**Практическая работа № 5**

**Распределенное программирование для систем с общей памятью**

**с использованием основ технологии MPI**

# **Цель работы**

* Отработка принципов разработки интерфейса передачи сообщений для распределенных приложений;
* Реализация простейшей распределенной программы;
* Анализ эффективности распределенных программ.

# **Постановка задачи**

Вариант 5.7

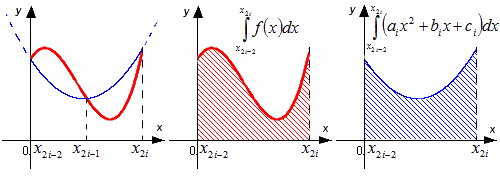
Составить программу последовательного и параллельного вычисления определённого интеграла методом Симпсона. Провести контрольные прогоны программы для числа разбиений отрезка интегрирования n = 104, 105, 106, 107, 108 и установленных для параллельного варианта количествах потоков p = 4, 8, 16, 32 и 64 с вычислением времени выполнения и ускорения. Полученные результаты свести в сводную таблицу.

Построить графики изменения ускорения при последовательном и параллельных вычислениях в зависимости от числа разбиений отрезка интегрирования. Построить графики изменения ускорения при параллельных вычислениях в зависимости от количества используемых потоков. Вычислить показатели эффективности и стоимости параллельной реализации программы. Провести анализ полученных результатов. Сделать выводы о проделанной работе, основанные на полученных данных.

# **Теоретическое введение**

Наиболее распространенной технологией программирования параллельных компьютеров с распределенной памятью является технология MPI [2,10]. Основным способом взаимодействия параллельных процессов в таких системах является передача сообщений друг другу. Это и отражено в названии технологии – Message Passing Interface. Под параллельной программой в рамках MPI понимается множество одновременно выполняемых процессов. Процессы могут выполняться на разных процессорах, но на одном процессоре могут располагаться и несколько процессов (в этом случае их исполнение осуществляется в режиме разделения времени).

Метод Симпсона заключается в интегрировании интерполяционного многочлена второй степени функции f(x) с узлами интерполяции a, b и m = (a+b)/2 — параболы p(x). Для повышения точности имеет смысл разбить отрезок интегрирования на N равных промежутков (по аналогии с методом трапеций), на каждом из которых применить метод Симпсона. Площадь параболы может быть найдена суммированием площадей 6 прямоугольников равной ширины. Высота первого из них должна быть равна f(a), с третьего по пятый — f(m), шестого — f(m). Таким образом, приближение методом Симпсона находим по формуле:



*Рисунок 1. Метод симпсона*

В предельном случае для выполнения параллельной программы может использоваться один процессор – как правило, такой способ применяется для начальной проверки правильности параллельной программы. Каждый процесс параллельной программы порождается на основе копии одного и того же программного кода. Данный программный код, представленный в виде исполняемой программы, должен быть доступен в момент запуска параллельной программы на всех используемых процессорах.

# **Описание алгоритмов, используемых для решения задачи**

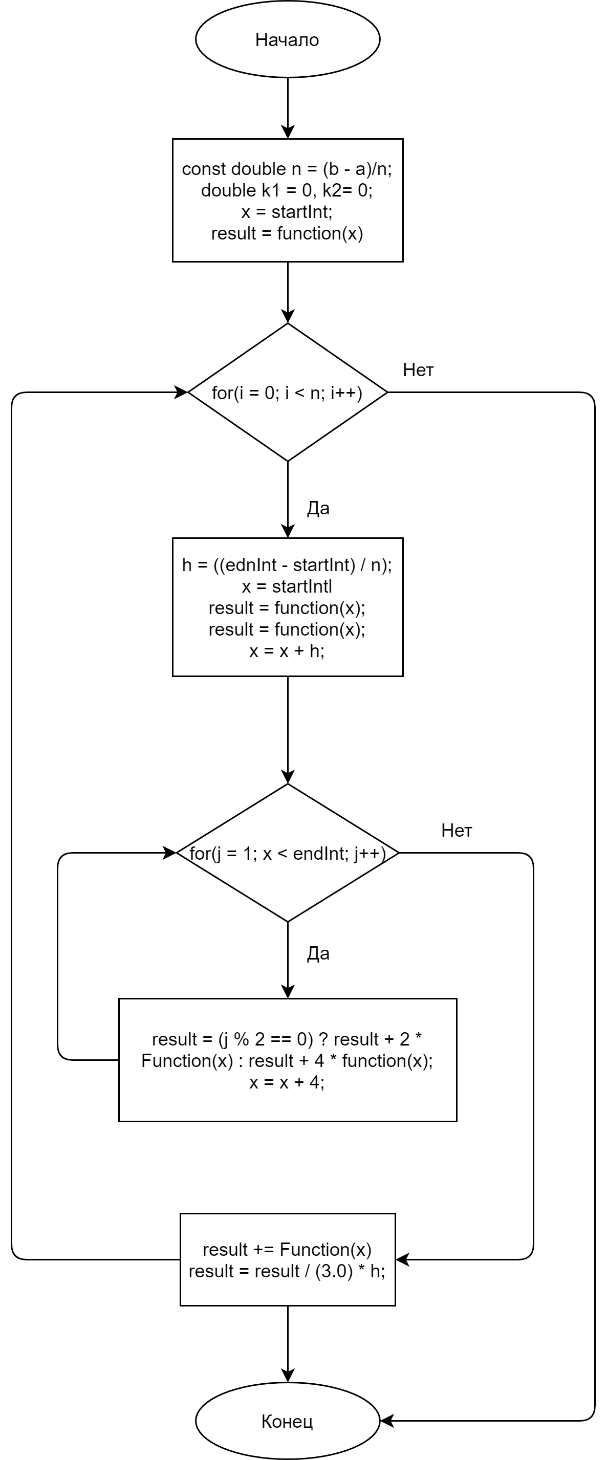
Для выполнения данной практической работы было решено провести декомпозицию общего процесса на более мелкие составляющие (рисунок 2).



