МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА №14

РАБОТА ЗАЩИЩЕНА С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

старший преподаватель		А. Ю. Сыщиков
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2
MPI, Распределенные вычисления

по дисциплине: Системы с параллельной обработкой информации

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. 1742		Д.В. Коробков
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2021

1. Цель работы

Сгенерировать в каждом из N процессов вектор чисел. Размерность вектора – M. Произвести поэлементную обработку всех векторов и поместить результирующий вектор в каком-либо процессе. В работе использовать средства MPI для организации распределенных вычислений

№ варианта	N	M	Тип элемента вектора	Тип поэлементной обработки
9	6	40	Без знаковый целый	Двоичное И

2. Текст программы

```
#include "mpi.h"
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define M 40
#define root 0
static int gsize, myid;
void initArray(unsigned int*a, int m) {
	int i;
	for (i = 0; i < m; i++) {
		a[i] = i+ myid;
	}
}
void printArray(unsigned int*a, int m) {
	int i;
	printf("[ ");
	for (i = 0; i < m; i++) {
		printf("%u ", a[i]);
	}
	printf("]\n");
}
int main(int argc, char *argv[]) {
	unsigned int *rbuf;
	unsigned int *sendbuf;
	int namelen;
	double startwtime = 0.0, endwtime;
	char processor_name[MPI_MAX_PROCESSOR_NAME];
	MPI_Init(&argc, &argv);
	MPI_Comm_size(MPI_COMM_WORLD, &gsize);
	MPI_Comm_rank(MPI_COMM_WORLD, &myid);
	MPI_Get_processor_name(processor_name, &namelen);
	printf("Process %d on %s\n", myid, processor_name);
	sendbuf = (unsigned int *)malloc(M * sizeof(unsigned int));
	initArray(sendbuf, M);
	printf("Massiv: ");
	printArray(sendbuf, M);
	if (myid == root) {
		startwtime = MPI_Wtime();
		rbuf = (unsigned int *)malloc(M * sizeof(unsigned int));
	}
	MPI_Reduce(sendbuf, rbuf, M, MPI_UNSIGNED, MPI_BAND, root, MPI_COMM_WORLD);
	if (myid == root) {
		printf("\nResult Massiv: ");
		printArray(rbuf, M);
	}
	fflush(stdout);
	if (myid == 0) {
		endwtime = MPI_Wtime();
		fprintf(stderr, "wall clock time = %f\n", endwtime - startwtime);
		fflush(stdout);
```

```
        }
        MPI_Finalize();
        return 0;
}
```

3. Результат работы программы

```
C:\Users\Admin\Documents\Visual Studio 2015\Projects\MPITest\Debug>mpiexec.exe -n 2 MPITest.exe
Process 1 on Danila.MYDNS
Massiv: [ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 ]
Process 0 on Danila.MYDNS
Massiv: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 ]

Result Massiv: [ 0 0 2 0 4 4 6 0 8 8 10 8 12 12 14 0 16 16 18 16 20 20 22 16 24 24 26 24 28 28 30 0 32 32 34 32 36 36 38 32 ]
wall clock time = 0.000350

C:\Users\Admin\Documents\Visual Studio 2015\Projects\MPITest\Debug>mpiexec.exe -n 1 MPITest.exe
Process 0 on Danila.MYDNS
Massiv: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 ]

Result Massiv: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
]
wall clock time = 0.000057

C:\Users\Admin\Documents\Visual Studio 2015\Projects\MPITest\Debug>mpiexec.exe -n 6 MPITest.exe
Process 3 on Danila.MYDNS
Massiv: [ 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 ]
Process 5 on Danila.MYDNS
Massiv: [ 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 ]
Process 1 on Danila.MYDNS
Massiv: [ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 ]
Process 2 on Danila.MYDNS
Massiv: [ 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 ]
Process 4 on Danila.MYDNS
Massiv: [ 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 ]
Process 0 on Danila.MYDNS
Massiv: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 ]

Result Massiv: [ 0 0 0 0 0 0 0 0 8 8 8 0 0 0 0 0 16 16 16 16 16 16 16 24 24 24 0 0 0 0 0 32 32 32 32 32 32 32 32 ]
wall clock time = 0.000692

C:\Users\Admin\Documents\Visual Studio 2015\Projects\MPITest\Debug>
```

Рисунок 1. Результат работы программы.

В ходе работы программы у каждого процесса создается свой массив из 40 элементов типа unsigned int. Формируются они по формуле a[i] = i+myid, где myid - это номер процесса. Сформированные массивы вводятся на экран. После вызывается функция MPI_Reduce, по окончанию которой главный процесс получает массив с результатами вычислений. Полученный массив выводиться на экран.

На рисунке 1 продемонстрированы 3 примера:

1) когда два процесса, то есть два массива. Результат вычислений побитового И выводится на экран, как Result Massiv;

2) когда один процесс, то есть один массив. Результат будет таким же, как и изначальный массив;

3) когда у нас 6 процессов, то есть 6 массивов. Результат будет состоять из большего числа нулей, чем в первой примере, так как побитовая разница между числами увеличилась. Чаще происходит ситуация, что в одном разряде по одному столбцу среди всех массивов имеется хотя бы один ноль.