Комплексное задание

Выполнил: Селиверстов Михаил

Группа: ИВТ-12М

Вариант 2

Все необходимые программные коды находятся на GitHub:

https://github.com/MichilMIET/complex_task.git

1. Последовательная программа по расчету интеграла

Аналитическое решение:

Для левых прямоуголников:

$$\int_0^1 \frac{4}{1+x^2}dx = h \cdot \sum_{i=0}^{n-1} f(x_i) = \frac{b-a}{n} \cdot \sum_{i=0}^{n-1} f\left(\frac{4}{1+x_i^2}\right)$$

Для правых прямоуголников:

$$\int_0^1 \frac{4}{1+x^2}dx = h \cdot \sum_{i=1}^{n} f(x_i) = \frac{b-a}{n} \cdot \sum_{i=1}^{n} f\left(\frac{4}{1+x_i^2}\right)$$

В результате выполнения программы, получаем для каждого из интервалов разбиения значения точности и времени выполнения. Точность я брал в процентах, на мой взгляд это более нагляднее, чем разница в значениях (где 100% полное соответствие с действительным значением). Ниже приведены результаты работы программы для каждого из интервалов.

```
Интервал разбиения: 100
Результат (правого): 3.131575986923128773753433
Точность (правого): 98.9983333333 %
Время выполнения (правого): 0.00002 сек
-----------------------------------------------
Результат (левого): 3.131575986923128773753433
Точность (левого): 98.9983333333 %
Время выполнения (левого): 0.00004 сек
```

Рисунок 1 - Результат работы программы в случае разбиения на 100

```
Интервал разбиения: 1000
Результат (правого): 3.140592486923122983455414 2
Точность (правого): 99.8999833333 %
Время выполнения (правого): 0.00012 сек
-----------------------------------------------
Результат (левого): 3.140592486923122983455414 2
Точность (левого): 99.8999833333 %
Время выполнения (левого): 0.00012 сек
```

Рисунок 2 - Результат работы программы в случае разбиения на 1000

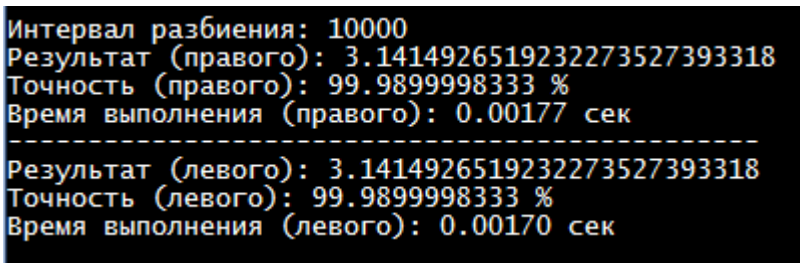

Рисунок 3 - Результат работы программы в случае разбиения на 10000

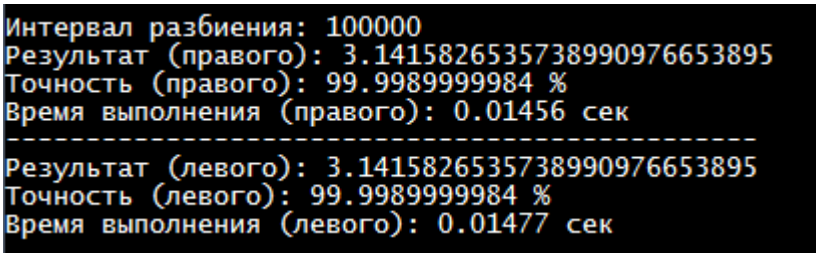

Рисунок 4 - Результат работы программы в случае разбиения на 100000

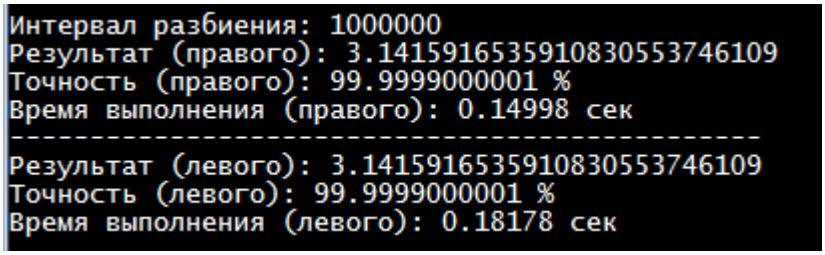

Рисунок 5 - Результат работы программы в случае разбиения на 1000000

Как видно, что с увеличением интервала, увеличивается время выполнения, поскольку алгоритму требуется больше времени на выполнения программы, но возрастает точность, поскольку если интервал сделать бесконечно большим, то число будет бесконечно близко к истинному.

