## Лабораторная работа №4
## Коллективные операции передачи данных

Функции MPI_Send и MPI_Recv, рассмотренные ранее, обеспечивают возможность выполнения парных операций передачи данных между двумя процессами параллельной программы. Для выполнения коммуникационных коллективных операций, в которых принимают участие все процессы коммуникатора, в MPI предусмотрен специальный набор функций (табл.1).

Таблица 1. Коллективные операции передачи данных

| Вид коллективной операции | Функции MPI |
|---|---|
| Передача от одного процесса всем процессам (широковещательная рассылка) | MPI_Bcast |
| Сбор и обработка данных на одном процессе от всех процессов (редукция данных) | MPI_Reduce |
| - то же с рассылкой результатов всем процессам | MPI_Allreduce<br>MPI_Reduce_scatter |
| - то же с получением частичных результатов обработки | MPI_Scan |
| Обобщенная передача от одного процесса всем процессам (распределение данных) | MPI_Scatter MPI_Scatterv |
| Обобщенная передача от всех процессов одному процессу (сбор данных) | MPI_Gather MPI_Gatherv |
| - то же с рассылкой результатов всем процессам | MPI_Allgather<br>MPI_Allgatherv |
| Обобщенная передача данных от всех процессов всем процессам | MPI_Alltoall MPI_Alltoallv |

### Передача данных от одного процесса всем процессам программы (широковещательная рассылка данных)

Достижение эффективного выполнения операции передачи данных от одного процесса всем процессам программы (широковещательная рассылка данных) может быть обеспечено при помощи функции MPI:

```
int MPI_Bcast(void *buf, int count, MPI_Datatype type, int root,
 MPI_Comm comm),
```

где

- buf, count, type — буфер памяти с отправляемым сообщением (для процесса с рангом 0) и для приема сообщений (для всех остальных процессов);
- root — ранг процесса, выполняющего рассылку данных;
- comm — коммуникатор, в рамках которого выполняется передача данных.

Функция MPI_Bcast осуществляет рассылку данных из буфера buf, содержащего count элементов типа type, с процесса, имеющего номер root, всем процессам, входящим в коммуникатор comm.

**Передача данных от всех процессов одному процессу. Операция редукции**

Операцией редукции данных называется операция обработки данных, полученных на одном процессе от всех процессов. Для наилучшего выполнения действий, связанных с редукцией данных, в MPI предусмотрена функция:

int MPI_Reduce(void *sendbuf, void *recvbuf, int count, MPI_Datatype type, MPI_Op op, int root, MPI_Comm comm),

где

- sendbuf — буфер памяти с отправляемым сообщением;
- recvbuf — буфер памяти для результирующего сообщения (только для процесса с рангом root);
- count — количество элементов в сообщениях;
- type — тип элементов сообщений;
- op — операция, которая должна быть выполнена над данными;
- root — ранг процесса, на котором должен быть получен результат;
- comm — коммуникатор, в рамках которого выполняется операция.

В качестве операций редукции данных могут быть использованы предопределенные в MPI операции – см. табл. 2.

Таблица 2. Базовые (предопределенные) типы операций MPI для функций редукции данных

| Операции | Описание |
|---|---|
| MPI_MAX | Определение максимального значения |
| MPI_MIN | Определение минимального значения |
| MPI_SUM | Определение суммы значений |
| MPI_PROD | Определение произведения значений |
| MPI_LAND | Выполнение логической операции "И" над значениями сообщений |

| | |
|---|---|
| MPI_BAND | Выполнение битовой операции "И" над значениями сообщений |
| MPI_LOR | Выполнение логической операции "ИЛИ" над значениями сообщений |
| MPI_BOR | Выполнение битовой операции "ИЛИ" над значениями сообщений |
| MPI_LXOR | Выполнение логической операции исключающего "ИЛИ" над значениями сообщений |
| MPI_BXOR | Выполнение битовой операции исключающего "ИЛИ" над значениями сообщений |
| MPI_MAXLOC | Определение максимальных значений и их индексов |
| MPI_MINLOC | Определение минимальных значений и их индексов |

Помимо данного стандартного набора операций могут быть определены и новые дополнительные операции непосредственно самим пользователем библиотеки MPI.

Таблица 3. Разрешенные сочетания операции типа операнда в операции редукции

| Операции | Допустимый тип операндов для языка С |
|---|---|
| MPI_MAX,MPI_MIN,MPI_SUM,MPI_PROD | Целый, вещественный |
| MPI_LAND, MPI_LOR, MPI_LXOR | Целый |
| MPI_BAND, MPI_BOR, MPI_BXOR | Целый, байтовый |
| MPI_MINLOC, MPI_MAXLOC | Целый, вещественный |

**Обобщенная передача данных от одного процесса всем процессам**

Обобщенная операция передачи данных от одного процесса всем процессам (распределение данных) отличается от широковещательной рассылки тем, что процесс передает процессам различающиеся данные (см. рис. 4).

