**Лабораторная работа №3: Распределение и сбор данных в MPI. Использование MPI\_Scatter, MPI\_Gather и MPI\_Allreduce**

**Теоретическая часть**

Эта лабораторная работа направлена на изучение нескольких новых коллективных операций MPI:

1. **MPI\_Scatter** — операция, которая позволяет одному процессу (обычно процессу с рангом 0) разделить массив данных на части и отправить каждый сегмент данных отдельному процессу. Например, если у нас есть массив из 16 элементов и 4 процесса, каждый процесс получит по 4 элемента. MPI\_Scatter полезен для распределения вычислительных задач и данных по нескольким процессам.
2. **MPI\_Gather** — операция, которая позволяет собирать данные от всех процессов и объединять их в один массив на одном процессе (обычно процесс 0). Каждый процесс отправляет свой сегмент данных, и MPI\_Gather объединяет их в один общий массив.
3. **MPI\_Allreduce** — это коллективная операция, объединяющая данные от всех процессов и передающая результат каждому процессу. Основное отличие MPI\_Allreduce от MPI\_Reduce в том, что результат передается не одному, а всем процессам. В данной работе мы используем эту операцию для получения общей суммы элементов, обработанных каждым процессом.

**Пример использования коллективных операций**

Предположим, у нас есть задача распределить массив данных, выполнить вычисления на каждом процессе и затем объединить результаты. Например, у нас есть массив из 16 элементов, который нужно разделить на 4 процесса. Каждый процесс обрабатывает свою часть данных (например, умножает на свой ранг), после чего результаты собираются и суммируются. Здесь полезны MPI\_Scatter для разделения данных, MPI\_Gather для сбора и MPI\_Allreduce для суммирования результатов.

**Задание**

1. **Создание данных:** Процесс 0 создает массив данных размером 4 \* N, где N — это число процессов. Это означает, что на каждый процесс будет приходиться по 4 элемента. Распределите этот массив данных с помощью MPI\_Scatter, так чтобы каждый процесс получил свой сегмент из 4 элементов.
2. **Модификация данных:** Каждый процесс изменяет свои данные, умножая элементы на свой ранг. Например, если процесс имеет ранг 1 и получает сегмент [2, 4, 6, 8], то он должен преобразовать его в [2 \* 1, 4 \* 1, 6 \* 1, 8 \* 1].
3. **Сбор данных:** Используйте MPI\_Gather, чтобы передать измененные сегменты обратно в процесс 0. Процесс 0 собирает все измененные сегменты в один массив и выводит его на экран.
4. **Сумма всех элементов:** Используйте MPI\_Allreduce, чтобы найти общую сумму всех обработанных данных. Все процессы должны вывести эту общую сумму.
5. **Ожидаемый вывод:** Каждый процесс должен вывести свою локальную сумму и глобальную сумму, а процесс 0 должен вывести исходные и собранные данные, чтобы была видна корректность выполнения каждой операции.

Запустите программу с 4 процессами. Ожидаемый вывод:

*Process 0 scatter data: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16*

*Process 0 gathered data: 0 0 0 0 5 6 7 8 18 20 22 24 39 42 45 48*

*Process 0 got total sum of all multiplied elements: 284*

*Process 1 got total sum of all multiplied elements: 284*

*Process 2 got total sum of all multiplied elements: 284*

*Process 3 got total sum of all multiplied elements: 284*