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 |  |  |  |  |
|  |  |  | 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 |
| 2 | 4 | 20 | Знаковый короткий целый | Поэлементное умножение |

1. **Текст программы**

#include "mpi.h"

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#define M 20 //размер массива

#define root 0 //какому процессу вернется результат умножения

static int gsize, myid;//количество процессов, номер процесса

void initArray(short int\* a) {//инициализация ассива размерности M со сдвигом

int i;

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

a[i] = i + myid;

}

}

void printArray(short int\* a) {//вывод массива размерности M

int i;

printf("[ ");

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

printf("%hd ", a[i]);

}

printf("]\n");

}

int main(int argc, char\* argv[]) {

short int\* rbuf; //выходной/результирующий массив, в нем результат умножения

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

short int\* sendbuf; //изначальны массив, у каждого процесса он свой

int namelen;//для названия ПК

double startwtime = 0.0, endwtime; //время

char processor\_name[MPI\_MAX\_PROCESSOR\_NAME]; //Взятие называния пк

MPI\_Init(&argc, &argv);//иницализация МПАЙ

MPI\_Comm\_size(MPI\_COMM\_WORLD, &gsize);//получение кол-ва процессов

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &myid);//получение номера процесса

MPI\_Get\_processor\_name(processor\_name, &namelen);//финальное имя процесса (ПК+номерпроцесса)

printf("Process %d on %s\n", myid, processor\_name);

/\*инициализация и вывод отправного буфера\*/

sendbuf = (short int\*)malloc(M \* sizeof(short int));

initArray(sendbuf);

printf("Massiv: ");

printArray(sendbuf);

if (myid == root) {

startwtime = MPI\_Wtime();

}

//значения всех sendbuf у каждого процесса поэлементно перемножаются и записывается всё в rbuf процесса номер root

MPI\_Reduce(sendbuf, rbuf, M, MPI\_SHORT, MPI\_PROD, root, MPI\_COMM\_WORLD);

if (myid == root) {

printf("\nResult Massiv: ");

printArray(rbuf);

}

fflush(stdout);

if (myid == root) {

endwtime = MPI\_Wtime();

fprintf(stderr, "wall clock time = %f\n", endwtime - startwtime);

fflush(stdout);

}

MPI\_Finalize();

return 0;

}

1. **Результат работы программы**

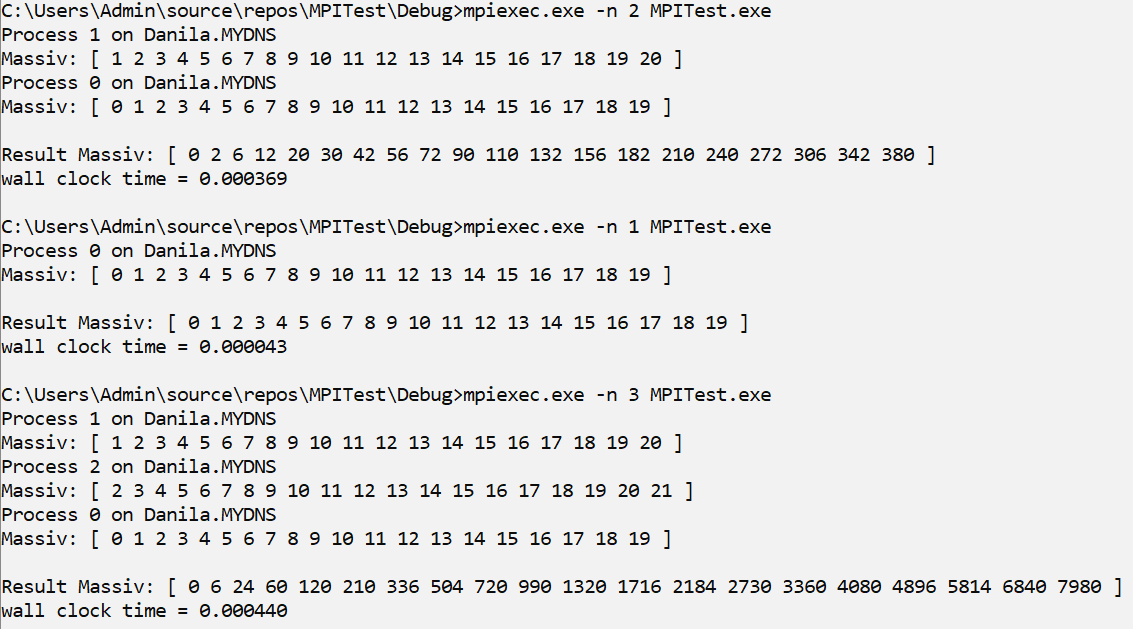


Рисунок 1. Результат работы программы