*Рисунок 2. Декомпозиция общей задачи на подзадачи*

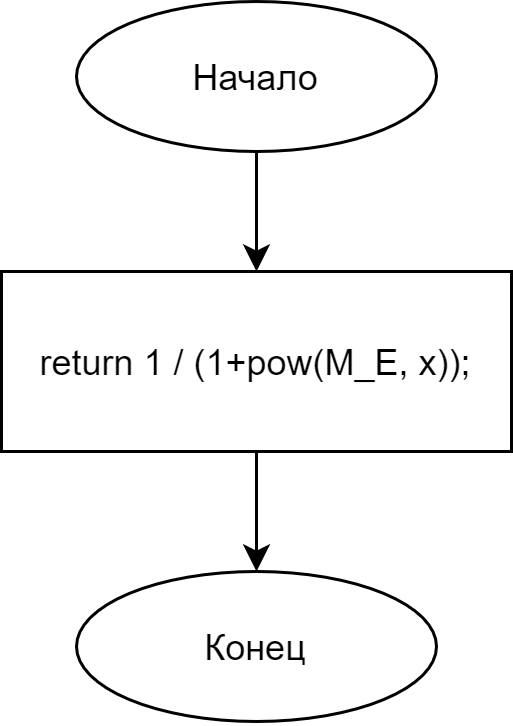
После проведения декомпозиции опишем использование всех алгоритмов, используемых в решении общей задачи.

Во-первых, ниже продемонстрирован рисунок схемы алгоритма работы программы (рисунок 3).



*Рисунок 3. Общего алгоритма работы программы*

Во-третьих, продемонстрирован рисунок схемы алгоритма вычисления значения функции в точке (рисунок 4).



