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](https://pro.guap.ru/inside_s#subjects/2436975) |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 1742 |  |  |  | В.А. Седов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2021

1. **Цель работы**

Сгенерировать в каждом из N процессов вектор чисел. Размерность вектора – М. 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 |
| 16 | 3 | 30 | Длинное целое | Двоичное И |

1. **Текст программы**

#include "mpi.h"

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#define M 40

#define root 0

static int gsize, myid;

constexpr auto M = 30;

void initArray(long int\* a, int m) {

int i;

for (i = 0; i < m; i++) {

a[i] = i + myid;

}

}

void printArray(long 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[]) {

long int\* rbuf;

long 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 = (long int\*)malloc(M \* sizeof(long int));

initArray(sendbuf, M);

printf("Massiv: ");

printArray(sendbuf, M);

if (myid == root) {

startwtime = MPI\_Wtime();

}

rbuf = (long int\*)malloc(M \* sizeof(long int));

MPI\_Reduce(sendbuf, rbuf, M, MPI\_LONG, 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;

}

1. **Результат работы программы**

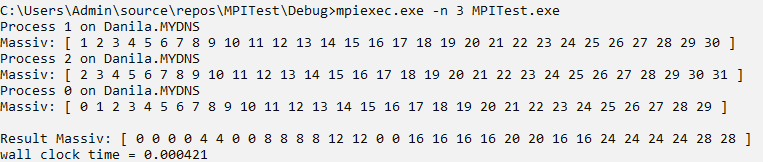


Рисунок 1. Результат работы программы