Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего профессионального образования

**«Национальный исследовательский ядерный университет «МИФИ»**

**ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ**

**Кафедра №42 (криптологии и кибербезопасности)**

ЛАБОРАТОРНАЯ РАБОТА №7  
по дисциплине: «Параллельное программирование»  
на тему: «Совместное применение OpenMP и MPI»

**Выполнил:**студент группы Б19-515  
Родионов Дмитрий Александрович

**Москва, 2021**

Оглавление

[Используемая рабочая среда 3](#_Toc91279102)

[Ход работы 4](#_Toc91279103)

[Результаты вычислительного эксперимента 6](#_Toc91279104)

[Заключение 7](#_Toc91279105)

# Используемая рабочая среда

**Процессор:** Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59 GHz

**Оперативная память:** 16.0 ГБ

**Тип системы:** 64-разрядная операционная система, процессор x64

**Версия:** 21H1

**Среда разработки:** WSL2 Ubuntu 20.04 LTS

**Набор компиляторов:** MPICH

# Ход работы

**Структура программы:**

1) инициализация MPI;

2) генерация массива натуральных чисел из главного потока;

3) рассылка определенных фрагментов массива от главного процесса всем остальным процессам;

4) перебор потенциальных делителей и нахождение остатка от деления данного числа на каждый из них;

5) сбор результатов перебора в главном потоке;

6) вывод проверенного диапазона и всех найденных в нем простых чисел;

7) завершение MPI.

**Назначение используемых функций MPI:**

1) *MPI\_Init* – в результате выполнения этой функции создается группа процессов, в которую помещаются все процессы приложения, и создается область связи, описываемая предопределенным коммуникатором *MPI\_COMM\_WORLD*;

2) *MPI\_COMM\_size* – функция возвращает количество процессов в области связи коммуникатора *MPI\_COMM\_WORLD*;

3) *MPI\_COMM\_rank* – функция возвращает номер процесса, вызвавшего эту функцию;

4) *MPI\_Wtime* – функция возвращает астрономическое время в секундах, прошедшее с некоторого момента в прошлом (точки отсчета);

5) *MPI\_Send* – функция производит блокирующую посылку массива в коммуникаторе *MPI\_COMM\_WORLD*; все элементы посылаемого сообщения должны быть расположены подряд в исходном буфере;

6) *MPI\_Recv* – функция производит блокирующий прием массива в коммуникаторе *MPI\_COMM\_WORLD* с заполнением массива атрибутов приходящего сообщения *STATUS*;

7) *MPI\_Finalize* – функция закрывает все MPI-процессы и ликвидирует все области связи.

**Назначение используемых директив OpenMP и их опций:**

1) *parallel* – задаёт параллельную область программы:

1. *num\_threads(threads)* – явное задание количества нитей, которые будут выполнять параллельную область;
2. *shared(i, length, threads)* – задает список переменных, общих для всех нитей;
3. *default(none)* – всем переменным в параллельной области класс должен быть назначен явно.

2) *for* – распределение итераций цикла между различными нитями: относится к идущему следом блоку, включающему оператор *for*.