2. Программа по расчету интеграла с использованием нескольких потоков и векторных инструкций

Далее я использовал потоки, а именно qpar, автоматическую векторизацию, совместно (thread, mutex).

Ниже представлены результаты работы программы для каждого интервала разбиения.

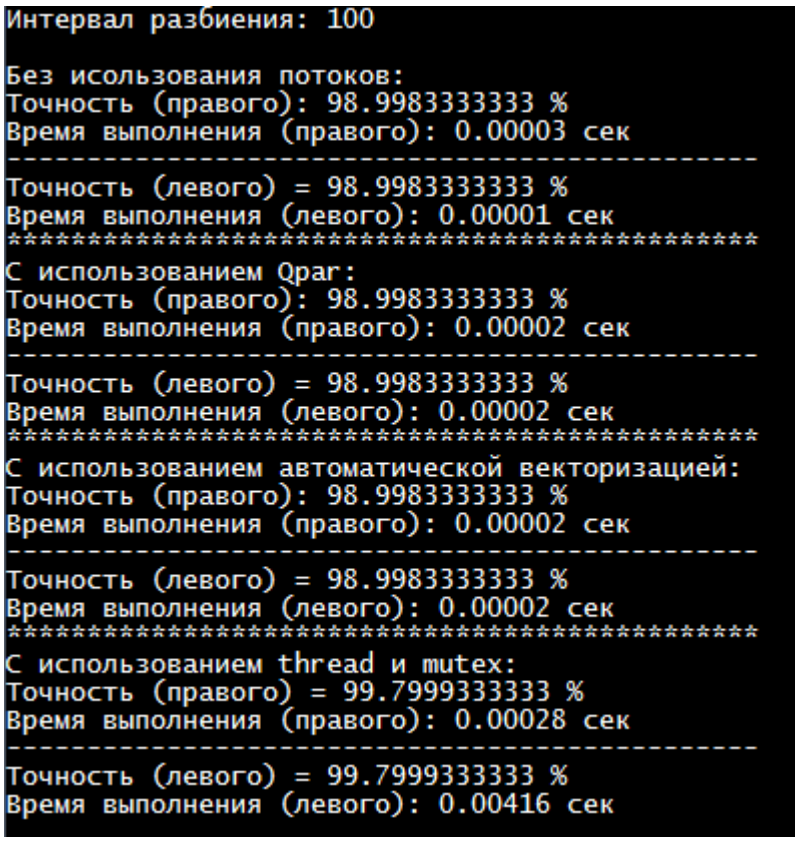

