Отчет по лабораторной работе №1

Сайфуллин Джамиль ИВТ-11М

Вариант 8

Технические характеристики

OS: Windows 10

Vs: Visual Studio 2017 v15.9.18

Ips: v2019

Железо: 2 ядра физических, 4 логических 3.4ГГц

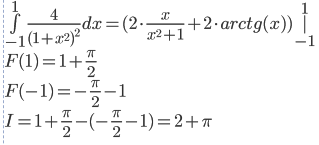
Интеграл

Решение методом центральных прямоугольников.

Разобьем отрезок на 4 части. Получаем отрезки: [-1;-0.5], [-0.5;0],[0,0.5],[0.5,1].

h =0.5

≈ h\*( f(-1+)+ f(-0.5+)+ f(0+)+ f(0.5+)) =5.18



Точное значение 2+π, полученная погрешность исправляется шагом разбиением.

# 1. Последовательная программа по расчету интеграла

В результате написания последовательной программы по расчету были получены следующие результаты, представленные на рисунке 1.

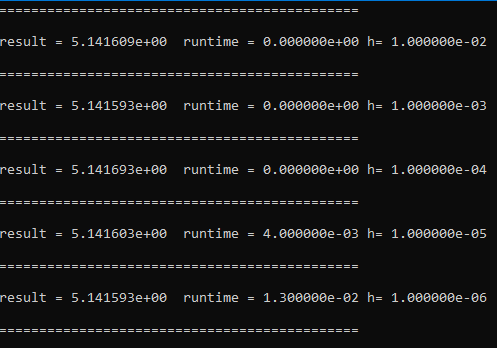


Рисунок 1 Вывод результатов работы последовательной программы

Как можно увидеть, с увеличением шага точность повысилась и приблизилась к аналитическому результату.

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

В результате написания программы по расчетуинтеграла с использованием нескольких потоков и векторных инструкций были получены следующие результаты, представленные на рисунке 2.

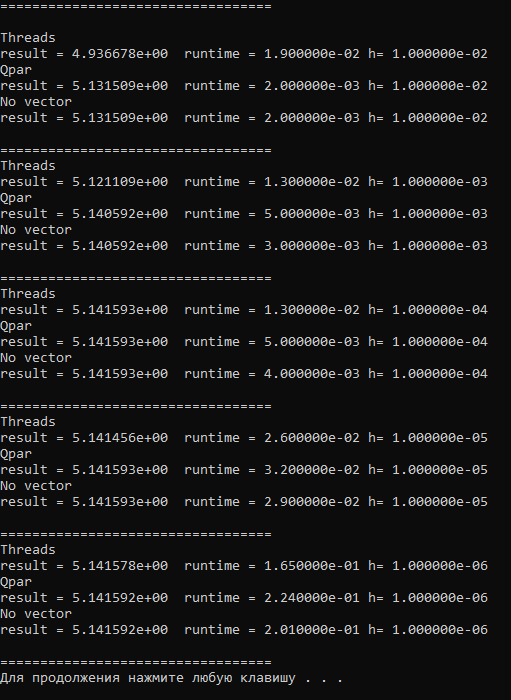


Рисунок 2 Вывод результатов работы программы по расчетуинтеграла с использованием нескольких потоков и векторных инструкций, количество потоков 8

Можно заметить, что с увеличением шага эффективность использования потоков возросла и интеграл наиболее быстро вычисляется при помощи потоков. Однако для использования потоков были необходимы дополнительные разбиения промежутка интегрирования, что вызвало погрешности на малых шагах. С уменьшением шага неточность устранилась. На рисунке 3 представлен результат работы программы, где количество создаваемых потоков было уменьшено с 8 до 4 и точность увеличилась.

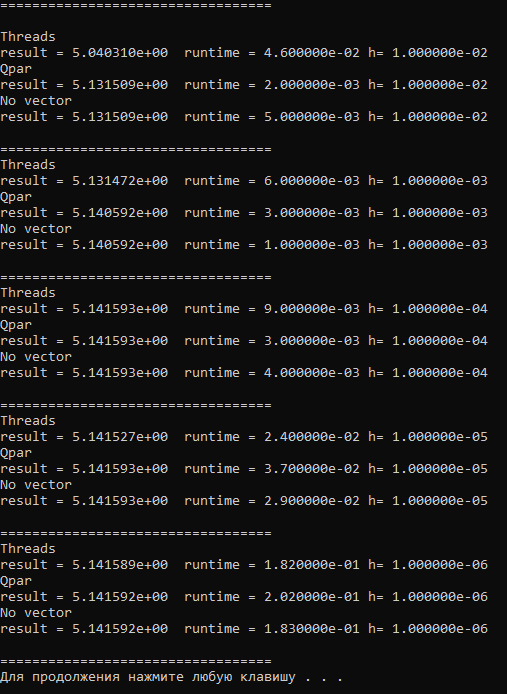


Рисунок 3 Вывод результатов работы программы по расчетуинтеграла с использованием нескольких потоков и векторных инструкций, количество потоков 4

