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 №5  
по дисциплине: «Параллельное программирование»  
на тему: «Технология MPI. Введение»

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

**Москва, 2021**

Оглавление

[Используемая рабочая среда 3](#_Toc90419291)

[Ход работы 4](#_Toc90419292)

[Результаты вычислительного эксперимента 5](#_Toc90419293)

[Заключение 6](#_Toc90419294)

# 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 из главного потока;

3) рассылка массива от главного процесса всем остальным процессам;

4) определение локального максимума внутри каждого потока;

5) определение максимального элемента массива с помощью операции редукции;

6) 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\_Bcast* – выполнение широковещательной рассылки данных; главный процесс рассылает сообщение из своего буфера передачи всем процессам области связи коммуникатора *MPI\_COMM\_WORLD*;

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

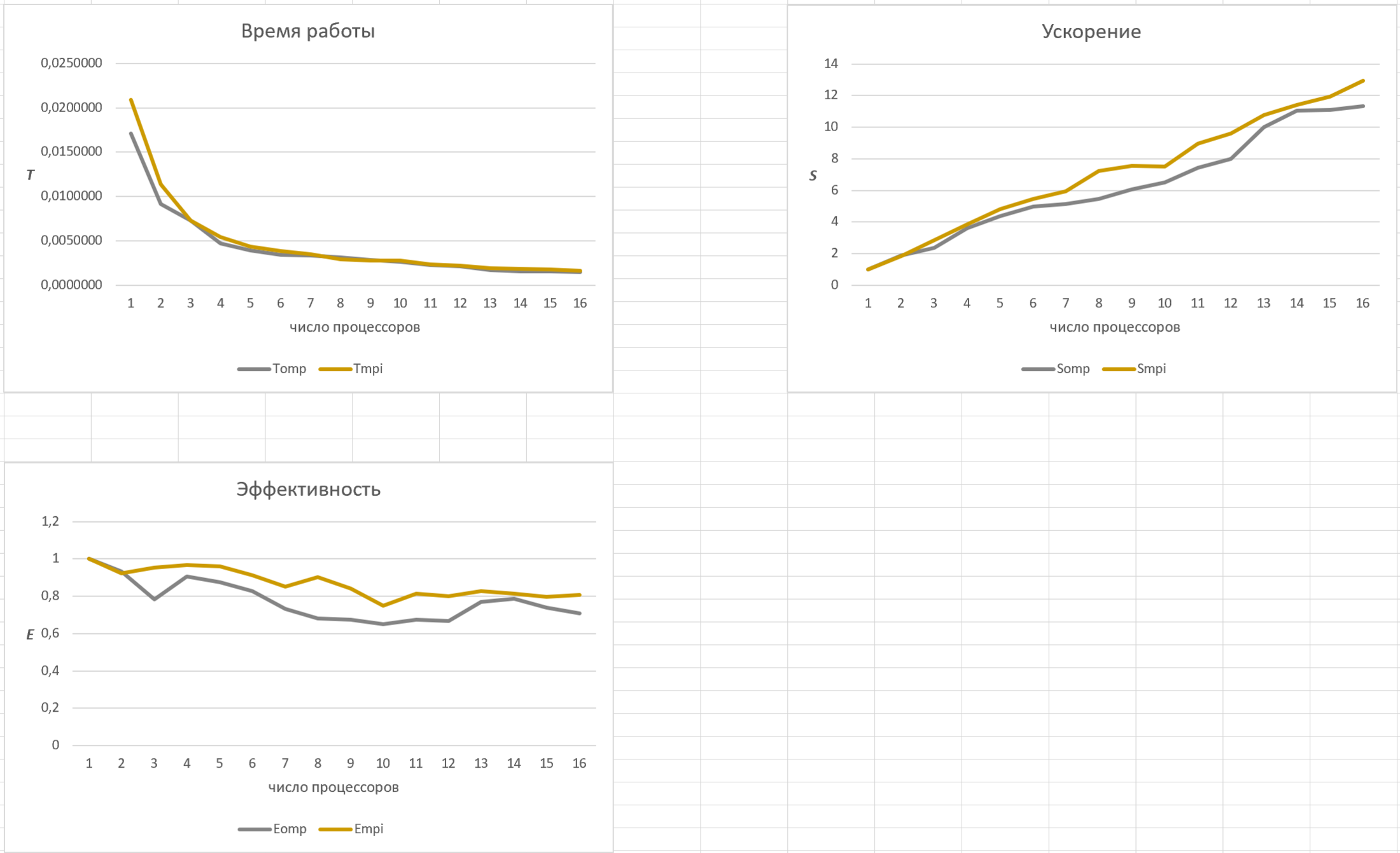
6) *MPI\_Reduce* – выполнение глобальной операции редукции с сохранением результата в адресном пространстве главного процесса;

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

# Результаты вычислительного эксперимента

При решении задачи нахождения максимального элемента массива модель передачи сообщений, предоставляемая интерфейсом параллельного программирования MPI, является менее эффективной по сравнению с моделью параллельных вычислений, предоставляемой стандартом параллельного программирования OpenMP. За счет идеи «инкрементального распараллеливания» OpenMP идеально подходит для разработчиков, желающих быстро распараллелить свои вычислительные программы с большими параллельными циклами. MPI, в свою очередь, не предоставляет механизмов задания начального размещения данных по процессорам. Более того, не существует реализаций MPI, в полной мере обеспечивающих совмещение обменов с вычислениями, вследствие чего процедура пересылки данных между ветвями значительно увеличивает время работы алгоритма.

Графики времени работы, ускорения и эффективности алгоритма, соответствующие результатам проведенного вычислительного эксперимента, приведены в приложении 1:



# Заключение

Интерфейс OpenMP задуман как стандарт для программирования в модели общей памяти. Это означает, что параллелизм возникает там, где каждый параллельный поток имеет доступ ко всем данным приложения. Технология MPI, в свою очередь, обеспечивает механизм программирования на устройствах с распределённой памятью. Это означает, что параллелизм возникает там, где каждый параллельный процесс работает в своём собственном пространстве памяти изолированно от других процессов. В данном контексте каждый написанный фрагмент кода выполняется каждым процессом независимо. Параллелизм возникает потому, что разработчик точно указывает каждому процессу, над какой частью глобальной программы он должен работать, полностью основываясь на своём идентификаторе процесса.

В данной лабораторной работе проведён теоретический и экспериментальный анализ параллельного алгоритма, реализованного с использованием стандарта параллельных вычислений MPI. Для качественного сравнения принципов работы используемых технологий параллельного программирования приведено чёткое обоснование результатов проведённого вычислительного эксперимента. Для достижения поставленной цели выполнено подробное моделирование написанной программы с применением базовых способов математического описания алгоритма. Также в ходе выполнения работы использовались средства автоматизации выполнения программ в операционной системе Linux Ubuntu 20.04 LTS.

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