Рисунок 6 - Результат работы программы в случае разбиения на 100

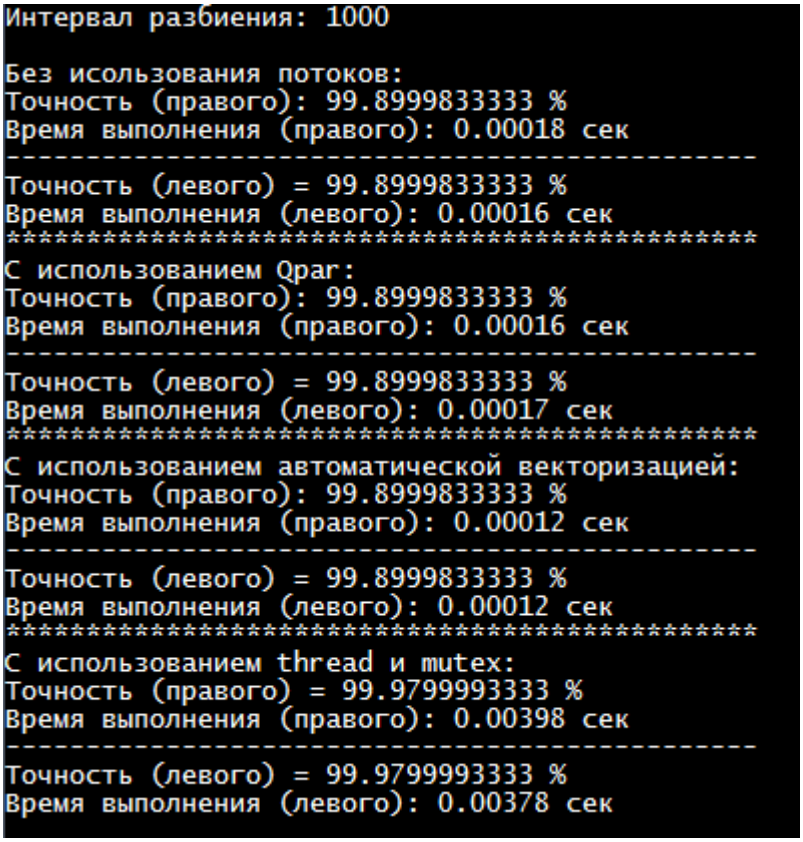

Рисунок 7 - Результат работы программы в случае разбиения на 1000

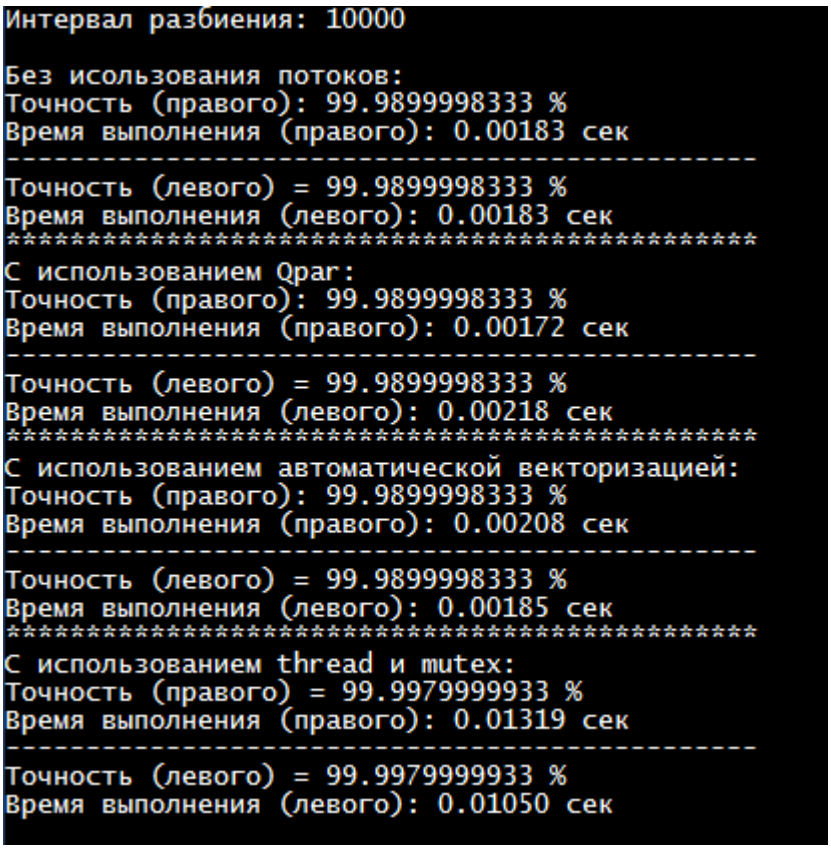

Рисунок 8 - Результат работы программы в случае разбиения на 10000

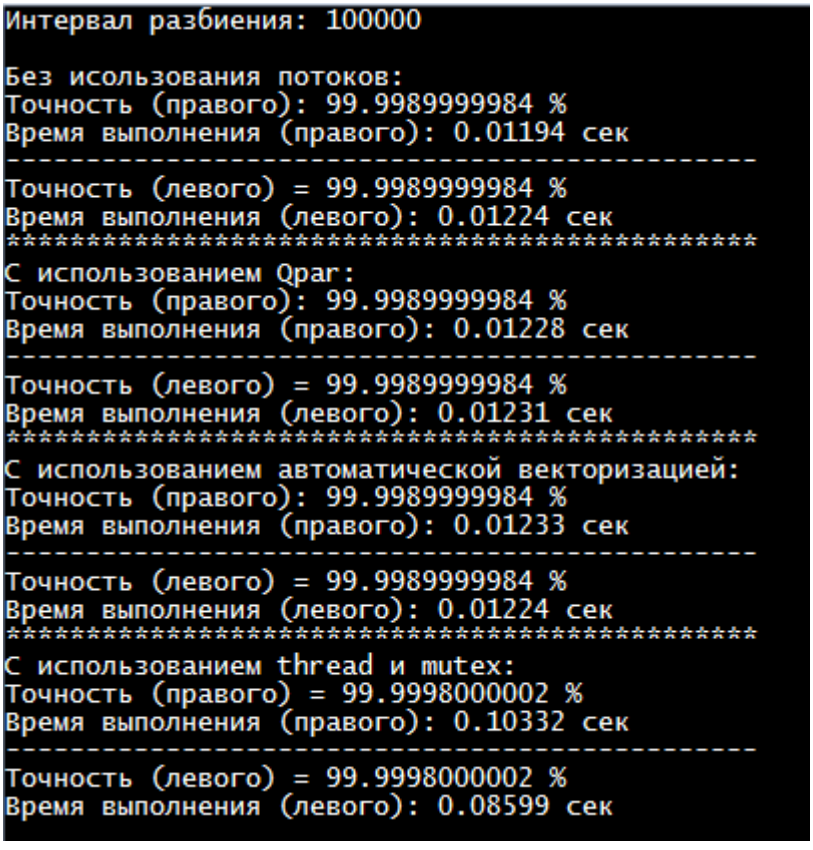