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

Для улучшения производительности была использована библиотека Intel Cilk Plus.  
Стандартный цикл for был заменен на cilk\_for, что позволило распараллелить вычисление интеграла на 4 потока.

Однако результат при маленьком шаге стал неверным. Результат представлен на рисунке 4. Многократный запуск показал, что последние два значения при маленьком шаге могут колебаться от 2 до 8, включая и правильное значение.

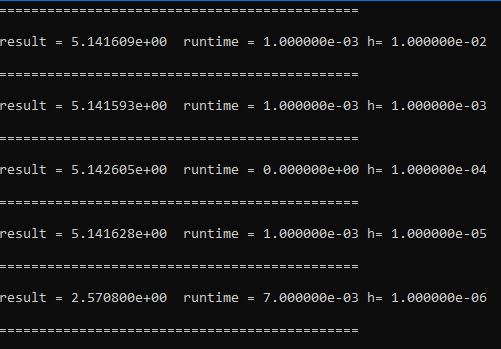


Рисунок 4 Вывод результатов работы программы *****с использованием cilk\_for*****

Для обнаружения ошибки распараллеливания был использован инструмент Intel Inspector. Однако никаких тупиков и гонок данных обнаружено не было. Результат использования инструмента представлен на рисунке 5.

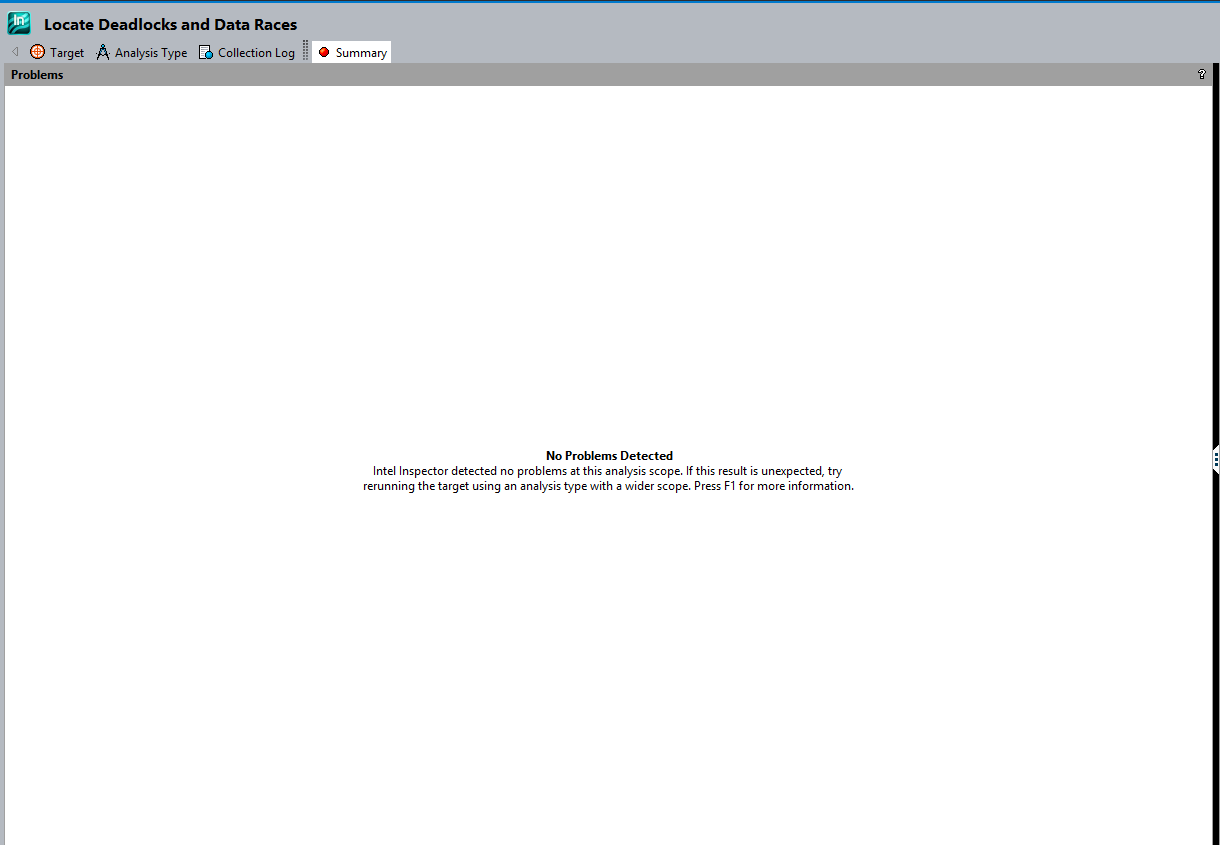


Рисунок 5 Результат работы Intel Inspector

Для устранения проблемы я попробовал добавить reducer. В результате ошибка исчезла. Вывод результатов программы на рисунке 6.

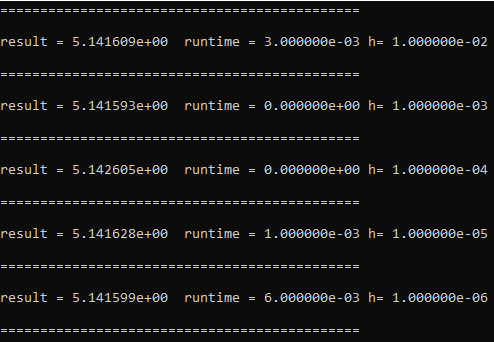


Рисунок 6 Вывод результатов работы программы *****с использованием cilk\_for и reducer*****

После этого я проанализировал программу на эффективность распараллеливания с помощью Vtune. Максимально мне удалось достичь значения распараллеливания в 41%, большую часть времени, как указывает программа, занимали выполнения вызовов функции cilk. Их распараллелить не представляется возможности. Результаты использования Vtune представлены на рисунках 7,8,9.

