Лабораторная работа № 4 Коллективные операции в MPI	ФИО	Титов А.К.
	Группа	ИВТ 460
	Предмет	Параллельное программирование
	Дата отчета	
	Оценка	
	Подпись преподавателя	

ХОД ВЫПОЛНЕНИЯ РАБОТЫ

1) Разошлите массив данных от одного процесса всем остальным с использованием функции MPI_Bcast. Используйте различные процессы в качестве корневого.

Код

```
void Task1()
{
    int procRank, procSize;

    MPI_Comm_size(MPI_COMM_WORLD, &procSize);
    MPI_Comm_rank(MPI_COMM_WORLD, &procRank);

    const int size = 100;
    int package[size];

    int senderRank = rand() % procSize;
    if (senderRank == procRank)
    {
        const int size = 100;
        for (int i = 0; i < size; i++)
            package[i] = i;
    }

    MPI_Bcast(package, size, MPI_INT, senderRank, MPI_COMM_WORLD);

    if (procRank != senderRank)
        cout << procRank << " got array from " << senderRank;
}
```

Результат работы для 16 процессов

```
D:\My\Workspace\C++\MPICollectiveOperations\Debug>mpiexec.exe -n 16 "MPICollectiveOperations.exe"
10 got array from 9
2 got array from 9
13 got array from 9
1 got array from 9
3 got array from 9
11 got array from 9
5 got array from 9
7 got array from 9
6 got array from 9
0 got array from 9
12 got array from 9
15 got array from 9
14 got array from 9
4 got array from 9
8 got array from 9
```

2) Разошлите матрицу построчно с использованием функций MPI_Scatter и MPI_Scatterv, выполните в процессах любые преобразования, затем соберите данные с использованием функций MPI_Gather и MPI_Gatherv.

Код

```
void Task2()
{
    int procRank, procSize;

    MPI_Comm_size(MPI_COMM_WORLD, &procSize);
    MPI_Comm_rank(MPI_COMM_WORLD, &procRank);


    const int rowLength = 10;
    int * matrix = new int[procSize * rowLength];
    int * vector = new int[rowLength];

    MPI_Scatter(matrix, rowLength, MPI_INT, vector, rowLength, MPI_INT, 0, MPI_COMM_WORLD);

    // modify array
    for (int i = 0; i < rowLength; i++)
        vector[i] = i;

    MPI_Gather(vector, rowLength, MPI_INT, matrix, rowLength, MPI_INT, 0, MPI_COMM_WORLD);

    bool isMain = procRank == 0;
    if (isMain)
    {
        for (int i = 0; i < procSize; i++)
        {
            for (int j = 0; j < rowLength; j++)
                cout << matrix[i*rowLength + j] << ' ';
            cout << endl;
        }
    }


    delete[] vector;
    delete[] matrix;
}
```

Результат работы для 4 процессов и матрицы (procSize, 10)

```
D:\My\Workspace\C++\MPICollectiveOperations\Debug>mpiexec.exe -n 4 "MPICollectiveOperations.exe"
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
```

3) Напишите программы с использованием функций MPI_Allgather и MPI_Alltoall.

Описание программы

1) Рассылает массив при помощи Scatter
2) Каждый поток суммирует свой участок массива
3) Все отправляют потоки результаты друг другу
4) Суммируют полученные результаты

Код

```
void Task3()
{
    int procRank, procSize;

    MPI_Comm_size(MPI_COMM_WORLD, &procSize);
    MPI_Comm_rank(MPI_COMM_WORLD, &procRank);

    int taskSize = 100;
    int * vector = new int[taskSize];
    for (int i = 0; i < taskSize; i++)
        vector[i] = 1;

    int subTaskSize = taskSize / procSize;
    int * subVector = new int[subTaskSize];

    MPI_Scatter(vector, subTaskSize, MPI_INT, subVector, subTaskSize, MPI_INT, 0, MPI_COMM_WORLD);
    int summ = 0;
    for (int i = 0; i < subTaskSize; i++)
        summ += subVector[i];


    int * summs = new int[procSize];
    MPI_Allgather(&summ, 1, MPI_INT, summs, 1, MPI_INT, MPI_COMM_WORLD);
    int totalSumm = 0;
    for (int i = 0; i < procSize; i++)
        totalSumm += summs[i];

    cout << "Proc " << procRank << " got " << totalSumm << endl;

    delete[] vector;
    delete[] subVector;
    delete[] summs;
}
```

Результат выполнения для 4 процессов и 100 элементных массивов

D:\My\Workspace\C++\MPICollectiveOperations\Debug>mpiexec.exe -n 4 "MPICollectiveOperations.exe"
Proc 3 got 100
Proc 0 got 100
Proc 2 got 100
Proc 1 got 100

4) Используя коллективные функции, найдите максимальный и минимальный элемент большого массива. Напишите программы с использованием MPI_Reduce, MPI_Allreduce, MPI_Reduce_scatter.

