МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ
УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**
(БГТУ им. В.Г. Шухова)

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ

Лабораторная работа №3

по дисциплине: Параллельное программирование
тема: «Гибридное параллельное программирование с использованием OpenMP + MPI»

Выполнил: ст. группы ПВ-223
Пахомов Владислав Андреевич

Проверили:
доц. Островский Алексей Мичеславович

Белгород 2025 г.

Цель работы: Изучить возможности гибридного подхода к параллельному программированию с использованием стандартов MPI и OpenMP, реализовать смешанную модель параллельных вычислений, оценить её эффективность по сравнению с отдельными технологиями, изучить механизмы межпроцессного взаимодействия (MPI) и многопоточного параллелизма (OpenMP).

Условие индивидуального задания:

Гибридная генерация и проверка простых чисел (решето Эратосфена). Реализовать гибридную версию алгоритма решета Эратосфена для поиска всех простых чисел до N. Разделить диапазон между процессами MPI. Внутри каждого процесса параллелизовать вычёркивание кратных с помощью OpenMP. После завершения — объединить частичные результаты. Провести измерение времени при разных N и конфигурациях потоков/процессов.

Ход выполнения работы

При решении задачи недостаточно использовать классическую схему решения решета Эратосфена. Для решения её в парадигме параллельного программирования необходимо использовать другой подход - сегментированное решето Эратосфена. Решение основывается на том, что для вычисления отрезка решета Эратосфена до N достаточно вычислить решето Эратосфена классическим способом только до $\sqrt{N}$. В однопоточном программировании такой подход даст улучшение относительно памяти, так как нам больше не нужно хранить весь массив.

Для параллельной задачи мы можем синхронно подготовить решето до $\sqrt{N}$ после чего разбить оставшийся массив и разделить между процессами MPI. При помощи OMP будем проставлять флаги того, что число является непростым.

Исходный код:

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <mpi.h>
#include <omp.h>
#include <math.h>

#define N 80000000

void classic_sieve(int *sieve, int sieve_size)
{
    for (int i = 2; i < sieve_size; i++)
    {
        if (sieve[i])
            continue;
        for (int j = 2 * i; j < N; j += i)
            sieve[j] = true;
    }
}

int main(int argc, char *argv[])
{
    // Инициализировать MPI, получить ранг
    // и количество процессов MPI
    int rank, size;
```

```
    // Просчёт локального решета Эратосфена
    int *local_sieve = (int *)calloc(N, sizeof(int));
    // Просчитываем первые sqrt(N) чисел
    classic_sieve(local_sieve, sqrt(N));

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    // Определяем границы для которых будет вычисляться локальное решето
    int left = sqrt(N) + ((N - sqrt(N)) / size) * rank;
    int right = rank == size - 1 ? N : (sqrt(N) + ((N - sqrt(N)) / size) * (rank + 1));

    // Проходим по всем простым числам до sqrt(N)
    for (int i = 2; i < sqrt(N); i++)
    {
        if (local_sieve[i])
            continue;

        // Находим первое число кратное i в рамках определённых границ
        int j = left / i * i;
        if (j < left)
        {
            j += i;
        }

// Устанавливаем кратные числа
#pragma omp parallel for schedule(guided)
        for (int jj = j; jj < right; jj += i)
        {
            local_sieve[jj] = true;
        }
    }

    int *global_sieve = (int *)calloc(N, sizeof(int));

    MPI_Reduce(
        local_sieve,
        global_sieve,
        N,
        MPI_INT,
        MPI_LOR,
        0,
        MPI_COMM_WORLD);

    // Очистить ресурсы MPI
    MPI_Finalize();
    return 0;
}
```

Результаты вычислений по времени:

		Количество потоков			
		1	2	3	4
Количество процессов	1	1,842	1,662	1,581	1,537
	2	2,103	1,998	2,009	2,01
	3	2,526	2,492	2,476	2,527
	4	2,772	2,791	2,765	2,761

Вывод: в ходе лабораторной работы изучили возможности гибридного подхода к параллельному программированию с использованием стандартов MPI и OpenMP, реализовать смешанную модель параллельных вычислений, оценить её эффективность по сравнению с отдельными технологиями, изучить механизмы межпроцессного взаимодействия (MPI) и многопоточного параллелизма (OpenMP).

С увеличением количества потоков уменьшалось и время работы, для генерации решета Эратосфена была использована стратегия static, так как присвоение - равномерная нагрузка.

Однако с увеличением процессов время выполнения работы программы увеличивалось, возможно это связано с тем что N недостаточно большой, и время по организации процессов перевешивает время вычислений.