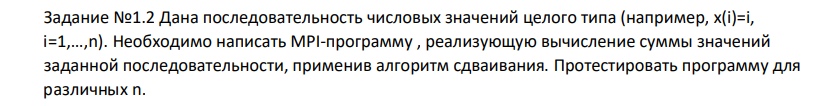
Отчет №3

Гренц Елизавета Алексеевна группа 932220



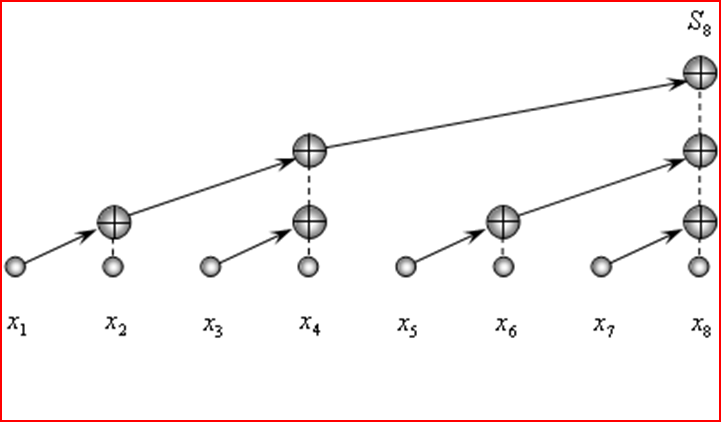
Описание работы:

Для начала я создаю последовательность х где x[i]=i , разбиваю последовательность между процессами.

Потом применяем алгоритм сдваивания, я его работу поняла как-то так:

1)каждый процесс содержит часть данных, которые мы суммируем.

2) Потом идет поэтапное суммирование, на каждом этапе «расстояние » между процессами удваивается



Картинка все идеально описывает

Пример:

N=16 , процессы - 4

Процесс 0: x={1,2,3,4} локальная сумма =1+2+3+4=10

Процесс 1: x={5,6,7,8} локальная сумма =5+6+7+8=26

Процесс 2: x={9,10,11,12} локальная сумма =9+10+11+12=42

Процесс 3: x={13,14,15,16} локальная сумма =13+14+15+16=58

Суммируем локальные суммы: Итог=10+26+42+58=136

Код для вставки :

#include <mpi.h>

#include <iostream>

#include <vector>

#include <cmath>

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

MPI\_Init(&argc, &argv);

int rank, size;

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

int n = 20;

int q = log2(size); //

std::vector<int> x(n / size);

for (int i = 0; i < n / size; ++i) {

x[i] = rank \* (n / size) + i + 1;// x[i]=i

}

int S=0;

for (int i = 0; i < n / size; ++i) {

S += x[i];

printf("x[%d ]= %d\n",i,x[i]);

}

printf("S=%d\n",S);

for (int k = 0; k < q; ++k) {

int step = 1 << k ;

if (rank % (2 \* step) == step-1 && rank + step < size) {

int dest = rank + step;

MPI\_Send(&S, 1, MPI\_INT, dest, 0, MPI\_COMM\_WORLD);

} else if (rank % (2 \* step) == step\*2-1) {

int source = rank - step;

int S1=0;

MPI\_Recv(&S1, 1, MPI\_INT, source, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

S+=S1;

}

}

int total\_sum = 0; // Declare and initialize total\_sum

if (rank == size-1) {

printf("Sum: %d\n ",S);

