

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

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

1. Описание проблемы и краткая характеристика инструментов параллелизации, используемых для решения задачи

Необходимо численно решить определенный интеграл, используя несколько потоков процессора. Для этого лучше использовать параллельный язык программирования (в данном случае дополнение для C++ - Cilk Plus). Во время написания программы нужно проверять код на различные ошибки, которые возникают в параллельной программе, а также провести оценку эффективности параллельной реализации. Это может выполнить набор программ от Intel – Intel Parallel Studio, в котором содержится VTune Amplifier и Intel Parallel Inspector.

Intel Cilk Plus – это дополнения языка C/C++, которые используют зарезервированные слова для описания параллелизма и новую систему обозначений для параллельной обработки массивов данных.

Конструкция `cilk_for` предназначена для введения параллелизма в циклы `for`. Reducer-ы используются для того, чтобы не возникало проблем с общей памятью в параллельных участках кода при выполнении какой-либо операции над общей для всех потоков переменной.

`Reducer_opadd` используется для суммирования.

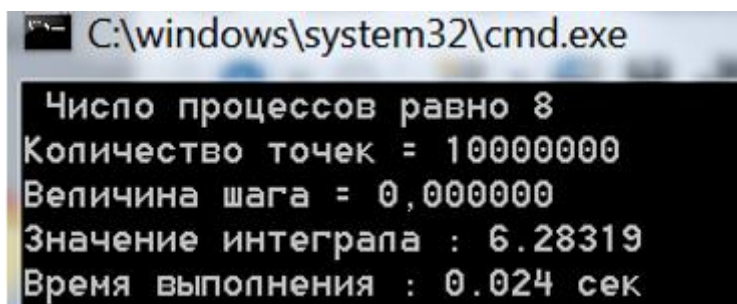
VTune Amplifier – это профилировщик, необходимый для поиска узких мест в работе программы.

Intel Parallel Inspector – инструмент, предназначенный для тестирования работающей программы с целью выявления основных ошибок, которые возникают при разработке параллельного кода.

```
>> int(4/(1+x^2), -1, 1)
```

```
ans =
```

