

**Напишите параллельную программу вычисления следующего интеграла с использованием дополнений Intel Cilk Plus языка C++:**

$$\int_0^1 \frac{4}{\sqrt{4-x^2}} dx$$

В математическом анализе обосновывается аналитический способ нахождения значения интеграла с помощью формулы Ньютона-Лейбница

$$\int_a^b f(x) dx = F(b) - F(a)$$

Однако применение данного подхода к вычислению наталкивается на несколько серьезных препятствий:

- Для многих функций не существует первообразной среди элементарных функций;
- Даже если первообразная для заданной функции существует, то вычисление двух ее значений  $F(a)$ ,  $F(b)$  может оказаться более трудоемким, чем вычисление существенно большего количества значений  $f(x)$ ;
- Для многих реальных приложений определенного интеграла характерная дискретность задания подынтегральной функции, что делает аналитический подход неприменимым.

Сказанное предопределяет необходимость использования приближенных формул для вычисления определенного интеграла на основе значений подынтегральной функции. Такие специальные формулы называются квадратурными формулами или формулами численного интегрирования.

В качестве инструментов параллелизации, используемых для решения задачи, мы будем использовать:

- Cilk Plus – это расширения языка C/C++, которое помогает с введением параллелизма в код программы;
- Intel Parallel Studio XE – это набор программных продуктов от компании Intel;
  1. Intel Parallel Inspector – инструмент, предназначенный для тестирования работающей программы с целью выявления основных ошибок, которые возникают при разработке параллельного кода;

2. Intel Amplifier – инструмент, который используется для профилирования приложения с целью выявления наиболее часто используемых участков программы (hotspots), а также узких мест (bottleneck) в работе программы. Этот инструмент также позволяет анализировать параллельные программы на эффективность использования ими ресурсов процессора;

**Реализуем вычисление интеграла с помощью формулы средних прямоугольников.**

```
// Последовательное вычисление интеграла
// по формуле средних прямоугольников
double intergal(int n)
{
    double sum = 0.0;
    double h = (PREDEL_B - PREDEL_A) / (n-1);
    for (auto i = 0; i < n; i++)
        sum += func(i * h) + func((i+1)*h);
    return sum * h / 2.0;
}
```

**С помощью инструмента Amplifier XE определим наиболее часто используемые участки кода.**

**Basic Hotspots** Hotspots viewpoint (change) ?

Analysis Target | Analysis Type | Collection Log | Summary | Bottom-up | Caller/Callee | Top-down Tree | Tasks and Frames

**Elapsed Time:** 0.930s

- CPU Time: 0.612s
- Total Thread Count: 1
- Paused Time: 0s

**Top Hotspots**

This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

Function	CPU Time
<a href="#">func</a>	0.329s
<a href="#">sqrt</a>	0.130s
<a href="#">intergal</a>	0.090s
<a href="#">RTC_CheckEsp</a>	0.037s
<a href="#">sqrt</a>	0.016s
[Others]	0.010s