Выполнение данной операции может быть обеспечено при помощи функции:

int MPI_Scatter(void *sbuf, int scount, MPI_Datatype stype, void *rbuf, int rcount, MPI_Datatype rtype, int root, MPI_Comm comm), где

- sbuf, scount, stype — параметры передаваемого сообщения (scount определяет количество элементов, передаваемых на каждый процесс);
- rbuf, rcount, rtype — параметры сообщения, принимаемого в процессах;
- root — ранг процесса, выполняющего рассылку данных;
- comm — коммуникатор, в рамках которого выполняется передача данных.

При вызове этой функции процесс с рангом root произведет передачу данных всем другим процессам в коммуникаторе. Каждому процессу будет отправлено scount элементов. Процесс с рангом 0 получит блок данных из sbuf элементов с индексами от 0 до scount-1, процессу с рангом 1 будет отправлен блок из sbuf элементов с индексами от scount до 2*scount-1 и т.д. Тем самым, общий размер отправляемого сообщения должен быть равен scount * p элементов, где p есть количество процессов в коммуникаторе comm.

Следует отметить, что поскольку функция MPI_Scatter определяет коллективную операцию, вызов этой функции при выполнении рассылки данных должен быть обеспечен на каждом процессе коммуникатора.

Отметим также, что функция MPI_Scatter передает всем процессам сообщения одинакового размера. Выполнение более общего варианта операции распределения данных, когда размеры сообщений для процессов могут быть различны, обеспечивается при помощи функции MPI_Scatterv.

**Обобщенная передача данных от всех процессов одному процессу**

Операция обобщенной передачи данных от всех процессов одному процессу (сбор данных) является двойственной к процедуре распределения данных.

Для выполнения этой операции в MPI предназначена функция:

int MPI_Gather(void *sbuf, int scount, MPI_Datatype stype, void *rbuf, int rcount, MPI_Datatype rtype, int root, MPI_Comm comm), где

- sbuf, scount, stype — параметры передаваемого сообщения;
- rbuf, rcount, rtype — параметры принимаемого сообщения;
- root — ранг процесса, выполняющего сбор данных;
- comm — коммуникатор, в рамках которого выполняется передача данных.

При выполнении функции MPI_Gather каждый процесс в коммуникаторе передает данные из буфера sbuf на процесс с рангом root. Процесс с рангомroot собирает все получаемые данные в буфере rbuf (размещение данных в буфере осуществляется в соответствии с рангами процессов – отправителей сообщений). Для того чтобы разместить все поступающие данные, размер буфера rbuf должен быть равен scount * p элементов, где ресть количество процессов в коммуникаторе comm.

Функция MPI_Gather также определяет коллективную операцию, и ее вызов при выполнении сбора данных должен быть обеспечен в каждом процессе коммуникатора.

Следует отметить, что при использовании функции MPI_Gather сборка данных осуществляется только на одном процессе. Для получения всех собираемых данных на каждом из процессов коммуникатора необходимо применять функцию сбора и рассылки:

int MPI_Allgather(void *sbuf, int scount, MPI_Datatype stype, void *rbuf, int rcount, MPI_Datatype rtype, MPI_Comm comm), где

- sbuf, scount, stype — параметры передаваемого сообщения;
- rbuf, rcount, rtype — параметры принимаемого сообщения;
- comm — коммуникатор, в рамках которого выполняется передача данных.

Выполнение общего варианта операции сбора данных, когда размеры передаваемых процессами сообщений могут быть различны, обеспечивается при помощи функций MPI_Gatherv и MPI_Allgatherv.

**Общая передача данных от всех процессов всем процессам**

Передача данных от всех процессов всем процессам является наиболее общей операцией передачи данных.

Выполнение данной операции может быть обеспечено при помощи функции:

int MPI_Alltoall(void *sbuf,int scount,MPI_Datatype stype, void *rbuf,int rcount,MPI_Datatype rtype,MPI_Comm comm), где

- sbuf, scount, stype — параметры передаваемых сообщений;
- rbuf, rcount, rtype — параметры принимаемых сообщений;
- comm — коммуникатор, в рамках которого выполняется передача данных.

При выполнении функции MPI_Alltoall каждый процесс в коммуникаторе передает данные из scount элементов каждому процессу (общий размер отправляемых сообщений в процессах должен быть равен scount * p элементов, где p есть количество процессов в коммуникаторе comm) и принимает сообщения от каждого процесса.

Вызов функции MPI_Alltoall при выполнении операции общего обмена данными должен быть выполнен в каждом процессе коммуникатора.

Вариант операции общего обмена данных, когда размеры передаваемых процессами сообщений могут быть различны, обеспечивается при помощи функций MPI_Alltoallv.

**Дополнительные операции редукции данных**

Рассмотренная ранее функция MPI_Reduce обеспечивает получение результатов редукции данных только на одном процессе. Для получения результатов редукции данных

на каждом из процессов коммуникатора необходимо использовать функцию редукции и рассылки:

int MPI_Allreduce(void *sendbuf, void *recvbuf, int count, MPI_Datatype type, MPI_Op op, MPI_Comm comm), где

- sendbuf — буфер памяти с отправляемым сообщением;
- recvbuf — буфер памяти для результирующего сообщения;
- count — количество элементов в сообщениях;
- type — тип элементов сообщений;
- op — операция, которая должна быть выполнена над данными;
- comm — коммуникатор, в рамках которого выполняется операция.

