**Вариант 19:**

Интеграл:

Метод центральных прямоугольников

**Аналитическое решение:**

Первообразная от

**Результаты:**

*Последовательная программа:*

|  |  |  |
| --- | --- | --- |
| *Число разбиений* | *Время выполнения в секундах* | *Ошибка* |
| 100 | 1.8e-06 | 3.105065e-06 |
| 1000 | 1.8e-05 | 7.031819e-08 |
| 10000 | 0.0001513 | 1.020724e-07 |
| 100000 | 0.0018103 | 1.023900e-07 |
| 1000000 | 0.0167176 | 1.023931e-07 |

*Программа с использованием потоков и векторных инструкций:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Число разбиений* | *Время выполнения в секундах* | | *Ошибка* | |
| *Без потоков* | *С потоками* | *Без потоков* | *С потоками* |
| 100 | 8.8e-06 | 6.15e-05 | 3.105064643627031273354078e-06 | 2.5907e-08 |
| 1000 | 4.52e-05 | 0.0002299 | 7.031818682889934279955924e-08 | 1.0111e-07 |
| 10000 | 0.0003913 | 0.0088852 | 1.020724460509825348708546e-07 | 1.0238e-07 |
| 100000 | 0.0041729 | 0.029731 | 1.023899560692598242894746e-07 | 1.0239e-07 |
| 1000000 | 0.044996 | 0.21609 | 1.023930811250295391801046e-07 | 1.0239e-07 |
| 10000000 | 0.44016 | 2.3523 | 1.023931095467389695841121e-07 | 1.0239e-07 |

Добавление #pragma loop(no\_vector) и #pragma loop(hint\_parallel(4)) не привело к изменению результатов => в данной задаче их нельзя применить.

*Программа с использованием дополнений Intel Cilk Plus:*

|  |  |  |
| --- | --- | --- |
| *Число разбиений* | *Время выполнения в секундах* | *Ошибка* |
| 100 | 0.0082439 | 3.10506e-06 |
| 1000 | 3.07e-05 | 7.03182e-08 |
| 10000 | 9.55e-05 | 1.02072e-07 |
| 100000 | 0.000218 | 1.0239e-07 |
| 1000000 | 0.0006417 | 1.02393e-07 |

*Программа с использованием TBB:*

|  |  |  |
| --- | --- | --- |
| *Число разбиений* | *Время выполнения в секундах* | *Ошибка* |
| 100 | 0.014253 | 0.0015405 |
| 1000 | 0.0002938 | 0.000154335 |
| 10000 | 0.0004165 | 1.5575e-05 |
| 100000 | 0.0016921 | 1.26984e-06 |
| 1000000 | 0.0089886 | 7.77435e-08 |

**Что такое Cilk Plus?**

Cilk Plus –это расширения языка C/C++, которое помогает с введением параллелизма в код программы, используя всего три ключевых слова:

