Задание 17. Коллективные коммуникации: широковещательная рассылка данных

1. Изучите MPI-функцию широковещательной рассылки данных MPI_Bcast. Напишите MPI-программу, которая в строке длины n определяет количество вхождений символов. Ввод данных должен осуществляться процессом с номером 0. Для рассылки строки поиска и ее длины по процессам используйте функцию MPI_Bcast.

2*. Перепишите программу, используя вместо функции MPI_Bcast функции коммуникации «точка-точка». Сравните эффективность выполнения программ с коллективными и точечными обменами.

Входные данные: целое число n ($1 \leq n \leq 100$), строка из n символов (каждый символ в строке может представлять собой только строчную букву английского алфавита).

Выходные данные: количество вхождений всех символов, имеющихся в строке в формате «<буква> = <значение>».

Пример входных и выходных данных

Входные данные	Выходные данные
9 aaaaaaaaa	a = 9
3 rvtabc	a = 1 b = 1 c = 1 r = 1 t = 1 v = 1

Задание 18. Коллективные коммуникации: операции редукции

1. Изучите MPI-функцию для выполнения операций редукции над данными, расположенными в адресных пространствах различных процессов, MPI_Reduce. Реализуйте программу вычисления числа π (см. задание 8), используйте функцию MPI_Reduce для суммирования результатов, вычисленных каждым процессом.

2*. Перепишите программу, используя вместо функции MPI_Reduce функции коммуникации «точка-точка». Сравните эффективность выполнения программ с коллективными и точечными обменами.

Входные данные: одно целое число N (точность вычисления).

Выходные данные: одно вещественное число pi.

Пример входных и выходных данных

Входные данные	Выходные данные
1000000000	3.14159265

Задание 19. Коллективные коммуникации: функции распределения и сбора данных

1. Изучите MPI-функции распределения и сбора блоков данных по процессам MPI_Scatter и MPI_Gather. Напишите программу, которая вычисляет произведение двух квадратных матриц $A \times B = C$ размера $n \times n$. Используйте формулу, приведенную в задании 9. Ввод данных и вывод результата должны осуществляться процессом с номером 0. Для распределения матриц *A* и *B* и сбора матрицы *C* используйте функций MPI_Scatter и MPI_Gather.

2*. Перепишите программу, используя вместо функций MPI_Scatter и MPI_Gather функции коммуникации «точка-точка». Сравните эффективность выполнения программ с коллективными и точечными обменами.

Входные данные: целое число *n,* $1 \leq n \leq 10$, n^2 вещественных элементов матрицы *A* и n^2 вещественных элементов матрицы *B.*

Выходные данные: n^2 вещественных элементов матрицы *C.*

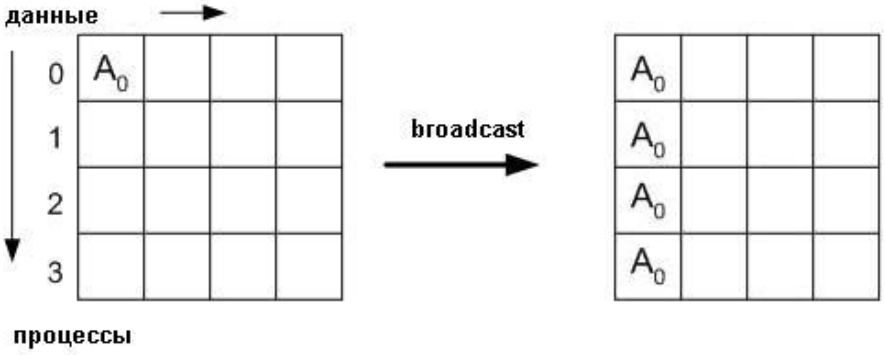
Пример входных и выходных данных

Входные данные	Выходные данные
2 1 3 4 8 5 4 3 0	14 4 44 16

Указания к заданию 17. Коллективные коммуникации: широковещательная рассылка данных

1. Создайте проект mpi_bcast в Microsoft Visual Studio 2010 с поддержкой MPI (см. указания к заданию 10).
2. Ознакомьтесь с функцией широковещательной рассылки данных MPI_Bcast:

   int MPI_Bcast (void* buffer, int count, MPI_Datatype datatype, int root, MPI_Comm comm) – процесс с номером root рассылает сообщение из своего буфера передачи всем процессам коммуникатора comm; count - число посылаемых элементов; datatype - тип посылаемых элементов.



3. Вставьте в код программы вызовы функций MPI_Init и MPI_Finalize.
4. Запишите номер каждого процесса в переменную rank.
5. Запишите количество параллельных процессов в программе в переменную size.
6. Объявите переменную buf в виде массива типа char длинной 100 элементов.
7. В процессе с номером 0 осуществите ввод числа n и строки длинны n в переменную buf.
8. Для передачи числа n в каждом процессе вызовите функцию MPI_Bcast, в качестве root укажите нулевой процесс, в качестве count – 1, в качестве datatype укажите тип данных переменной n.
9. Для передачи строки в каждом процессе вызовите функцию MPI_Bcast, в качестве root укажите нулевой процесс, в качестве count – число n, в качестве datatype укажите тип данных MPI_CHAR.

10. Определите, какой процесс, какие буквы английского алфавита будет искать в строке. Напишите цикл for, в котором каждый процесс перебирает все свои буквы. Например, следующим образом:

```
char a = 'a'; // Первая буква английского алфавита (строчная)
for (i=rank; i<26; i=i+size) {
  // Доступ к символу осуществляется через ASCII код:
  // (int)a + i
}
```

11. Затем внутри данного цикла напишите еще один цикл, который будет перебирать последовательно все символы входной последовательности длины n и сравнивать их с символом (char)((int)a + i). При совпадении необходимо увеличивать на единицу счетчик количества вхождений для каждой буквы.

