## Индивидуальное задание по разработке параллельных программ с использованием Cilk Plus

Выполнил ст. гр. ВМ 22 Леонтьев Алексей

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

$$\int_{-1}^{1} \frac{8}{2 + 2x^2} dx$$

1. Данная задача основана на вычислении площади криволинейной трапеции с помощью метода прямоугольников. Область интегрирования разбивается на множество маленьких отрезков. Сумма площадей прямоугольников, образованных этими отрезками и есть искомый интеграл. Точность вычисления обусловлена выбором мелкости разбиения. Поскольку количество вычислений зависит от количества отрезков, то для оптимизации выполнения программы ее необходимо распараллелить. Для этого воспользуемся пакетом *Intel Parallel Studio XE*.

При решении задачи будем пользоваться следующими средствами:

- *Amplifier XE* определение наиболее часто используемых участков кода программы;
- *Inspector XE* определите тех данные (если таковые имеются), которые принимают участие в гонке данных или в других основных ошибках, возникающих при разработке параллельных программ.
- 2. Описание и анализ программной реализации.

Написав первоначальную версию программы, убедился в правильности нахождения интеграла и узнал время, затраченное на вычисления с помощью стандартного цикла for. Количество интервалов разбиения равно  $10^9$ . Результат показан на рис. 1.

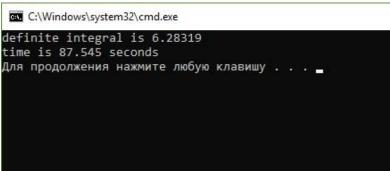


Рис. 1 — Найденное значение интеграла и время выполнения его вычисления с помощью цикла *for* 

Проанализировал последовательный программный код с помощью инструмента INTEL VTUNE AMPLIFIER 2019, получил следующие данные (рис. 2–4).

☼ CPU Time <sup>③</sup> : 28.259s			
Total Thread Count: 1			
Paused Time <sup>②</sup> : 0s			
Top Hotspots			
This section lists the most active f	unctions in your a	pplication. Optim	izing these
This section lists the most active f hotspot functions typically results Function			_
hotspot functions typically results	in improving overa	All application per Module	rformance.
hotspot functions typically results	in improving overa	Module individual.exe	rformance.  CPU Time  19.181
hotspot functions typically results  Function  f	in improving overa Module individual.exe	Module individual.exe	rformance.  CPU Time  19.181  8.263
hotspot functions typically results  Function  f  main	Module  Individual.exe individual.exe individual.exe	all application per Module individual.exe individual.exe	rformance.  CPU Time  19.181

Рис. 2 — Время, потраченное на выполнение последовательного программного кода при анализе

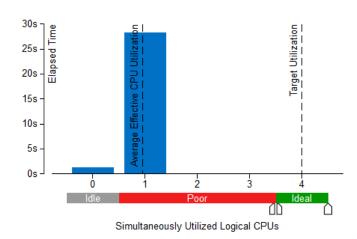


Рис. 3 — Зависимость времени выполнения программы от количества используемых процессов

So	ource	Assembly III = 67 6+ 6+			
Soı ▲		Source	♠ CPU Time: Total	CPU Time: Self 🔌	Source File
20	i	1//			
21		<pre>srand((unsigned)time(0));</pre>			
22		double a = -1.0;			
23		double b = 1.0;			
24		int N = 1e+9;			
25		double h = (b - a) / N;			
26		// cilk::reducer_opadd <double> I(0.0);</double>			
27		double I = 0.0;			
28		high_resolution_clock::time_point t1 = high_resolution_clock::now();			
29		for (int i = 0; i < N; i++)	2.7%	0.753s	main.cpp
30		{			
31		I += f(i*h + a) * h;	26.6%	7.511s	main.cpp
32		}			
33		high_resolution_clock::time_point t2 = high_resolution_clock::now();			
34		duration <double> duration = (t2 - t1);</double>			
35		double Pi = 3.1415926535897932384626433832795028841971693993751058209749445923078			
36		cout << "definite integral is " << I << endl;			
37		<pre>cout &lt;&lt; "time is " &lt;&lt; duration.count() &lt;&lt; " seconds" &lt;&lt; endl;</pre>			
38		return 0;			
39	}				

Рис. 4 – Окно с изображением наиболее часто используемых участков кода

На основании полученной информации можно сделать вывод о возможности распараллеливания стандартного цикла *for* с помощью *cilk\_for*. Результат выполнения программы с использованием *cilk\_for* приведён на рис. 5.

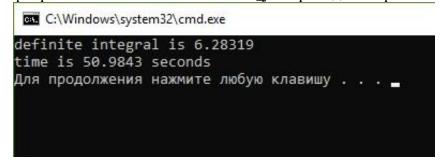


Рис. 5 – Результат выполнения программы после применения конструкции cilk\_for

Последним этапом в анализе программного кода стало использование инструмента *INTEL INSPECTOR 2019*. С помощью данного инструмента я проанализировал программу на наличие различных ошибок, таких как, например, гонка данных. Результат анализа представлен на рис. 6.



Рис. 6 Окно ошибок инструментария INTEL INSPECTOR 2019

В результате анализа ошибок не обнаружено.

Ссылка на GitHub: https://github.com/alexleo20/individual task1