Рисунок 9 - Результат работы программы в случае разбиения на 100000

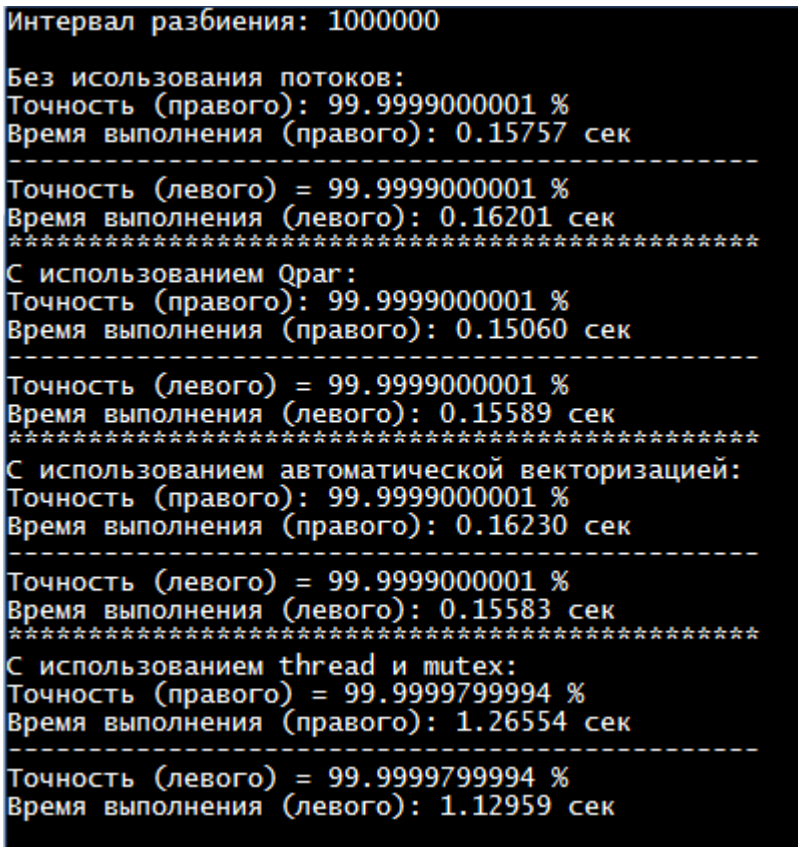

Рисунок 10 - Результат работы программы в случае разбиения на 1000000

Как видно, распараллеливание дает выигрыш во времени, но с увеличением интервала выигрыш становится не столь сильно заметен.