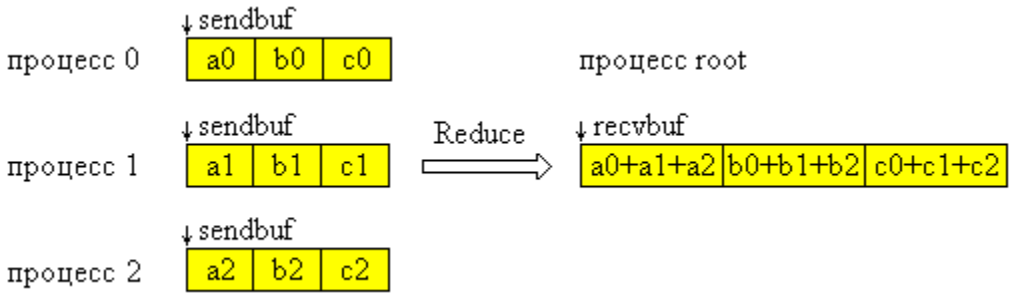
12. Для всех процессов выведите значения счетчиков букв на экран с помощью printf.

13. Скомпилируйте и запустите ваше приложение. Убедитесь, что выводится верный результат.

Указания к заданию 18. Коллективные коммуникации: операции редукции

1. Создайте проект mpi_reduce в Microsoft Visual Studio 2010 с поддержкой MPI (см. указания к заданию 10).
2. Ознакомьтесь с функцией, осуществляющей редукцию данных:

   int MPI_Reduce (void* sendbuf, void* recvbuf, int count, MPI_Datatype datatype, MPI_Op op, int root, MPI_Comm comm). Операция глобальной редукции, указанная параметром op, выполняется над первыми элементами входного буфера, и результат посылается в первый элемент буфера приема процесса root. Затем то же самое делается для вторых элементов буфера и т.д.
   MPI_Op определяет следующие основные операции: MPI_MAX (максимум), MPI_MIN (минимум), MPI_SUM (сумма), MPI_PROD (произведение).



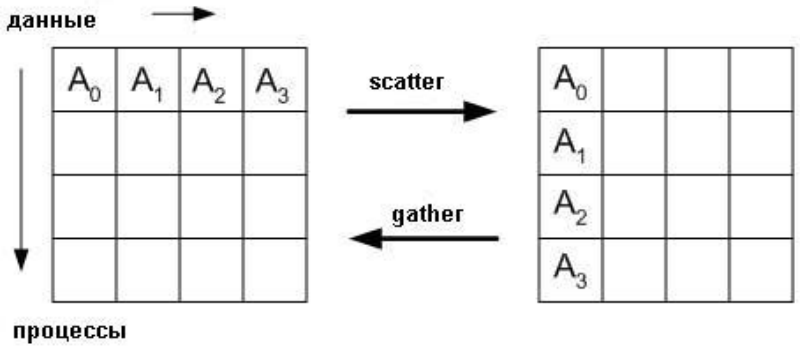
3. Вставьте код программы, вычисляющей число π.
4. Распределите все итерации цикла по процессам.
5. Для рассылки параметра N используйте коллективную функцию MPI_Bcast.
6. Для сбора и суммирования на нулевом процессе всех частичных сумм, посчитанных каждым процессом, используйте коллективную функцию MPI_Reduce. Операцию op определите как MPI_SUM.
7. Скомпилируйте и запустите ваше приложение. Убедитесь, что выводится верный результат.

Указания к заданию 19. Коллективные коммуникации: функции распределения и сбора данных

1. Создайте проект mpi_scattergather в Microsoft Visual Studio 2010 с под-держкой MPI (см. указания к заданию 10).
2. Ознакомьтесь с функциями распределения и сбора блоков данных MPI_Scatter и MPI_Gather:

   int MPI_Scatter (void *sendbuf, int sendcount, MPI_Datatype sendtype, void *recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm) – разбивает сообщение из буфера посылки процесса root на равные части размером sendcount и посылает i-ю часть в буфер приема процесса с номером i (в том числе и самому себе). Процесс root использует оба буфера (посылки и приема), поэтому в вызываемой им подпрограмме все параметры являются существенными. Остальные процессы группы с коммуникатором comm являются только получателями, поэтому для них параметры, специфицирующие буфер посылки, не существенны.

   int MPI_Gather (void* sendbuf, int sendcount, MPI_Datatype sendtype, void* recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm) – производит сборку блоков данных, посылаемых всеми процессами группы, в один массив процесса с номером root. Длина блоков предполагается одинаковой. Объединение происходит в порядке увеличения номеров процессов-отправителей. То есть данные, посланные процессом i из своего буфера sendbuf, помещаются в i-ю порцию буфера recvbuf процесса root. Длина массива, в который собираются данные, должна быть достаточной для их размещения.



3. Вставьте в код программы вызовы функций MPI_Init и MPI_Finalize.
4. Запишите номер каждого процесса в переменную rank.
5. Запишите количество параллельных процессов в программе в переменную size.
6. Разошлите параметр n всем процессам с помощью функции MPI_Bcast.
7. Распределите строки матрицы *A* равномерно по процессам с помощью функции MPI_Scatter.
8. Разошлите матрицу *B* всем процессам с помощью функции MPI_Bcast.
9. Произведите вычисления строки матрицы C на каждом процессе.
10. С помощью функции MPI_Gather соберите результат вычислений матрицы C со всех процессов на нулевом процессе.
11. Скомпилируйте и запустите ваше приложение. Убедитесь, что выводится верный результат.