**Collection and Platform Info**

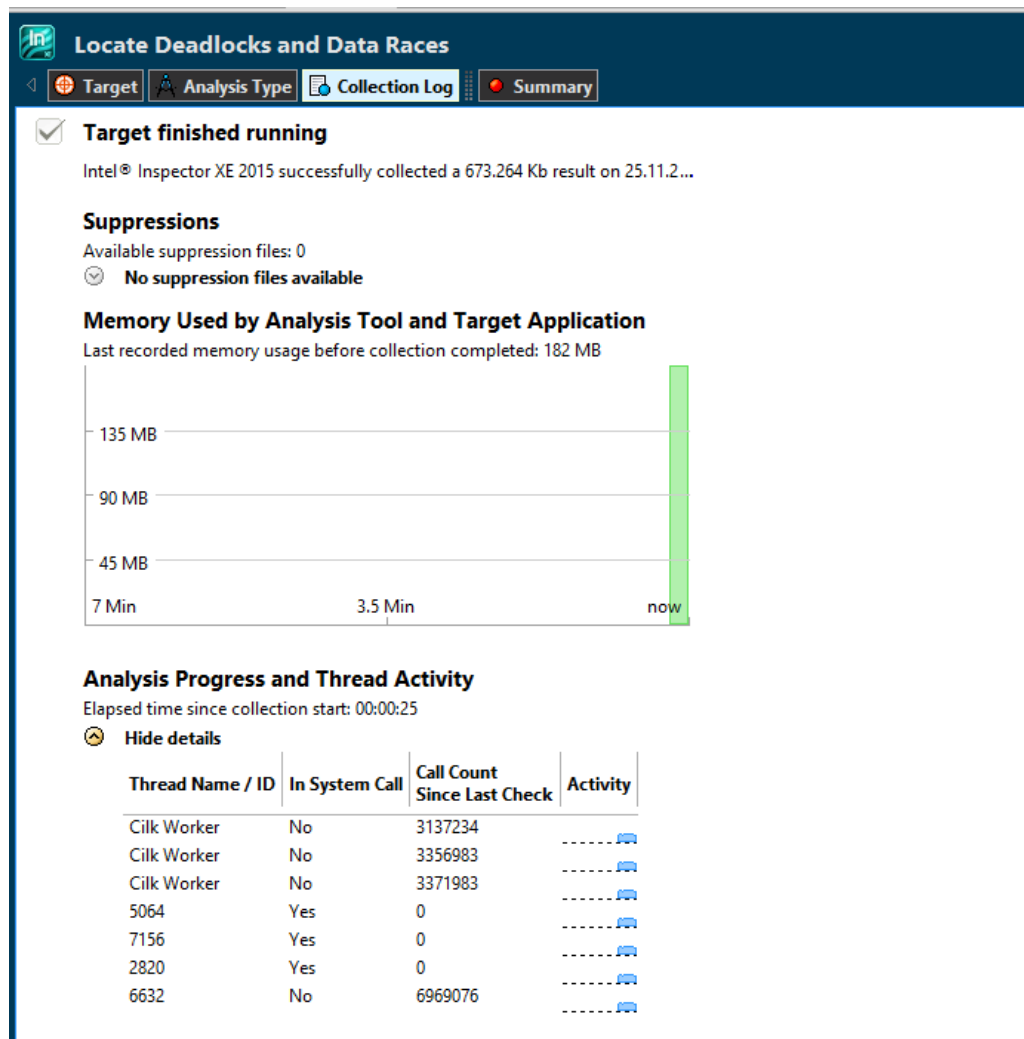
This section provides information about this collection, including result set size and collection platform data.

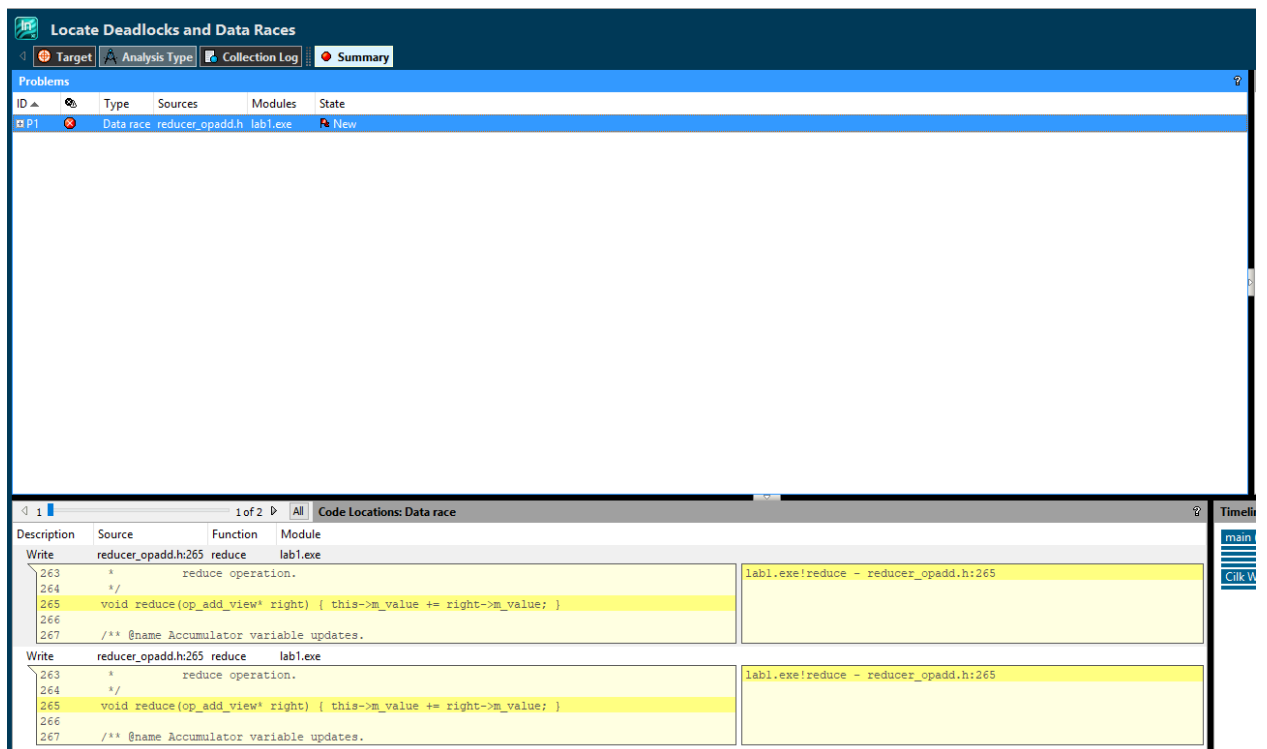
- Application Command Line: D:\Документы\11 semester\131045\IPS\ips\_lab\Debug\lab1.exe
- Environment Variables:
- Operating System: Microsoft Windows 8
- Computer Name: DESKTOP-LO22S9L
- Result Size: 1 MB
- Collection start time: 08:50:48 25/11/2018 UTC
- Collection stop time: 08:50:49 25/11/2018 UTC
- CPU**
  - Name: 3rd generation Intel(R) Core(TM) Processor family
  - Frequency: 2.5 GHz
  - Logical CPU Count: 4