*Рисунок 4. Схема алгоритма вывода значения функции в точке*

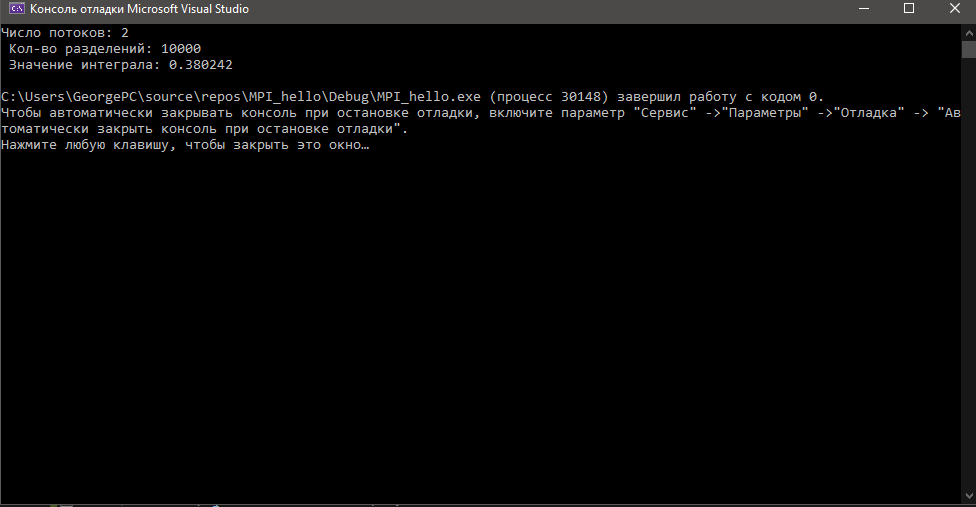
# **Текст исходного кода программы**

|  |
| --- |
| #include <math.h>  #include <mpi.h>  #include <iostream>  using namespace std;  #define MASTER 0  double Function(double x);  int main(int argc, char\*\* argv) {  int myrank, nprocs, a, b, i;  long n, j;  double startInt, endInt, result, h, x, endresult;  setlocale(LC\_ALL, "Rus");  MPI\_Status recv\_status;  MPI\_Request request;  MPI\_Init(&argc, &argv);  MPI\_Comm\_rank(MPI\_COMM\_WORLD, &myrank);  MPI\_Comm\_size(MPI\_COMM\_WORLD, &nprocs);  a = 0;  b = 1;  startInt = (double)(b - a) / nprocs \* myrank + a;  endInt = (double)(b - a) / nprocs \* (myrank + 1.0) + a;  for (i = 0; i < n; ++i) {  MPI\_Barrier(MPI\_COMM\_WORLD);  time = MPI\_Wtime();  h = ((endInt - startInt) / n);  x = startInt;  result = function(x);  x = x + h;  for (j = 1; x < endInt; ++j) {  result = (j % 2 == 0) ? result + 2 \* Function(x) :  result + 4 \* function(x);  x = x + h;  }  result += Function(x);  result = result / (3.0) \* h;  MPI\_Reduce(&result, &endresult, 1, MPI\_DOUBLE, MPI\_SUM,  MASTER, MPI\_COMM\_WORLD);  time = MPI\_Wtime() - time;  MPI\_Barrier(MPI\_COMM\_WORLD);  MPI\_Reduce(&time, &slowest, 1, MPI\_DOUBLE, MPI\_MAX, MASTER,  MPI\_COMM\_WORLD);  if (myrank == MASTER) {  times[i] = slowest;  }  }  if (myrank == MASTER) {  cout << "Num thread: " << nprocs << " Partition: " << n  << " Min in thread: " << time << " Res: " <<  endresult;  }  MPI\_Finalize();  }  double function(double x) {  double result = 1 / (1 + pow(2.71, x));  if (isnan(result)) {  result = 0.0f;  }  return result;  } |
|  |

Листинг файла index.cpp. Исходный код программы.

# **Контрольные прогоны программы**

Контрольные прогоны программы происходят при количестве разбиений n = 10000, при количестве потоков p = 2. Результат выполнения программы представлен на рисунке 5.



*Рисунок 5. Результаты тестирования*.

# **Анализ полученных результатов**

Далее – были составлены сводные таблицы по сравнению ключевых параметров работы алгоритмов.

*Таблица 1. Время работы последовательного и параллельного алгоритма вычисления интеграла методом Симпсона*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Количество элементов, n | Время при последовательном вычислении, с | Время работы при  P = 4 | Время работы при  P=8 | Время работы при  P=16 | Время работы при  P=32 | Время работы при  P=64 |
| 10000 | 4.273 | 2.268 | 1.667 | 0.997 | 0.989 | 0.844 |
| 100000 | 14.275 | 7.386 | 5.448 | 3.329 | 3.258 | 2.71 |
| 1000000 | 33.22 | 17.083 | 12.292 | 7.845 | 7.653 | 6.956 |
| 10000000 | 64.128 | 33.126 | 23.835 | 15.145 | 14.85 | 12.134 |
| 100000000 | 110.822 | 56.774 | 41.038 | 26.058 | 25.422 | 23.095 |

На основании этих данных был построен общий график зависимости время выполнения алгоритма от размера массива (рисунок 6).

На основании этого графика можно сделать следующие вывод, что наиболее быстрая работа достигается при работе программы с числом потоков равным 64. Медленнее же всего работает последовательный алгоритм

*Рисунок.6. Графики зависимости время работы программы от размерности массива*

Далее, была составлена сводная таблица изменения ускорения от количества элементов массива

*Таблица 2. Изменение ускорения параллельной реализацией вычисления интеграла методом Симпсона*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Количество элементов, n | Ускорение Sn, при P = 4 | Ускорение Sn, при P=8 | Ускорение Sn, при P=16 | Ускорение Sn, при P=32 | Ускорение Sn, при P=64 |
| 1000 | 1,8779 | 1,618 | 4,293 | 4,3395 | 4,0626 |
| 10000 | 1,9352 | 2,623 | 4,29 | 4,3849 | 5,1527 |
| 100000 | 1,9344 | 2,691 | 4,225 | 4,3261 | 5,1659 |
| 1000000 | 1,9334 | 2,679 | 4,238 | 4,3174 | 5,2838 |
| 10000000 | 1,9353 | 2,677 | 4,214 | 4,3192 | 5,2853 |

На основании этих данных был построен общий график зависимости ускорения алгоритма от размера массива (рисунок 7).

