Федеральное государственное автономное образовательное учреждение высшего образования Санкт-Петербургский политехнический университет Петра Великого

Физико-механический институт

Высшая школа прикладной математики и вычислительной физики

Лабораторная работа №2

**Освоение процедур приема-передачи данных между параллельно исполняющимися процессами в MPI**

Выполнил:

Кучиев Д.Ю.

Преподаватель:

Абрамов А.Г.

Санкт-Петербург

2025

**Цель работы**

Освоить принципы работы процедур приема-передачи сообщений в MPI, реализующими двухточечные обмены (с блокировкой и без блокировки) и коллективные обмены данными. В данной работе рассматривается использование различных процедур приема-передачи данных для дальнейшего использования их в рабочей практике.

Программные коды с комментариями представлены в репозитории: https://github.com/MrDionisio/MPI\_CC

**Часть 1: *Неблокирующий обмен сообщениями между всеми процессами в соответствии с топологией кольца***

Написать программу, в которой все исполняющие ее процессы при помощи неблокирующих коммуникационных операций (процедуры MPI\_Isend()/MPI\_Irecv()) однократно обмениваются сообщениями с ближайшими соседями (по номерам процессов) в соответствии с топологией кольца. Напечатать значения передаваемых переменных (например, номера процесса отправителя) после получения сообщений. Реализовать проверку состояния приема с помощью процедур MPI\_Wait() и MPI\_Test().

**MPI\_Isend** (Неблокирующая отправка). Инициирует отправку данных без блокировки программы. Возвращает управление сразу после начала операции. Состояние отправки отслеживается с помощью запроса (MPI\_Request).

**MPI\_Irecv** (Неблокирующий приём). Инициирует приём данных без блокировки программы. Возвращает управление сразу после начала операции. Состояние приёма отслеживается с помощью запроса (MPI\_Request).

**MPI\_Test** (Проверка завершения операции). Проверяет, завершена ли операция, связанная с запросом. Если операция завершена, возвращает flag = 1, иначе flag = 0.

**MPI\_Wait** (Ожидание завершения операции). Блокирует выполнение программы до завершения операции, связанной с запросом.

Результаты работы программы:



Рис. 1 Пример выполнения программы

**Часть 2: *Рассылка данных при помощи коллективных коммуникационных операций***

Написать программу, в которой нулевой процесс инициализирует двумерный массив (матрицу) вещественных чисел и рассылает ее, разбивая последовательно на столбцы, другим процессам приложения (использовать процедуру MPI\_Scatter()). После рассылки в процессах производится однотипное изменение элементов массива (например, умножение на 2). Напечатать исходный и модифицированный массив данных.

**MPI\_Scatter (Рассылка данных).** Разделяет данные из массива на нулевом процессе и равномерно распределяет их между всеми процессами. Каждый процесс получает часть данных.

**MPI\_Gather (Сбор данных).** Собирает данные от всех процессов в массив на нулевом процессе. Каждый процесс отправляет свою часть данных.

Результаты работы программы:

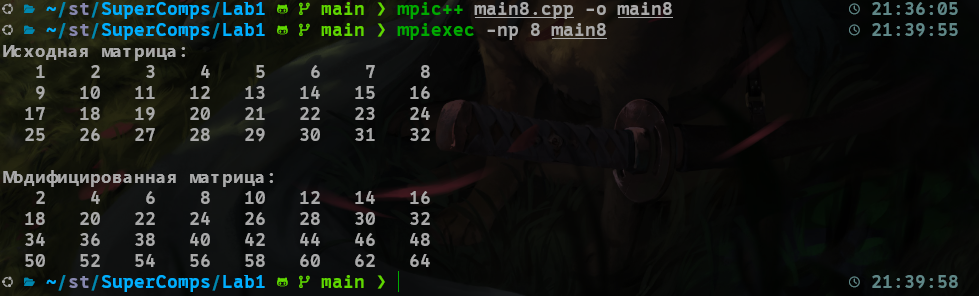


Рис. 2 Пример выполнения программы №2

**Вывод:**

Неблокирующие операции (MPI\_Isend, MPI\_Irecv) полезны для повышения производительности, так как позволяют перекрывать вычисления и коммуникации.

Коллективные операции (MPI\_Scatter, MPI\_Gather) упрощают распределение и сбор данных между процессами, что особенно полезно для задач, связанных с обработкой матриц или больших массивов данных.

Правильное управление памятью (например, выделение и освобождение памяти только на нулевом процессе) важно для избежания ошибок, таких как утечки памяти или повреждение данных.

Проверка состояния операций с помощью MPI\_Test и MPI\_Wait обеспечивает корректное завершение неблокирующих операций.

Выполненные задания продемонстрировали ключевые аспекты работы с MPI:

* Использование неблокирующих операций для эффективного обмена данными.
* Применение коллективных операций для распределения и сбора данных.
* Важность правильного управления памятью и синхронизации процессов.
* Эти навыки необходимы для разработки высокопроизводительных параллельных приложений, работающих на кластерах и суперкомпьютерах.