МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСТИТЕТ

Институт прикладной математики и механики

Кафедра теоретической механики

ОТЧЕТ ПО КУРСОВОМУ ПРОЕКТУ

MPI-программа вычисления определенного интеграла

Выполнила:

студентка группы № 63604/1

А. С. Гнездилова

Проверил:

\_\_\_\_\_\_\_\_\_\_\_\_ А. А. Ле-Захаров

Санкт-Петербург

2015

**1. Программирование в Message Passing Interface (MPI)**

MPI - это библиотека передачи сообщений, собрание функций на C/C++ (или подпрограмм в Фортране, которые, зная MPI для C/C++, легко изучить самостоятельно), облегчающих коммуникацию (обмен данными и синхронизацию задач) между процессами параллельной программы с распределенной памятью. Разделенная память - память, которая разделена на сегменты, каждый из которых может быть напрямую доступен только одному узлу из параллельных процессоров. Акроним (сокращение по первым буквам) установлен для Message Passing Interface (интерфейс передачи сообщений).

MPI является на данный момент фактическим стандартом и самой развитой переносимой библиотекой параллельного программирования с передачей сообщений предлагает переносимость, стандартизацию, эффективную работу, функциональность.

Стандартизация

MPI стандартизован на многих уровнях. Например, поскольку синтаксис стандартен, вы можете положиться на ваш MPI код при запуске в любой реализации MPI, работающей на вашей архитектуре. Поскольку функциональное поведение вызовов MPI довольно стандартизовано, нет нужды беспокоиться о том, какая реализация MPI установлена сейчас на вашей машине; ваши вызовы MPI должны вести себя одинаково независимо от реализации. Эффективность работы, тем не менее, слегка меняется в зависимости от реализации.

Переносимость

В быстро изменяющемся окружении высокопроизводительных компьютеров и технологий коммуникации, переносимость (мобильность) почти всегда важна. Кто захочет развивать программу, которая может выполняться только на одной машине, или только с дополнительными затратами труда на других машинах? Все системы массивной параллельной обработки обеспечивают своего рода библиотеку передачи сообщений, которая точно определена аппаратными средствами используемой ЭВМ. Они обеспечивают прекрасную эффективность работы, но прикладной код, написанный для одной платформы не может быть легко перенесен на другую позволяет вам писать портативные программы, которые все еще используют в своих интересах спецификации аппаратных средств ЭВМ и программного обеспечения, предлагаемого поставщиками. К счастью, эти заботы в основном берут на себя запросы MPI, потому что конструкторы настроили эти вызовы на основные аппаратные средства ЭВМ и окружающую среду программного обеспечения.

# Постановка задачи

Приведенная ниже программа представляет собой консольное приложение Windows, вычисляющее определенный интеграл методом Гаусса (прямоугольников) с шагом 10-4.

#include "mpi.h"

#include <stdlib.h>

#include <time.h>

#include <iostream>

#include <math.h>

using namespace std;

double f(double x)

{

return pow(x, 8) + pow(x, 2)+pow(x,-2);

}

double integral(double x, double h)

{

return f(x + 0.5 \* h) \* h;

}

double sum(int rank, int size, double a, double b, double h)

{

double sum = 0;

double x = a + rank \* h;

double bb = b - h;

while(x <= bb)

{

sum += integral(x, h);

x += size \* h;

}

if (x < b) sum += integral(x, b - x);

return sum;

}

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

{

double dt;

long i;

double a(-2), b(2);

double h = 0.0001;

double I, dI,t1,t2,z;

int rank;

int size;

const int root = 0;

MPI\_Init(&argc, &argv);

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

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

t1=MPI\_Wtime();

dI = sum(rank, size, a, b, h);

if (rank != root)

{

MPI\_Send(&dI, 1, MPI\_DOUBLE, root, 0, MPI\_COMM\_WORLD);

}

else

{

I = dI;

MPI\_Status status;

for (int sender = 0; sender < size; sender++)

if (sender != root)

{

MPI\_Recv(&dI, 1, MPI\_DOUBLE, sender, 0, MPI\_COMM\_WORLD, &status);

I += dI;

}

cout << " I = " << I << endl;

}

t2=MPI\_Wtime();

if(rank==0)

printf("N= %d, Nproc=%d, z = %lf, Time=%lf \n", i, size, z, t2-t1);

MPI\_Finalize();

getchar();

return 0;

}  
На рис.1. и рис.2. приведены значения интеграла и время, которое было затрачено на расчет в одном и двух потоках соответственно.

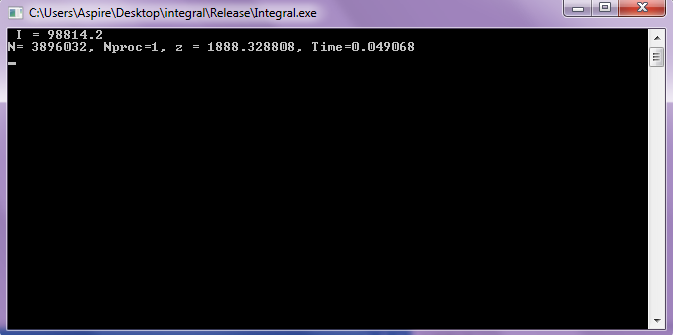


Рис.1 Запуск программы в одном потоке.

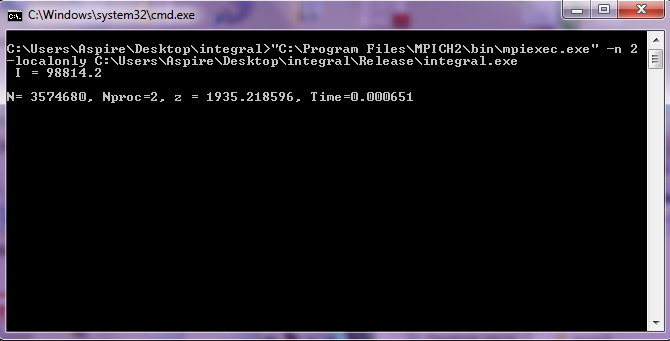


Рис.2 Запуск программы в двух потоках.

**Заключение**

В ходе работы с помощью библиотеки MPI было реализовано вычисление определенного интеграла в одном и двух потоках. Выявлено увеличение производительности в двух потоках.