Министерство науки и высшего образования Российской Федерации
Новосибирский Государственный технический университет
Кафедра автоматизированных систем управления

Отчет по лабораторной работе №5
по дисциплине «Параллельное программирование»
«Знакомство с MPI»
Вариант - 7

Выполнили
студенты группы АВТ-813:
Кинчаров Данил
Пайхаев Алексей
Чернаков Кирилл
Преподаватель:
Ландовский Владимир Владимирович,
к.т.н., доцент кафедры АСУ

г. Новосибирск
2020 г.

Содержание

1. Постановка задачи:

Реализовать параллельный алгоритм численного интегрирования методом трапеций с помощью MPI.

2. Задание:

Нулевой процесс отправляет границы интервалов остальным процессам, использовать рассылку (Scatter);

Для получения окончательного результата выполняется сборка данных (Gather) и суммирование нулевым процессом.

3. Пример работы программы:

В нашем случае MPI с помощью нескольких процессов считает определённый интеграл методом трапеций, и нулевой процесс рассылает границы интервалов остальным процессам, использовать рассылку (Scatter), а для получения окончательного результата выполняется сборка данных (Gather) и суммирование нулевым процессом.

Подынтегральная функция:

$$f(x) = \frac{x * \sqrt{x}}{\log x}$$

Пределы интегрирования:

$$a = 2, b = 5$$

Шаг:

$$1 \text{ x } 10^{-3}$$

В таблице на рисунке 1 приведены результаты выполнения программы, выполнявшейся с использованием процессора 4/8 (Intel core i7-7700HQ), а также график зависимости времени от количества процессов.

Кол-во потоков	Время, мс	Сумма
1	755	23.4974
2	386	23.4974
3	261	23.4974
4	200	23.4974
6	136	23.4974
8	129	23.4974
16	145	23.4974
20	145	23.4974

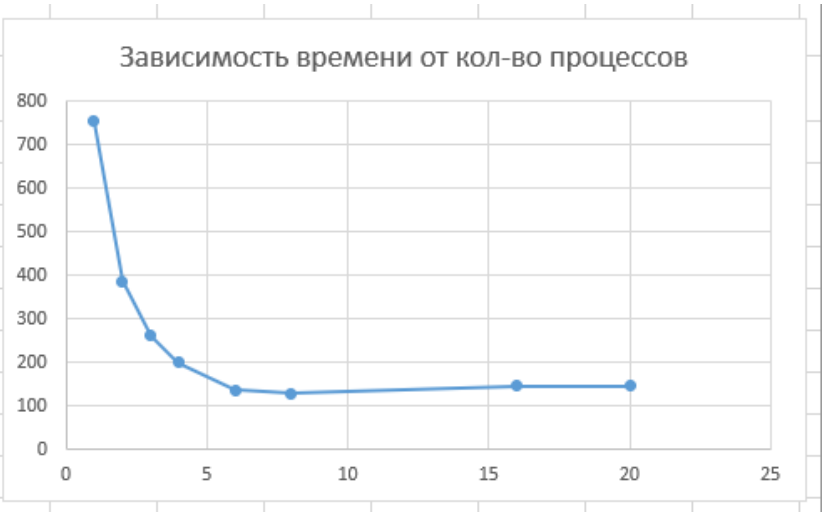

Рисунок 1 – Таблица с полученными данными и график зависимости времени от количества процессов

```
D:\Project\AVT_813_5SEM\parallel programming\LAB5\code\Debug>mpiexec -np 1 code.exe
-a = 2 -b = 6 -s = 4e-07
0- Trapes : 23.4974
Trapes : 23.4974
Time : 755 ms
```

Рисунок 2 – Результат работы программы при использовании 1 процесса

```
D:\Project\AVT_813_5SEM\parallel programming\LAB5\code\Debug>mpiexec -np 2 code.exe
-a = 2 -b = 4 -s = 4e-07
-a = 4 -b = 6 -s = 4e-07
0- Trapes : 9.57757
1- Trapes : 13.9198
Trapes : 23.4974
Time : 386 ms
```

Рисунок 3 – Результат работы программы при использовании 2 процесса

```
D:\Project\AVT_813_5SEM\parallel programming\LAB5\code\Debug>mpiexec -np 3 code.exe
-a = 2 -b = 3.33333 -s = 4e-07
-a = 4.66667 -b = 6 -s = 4e-07
-a = 3.33333 -b = 4.66667 -s = 4e-07
2- Trapes : 9.81733
1- Trapes : 7.70701
0- Trapes : 5.97303
Trapes : 23.4974
Time : 261 ms
```