3. Программа с использованием дополнений Intel Cilk Plus языка C++

Далее я добавил цикл cilk_for, для распараллеливания вычислений в программе и объединил интервалы на одном скриншоте.

```
Интервал разбиения: 100

Результат (правого): 3.13959198692313 Время выполнения: 0.020171271 сек
Результат (левого): 3.13959198692313 Время выполнения: 0.001588887 сек
************************************************

Интервал разбиения: 1000

Результат (правого): 3.14139264692314 Время выполнения: 0.000395515 сек
Результат (левого): 3.14139264692314 Время выполнения: 0.000235517 сек
************************************************

Интервал разбиения: 10000

Результат (правого): 3.14157265352296 Время выполнения: 0.002607328 сек
Результат (левого): 3.14157265352296 Время выполнения: 0.002637194 сек
************************************************

Интервал разбиения: 100000

Результат (правого): 3.14159065359186 Время выполнения: 0.018837527 сек
Результат (левого): 3.14159065359186 Время выполнения: 0.017865592 сек
************************************************

Интервал разбиения: 1000000

Результат (правого): 3.14159245358412 Время выполнения: 0.184200815 сек
Результат (левого): 3.14159245358412 Время выполнения: 0.203764092 сек
************************************************
```

Рисунок 11 - Результат работы программы для всех интервалов разбиения

Как видно время затрачено значительно меньше, чем в предыдущих решениях, но все равно время увеличивается при увеличении интервала разбиения.

Далее использовал Intel Inspector для обнаружения ошибок распараллеливания. Ошибок не обнаружено.