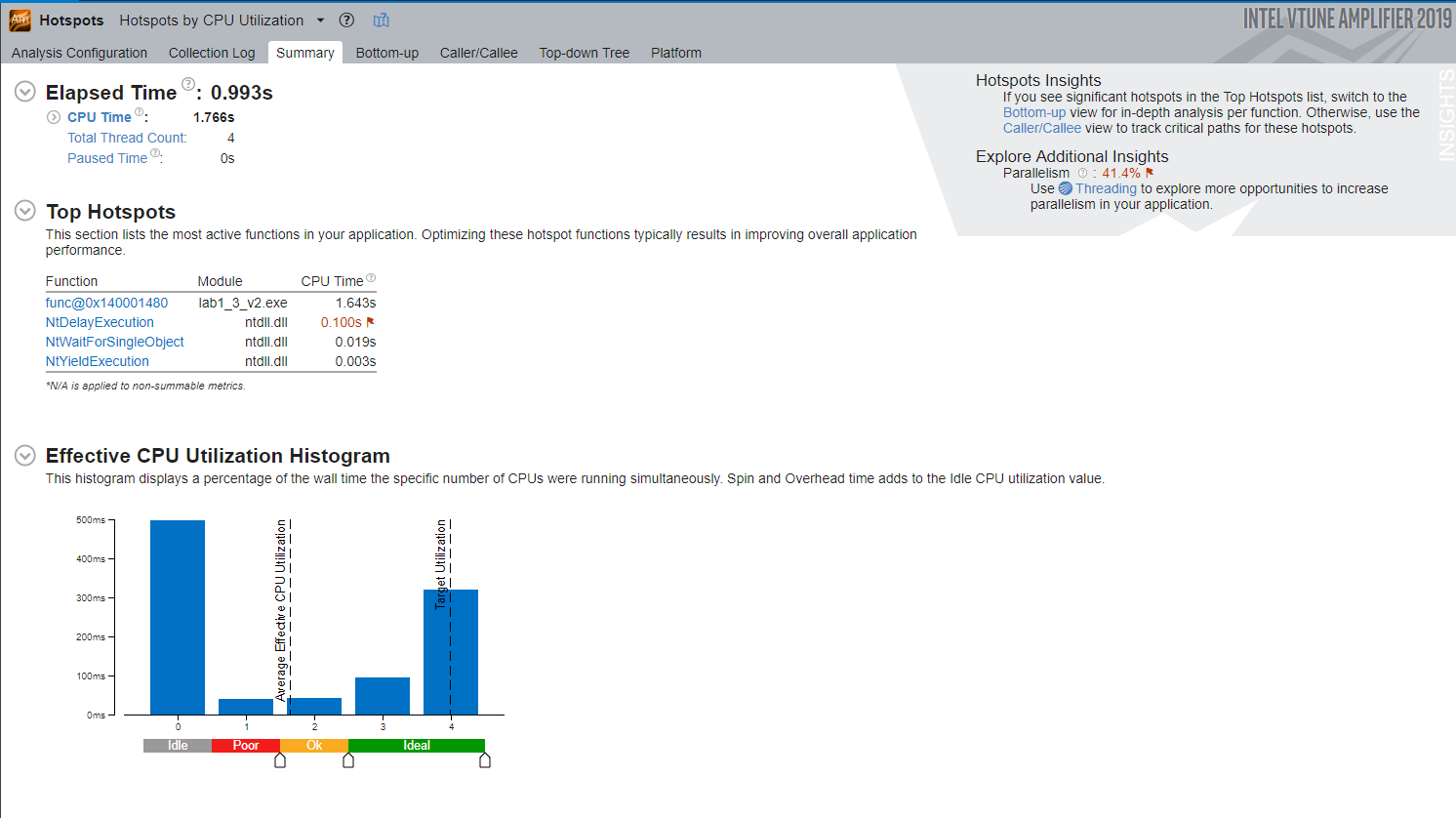


Рисунок 7 Вывод результатов работы Vtune по затраченному времени и используемым CPU

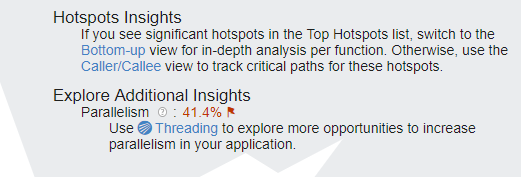


Рисунок 8 Оценка параллелизма

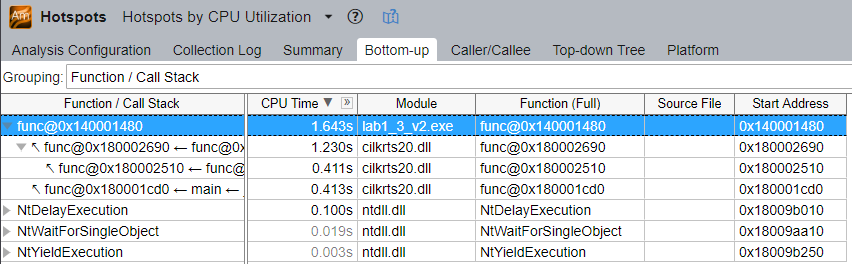


Рисунок 9 Вывод результатов работы Vtune с указанием затраченного времени на конкретные функции

# ****4.**** Программа с использованием шаблонов TBB

Для распараллеливания программы с помощью библиотеки TBB был использован библиотечный цикл parallel\_for. В качестве аргументов он принимает итератор на начало и конец коллекции, а также лямбда функцию. Полученные результаты представлены на рисунке 10.

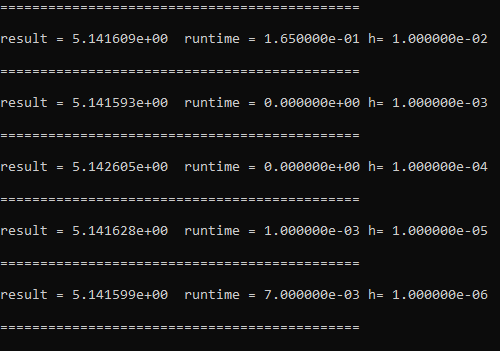


Рисунок 10 Вывод результатов работы программы с использованием parallel\_for

***Вывод:*** можно заметить, что результат каждой из программ напрямую зависит от величины шага, чем он меньше, тем дольше работает программа. Исходя из поставленных изначально условий (разбиение на 100,1000,10000,100000,1000000), не удалось объективно оценить какой метод лучше. Я дополнительно провел запуски с увеличенным разбиением на 10000000 и 100000000, что позволило оценить производительность лучше (результаты представлены в приложении). В итоге наилучшей по скорости оказалась программа с использованием библиотеки tbb, но она лишь немного опережает cilk и стандартные способы распараллеливания (потоки, qpar и отключенную векторизацию). Хоть результаты могут и меняться от запуска к запуску, очевидно, что последовательная программа уступает по скорости программам с распараллеливанием вычислений.