Рисунок 4 – Результат работы программы при использовании 3 процесса

```
D:\Project\AVT_813_5SEM\parallel programming\LAB5\code\Debug>mpiexec -np 4 code.exe
-a = 2 -b = 3 -s = 4e-07
-a = 3 -b = 4 -s = 4e-07
-a = 4 -b = 5 -s = 4e-07
-a = 5 -b = 6 -s = 4e-07
0- Trapes : 4.34307
3- Trapes : 7.56908
2- Trapes : 6.35071
1- Trapes : 5.23451
Trapes : 23.4974
Time : 200 ms
```

Рисунок 5 – Результат работы программы при использовании 4 процесса

```
D:\Project\AVT_813_5SEM\parallel programming\LAB5\code\Debug>mpiexec -np 6 code.exe
-a = 2 -b = 2.66667 -s = 4e-07
-a = 2.66667 -b = 3.33333 -s = 4e-07
-a = 4.66667 -b = 5.33333 -s = 4e-07
-a = 3.33333 -b = 4 -s = 4e-07
-a = 5.33333 -b = 6 -s = 4e-07
-a = 4 -b = 4.66667 -s = 4e-07
0- Trapes : 2.81601
4- Trapes : 4.63212
5- Trapes : 5.18521
2- Trapes : 3.60455
3- Trapes : 4.10246
1- Trapes : 3.15702
Trapes : 23.4974
Time : 136 ms
```

Рисунок 6 – Результат работы программы при использовании 6 процессов

```
D:\Project\AVT_813_5SEM\parallel programming\LAB5\code\Debug>mpiexec -np 8 code.exe
-a = 2 -b = 2.5 -s = 4e-07
-a = 2.5 -b = 3 -s = 4e-07
-a = 3 -b = 3.5 -s = 4e-07
-a = 4 -b = 4.5 -s = 4e-07
-a = 5 -b = 5.5 -s = 4e-07
-a = 4.5 -b = 5 -s = 4e-07
-a = 3.5 -b = 4 -s = 4e-07
-a = 5.5 -b = 6 -s = 4e-07
1- Trapes : 2.25631
2- Trapes : 2.48669
0- Trapes : 2.08675
4- Trapes : 3.02824
6- Trapes : 3.62752
5- Trapes : 3.32247
3- Trapes : 2.74783
7- Trapes : 3.94156
Trapes : 23.4974
Time : 129 ms
```

