МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО» ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Кафедра інформатики та програмної інженерії

Звіт

З компʼютерного практикуму № 7 з дисципліни

«Технології паралельних обчислень»

Тема: «Розробка паралельного алгоритму множення матриць з використанням МРІ-методів колективного обміну повідомленнями («один-до-багатьох», «багато-до-одного», «багато-до-багатьох») та дослідження його ефективності»

**Виконав(ла)** *ІП-15 Мєшков Андрій*

(шифр, прізвище, ім'я, по батькові)

**Перевірив** *Дифучина О. Ю.*

(шифр, прізвище, ім'я, по батькові)

Київ 2024

# ЗАВДАННЯ

1. Ознайомитись з методами колективного обміну повідомленнями типу «один-до-багатьох», «багато-до-одного», «багато-до-багатьох» (див. лекцію та документацію стандарту MPI).

2. Реалізувати алгоритм паралельного множення матриць з використанням розподілених обчислень в MPI з використанням методів колективного обміну повідомленнями. 40 балів.

3. Дослідити ефективність розподіленого обчислення алгоритму множення матриць при збільшенні розміру матриць та при збільшенні кількості вузлів, на яких здійснюється запуск програми. Порівняйте ефективність алгоритму при використанні методів обміну повідомленнями «один-до-одного», «один-до-багатьох», «багато-до-одного», «багато-до-багатьох». 60 балів.

# ХІД РОБОТИ

Лістинг коду:

Mult.py

*from* mpi4py *import* MPI

*import* numpy *as* np

*import* sys

def main():

comm = MPI.COMM\_WORLD

rank = comm.Get\_rank()

size = comm.Get\_size()

n = 3000

master = 0

*if* size < 2:

*if* rank == master:

print("Cannot mult")

MPI.COMM\_WORLD.Abort(0)

sys.exit(0)

workers = size - 1

rows\_for\_worker = n // workers

extra\_rows = n % workers

*if* rank == master:

A = np.ones((n, n), dtype=int)

B = np.ones((n, n), dtype=int)

C = np.zeros((n, n), dtype=int)

start\_time = MPI.Wtime()

requests = []

start\_row\_a = 0

*for* dest *in* range(1, workers + 1):

rows = rows\_for\_worker + 1 *if* dest <= extra\_rows *else* rows\_for\_worker

req\_a = comm.Isend([A[start\_row\_a:start\_row\_a + rows, :], MPI.INT], dest=dest, tag=0)

req\_b = comm.Isend([B, MPI.INT], dest=dest, tag=0)

requests.append(req\_a)

requests.append(req\_b)

start\_row\_a += rows

MPI.Request.Waitall(requests)

requests = []

start\_row = 0

*for* source *in* range(1, workers + 1):

rows = rows\_for\_worker + 1 *if* source <= extra\_rows *else* rows\_for\_worker

req\_c = comm.Irecv([C[start\_row:start\_row + rows, :], MPI.INT], source=source, tag=0)

requests.append(req\_c)

start\_row += rows

MPI.Request.Waitall(requests)

end\_time = MPI.Wtime()

elapsed\_time\_ms = end\_time - start\_time

print(f"Time elapsed: {elapsed\_time\_ms:.2f} s - not-blocked matrix multiplication {n}x{n}")

*# Uncomment below lines to print matrix C (Not recommended for large n)*

*# for i in range(n):*

*# for j in range(n):*

*# print(C[i, j], end=" ")*

*# print()*

*else*:

rows = rows\_for\_worker + 1 *if* rank <= extra\_rows *else* rows\_for\_worker

a\_rows = np.empty((rows, n), dtype=int)

b = np.empty((n, n), dtype=int)

requests = []

req\_a\_rows = comm.Irecv([a\_rows, MPI.INT], source=master, tag=0)

req\_b = comm.Irecv([b, MPI.INT], source=master, tag=0)

requests.append(req\_a\_rows)

requests.append(req\_b)

MPI.Request.Waitall(requests)

result = np.zeros((rows, n), dtype=int)

*for* row *in* range(rows):

*for* column *in* range(n):

*for* k *in* range(n):

result[row, column] += a\_rows[row, k] \* b[k, column]

comm.Isend([result, MPI.INT], dest=master, tag=0)

*if* \_\_name\_\_ == "\_\_main\_\_":

main()

**Результат:**



Рисунок 1 – Результат запуску програми

В таблиці зображено порівняння алгоритмів.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Розмір | | Звичайний, с | | 8 | | | | 9 | | | | 10 | | |
| Один до одного, с | | Прискорення | | Один до одного, с | | Прискорення | | Один до одного, с | | Прискорення |
| 1000 | | 567 | | 205 | | 2,7659 | | 185 | | 3,0648 | | 181 | | 3,1326 |
| 1500 | | 2167 | | 765 | | 2,8291 | | 683 | | 3,1728 | | 648 | | 3,3441 |
| 2000 | | 4698 | | 1583 | | 2,9679 | | 1392 | | 3,375 | | 1256 | | 3,7404 |
| 2500 | | 10864 | | 3644 | | 2,9815 | | 3291 | | 3,3011 | | 2892 | | 3,7566 |
| 3000 | | 17792 | | 5948 | | 2,9911 | | 5301 | | 3,3563 | | 4647 | | 3,8287 |
| Розмір | | Звичайний, с | | 8 | | | | | | | |
| Один До Багатьох, с | | Прискорення | | Багато До Багатьох, с | | Прискорення | |
| 1000 | | 567 | | 192 | | 2,953125 | | 193 | | 2,93782383 | |
| 1500 | | 2167 | | 770 | | 2,81428571 | | 766 | | 2,82898172 | |
| 2000 | | 4698 | | 1543 | | 3,04471808 | | 1545 | | 3,0407767 | |
| 2500 | | 10864 | | 3602 | | 3,01610217 | | 3599 | | 3,01861628 | |
| 3000 | | 17792 | | 5121 | | 3,47432142 | | 4983 | | 3,57053984 | |
| Розмір | | Звичайний, с | | 9 | | | | | | | |
| Один До Багатьох, мс | | Прискорення | | Багато До Багатьох, с | | Прискорення | |
| 1000 | | 567 | | 190 | | 2,98421053 | | 192 | | 2,953125 | |
| 1500 | | 2167 | | 754 | | 2,87400531 | | 761 | | 2,84756899 | |
| 2000 | | 4698 | | 1537 | | 3,05660377 | | 1519 | | 3,09282423 | |
| 2500 | | 10864 | | 3194 | | 3,40137758 | | 3084 | | 3,5226978 | |
| 3000 | | 17792 | | 4702 | | 3,78392174 | | 4302 | | 4,13575081 | |
| Розмір | | Звичайний, с | | 10 | | | | | | | |
| Один До Багатьох, с | | Прискорення | | Багато До Багатьох, с | | Прискорення | |
| 1000 | | 567 | | 188 | | 3,01595745 | | 190 | | 2,98421053 | |
| 1500 | | 2167 | | 750 | | 2,88933333 | | 748 | | 2,89705882 | |
| 2000 | | 4698 | | 1530 | | 3,07058824 | | 1501 | | 3,12991339 | |
| 2500 | | 10864 | | 2781 | | 3,90650845 | | 2650 | | 4,09962264 | |
| 3000 | | 17792 | | 4203 | | 4,23316679 | | 3983 | | 4,46698468 | |

# ВИСНОВКИ

В результаті роботи над комп’ютерним практикумом було розроблено програму, що реалізує паралельне множення матриць з використанням MPI методів обміну повідомленнями один-до-багатьох та багато-до-багатьох.