

Индивидуальное задание

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 правых прямоугольников.

```
double integrate(double a, double b, size_t N)
{
    const double h = (b - a) / N;
    double sum = 0.0f;
    for (size_t i = 0; i < N; ++i)
        sum += fun(a + i * h);

    return sum * h;
}
```

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

Top Hotspots

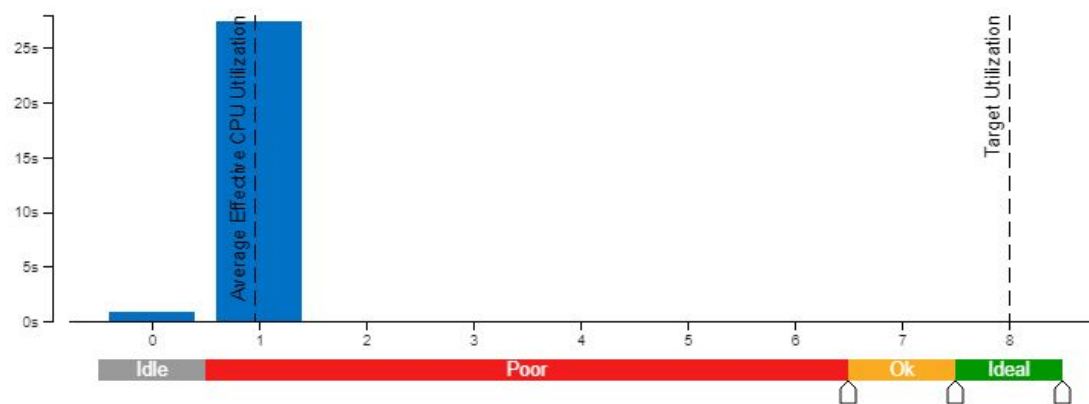
This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improved performance.

Function	Module	CPU Time [?]
fun	IPS_labs.exe	11.644s
integrate	IPS_labs.exe	9.061s
_libm_sqrt_ex	libmdd.dll	5.481s
sqrt	libmdd.dll	1.009s
[Import thunk sqrt]	IPS_labs.exe	0.020s
[Others]		0.013s

*N/A is applied to non-summable metrics.

Effective CPU Utilization Histogram

This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and



Распараллелим подсчет суммы:

```
double parallel_integrate(double a, double b, size_t n)
{
    const double h = (b - a) / n;
    cilk::reducer_opadd<double> sum(0.0);
    cilk_for(size_t i = 0; i < n; ++i)
        sum += fun(a + i * h);

    return sum->get_value() * h;
}
```

Top Hotspots

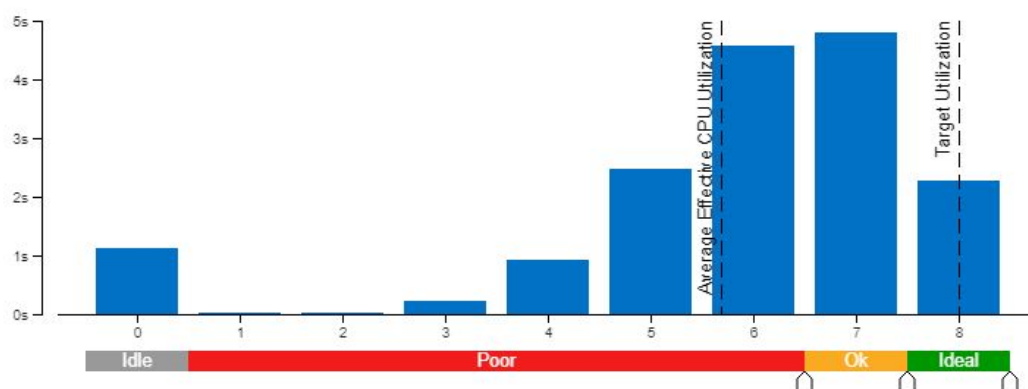
This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving

Function	Module	CPU Time ^③
func@0x140005085	IPS_labs.exe	19.943s
cilk::internal::reducer_base<struct cilk::op_add<double,1> >::view	IPS_labs.exe	17.733s
[Cilk reducer]	cilkrts20.dll	15.018s
fun	IPS_labs.exe	14.503s
cilk::op_add_view<double>::operator+=	IPS_labs.exe	11.039s
[Others]		34.452s

*N/A is applied to non-summable metrics.

Effective CPU Utilization Histogram

This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Ov



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

Description	Source	Function	Module	Variable
Write	reducer.h:484	scalar_view	ips_labs.exe	0x176caa82a00
482	/** Default constructor.		ips_labs.exe!scalar_view - reducer.h:484	
483	*/		ips_labs.exe!op_add_view - reducer_opadd.h:245	
484	scalar_view() : m_value() {}		ips_labs.exe!identity - reducer.h:441	
485			ips_labs.exe!identity_wrapper - reducer.h:927	
486	/** Value constructor.		ips_labs.exe!view - reducer.h:890	
Write	reducer.h:201	allocate	ips_labs.exe	0x176caa82a00
199	* @return An untyped pointer to the allocated memory.		ips_labs.exe!allocate - reducer.h:201	
200	*/		ips_labs.exe!allocate_wrapper - reducer.h:941	
201	void* allocate(size_t s) const { return operator new(s); }		ips_labs.exe!view - reducer.h:890	
202			ips_labs.exe!view - reducer.h:1232	
203	/** Deallocates raw memory pointed to by @a p		ips_labs.exe!operator+= - reducer_opadd.h:430	

Ошибок не обнаружено.

Скомпилируем программу в режиме Release и запустим для различных значений N

```
n = 100000
Serial ans: 2.09439
Spended time: 0.000906209
Parallel ans: 2.09439
Spended time: 0.000630152
Boost: 1.43808
-----
n = 1000000
Serial ans: 2.09439
Spended time: 0.00301109
Parallel ans: 2.09439
Spended time: 0.000685218
Boost: 4.39436
-----
n = 10000000
Serial ans: 2.0944
Spended time: 0.0316068
Parallel ans: 2.0944
Spended time: 0.00459559
Boost: 6.87764
-----
n = 100000000
Serial ans: 2.0944
Spended time: 0.30916
Parallel ans: 2.0944
Spended time: 0.0450037
Boost: 6.86965
-----
n = 1000000000
Serial ans: 2.0944
Spended time: 3.12899
Parallel ans: 2.0944
Spended time: 0.445587
Boost: 7.02219
-----
```

Как видим, ответ во всех случаях корректный, а скорость вычислений выше в параллельной реализации, причем выигрыш увеличивается пропорционально N.