Министерство науки и высшего образования Российской Федерации НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ) Институт прикладной математики и компьютерных наук

**ЛАБОРАТОРНАЯ РАБОТА №3**

по дисциплине «Параллельное программирование»

по теме «Вычисление определенных интегралов»

Выполнил:

Студент гр. 931801:

Рудов В.А

Проверил:

Преподаватель:

Данилкин Е. А.

Томск – 2021

Задача

Написать MPI-программу приближенного вычисления определенного интеграла с точностью ɛ=10e-5, используя обобщенную **формулу средних прямоугольников.**

Для контроля точности использовать результат расчета на MathCad. Оценить ускорение и эффективность параллельной программы.

**Вариант работы:**



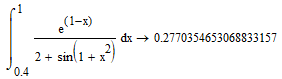
Код решения:

1. #include <cstdlib>
2. #include <stdio.h>
3. #include <mpi.h>
4. #include <cmath>
6. double f(double x)
7. {
8. return exp(1 - x) / (2 + sin(1 + x \* x));
9. }
11. int main(int argc, char\*\* argv)
12. {
13. int rank, size;
15. **MPI\_Status status;**
16. MPI\_Init(&argc, &argv);
17. MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);
18. MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);
20. **const double int\_len = 1e-9;**
21. const double a = 0.4;
22. const double b = 1;
23. const int int\_n = (int)((b - a) / int\_len);
25. **double int\_h = 0.0;**
26. double X = 0.0;
27. double local\_sum;
28. double global\_sum;
30. **double zerotime;**
32. if (rank == 0)
33. {
34. printf("a = %.3f**\n**", a);
35. **printf("b = %.3f\n", b);**
36. printf("Number of intervals: = %i**\n**", int\_n);
37. zerotime = MPI\_Wtime();
38. }
40. **for (int interval = 1 + rank; interval <= int\_n; interval += size)**
41. {
42. X = a + (int\_len \* ((double)interval - 0.5));
43. int\_h += f(X);
44. }
46. local\_sum = int\_len \* int\_h;
47. MPI\_Reduce(&local\_sum, &global\_sum, 1, MPI\_DOUBLE, MPI\_SUM, 0, MPI\_COMM\_WORLD);
49. if (rank == 0)
50. **{**
51. printf("Integral value: %.16f**\n**", global\_sum);
52. printf("Time: %f sec **\n**", MPI\_Wtime() - zerotime);
53. }
54. MPI\_Finalize();
55. **}**

Результаты работы алгоритма:

|  |  |  |  |
| --- | --- | --- | --- |
| **Установки** | **Результат** | **Ускорение** | **Эффективность** |
| Size = 1  N = |  | 1.00 | 13.55 |
| Size = 2  N = |  | 2.00 | 3.38 |
| Size = 4  N = |  | 3.97 | 0.85 |
| Size = 8  N = |  | 7.97 | 0.21 |
| Size = 16  N = |  | 13.96 | 0.06 |
| Size = 1  N = |  | 1.00 | 0.013 |
| Size = 2  N = |  | 1.94 | 0.0035 |
| Size = 4  N = |  | 3.31 | 0.001 |
| Size = 8  N = |  | 5.0 | 0.00025 |
| Size = 16  N = |  | 1.7 | 0.0005 |

Проверка в среде MathCad:



**Результаты верны.**

Вывод:

Использование распараллеливания для вычисления определенных интегралов является вполне эффективным. В ходе исследования было замечено, что на время подсчёта влияет 2 фактора:

1. Количество выделяемых потоков
2. Конечная точность интегрирования

Для получения результатов с высокой точностью за максимально короткое время выполнения рекомендуется использовать наибольшее число вычислительных потоков.