Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Лабораторная работа № 1

по курсу «Распределение параллельных и распределённых программ»

«Распараллеливание алгоритма вычисления произведения двух матриц»

Студент группы ИУ9-51Б Горбунов А. Д.

Преподаватель Царёв А. С.

Москва 2024

1 Задача

Замерить время вычисления, сравнить с временем при вычислении элементов матрицы C не по строкам, а по столбцам. Размер матриц подобрать таким образом, чтобы время выполнения на вашей машине было не слишком непоказательно малым (меньше нескольких минут), но и не чересчур большим (несколько часов). Использовать библиотечные функции для вычисления произведений матриц нельзя.

Затем конечную матрицу C условно разделить на примерно равные прямоугольные подматрицы и распараллелить программу таким образом, чтобы каждый поток занимался вычислением своей подматрицы. Матрицы A и B для этого разделить на примерно равные группы строк и столбцов соответственно. Сделать для разного количества потоков (разных разбиений), также замерить время вычисления, сравнить с вычислениями стандартным алгоритмом. Также по окончании вычислений сравнивать получившуюся матрицу с той, что была вычислена стандартным алгоритмом, для проверки правильности вычислений.

2 Характеристики устройства


Ubuntu
Device Name
GoComp
Hardware Model
HP HP Pavilion Gaming Laptop 15-ec0xxx
Memory
8,0 GiB
Processor
AMD® Ryzen 5 3550h with radeon vega mobile gfx × 8
Graphics
AMD® Radeon vega 8 graphics
Disk Capacity
256,1 GB
OS Name
Ubuntu 22.04.2 LTS
OS Type
64-bit
GNOME Version
42.5
Windowing System
Wayland

3 Код решения

Файл main.py

```
import random
import numpy as np
import threading, time
n_1 = 501  # Размерность матрицы
m_1 = 520
matrix_1 = np.array(
[[random.randint(-1000, 1000) for _ in range(m_1)] for _ in range(n_1)])

n_2 = 520  # Размерность матрицы
m_2 = 515
matrix_2 = np.array(
[[random.randint(-1000, 1000) for _ in range(m_2)] for _ in range(n_2)])

print("ожидаемый результат:")
print(np.matmul(matrix_1, matrix_2))

def multiplicationOfCube(n, final_matrix):
    for i in range(n):
        for j in range(m_2):
            c = 0
            for k in range(len(matrix_1[i])):
                c += matrix_1[i][k] * matrix_2[j][k]
            final_matrix[i, j] = c

def multiplicationOfCubeThreading(n, h, b, final_matrix):
    n_min = 0
    if h > 0:
        if h == 1:
            n_min = n-1
        else:
```

```
            n_min = b*(h-1)*n
        n = b*h*n
    if n == 0:
        n = n_1

    for i in range(n_min, n):
        if i >= len(matrix_1):
            break
        for j in range(m_2):
            c = 0
            for k in range(len(matrix_2[j])):
                c += matrix_1[i][k] * matrix_2[j][k]
            final_matrix[i, j] = c


def NThreadings(b):
    final_matrix = np.array(
    [[random.randint(-1000, 1000) for _ in range(m_2)] for _ in range(n_1)])
    #print(final_matrix)

    for h in range(b):
        if n_1 %2 !=0:
            thread = threading.Thread(name='worker',
                                      target=multiplicationOfCubeThreading,
                                      args=(n_1//b +1, h, b, final_matrix,))
        else:
            thread = threading.Thread(name='worker',
                                      target=multiplicationOfCubeThreading,
                                      args=(n_1//b, h, b, final_matrix,))
        thread.start()

    t_start = time.time()
    while any(th.is_alive()
```

```
            for th in threading.enumerate()
            if th.name == 'worker'):
        continue

    tm = round(time.time() - t_start, 2)
    print(f'{b} thread время: {tm}')
    print(final_matrix)


if __name__ == '__main__':
    print("1-я матрица:")
    print(matrix_1)
    print("2-я матрица:")
    print(matrix_2)
    matrix_2 = matrix_2.T

    final_matrix = np.array(
    [[random.randint(-1000, 1000) for _ in range(m_2)] for _ in range(n_1)])

    print("заготовка финальной матрицы:")
    print(final_matrix)

    t_start = time.time()

    multiplicationOfCube(n_1, final_matrix)

    tm = round(time.time() - t_start, 2)

    print(f'1 thread время: {tm}')
    print(final_matrix)

    NThreadings(1)
    NThreadings(2)
```

NThreadings(4)
NThreadings(6)
NThreadings(8)
NThreadings(10)

4 Времена работы программы

Потоки	Время
1(без использования потоков)	76.43 с
1(с использованием потоков)	380.94 с
2	295.35 с
4	251.84 с
6	285.46 с
8	341.22 с
10	370.03 с

5 Заключение

В данной работе я изучил возможности языка python в работе с библиотекой threading, а именно научился распределять вычисления на потоки.

