**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»

Отчёт о лабораторной работе № 2 по курсу «Разработка параллельных и распределенных программ»

Параллельная реализация решения системы линейных алгебраических

уравнений с помощью MPI

Студент: Жовтяк Я.Е. Группа: ИУ9-51Б

Преподаватель: Царев А.С.

Москва, 2022

**Содержание**

Постановка задачи 2

Практическая реализация 2

Результаты 9

Выводы 10

**Постановка задачи**

Пусть есть система из N линейных алгебраических уравнений в виде Ax=b,

где А – матрица коэффициентов уравнений размером N×N, b – вектор правых

частей размером N, x – искомый вектор решений размером N. Решение

системы уравнений итерационным методом состоит в выполнении

следующих шагов.

1. Задается x 0 – произвольное начальное приближение решения (вектор с

произвольными начальными значениями).

2. Приближение многократно улучшается с использованием формулы вида

x n+1 = f(x n ), где функция f определяется используемым методом 1 .

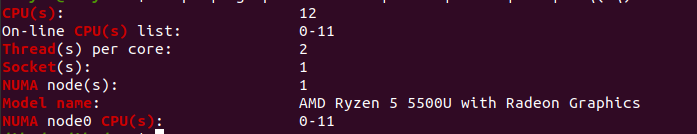
3. Процесс продолжается, пока не выполнится условие g(x n ) < ε, где функция

g определяется используемым методом, а величина ε задает требуемую

точность.

**Практическая реализация**

Лабораторная работа выполнялась на ноутбуке с установленной ОС Ubuntu 20.04.4 LTS, процессор имеет характеристики, указанные на скриншоте:



Для реализации поставленной задачи использовался язык программирования Python.

Была создана программа, запускаемая в консоли при помощи bash-скрипта, который запускает программу на разном количестве потоков и замеряет время выполнения.

Ниже приведен листинг кода программы:

from mpi4py import MPI

import numpy as np

from math import sin, pi

def main():

comm = MPI.COMM\_WORLD

rank = comm.Get\_rank()

size = comm.Get\_size()

N = 1024

epsilon = 0.00001

tau = 0.1 / N

if rank == 0:

A = np.empty(shape=(N, N), dtype='d')

u = np.empty(shape=(N, 1), dtype='d')

x = np.zeros(shape=(N, 1), dtype='d')

b = np.empty(shape=(N, 1), dtype='d')

for i in range(N):

for j in range(N):

if i == j:

A[i][j] = 2.0

else:

A[i][j] = 1.0

for i in range(N):

u[i] = sin((2 \* pi \* i) / N)

b = A @ u

else:

b = None

x = None

A = None

u = None

b = comm.bcast(b, root=0)

x = comm.bcast(x, root=0)

part = np.empty(shape=(N // size, N), dtype='d')

comm.Scatter(A, part, root=0)

start\_time = MPI.Wtime()

while True:

part\_y = part @ x

y = None

if rank == 0:

y = np.empty(shape=(N, 1), dtype='d')

comm.Gather(part\_y, y, root=0)

y = comm.bcast(y, root=0)

y = y - b

if (np.linalg.norm(y) / np.linalg.norm(b)) < epsilon:

break

if rank == 0:

x = x - tau \* y

x = comm.bcast(x, root=0)

finish\_time = MPI.Wtime()

print(finish\_time - start\_time)

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

main()

**Результаты**

В ходе выполнения лабораторной работы были изучены способы распараллеливания вычислений с помощью потоков. Для каждой конфигурации исходных данных проводилось измерение затраченного время на вычисление. Результаты измерений прилагаются в виде таблицы и графика (указывается время в секундах).

|  |  |
| --- | --- |
| Количество потоков | Время |
| 2 | 60.2 |
| 4 | 52.3 |
| 8 | 36.3 |
| 12 | 22.2 |

**Выводы**

Из приведённых данных видно, что использование потоков для вычисления произведения матриц позволило сократить время, затрачиваемое на вычисление. Также наглядно демонстрируется влияние количества потоков, на которых исполняется программа, на время выполнения.