Функция MPI_Allreduce выполняет рассылку между процессами всех результатов операции редукции. Возможность управления распределением этих данных между процессами предоставляется функций MPI_Reduce_scatter.

И еще один вариант операции сбора и обработки данных, при котором обеспечивается получение всех частичных результатов редуцирования, может быть реализован при помощи функции:

int MPI_Scan(void *sendbuf, void *recvbuf, int count, MPI_Datatype type, MPI_Op op, MPI_Comm comm), где

- sendbuf — буфер памяти с отправляемым сообщением;
- recvbuf — буфер памяти для результирующего сообщения;
- count — количество элементов в сообщениях;
- type — тип элементов сообщений;
- op — операция, которая должна быть выполнена над данными;
- comm — коммуникатор, в рамках которого выполняется операция.

Элементы получаемых сообщений представляют собой результаты обработки соответствующих элементов передаваемых процессами сообщений, при этом для получения результатов на процессе с рангом i, $0 \leq i < n$, используются данные от процессов, ранг которых меньше или равен i.

## Задание на лабораторную работу

Используя коллективные операции, решите поставленную задачу. Рекомендуется использовать операции из набора: широковещательная рассылка, редукция, распределение и сборка данных. Считать, что количество данных и количество доступных

вычислительных узлов (запускаемых процессов) может быть произвольным. То есть считать, что в общем случае количество строк (столбцов, элементов и т.п.) может не делиться нацело на количество рабочих процессов, за исключением случаев, где это специально оговорено заданием. Обеспечить корректную работу программы при любых разумных размерностях данных и количествах процессов.

**Варианты заданий**

1. Дана матрица MxN, сгенерированная случайным образом, и вектор из N элементов. Считая каждую строку матрицы вектором, найти такую, которая бы имела наименьшее скалярное произведение на заданный вектор.
2. В матрице MxN определить сумму скалярных произведений всех возможных пар соседних строк.
3. Имеется вещественная матрица размера MxN. Определить номер строки, в которой необходимо произвести максимальное количество перестановок для её сортировки по возрастанию.
4. Дана матрица размером MxN, где M - чётное. Определить максимальную сумму элементов чётных строк.
5. Дан вектор натуральных чисел. Получить новый вектор, в котором элемент равен 1, если соответствующий элемент исходного вектора – простое число, и 0 в противном случае.
6. Дана матрица размером MxN. Определить строку, имеющую минимальную сумму элементов.
7. Реализовать алгоритм поиска простых чисел с равномерным распределением проверяемых чисел между имеющимися вычислительными узлами.
8. Дан вектор из N элементов. Найти его сумму с использованием модифицированной каскадной схемы и групповых операций. N задать самостоятельно.
9. Дан вектор, в котором каждая группа из 4х элементов представляет собой координаты прямоугольника на плоскости. Найти прямоугольник с наибольшим периметром.
10. Считая строки матрицы размером MxN векторами N-мерного пространства, определить, какая из них ближе всего к вектору, введенному пользователем. Используйте метрику Евклида.
11. Дан вектор размером N, где N кратно 6. Каждая группа из 6 элементов может быть рассмотрена как координаты вершин треугольника на плоскости. Среди групп, которые формируют треугольники, найти таковой с минимальной площадью.
12. Дан вектор размером N элементов, где N кратно 100. Распределить вектор блоками по 100 элементов и найти блок с минимальным значением оценки локальной дисперсии.
13. Дана матрица размером MxN. Используя коллективные операции рассчитать сумму элементов столбца, номер которого задан пользователем.
14. В матрице размером MxN, хранящей значения бит, сформировать N+1й столбец проверки на чётность.

15. К матрице размером MxN добавить N+1й столбец, содержащий суммы элементов строк. Найти строку с максимальной суммой.
16. Имеется матрица MxN, хранящая значения яркости пикселей изображения (полутоновое, 8 бит на пиксель). Выполнить линейную фильтрацию изображения фильтром, имеющим следующее ядро:

$$\begin{matrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{matrix}$$

17. Матрица размером MxN содержит пары строк. Каждая пара задаёт координаты множества точек на плоскости. Определите два множества, расстояние между центрами масс которых минимально.
18. Даны два множества точек на плоскости. Определить точку второго множества, расстояние от которой до центра масс первого минимально.
19. Дан фрагмент текста. Определить в тексте самое длинное слово.
20. В квадратной матрице выполнить циклический сдвиг элементов каждой строки таким образом, чтобы максимальный элемент строки оказался на главной диагонали.
21. Найдите в квадратной матрице размером MxM такое i, что скалярное произведение i-го столбца на i-ю строку минимально.
22. Рассматривая группы по 4 элемента вектора как координаты двух противоположных вершин прямоугольника, найти прямоугольник, центр которого расположен наиболее близко по отношению к началу координат.