На основе последовательной функции создадим новую функцию, используя `cilk_for`.

```
// Параллельное вычисление интеграла
// по формуле средних прямоугольников
double parallel_integral(int n)
{
    double h = (PREDEL_B - PREDEL_A) / n;
    cilk::reducer_opadd<double> sum(0.0);
    cilk_for (auto i = 0; i < n; i++)
        sum += func(i * h) + func((i + 1)*h);
    return sum->get_value()*h / 2.0;
}
```

Далее, используя Inspector XE, определим те данные, которые принимают участие в гонке данных или в других основных ошибках, возникающих при разработке параллельных программ.





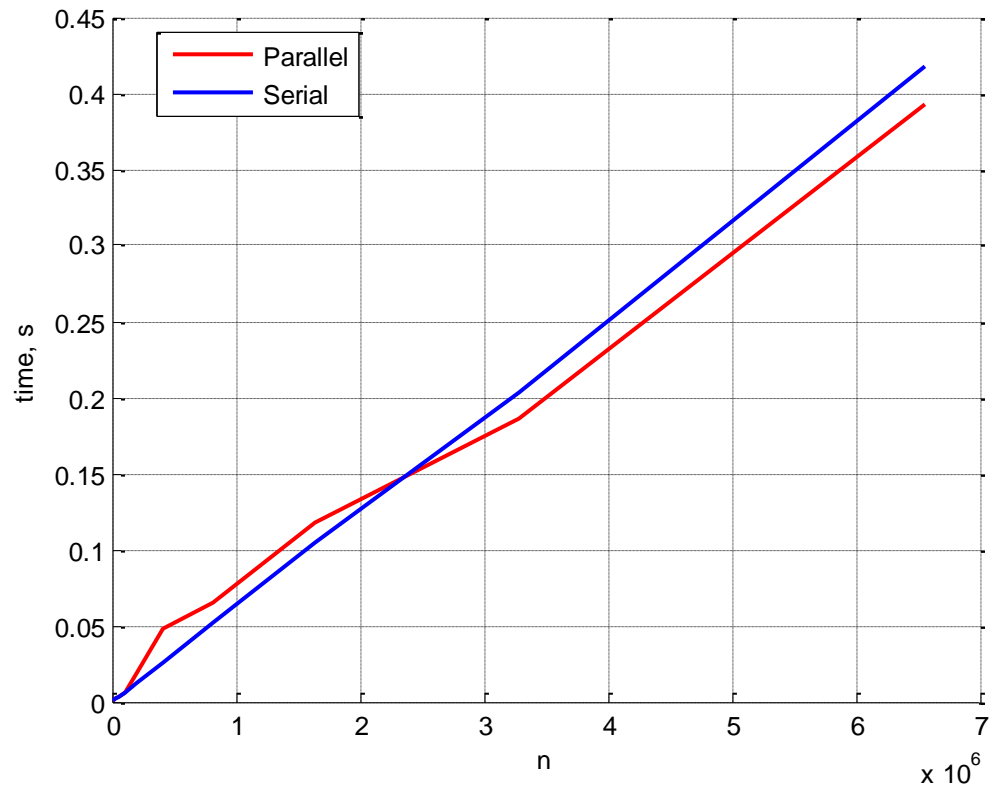
Как видно, ошибок не обнаружено.

Запустим полученную программу

```
C:\Windows\system32\cmd.exe

Integral = 2.0944
Elapsed time = 0.644898
Parallel Integral = 2.0944
Elapsed time = 0.561304
Boost = 1.14893
Для продолжения нажмите любую клавишу . . .
```

Построим график зависимости времени выполнения от заданных параметров алгоритма.



Как видно, при увеличении количества точек разбиения, параллельная реализация работает быстрее, чем последовательная.