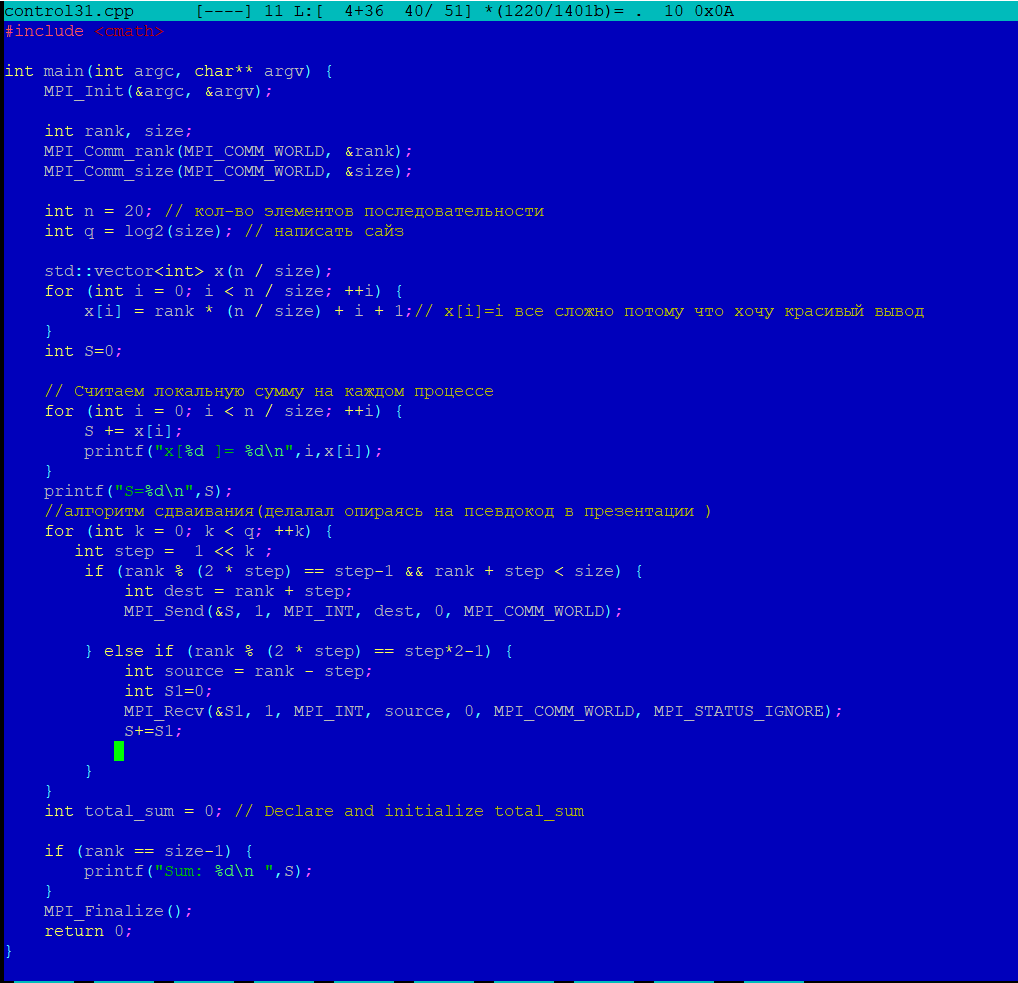
}

MPI\_Finalize();

return 0;

}

Скрин\_для \_просмотра



**Результаты**

**2-процесса**

**N=20**

x[0 ]= 1

x[1 ]= 2

x[2 ]= 3

x[3 ]= 4

x[4 ]= 5

x[5 ]= 6

x[6 ]= 7

x[7 ]= 8

x[8 ]= 9

x[9 ]= 10

S=55

Total sum: 210

x[0 ]= 11

x[1 ]= 12

x[2 ]= 13

x[3 ]= 14

x[4 ]= 15

x[5 ]= 16

x[6 ]= 17

x[7 ]= 18

x[8 ]= 19

x[9 ]= 20

S=155

**4-процесса**

**N=20**

x[0 ]= 11

x[1 ]= 12

x[2 ]= 13

x[3 ]= 14

x[4 ]= 15

S=65

x[0 ]= 16

x[1 ]= 17

x[2 ]= 18

x[3 ]= 19

x[4 ]= 20

S=90

x[0 ]= 1

x[1 ]= 2

x[2 ]= 3

x[3 ]= 4

x[4 ]= 5

S=15

x[0 ]= 6

x[1 ]= 7

x[2 ]= 8

x[3 ]= 9

x[4 ]= 10

S=40

Total sum: 210

**10-процессов**

N =20

x[0 ]= 5

x[1 ]= 6

S=11

x[0 ]= 1

x[1 ]= 2

S=3

x[0 ]= 3

x[1 ]= 4

S=7

x[0 ]= 7

x[1 ]= 8

S=15

x[0 ]= 9

x[1 ]= 10

S=19

x[0 ]= 11

x[1 ]= 12

S=23

x[0 ]= 19

x[1 ]= 20

S=39

x[0 ]= 17

x[1 ]= 18

S=35

Total sum: 210

x[0 ]= 13

x[1 ]= 14

S=27

x[0 ]= 15

x[1 ]= 16

S=31

**Вывод :**

Я поняла, как работает алгоритм сдваивания.(ну мне так кажется)