# Результаты вычислительного эксперимента

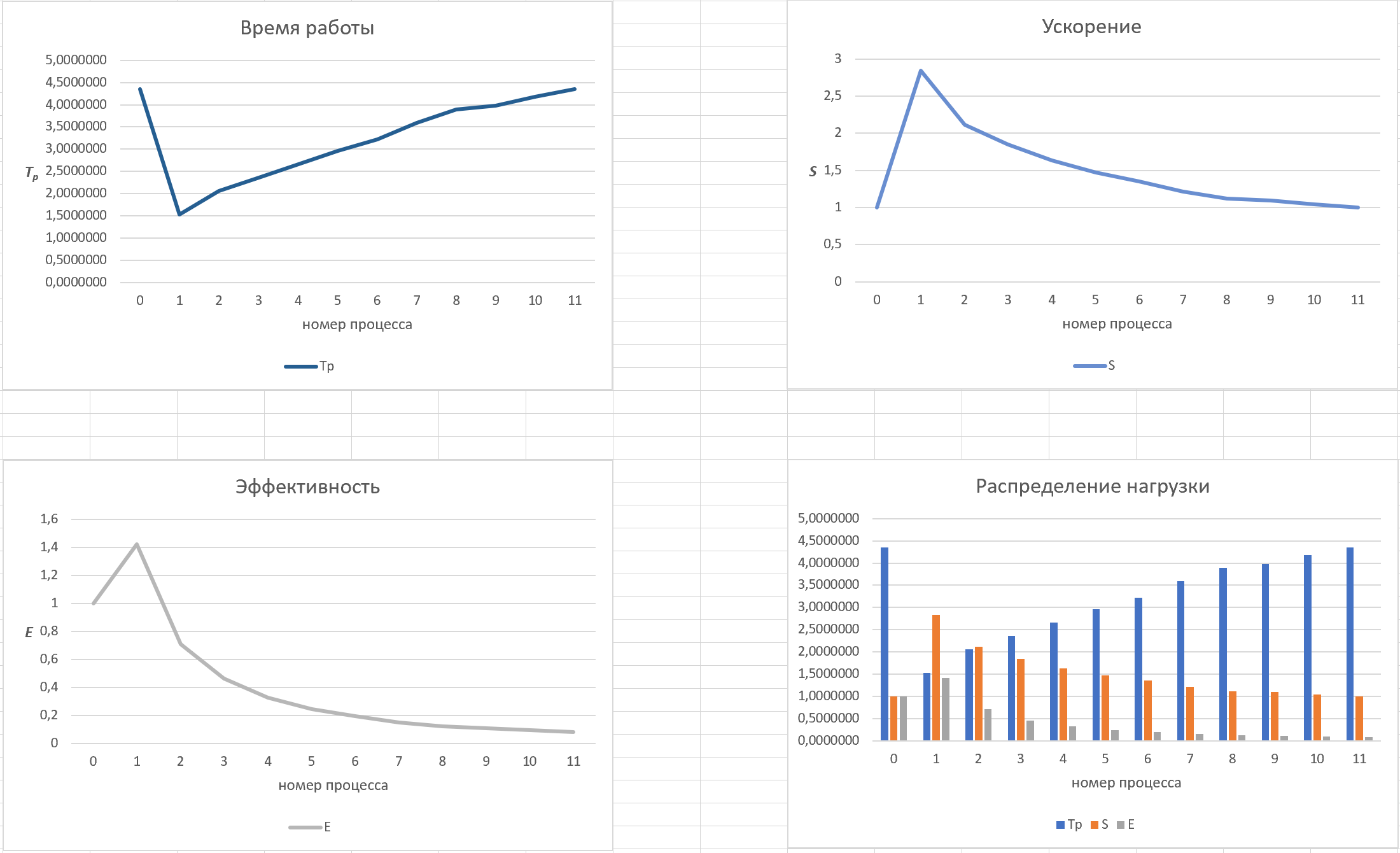
На кластере мультипроцессоров (или многоядерных процессоров) мы можем выполнить параллельную программу, имея процесс MPI, выполняющийся в каждом процессоре (или ядре). Может быть и так, что несколько процессов MPI выполняются в одном мультипроцессоре (или многоядерном процессоре), но все равно взаимодействие между этими процессами основано на передаче сообщений.

Другой подход заключается в разработке гибридной программы, в которой только один процесс MPI выполняется в каждом мультипроцессоре (или многоядерном процессоре), а затем запускается набор потоков, равный количеству процессоров (или ядер) в каждой машине, для выполнения параллельных областей программы.

В качестве альтернативы можно комбинировать обе стратегии и адаптировать разделение между MPI-процессами и потоками таким образом, чтобы оптимизировать использование доступных ресурсов без нарушения возможных ограничений или требований решаемой задачи.

При разработке алгоритма поиска простых чисел в заданном диапазоне данный метод существенно сокращает среднее время работы программы без потери ее производительности и масштабируемости.

Графики времени работы, ускорения и эффективности алгоритма, а также диаграмма распределения вычислительной нагрузки, соответствующие результатам вычислительного эксперимента, приведены в приложении 1:



# Заключение

В данной лабораторной работе проведен теоретический и экспериментальный анализ параллельного алгоритма, реализованного с использованием комбинации стандартов параллельного программирования MPI и OpenMP. Для четкого обоснования результатов проведенного вычислительного эксперимента описаны подходы к применению используемых средств гибридного программирования MPI-OpenMP. Для достижения поставленной цели выполнено подробное моделирование написанной программы с применением базовых способов математического описания алгоритма. Также в ходе выполнения работы использовалась одна из основных теорем арифметики: теорема о делителе составного числа.

Цель, поставленная в начале работы, достигнута; задачи выполнены.