Описание программы

- Поиск минимумов и максимумов подмассивов
- Объединение результатов с помощью MPI_Reduce для операций MPI_MIN и MPI_MAX

Код

```
void Task4(int taskSize)
{
    int procRank, procSize;

    MPI_Comm_size(MPI_COMM_WORLD, &procSize);
    MPI_Comm_rank(MPI_COMM_WORLD, &procRank);

    int * vector = new int [taskSize];
    srand(procRank);
    for (int i = 0; i < taskSize; i++)
        vector[i] = rand();

    // local min and max
    int localMin = vector[0];
    int localMax = vector[0];

    for (int i = 0; i < taskSize; i++)
    {
        if (localMin > vector[i]) localMin = vector[i];
        if (localMax < vector[i]) localMax = vector[i];
    }

   int globalMin, globalMax;
   MPI_Reduce(&localMin, &globalMin, 1, MPI_INT, MPI_MIN, 0, MPI_COMM_WORLD);
   MPI_Reduce(&localMax, &globalMax, 1, MPI_INT, MPI_MAX, 0, MPI_COMM_WORLD);


   bool isMain = procRank == 0;
   if (isMain)
   {
       cout << "Global min: " << globalMin << endl;
       cout << "Global max: " << globalMax << endl;
   }

   delete[] vector;
}
```

Результат выполнения для 4 процессов

```
D:\My\Workspace\C++\MPICollectiveOperations\Debug>mpiexec.exe -n 4 "MPICollectiveOperations.exe"
Global min: 0
Global max: 32767
```

Индивидуальное задание.

Задача построения массивов значений нескольких функций на определенном отрезке. Подзадача – обработка одной функции.

Код

```
#include <vector>
typedef double(*matrFunc)(double); // math func
void IndividualTask(int rangeLength)
{
    int procSize, procRank;

    MPI_Comm_size(MPI_COMM_WORLD, &procSize);
    MPI_Comm_rank(MPI_COMM_WORLD, &procRank);

    std::vector<matrFunc> functions(procSize, sin); // many sinuses
    double * range = new double[rangeLength];

    bool isMain = procRank == 0;
    if (isMain)
    {
        double start = 0.0;
        double finish = 150.0;
        double step = abs(start - finish) / rangeLength;
        for (int i = 0; i < rangeLength; i++)
            range[i] = start + i*step;
    }

    MPI_Bcast(range, rangeLength, MPI_DOUBLE, 0, MPI_COMM_WORLD);

    // Calculate function values
    double * functionValues = new double[rangeLength];
    for (int i = 0; i < rangeLength; i++)
        functionValues[i] = functions[procRank](range[i]);


    double * allFunctionsValues = NULL; // NULL is very important here
    if (isMain)
        allFunctionsValues = new double[rangeLength * procSize];

    MPI_Gather(functionValues, rangeLength, MPI_DOUBLE,
               allFunctionsValues, rangeLength, MPI_DOUBLE, 0, MPI_COMM_WORLD);

    if (isMain)
    {
        for (int i = 0; i < procSize; i++)
        {
            cout << "Function " << i << " values: ";
            for (int j = 0; j < rangeLength; j++)
                cout << allFunctionsValues[i*rangeLength + j] << ' ';
            cout << endl;
        }
    }

    if (isMain)
        delete[] allFunctionsValues;

    delete[] functionValues;
    delete[] range;
}
```

Результат выполнения для 4 процессов с размерами диапазонов 20

```
D:\My\Workspace\C++\MPICollectiveOperations\Debug>mpiexec.exe -n 4 "MPICollectiveOperations.exe"
Function 0 values: 0 0.938 0.650288 -0.487175 -0.988032 -0.197799 0.850904 0.787705 -0.304811 -0.999021 -0.387782 0.730184 0.893997 -0.110402 -0.970535 -0.562441 0.580611 0.964962 0.0883687 -0.903699
Function 1 values: 0 0.938 0.650288 -0.487175 -0.988032 -0.197799 0.850904 0.787705 -0.304811 -0.999021 -0.387782 0.730184 0.893997 -0.110402 -0.970535 -0.562441 0.580611 0.964962 0.0883687 -0.903699
Function 2 values: 0 0.938 0.650288 -0.487175 -0.988032 -0.197799 0.850904 0.787705 -0.304811 -0.999021 -0.387782 0.730184 0.893997 -0.110402 -0.970535 -0.562441 0.580611 0.964962 0.0883687 -0.903699
Function 3 values: 0 0.938 0.650288 -0.487175 -0.988032 -0.197799 0.850904 0.787705 -0.304811 -0.999021 -0.387782 0.730184 0.893997 -0.110402 -0.970535 -0.562441 0.580611 0.964962 0.0883687 -0.903699
```