Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»

(МГТУ им. Н.Э. Баумана)

Факультет: Информатика и системы управления

Кафедра: Теоретическая информатика и компьютерные технологии

**Лабораторная работа №1**

**«Распараллеливание алгоритма вычисления произведения двух матриц»**

**по курсу: «Разработка параллельных и распределенных программ»**

Выполнил:

Студент группы ИУ9-52Б

Джабаров Р.А.

Проверил:

Царев А.С.

**1. Цель работы**

Распараллеливание алгоритма вычисления произведения двух матриц.

**2. Реализация**

Код программы:

from random import randint

from threading import Thread

import numpy as np

import time

a, b, c = [], [], []

n = 500

p = 2

def printt():

print(a)

print(b)

print(c, '\n')

def generateMatrix():

for i in range(n):

rowsA = []

rowsB = []

rowsC = []

for j in range(n):

rowsA.append(randint(0, 9))

rowsB.append(randint(0, 9))

rowsC.append(0)

a.append(rowsA)

b.append(rowsB)

c.append(rowsC)

def defaultAlg():

start\_time = time.time()

for i in range(n):

for j in range(n):

cij = 0

for k in range(n):

cij += a[i][k] \* b[k][j]

c[i][j] = cij

end\_time = time.time()

print("Время вычисления по строкам: ", round((end\_time - start\_time) \* 1000, 3), "ms")

#print(c, "default")

def notDefaultAlg():

start\_time = time.time()

for i in range(n):

for j in range(n):

cji = 0

for k in range(n):

cji += a[j][k] \* b[k][i]

c[j][i] = cji

end\_time = time.time()

print("Время вычисления по столбцам: ", round((end\_time - start\_time) \* 1000, 3), "ms")

#printt()

def calculatetResElem(i, j):

res = 0

for ii in range(n):

res += a[i][ii] \* b[ii][j]

return res

def threadAlg(ifrom, jfrom, ito, jto):

for i in range(n // p):

jFproc = jfrom

iFproc = ifrom

jTproc = jto

iTproc = ito

while (jFproc != -1):

c[iFproc][jFproc] = calculatetResElem(iFproc, jFproc)

jFproc -= 1

iFproc += 1

while (iTproc != n):

c[iTproc][jTproc] = calculatetResElem(iTproc, jTproc)

iTproc += 1

jTproc -= 1

jfrom += 1

ito -= 1

#printt()

generateMatrix()

start\_time = time.time()

mas = []

# 2 потока

thread = (Thread(target=threadAlg, args=(0,0, n - 1, n-1)))

thread2 = (Thread(target=threadAlg, args=(0,n // p, n - 1 - (n // p), n-1)))

mas.append(thread)

mas.append(thread2)

for i in range(len(mas)):

mas[i].start()

for i in range(len(mas)):

mas[i].join()

end\_time = time.time()

print(f"Время вычисления по {p} потокам: ", round((end\_time - start\_time) \* 1000, 3), "ms")

# 3 потока

start\_time = time.time()

p = 3

mas = []

thread = (Thread(target=threadAlg, args=(0,0, n - 1, n-1)))

thread2 = (Thread(target=threadAlg, args=(0,n // p, n - 1 - (n // p), n-1)))

thread3 = (Thread(target=threadAlg, args=(0,2 \* n // p, n - 1 - 2 \* (n // p), n-1)))

mas.append(thread)

mas.append(thread2)

mas.append(thread3)

for i in range(len(mas)):

mas[i].start()

for i in range(len(mas)):

mas[i].join()

end\_time = time.time()

print(f"Время вычисления по {p} потокам: ", round((end\_time - start\_time) \* 1000, 3), "ms")

# 4 потока

start\_time = time.time()

p = 4

mas = []

thread = (Thread(target=threadAlg, args=(0,0, n - 1, n-1)))

thread2 = (Thread(target=threadAlg, args=(0,n // p, n - 1 - (n // p), n-1)))

thread3 = (Thread(target=threadAlg, args=(0,2 \* n // p, n - 1 - 2 \* (n // p), n-1)))

thread4 = (Thread(target=threadAlg, args=(0,3 \* n // p, n - 1 - 3 \* (n // p), n-1)))

mas.append(thread)

mas.append(thread2)

mas.append(thread3)

mas.append(thread4)

for i in range(len(mas)):

mas[i].start()

for i in range(len(mas)):

mas[i].join()

end\_time = time.time()

print(f"Время вычисления по {p} потокам: ", round((end\_time - start\_time) \* 1000, 3), "ms")

# 5 потока

start\_time = time.time()

p = 5

mas = []

thread = (Thread(target=threadAlg, args=(0,0, n - 1, n-1)))

thread2 = (Thread(target=threadAlg, args=(0,n // p, n - 1 - (n // p), n-1)))

thread3 = (Thread(target=threadAlg, args=(0,2 \* n // p, n - 1 - 2 \* (n // p), n-1)))

thread4 = (Thread(target=threadAlg, args=(0,3 \* n // p, n - 1 - 3 \* (n // p), n-1)))

thread5 = (Thread(target=threadAlg, args=(0,4 \* n // p, n - 1 - 4 \* (n // p), n-1)))

mas.append(thread)

mas.append(thread2)

mas.append(thread3)

mas.append(thread4)

mas.append(thread5)

for i in range(len(mas)):

mas[i].start()

for i in range(len(mas)):

mas[i].join()

end\_time = time.time()

print(f"Время вычисления по {p} потокам: ", round((end\_time - start\_time) \* 1000, 3), "ms")