***Приложение***

***Результаты работы программ с уменьшенным шагом***

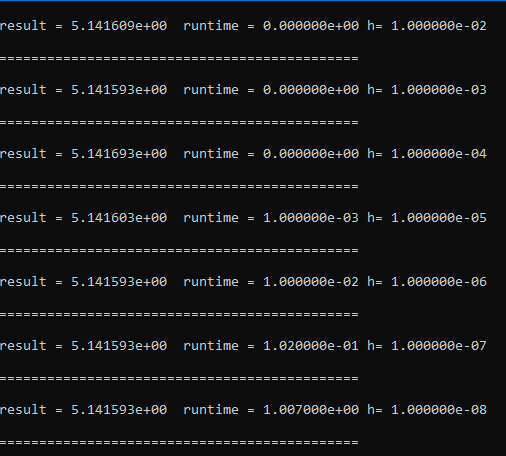
******

Рисунок 11 Вывод результатов работы последовательной программы

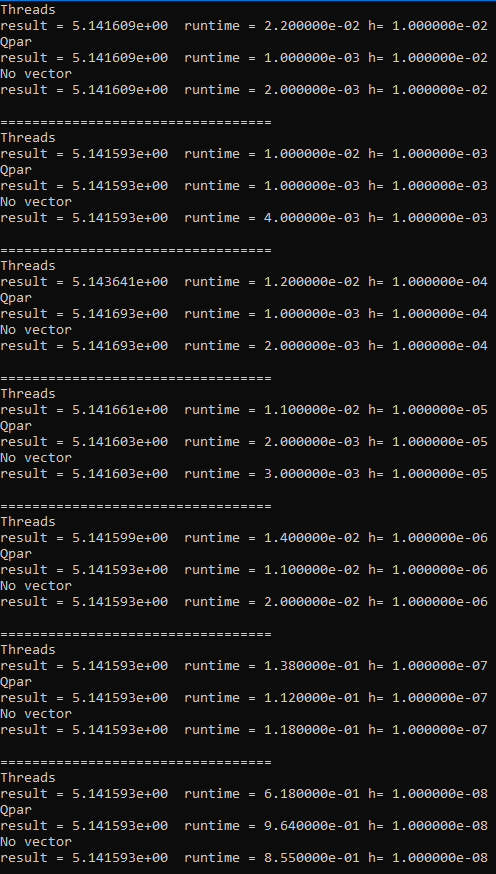
******

Рисунок 12 Вывод результатов работы программы по расчетуинтеграла с использованием нескольких потоков и векторных инструкций, количество потоков 8

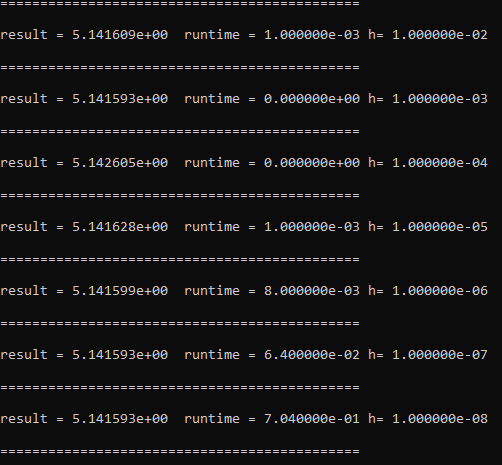
******

Рисунок 13 Вывод результатов работы программы *****с использованием cilk\_for и reducer*****

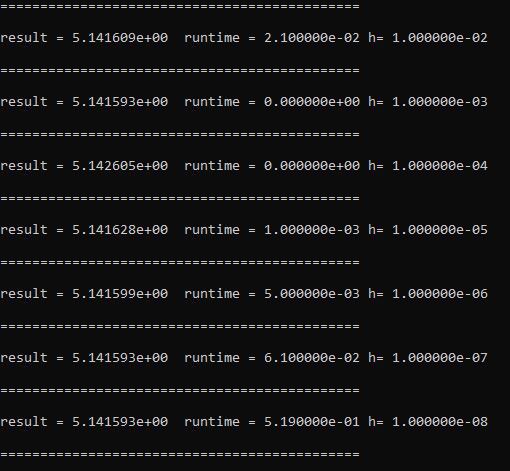
**

Рисунок 14 Вывод результатов работы программы с использованием parallel\_for