Рисунок 7 – Результат работы программы при использовании 8 процессов

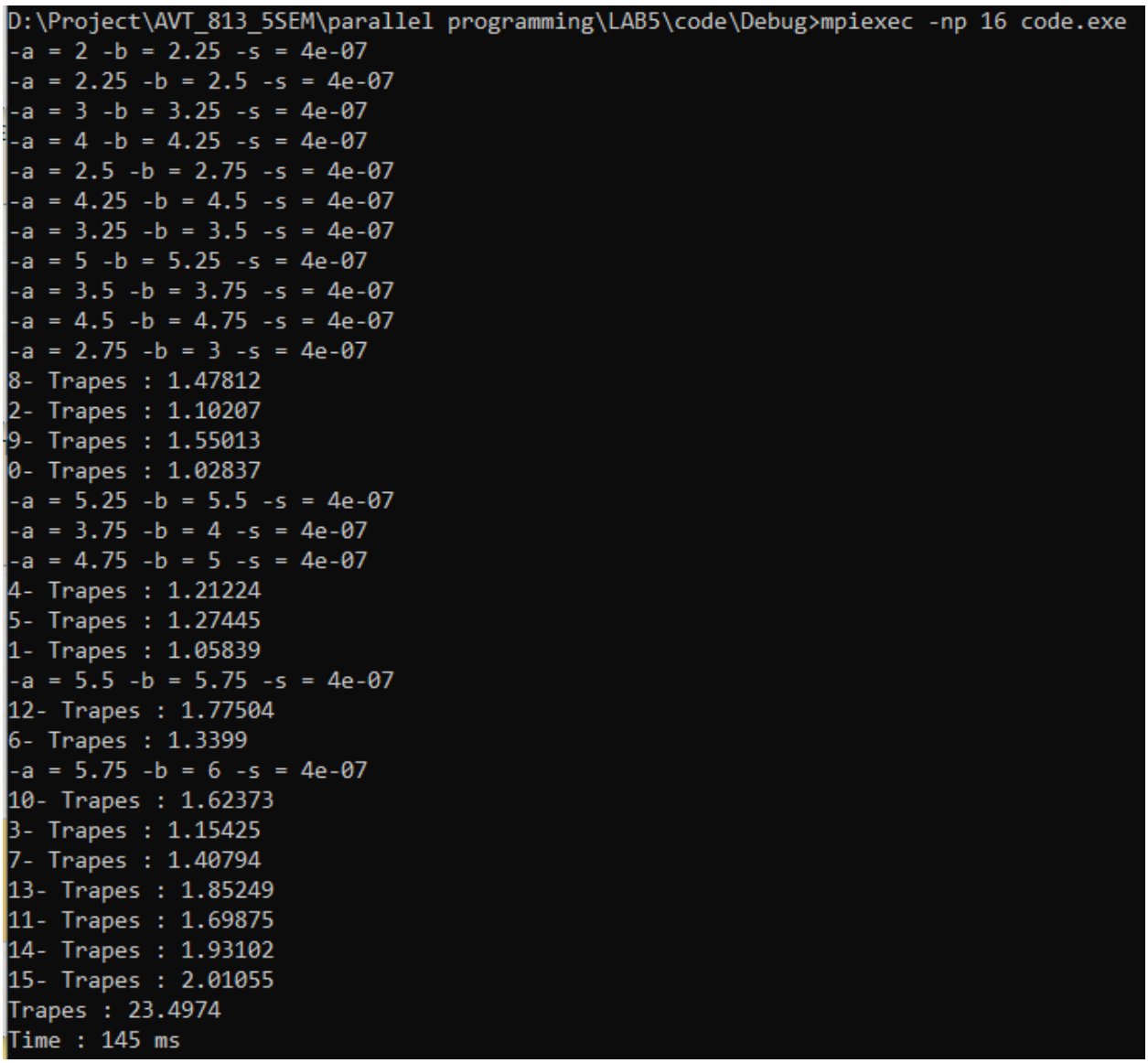

Рисунок 8 – Результат работы программы при использовании 16 процессов

4. Листинг программы:

```
#include <mpi.h>
#include <cmath>
#include <iostream>
#include <ctime>

double Fx(double x)
{
   return (sqrt(x) * x) / log(x);
}

double trapesIntegr(double a, double b, double shift)
{
   std::cout << "-a = " << a << " -b = " << b << " -s = " << shift << std::endl;

   double C0 = 3. / 8.;

   double h = (b - a) / shift;
   double result = 0;
   double x1 = a;
   double S = C0 * shift * result;

   for (int i = 0; i < h / 3; ++i)
   {
      result += (Fx(x1) + 3 * Fx(x1 + shift) + 3 * Fx(x1 + shift * 2) + Fx(x1 + s
hift * 3));
      x1 = x1 + shift * 3;
   }

   S = C0 * shift * result;

   return S;

}

void main(int argc, char** argv)
{
   MPI_Init(&argc, &argv);

   double a = 2, b = 6;

   int rank;
   int numTasks;

   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   MPI_Comm_size(MPI_COMM_WORLD, &numTasks);

   const int numArgs = 3;
   double* sendBuf = new double[numArgs * numTasks];
```

```
   double recvBuffer[numArgs];

   double step = (b - a) / numTasks;

   double shift = (b - a) / 10000000.;


   clock_t time = clock();

   if (rank == 0) {
      for (size_t i = 0; i < numTasks * 3; i += 3) {
         sendBuf[i] = a + i / 3 * step;
         sendBuf[i + 1] = sendBuf[i] + step;
         sendBuf[i + 2] = shift;
      }
   }

   MPI_Scatter(sendBuf, 3, MPI_DOUBLE, recvBuffer, 3, MPI_DOUBLE, 0, MPI_COMM_WOR
LD);

   double res = trapesIntegr(recvBuffer[0], recvBuffer[1], recvBuffer[2]);

   std::cout << rank << "- Trapes : " << res << std::endl;

   double *curRes = new double[numTasks + 1];

    MPI_Gather(&res, 1, MPI_DOUBLE, curRes, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);

    if (rank == 0)
    {

       time = clock() - time;
       res = 0;

       for (int i = 0; i < numTasks; i++)
          res += curRes[i];

        std::cout << "Trapes : " << res << std::endl;
        std::cout << "Time : " << time << " ms" << std::endl;
     }

     delete[] curRes;

   MPI_Finalize();
}
```

6. Вывод:

В ходе лабораторной работы была написана программа, с помощью которой осуществляется подсчёт определённого интеграла при помощи MPI с множеством процессов и метода трапеций.

Исходя из результатов тестирования, можно сделать вывод о том, что в результате работы программы при увеличении числа процессов уменьшается время работы программы, а после превышения числа доступных для процессора процессов идёт увеличение времени работы программы, так как процессы ожидают освобождения вычислительной мощности процессора.