**Практическая работа №4**

**Вариант №2**

**Формулировка задания.**

Написать параллельную MPI программу вычисления скалярного произведение двух векторов X и Y длиной n. Пусть оба вектора состоят из случайных вещественных чисел из диапазона [0, 1]. Требуется в зависимости от варианта задания реализовать алгоритм суммирования (ас, кс, мкс). Оценить ускорение и эффективность параллельной программы, подготовить отчет.

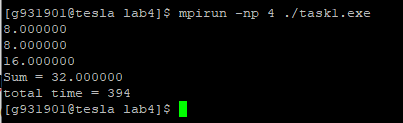
**Ход работы.**

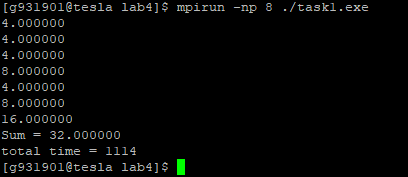
Схема вычислений:

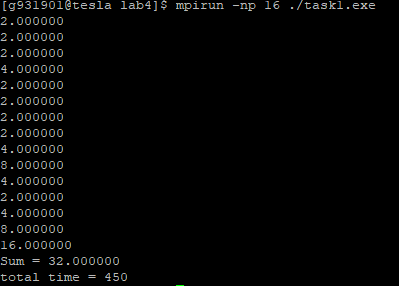
Сначала мы формируем вектора x и y, которые имеют случайные значения. После чего считаем их скалярное произведение и храним сумму их всех в local\_sum. В случае, если элементов в массиве не будет хватать, на первом процессе предусмотрено резерв места. Потом переходим к каскадному суммированию.

Каскадное суммирование реализовано как цикл, в котором шаги представляют из себя степени двойки начиная с 0. После чего идет проверка, в которой должны выполняться условия: во-первых, принимают числа только процессы кратные 2 и, во-вторых, это не последний элемент, в других случаях мы отправляем элементы процессам rank – step.

Пример(size = 4, 8, 16):

****

****

****

**Заключение**

В данной работе была написана программа для распараллеливание векторных операций по процессам, с последующим выводом на экран.

**Код программы**

#include <stdio.h>

#include <mpi.h>

#include <cstdlib>

#include <time.h>

#include <math.h>

#include <random>

const int N = 32;

using namespace std;

void ScolarVectors(double x[], double y[], double scolar[], double& sum, int length, int rank)

{

for(int i = 0; i < length; i++)

{

x[i] = 1;

y[i] = 1;

scolar[i] = x[i]\*y[i];

sum += scolar[i];

//printf("Process %d: Xi = %f, Yi = %f, S = %f\n", rank, x[i], y[i], sum);

}

}

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

{

int rank, size, rc, tag = 11;

MPI\_Status status;

rc = MPI\_Init(&argc, &argv);

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

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

int length = N / size;

int overflowLength = N % size;

double local\_sum = 0.0;

double x\_extra[length + overflowLength];

double y\_extra[length + overflowLength];

double scolar\_extra[length + overflowLength];

double x\_local[length];

double y\_local[length];

double scolar\_local[length];

int start\_time = clock();

if(rank == 0)

{

ScolarVectors(x\_extra, y\_extra, scolar\_extra, local\_sum, length + overflowLength, rank);

}

else

{

ScolarVectors(x\_local, y\_local, scolar\_local, local\_sum, length, rank);

}

double global\_sum = local\_sum;

double recieved\_message = 0.0;

for (int step = 1; step < size; step \*= 2)

{

if (rank % (2 \* step) == 0)

{

if (rank + step < size)

{

MPI\_Recv(&recieved\_message, 1, MPI\_DOUBLE, rank + step, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

printf("%f\n", recieved\_message);

global\_sum += recieved\_message;

}

}

else

{

MPI\_Send(&global\_sum, 1, MPI\_DOUBLE, rank - step, 0, MPI\_COMM\_WORLD);

break;

}

}

int finish\_time = clock();

if(rank == 0)

{

printf("Sum = %f\n", global\_sum);

printf("total time = %d\n", finish\_time - start\_time);

}

rc = MPI\_Finalize();

}

**Ссылка на код на кластере**

/home/g931901/g932204/Sobol/lab4/task1.cpp