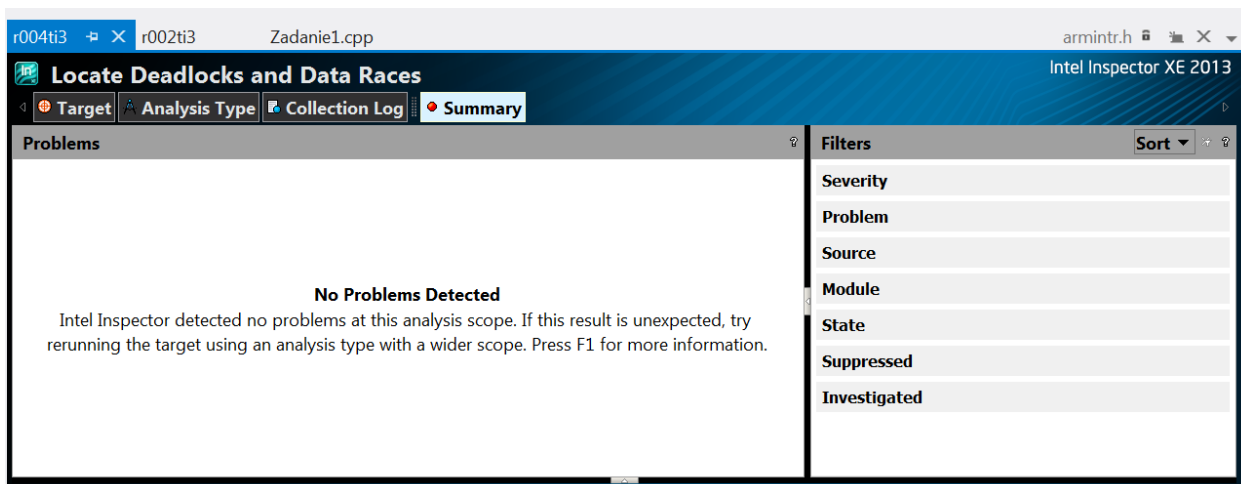
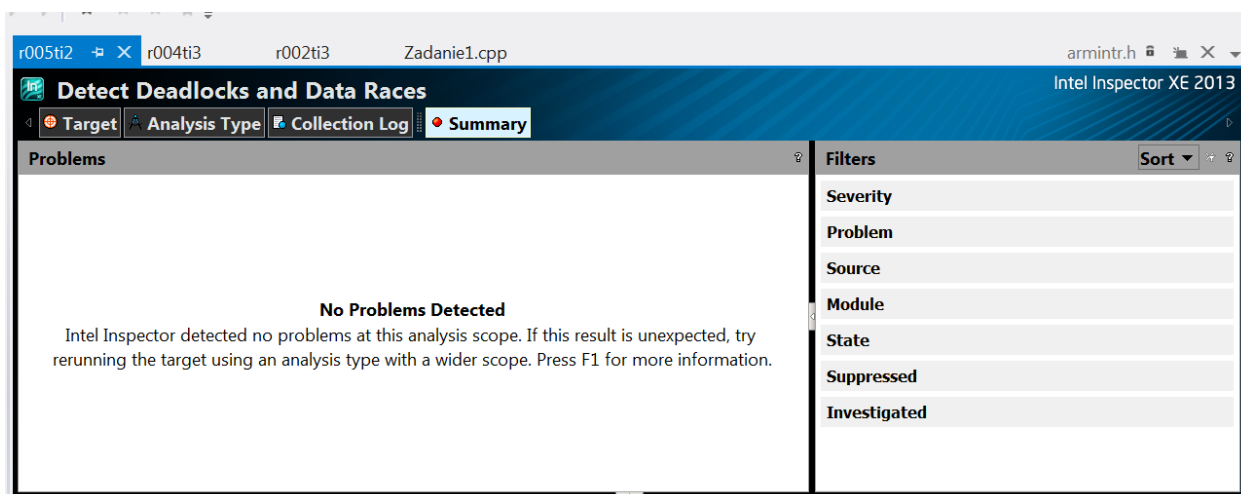
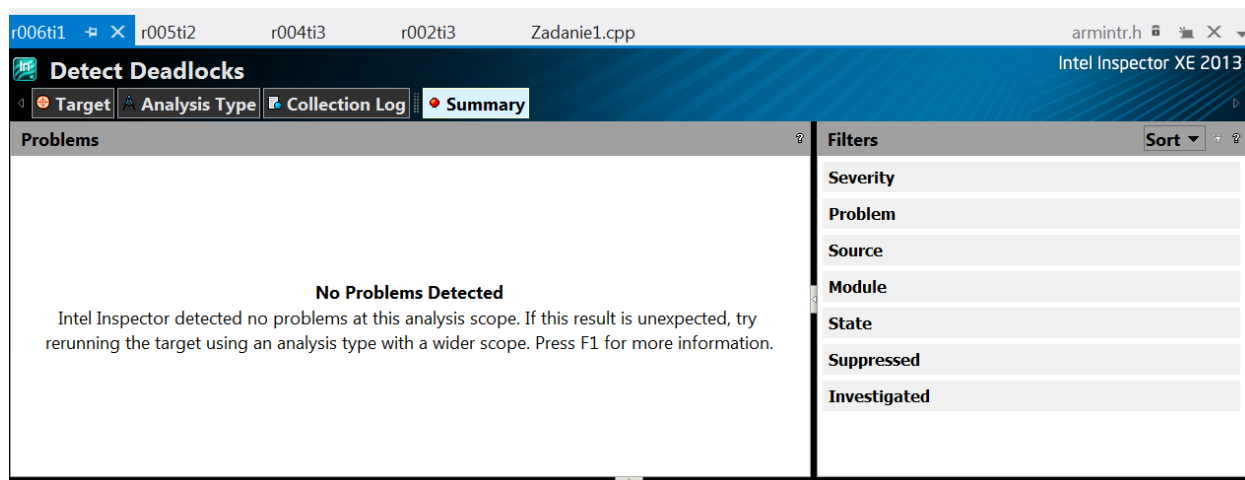
```
2*pi
```



```
C:\windows\system32\cmd.exe
Число процессов равно 8
Количество точек = 10000000
Величина шага = 0,000000
Значение интеграла : 6.28319
Время выполнения : 0.024 сек
```

2. Описание и анализ программной реализации

- Анализ работы программы с использованием *Intel Parallel Inspector XE*;



Intel Inspector XE 2013

Detect Leaks

Target Analysis Type Collection Log Summary

ID	Problem	Sources	Modules	Object Size	State
# P1	Missing allocation	[Unknown]	nvd3d9wrap.dll		Not fixed
# P2	Missing allocation	[Unknown]	nvd3d9wrap.dll		Not fixed
# P3	Missing allocation	[Unknown]	nvd3d9wrap.dll		Not fixed
# P4	Missing allocation	popen.c	MSVCR110.dll		Not fixed