# 6 потока

start\_time = time.time()

p = 6

mas = []

thread = (Thread(target=threadAlg, args=(0,0, n - 1, n-1)))

thread2 = (Thread(target=threadAlg, args=(0,n // p, n - 1 - (n // p), n-1)))

thread3 = (Thread(target=threadAlg, args=(0,2 \* n // p, n - 1 - 2 \* (n // p), n-1)))

thread4 = (Thread(target=threadAlg, args=(0,3 \* n // p, n - 1 - 3 \* (n // p), n-1)))

thread5 = (Thread(target=threadAlg, args=(0,4 \* n // p, n - 1 - 4 \* (n // p), n-1)))

thread6 = (Thread(target=threadAlg, args=(0,5 \* n // p, n - 1 - 5 \* (n // p), n-1)))

mas.append(thread)

mas.append(thread2)

mas.append(thread3)

mas.append(thread4)

mas.append(thread5)

mas.append(thread6)

for i in range(len(mas)):

mas[i].start()

for i in range(len(mas)):

mas[i].join()

end\_time = time.time()

print(f"Время вычисления по {p} потокам: ", round((end\_time - start\_time) \* 1000, 3), "ms")

# 7 потока

start\_time = time.time()

p = 7

mas = []

thread = (Thread(target=threadAlg, args=(0,0, n - 1, n-1)))

thread2 = (Thread(target=threadAlg, args=(0,n // p, n - 1 - (n // p), n-1)))

thread3 = (Thread(target=threadAlg, args=(0,2 \* n // p, n - 1 - 2 \* (n // p), n-1)))

thread4 = (Thread(target=threadAlg, args=(0,3 \* n // p, n - 1 - 3 \* (n // p), n-1)))

thread5 = (Thread(target=threadAlg, args=(0,4 \* n // p, n - 1 - 4 \* (n // p), n-1)))

thread6 = (Thread(target=threadAlg, args=(0,5 \* n // p, n - 1 - 5 \* (n // p), n-1)))

thread7 = (Thread(target=threadAlg, args=(0,6 \* n // p, n - 1 - 6 \* (n // p), n-1)))

mas.append(thread)

mas.append(thread2)

mas.append(thread3)

mas.append(thread4)

mas.append(thread5)

mas.append(thread6)

mas.append(thread7)

for i in range(len(mas)):

mas[i].start()

for i in range(len(mas)):

mas[i].join()

end\_time = time.time()

print(f"Время вычисления по {p} потокам: ", round((end\_time - start\_time) \* 1000, 3), "ms")

# 8 потоков

start\_time = time.time()

p = 8

mas = []

thread = (Thread(target=threadAlg, args=(0,0, n - 1, n-1)))

thread2 = (Thread(target=threadAlg, args=(0,n // p, n - 1 - (n // p), n-1)))

thread3 = (Thread(target=threadAlg, args=(0,2 \* n // p, n - 1 - 2 \* (n // p), n-1)))

thread4 = (Thread(target=threadAlg, args=(0,3 \* n // p, n - 1 - 3 \* (n // p), n-1)))

thread5 = (Thread(target=threadAlg, args=(0,4 \* n // p, n - 1 - 4 \* (n // p), n-1)))

thread6 = (Thread(target=threadAlg, args=(0,5 \* n // p, n - 1 - 5 \* (n // p), n-1)))

thread7 = (Thread(target=threadAlg, args=(0,6 \* n // p, n - 1 - 6 \* (n // p), n-1)))

thread8 = (Thread(target=threadAlg, args=(0,7 \* n // p, n - 1 - 7 \* (n // p), n-1)))

mas.append(thread)

mas.append(thread2)

mas.append(thread3)

mas.append(thread4)

mas.append(thread5)

mas.append(thread6)

mas.append(thread7)

mas.append(thread8)

for i in range(len(mas)):

mas[i].start()

for i in range(len(mas)):

mas[i].join()

end\_time = time.time()

print(f"Время вычисления по {p} потокам: ", round((end\_time - start\_time) \* 1000, 3), "ms")

defaultAlg()

notDefaultAlg()

'''

thread1.start()

thread2.start()

thread1.join()

thread2.join()

'''

**4. Время работы:**

Время вычисления по 2 потокам: 5900.327 ms

Время вычисления по 3 потокам: 5779.321 ms

Время вычисления по 4 потокам: 5606.623 ms

Время вычисления по 5 потокам: 5121.251 ms

Время вычисления по 6 потокам: 4902.123 ms

Время вычисления по 7 потокам: 4890.733 ms

Время вычисления по 8 потокам: 4812.461 ms

Время вычисления по строкам: 5914.396 ms

Время вычисления по столбцам: 5915.573 ms