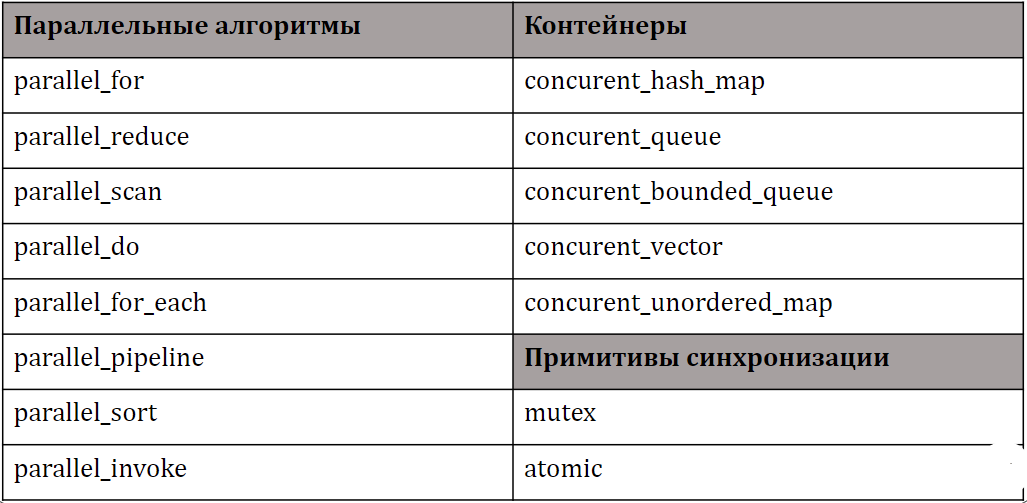
* cilk\_spawn
* cilk\_sync
* cilk\_for

Планировщик CilkPlus автоматически решает проблему балансировки нагрузки на ядра процессора. Ключевое слово cilk\_spawn определяет возможность асинхронного запуска. Ключевое слово cilk\_sync предписывает завершение всех инструкций, заключенных между слов cilk\_spawn и cilk\_sync, перед тем как продолжить исполнение программы. Ключевое слово reducer используется для того, чтобы избежать такой ошибки параллельного программирования как гонка данных -ошибка, которая связана с несинхронизованностью потоков команд, работающих с общими данными.

**Что такое TBB?**

Treading Building Blocks (TBB) является частью Intel Parallel Studio, содержит следующие компоненты:

* Параллельные алгоритмы
* Планировщик задач
* Контейнеры
* Инструменты для работы с памятью
* Низкоуровневые примитивы синхронизации



**Реализация с Cilk:**

#include <cilk/cilk.h>

#include <stdlib.h>

#include <iostream>

#include <math.h>

#include <chrono>

#include <cilk/reducer\_opadd.h>

using namespace std;

using namespace std::chrono;

# define M\_PI 3.14159265358979323846 /\* pi, #include <math.h> \*/

double refer = 2.094395;

double func(double x)//Функция, от которой вычисляем интеграл

{

return 4 / (sqrt(4 - x \* x));

}

int main()

{

double double\_integral;

int a = 0;

int b = 1;

int q[5] = { 100, 1000, 10000, 100000, 1000000 };

double h[5];

for (int i = 0; i < 5; i++)

h[i] = (double)(b - a) / (double)q[i];

cout << " Value " << "|" << " Error " << "|" << "Time" << endl;

for (int j = 0; j < 5; j++)

{

high\_resolution\_clock::time\_point t1 = high\_resolution\_clock::now();

cilk::reducer\_opadd<double> integral(0); //Объявляем reducer, инициализируем значением 0

cilk\_for (int i = 0; i < q[j]; i++)//Параллельный цикл

{

integral = integral + func((double)(a + i \* h[j] + h[j]/2))\*h[j];

}

double\_integral = integral.get\_value();

high\_resolution\_clock::time\_point t2 = high\_resolution\_clock::now();

duration<double> duration = (t2 - t1);

cout << double\_integral << "|" << abs(refer - double\_integral) << "|" << duration.count() << " seconds" << endl;

}

return 0;

}

**Реализация с TBB:**

#include <cilk/cilk.h>

#include <stdlib.h>

#include <iostream>

#include <math.h>

#include <chrono>

#include "tbb/task\_scheduler\_init.h"

#include "tbb/parallel\_for.h"

#include "tbb/blocked\_range.h"

#include "tbb/scalable\_allocator.h"

#include "tbb/parallel\_reduce.h"

using namespace tbb;

using namespace std;

using namespace std::chrono;

# define M\_PI 3.14159265358979323846 /\* pi, #include <math.h> \*/

double a = 0.0;

double b = 1.0;

double refer = 2.094395;

double func(double x) //Функция, от которой вычисляем интеграл

{

return 4 / (sqrt(4 - x \* x));

}

int main()

{

int q[5] = { 100, 1000, 10000, 100000, 1000000 };

double h[5];

for (int i = 0; i < 5; i++)

h[i] = (double)(b - a) / (double)q[i];

cout << " Value " << "|" << " Error " << "|" << "Time" << endl;

for (int j = 0; j < 5; j++)