Filters

Severity

Error 4 item(s)

Problem

Missing allocation 4 item(s)

Source

[Unknown] 3 item(s)

Code Locations: Missing allocation

Description	Source	Function	Module	Object Size
Invalid deallocation site	nvd3d9wrap.dll:0x9fd4	setDeviceHandle	nvd3d9wrap.dll	

[No Source]

Timeline

mainCRTStartup (10200)

Intel Inspector XE 2013

Detect Memory Problems

Target Analysis Type Collection Log Summary

ID	Problem	Sources	Modules	Object Size	State
# P1	Memory not deallocated	istream	MSVCP110.dll	16	Not fixed

Filters

Severity

Warning 1 item(s)

Problem

Memory not deallocated 1 item(s)

Source

istream 1 item(s)

Code Locations: Memory not deallocated

Description	Source	Function	Module	Object Size	Offset
Allocation site	istream:617	get	MSVCP110.dll	8	

```

615 { // state okay, use facet to
616 _TRY_IO_BEGIN
617 int_type _Meta = _Myios::rdbuf()
618
619 for (; ; _Meta = _Myios::rdbuf())
  
```

MSVCP110.dll!get - istream:617
 MSVCP110.dll!time_put<wchar_t,cla
 MSVCP110.dll!operator<< - ostream
 Zadaniel.exe!main - Zadaniel.cpp:
 Zadaniel.exe! tmainCRTStartup - c

Timeline

RtlQueryEnvironmentVariable (5752)

Intel Inspector XE 2013

Locate Memory Problems

Target Analysis Type Collection Log Summary

ID	Problem	Sources	Modules	Object Size	State
# P1	Memory not deallocated	istream	MSVCP110.dll	16	Not fixed

Filters

Severity

Warning 1 item(s)

Problem

Memory not deallocated 1 item(s)

Source

istream 1 item(s)

Code Locations: Memory not deallocated

Description	Source	Function	Module	Object Size	Offset
Allocation site	istream:617	get	MSVCP110.dll	8	

```

615 { // state okay, use facet to
616 _TRY_IO_BEGIN
617 int_type _Meta = _Myios::rdbuf()
618
619 for (; ; _Meta = _Myios::rdbuf())
  
```

MSVCP110.dll!get - istream:617
 MSVCP110.dll!time_put<wchar_t,cla
 MSVCP110.dll!operator<< - ostream
 Zadaniel.exe!main - Zadaniel.cpp:
 Zadaniel.exe! tmainCRTStartup - c

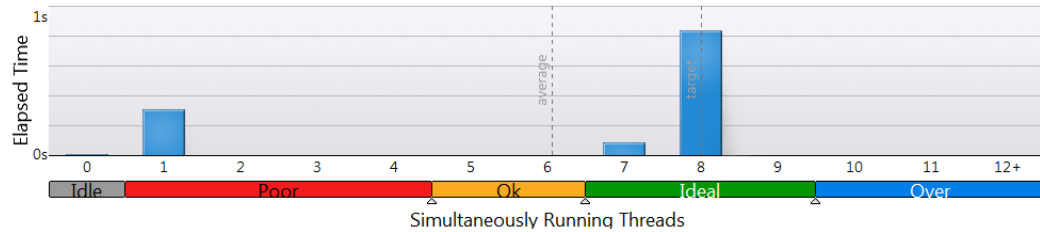
Timeline

RtlQueryEnvironmentVariable (8644)

- Оценка эффективности программной реализации;
- Проверка выполнения работы программы с использованием **Intel VTune Amplifier XE**;

