



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
распределённых программ»

Параллельная реализация решения системы линейных
алгебраических
уравнений с помощью MPI

Студент: Белецкий В. С.

Группа: ИУ9-51Б

Преподаватель: Царев А.С.

Москва, 2022

Содержание

Постановка задачи	2
Практическая реализация	2
Результаты	4
Выводы	5

Постановка задачи

Пусть есть система из N линейных алгебраических уравнений в виде $Ax=b$, где A – матрица коэффициентов уравнений размером $N \times N$, 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 $g(x_n) < \epsilon$, где функция g определяется используемым методом, а величина ϵ задает требуемую точность.

Практическая реализация

Лабораторная работа выполнялась на ноутбуке с установленной ОС Windows 11, процессор имеет характеристики, указанные на скриншоте:

Процессор	Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz, 1801 МГц, ядер: 4, логических процессоров: 8
-----------	--

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.

Код программы :

```
from mpi4py import MPI
import numpy as np
from math import sin, pi
```

```

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()

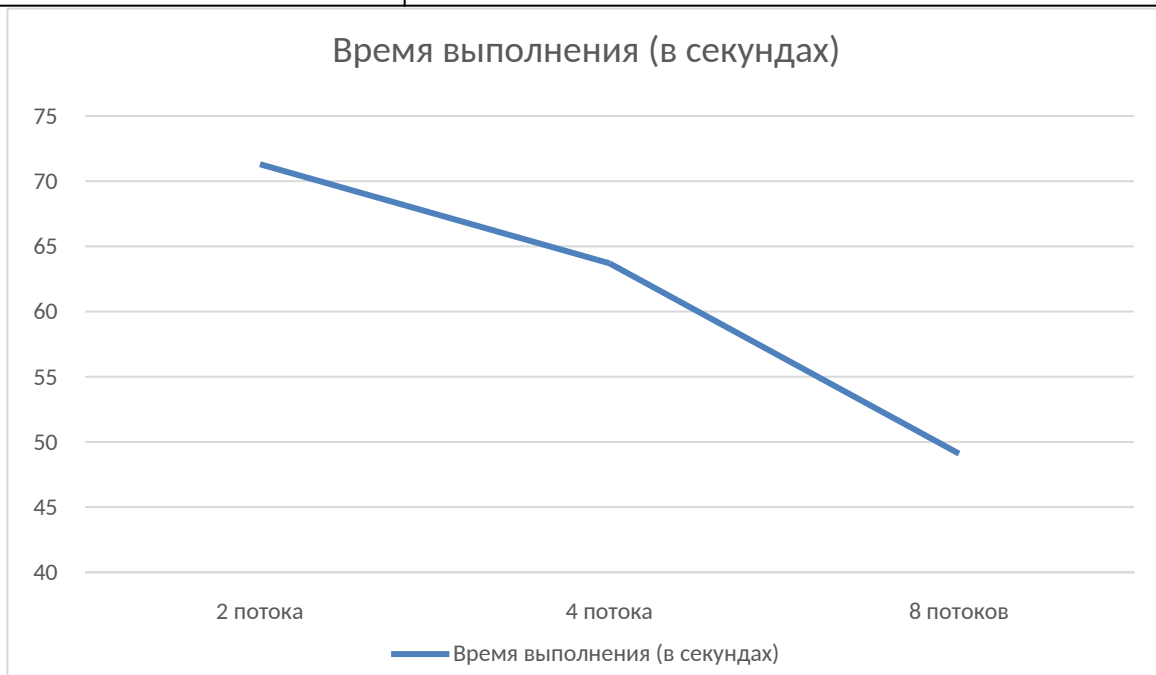
print(finish_time - start_time)

```

Результаты

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

Количество потоков	Время
2	71.3
4	60.7
8	49.1



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.