{

high\_resolution\_clock::time\_point t1 = high\_resolution\_clock::now();

double sum;

sum = parallel\_reduce//Используем алгоритм parallel\_reduce

(

blocked\_range<int>(0, q[j]), //Диапазон

0.f,

[&](blocked\_range<int> range, double integral) -> double

{

for (int i = range.begin(); i < range.end(); i++)

{

integral = integral + func((double)(a + (double)i \* h[j])) \* h[j];

}

return integral;

},

std::plus<double>()

);

double integral = sum;

high\_resolution\_clock::time\_point t2 = high\_resolution\_clock::now();

duration<double> duration = (t2 - t1);

cout << integral << "|" << refer - integral << "|" << duration.count() << " seconds" << endl;

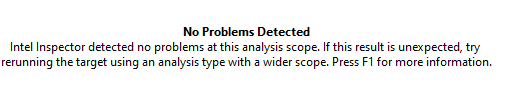
}

return 0;

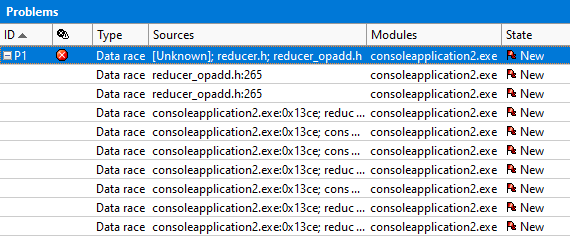
}

**Анализ реализации с использованием дополнений Intel Cilk Plus:**

Intel Inspector. Locate Memory Problems

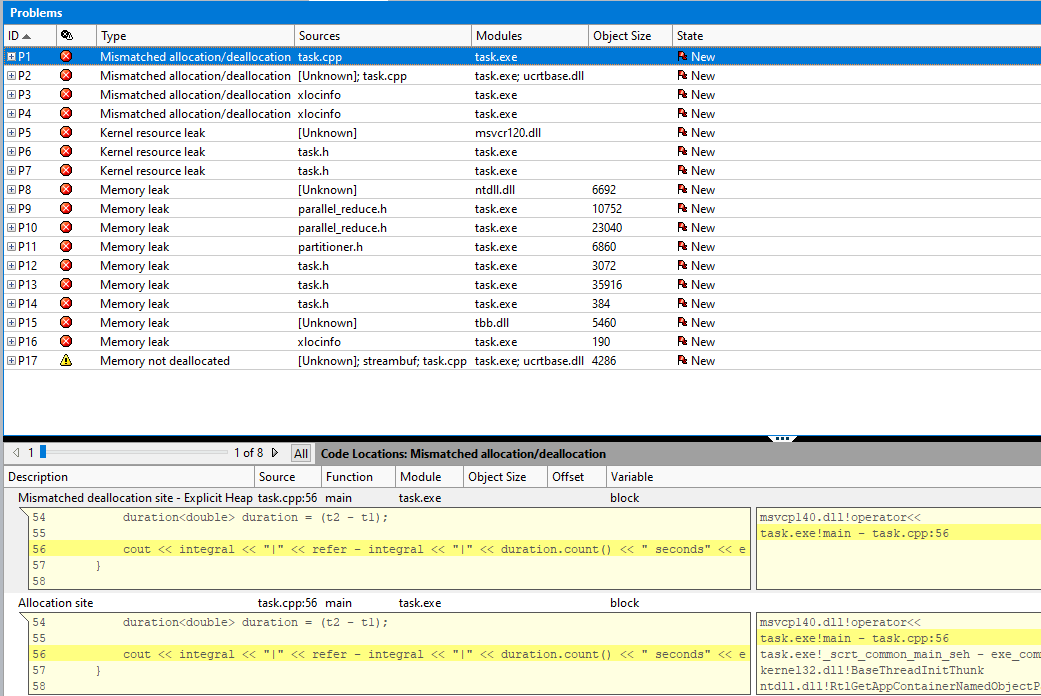


Intel Inspector. Locate Deadlocks and Data Races



**Анализ реализации с использованием дополнений TBB:**

Intel Inspector. Locate Memory Problems



Intel Inspector. Locate Deadlocks and Data Races