*Рисунок.7. Графики зависимости ускорения от кол-ва разбиений*

Далее, была составлена сводная таблица изменения эффективности от количества разбиений отрезков

*Таблица 3. Изменение эффективности параллельной реализации вычисления интеграла методом Симпсона*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Количество элементов, n | Эффективность En, при P = 4 | Эффективность En, при P=8 | Эффективность En, при P=16 | Эффективность En, при P=32 | Эффективность En, при P=64 |
| 1000 | 0,4679 | 0,327 | 0,2684 | 0,1356 | 0,0791 |
| 10000 | 0,4833 | 0,3277 | 0,2681 | 0,137 | 0,08051 |
| 100000 | 0,4846 | 0,3365 | 0,2638 | 0,1352 | 0,07437 |
| 1000000 | 0,4838 | 0,3362 | 0,2645 | 0,14492 | 0,08255 |
| 10000000 | 0,4835 | 0,3345 | 0,2634 | 0,13513 | 0,08335 |

На основании этих данных был построен общий график зависимости

эффективности алгоритма от кол-ва разбиений отрезка интегрирования (рисунок 8).

*Рис.8. Графики зависимости эффективности от количества разбиений*

*Таблица 4. Изменение стоимости параллельной реализации вычисления интеграла методом Симпсона*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Количество элементов, n | Стоимость Cn, при P = 4 | Стоимость Cn, при P=8 | Стоимость Cn, при P=16 | Стоимость Cn, при P=32 | Стоимость Cn, при P=64 |
| 1000 | 9,152 | 13,096 | 15,952 | 31,584 | 54,144 |
| 10000 | 29,532 | 43,544 | 53,232 | 104,16 | 177,28 |
| 100000 | 68,324 | 98,368 | 125,504 | 244,896 | 445,184 |
| 1000000 | 132,504 | 190,68 | 242,32 | 475,2 | 776,576 |
| 10000000 | 227,096 | 328,304 | 416,928 | 813,504 | 1478,08 |

На основании этих данных был построен общий график зависимости

стоимости алгоритма от кол-ва разбиений отрезка интегрирования (рисунок 9).

*Рисунок.9. Графики зависимости стоимости от количества элементов массива*

# **Выводы**

В ходе выполнения данной практической работы реализовал цели данной работы и соответствующего раздела, углубленно познакомился с пакетом MPI, изменил алгоритм работы программы в соответствии с синтаксисом и принципом работы пакета, провел результаты, по которым провел соответствующие выводы.

1. Максимальное ускорение для n = 1000 достигается при p = 32
2. Максимальное ускорение для n = 10000 достигается при p = 64
3. Максимальное ускорение для n = 100000 достигается при p = 64
4. Максимальное ускорение для n = 1000000 достигается при p = 64
5. Максимальное ускорение для n = 10000000 достигается при p = 64
6. Самым эффективный алгоритм, работающий на p = 4, самый неэффективны работающий на p = 64
7. Самым не затратным алгоритм, работающий на p = 4, самый затратный алгоритм работающий на p = 64

# **Список используемых информационных источников**

1. Сыромятников В. П. Курс лекций по дисциплине «Параллельное программирование». – РТУ МИРЭА, 2020-2021 г.

2. Руководство по использованию пакета MPI [Электронный ресурс]. – Режим доступа: <http://appmath.mrsu.ru/assets/templates/appmath/pdf_docs/ParProg_MPI_OpenMP.pdf> (дата обращения 30.10.2020)

3. Блок-схемы алгоритмов ГОСТ 19.701-90 «Схемы алгоритмов программ, данных и систем» [Электронный ресурс]. – Режим доступа: https://pro-prof.com/archives/1462 (дата обращения 01.11.2020)

4. Алгоритм вычисления интеграла методов Симпсона [Электронный ресурс]. – Режим доступа:[https://zaochnik.com/spravochnik/matematika/integraly-integrirovanie/metod-simpsona-parabol/](https://zaochnik.com/spravochnik/matematika/integraly-integrirovanie/metod-simpsona-parabol/%20) (дата обращения 13.11.2020).