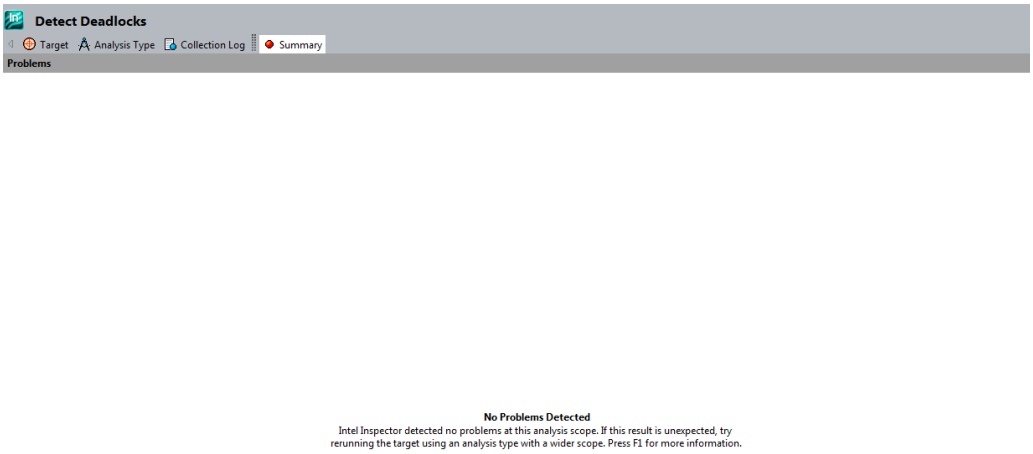

Рисунок 12 - Результат работы Inspector в режиме обнаружения ошибок в потоках

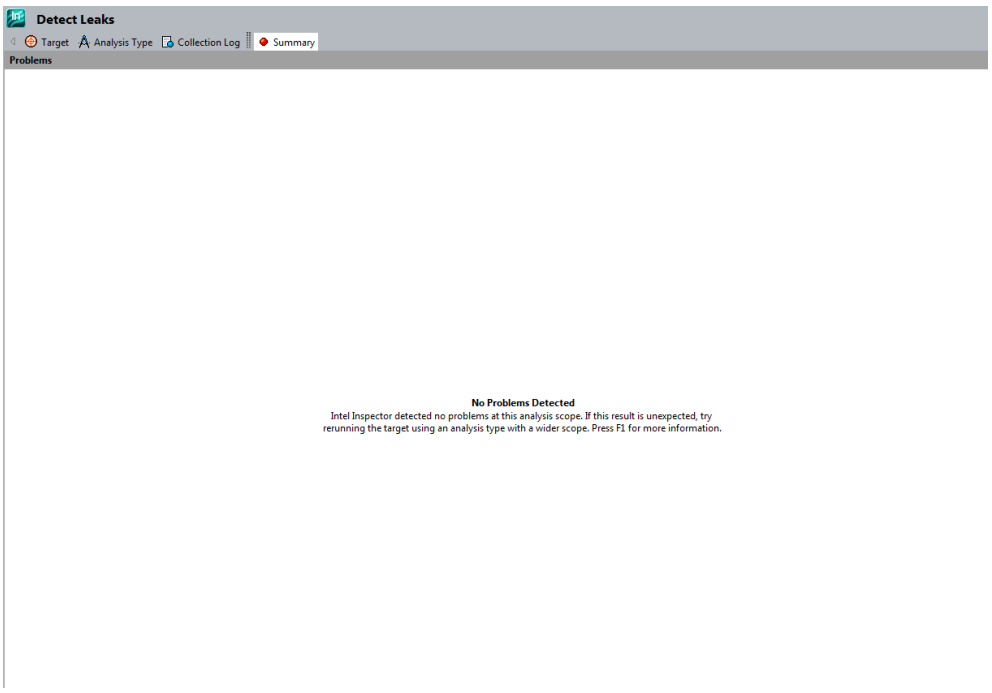

Рисунок 13 - Результат работы Inspector в режиме обнаружения ошибок в памяти

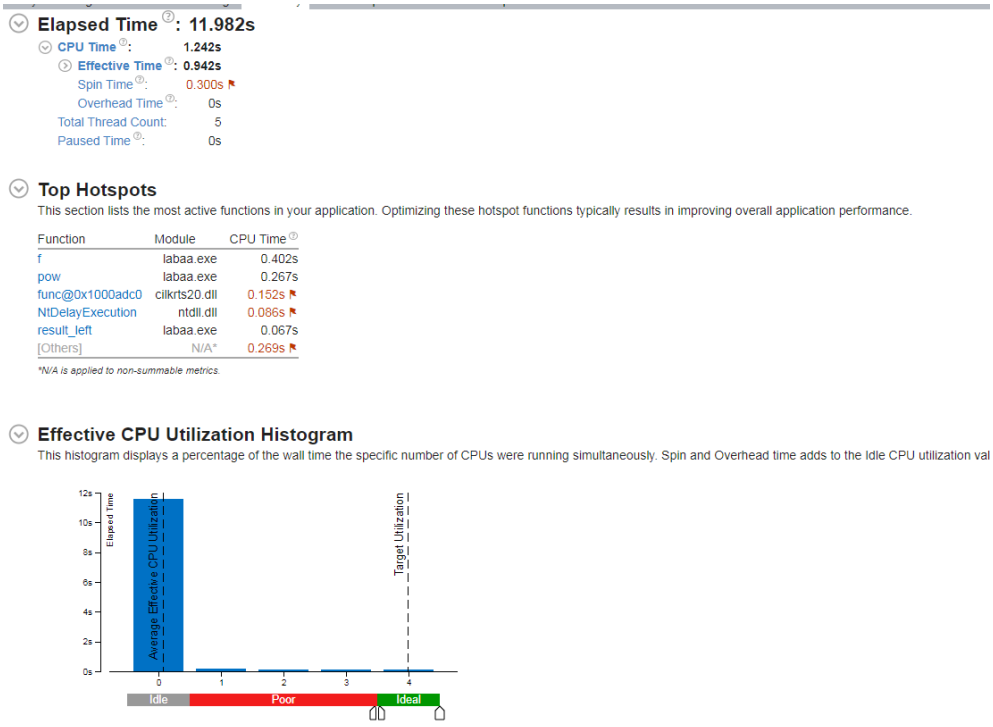

Рисунок 14 - Результат работы VTune, оценка затраченного времени

Вывод: в результате проделанной работы я выяснил как влияют на скорость работы программы параллельные функции. Использование библиотеки Intell cilk оказалось самым оптимальным в плане скорости работы методом.