mpi4py import MPI

import random

from functools import reduce

import numpy as np

comm = MPI.COMM\_WORLD

rank = comm.Get\_rank()

n\_procs = comm.Get\_size()

root = 0

m\_size = 500

def generate\_matrix():

    return [[ random.randint(0, 100) for \_ in range(m\_size)] for \_ in range(m\_size)]

def scalar\_mult(v1, v2):

    return sum(map(lambda x: x[0]\*x[1], zip(v1, v2)))

if rank == root:

    matrix1 = generate\_matrix()

    matrix2 = np.array(generate\_matrix())

    matrix1\_chunks = list(map(lambda l: l.tolist(), np.array\_split(matrix1, n\_procs)))

    start = MPI.Wtime()

else:

    matrix1\_chunks = None

    matrix2 = None

matrix1\_chunk = comm.scatter(matrix1\_chunks, root=0)

matrix2 = comm.bcast(matrix2, root=root)

proc\_rows = [[scalar\_mult(row, matrix2[:, i]) for i in range(m\_size)] for row in matrix1\_chunk]

result\_chunks = comm.gather(proc\_rows, root=0)

if rank == root:

    result = np.array(reduce(list.\_\_add\_\_, result\_chunks))

    time = MPI.Wtime() - start

    print(f'Elapsed time: {time}')

Приклад результату для малого розміру матриці:

Table

Description automatically generated

## Дослідження ефективності програми

При множенні квадратних матриць розміром NxN, для кожного з NxN елементів результуючої матриці потрібно виконати N операцій множення та N-1 операцій суми. Отже загалом, потрібно виконати

N\*N\*(N+N-1) = 2\*N3 – N2 операцій.

Переслати треба 3N чисел. Якщо припустити, що довжина слова з плаваючою комою дорівнює 10 байт, використовувати технологію Fast Ethernet з пропускною здатністю 100 Мб/сек, та вважати час обчислення однієї операції рівним 20 нс, то коефіцієнт цифрової деградації буде дорівнювати

## Висновки

В даній роботі була розроблена програма для множення двох матриць та обчислена її ефективність. Розрахунки показали, що прискорення роботи програми буде тим більшим, чим більшим буде розмір обчислюваних матриць, що є гарним результатом для програми розподілених обчислень.