6 Результат запуска

```
[[ -1424618  -4407811   1275165 ...  -5148783  -8766189 -11676610]
 [  -612326  10094134  -2961016 ...   1360867   5297354 -10763282]
 [ -3386202   7647572  -9473926 ...   -758703  12420372 -12669343]
 ...
 [  1887726   5145427  -4251790 ...   4197329   1997225  -8007975]
 [ -2721980  -6574146  -3202875 ...  -6496643   6963519  -4742981]
 [ -5018737 -13749236  -7157223 ...  -2511099   -770869  -2213711]]
1-я матрица:
[[ 915  784 -376 ... -368 -670  955]
 [-688 -404  769 ...  853  498  652]
 [-358  465  151 ...  388  443  645]
 ...
 [-497  283  324 ... -423  130 -339]
 [ 919 -757   94 ... -574 -302 -998]
 [ -79  758  971 ... -494  102  866]]
2-я матрица:
[[-807  -57  420 ... -711 -993 -327]
 [-312 -422  283 ...  496 -409 -198]
 [  -1    9 -819 ...  509  301  569]
 ...
 [-81  481  -74 ...  892 -557 -839]
 [ 838  -78 -127 ... -732   -9  791]
 [-717   40 -473 ...  -74 -274 -302]]
заготовка финальной матрицы:
[[ 927 -855  702 ...  185 -241 -833]
 [ 764  567  504 ... -512 -575  201]
 [ 974  489 -580 ...  373  931  560]
 ...
 [-606  977 -343 ... -139  135 -730]
 [-988  522 -127 ...  269   -1 -580]
 [ 665 -909  -55 ...   83  644   34]]
1 thread время: 76.43
[[ -1424618  -4407811   1275165 ...  -5148783  -8766189 -11676610]
 [  -612326  10094134  -2961016 ...   1360867   5297354 -10763282]
 [ -3386202   7647572  -9473926 ...   -758703  12420372 -12669343]
 ...
 [  1887726   5145427  -4251790 ...   4197329   1997225  -8007975]
 [ -2721980  -6574146  -3202875 ...  -6496643   6963519  -4742981]
 [ -5018737 -13749236  -7157223 ...  -2511099   -770869  -2213711]]
1 thread время: 380.94
[[ -1424618  -4407811   1275165 ...  -5148783  -8766189 -11676610]
 [  -612326  10094134  -2961016 ...   1360867   5297354 -10763282]
 [ -3386202   7647572  -9473926 ...   -758703  12420372 -12669343]
 ...
 [  1887726   5145427  -4251790 ...   4197329   1997225  -8007975]
 [ -2721980  -6574146  -3202875 ...  -6496643   6963519  -4742981]
 [ -5018737 -13749236  -7157223 ...  -2511099   -770869  -2213711]]
2 thread время: 295.35
[[ -1424618  -4407811   1275165 ...  -5148783  -8766189 -11676610]
 [  -612326  10094134  -2961016 ...   1360867   5297354 -10763282]
```

```
2 thread время: 295.35
[[ -1424618  -4407811   1275165 ...  -5148783  -8766189 -11676610]
 [  -612326  10094134  -2961016 ...   1360867   5297354 -10763282]
 [ -3386202   7647572  -9473926 ...   -758703  12420372 -12669343]
 ...
 [  1887726   5145427  -4251790 ...   4197329   1997225  -8007975]
 [ -2721980  -6574146  -3202875 ...  -6496643   6963519  -4742981]
 [ -5018737 -13749236  -7157223 ...  -2511099   -770869  -2213711]]
4 thread время: 251.84
[[ -1424618  -4407811   1275165 ...  -5148783  -8766189 -11676610]
 [  -612326  10094134  -2961016 ...   1360867   5297354 -10763282]
 [ -3386202   7647572  -9473926 ...   -758703  12420372 -12669343]
 ...
 [  1887726   5145427  -4251790 ...   4197329   1997225  -8007975]
 [ -2721980  -6574146  -3202875 ...  -6496643   6963519  -4742981]
 [ -5018737 -13749236  -7157223 ...  -2511099   -770869  -2213711]]
6 thread время: 285.46
[[ -1424618  -4407811   1275165 ...  -5148783  -8766189 -11676610]
 [  -612326  10094134  -2961016 ...   1360867   5297354 -10763282]
 [ -3386202   7647572  -9473926 ...   -758703  12420372 -12669343]
 ...
 [  1887726   5145427  -4251790 ...   4197329   1997225  -8007975]
 [ -2721980  -6574146  -3202875 ...  -6496643   6963519  -4742981]
 [ -5018737 -13749236  -7157223 ...  -2511099   -770869  -2213711]]
8 thread время: 341.22
[[ -1424618  -4407811   1275165 ...  -5148783  -8766189 -11676610]
 [  -612326  10094134  -2961016 ...   1360867   5297354 -10763282]
 [ -3386202   7647572  -9473926 ...   -758703  12420372 -12669343]
 ...
 [  1887726   5145427  -4251790 ...   4197329   1997225  -8007975]
 [ -2721980  -6574146  -3202875 ...  -6496643   6963519  -4742981]
 [ -5018737 -13749236  -7157223 ...  -2511099   -770869  -2213711]]
10 thread время: 370.03
[[ -1424618  -4407811   1275165 ...  -5148783  -8766189 -11676610]
 [  -612326  10094134  -2961016 ...   1360867   5297354 -10763282]
 [ -3386202   7647572  -9473926 ...   -758703  12420372 -12669343]
 ...
 [  1887726   5145427  -4251790 ...   4197329   1997225  -8007975]
 [ -2721980  -6574146  -3202875 ...  -6496643   6963519  -4742981]
 [ -5018737 -13749236  -7157223 ...  -2511099   -770869  -2213711]]
```