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 №6  
по дисциплине: «Параллельное программирование»  
на тему: «Коллективные операции в MPI»

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

**Москва, 2021**

Оглавление

[Используемая рабочая среда 3](#_Toc91331224)

[Ход работы 4](#_Toc91331225)

[Результаты вычислительного эксперимента 5](#_Toc91331226)

[Заключение 6](#_Toc91331227)

# 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 определенных фрагментов массива от главного процесса всем остальным процессам;

4) сортировка определенного фрагмента массива внутри каждого потока;

5) сбор результатов сортировки в главном потоке;

6) завершение MPI.

**Оценка временной сложности алгоритма:**

# Результаты вычислительного эксперимента

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 распределенной памятью. Это означает, что параллелизм возникает там, где каждый параллельный процесс работает в своем 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 своем идентификаторе процесса. Стоит отметить, что не существует реализаций MPI, в полной мере обеспечивающих совмещение обменов с вычислениями, вследствие чего процедура пересылки данных между ветвями значительно увеличивает время работы программы. Таким образом, при реализации параллельного алгоритма сортировки Шелла модель передачи сообщений, предоставляемая интерфейсом параллельного программирования MPI, является менее эффективной по сравнению с моделью параллельных вычислений, предоставляемой стандартом параллельного программирования OpenMP.

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



# Заключение

В данной лабораторной работе проведен теоретический и экспериментальный анализ параллельного алгоритма, реализованного с использованием стандарта параллельных вычислений MPI. Для четкого обоснования результатов проведенного вычислительного эксперимента дополнительно описаны принципы работы используемых технологий параллельного программирования. 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.