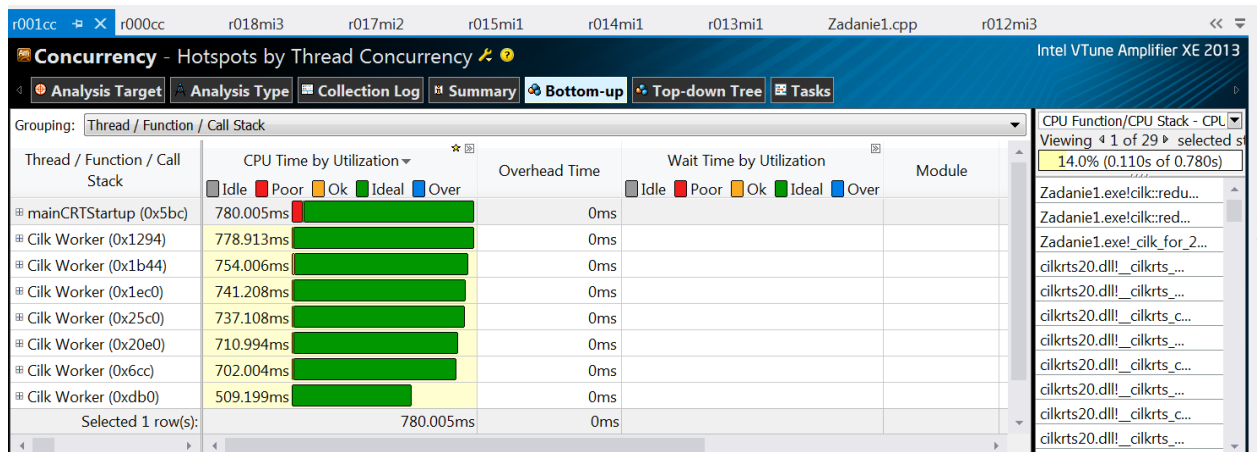
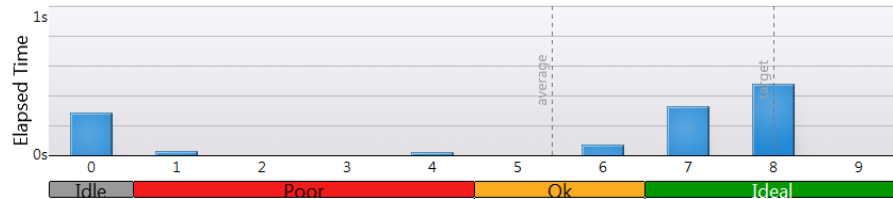
Thread Concurrency Histogram

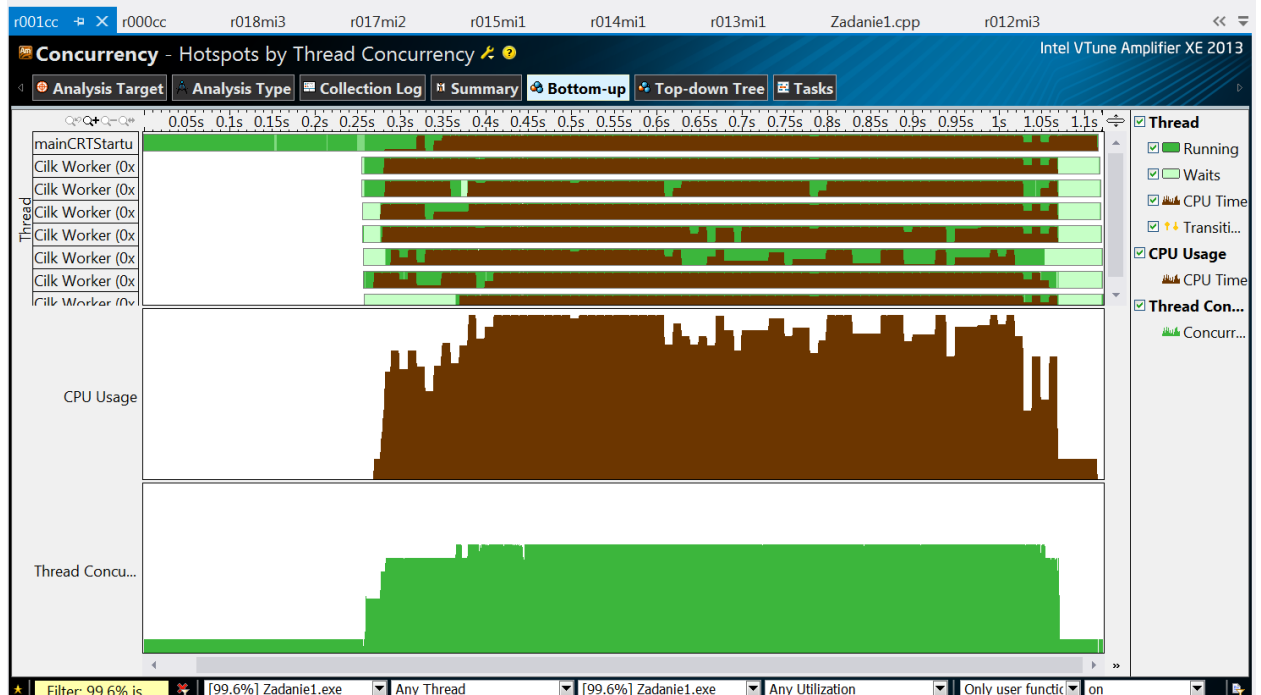
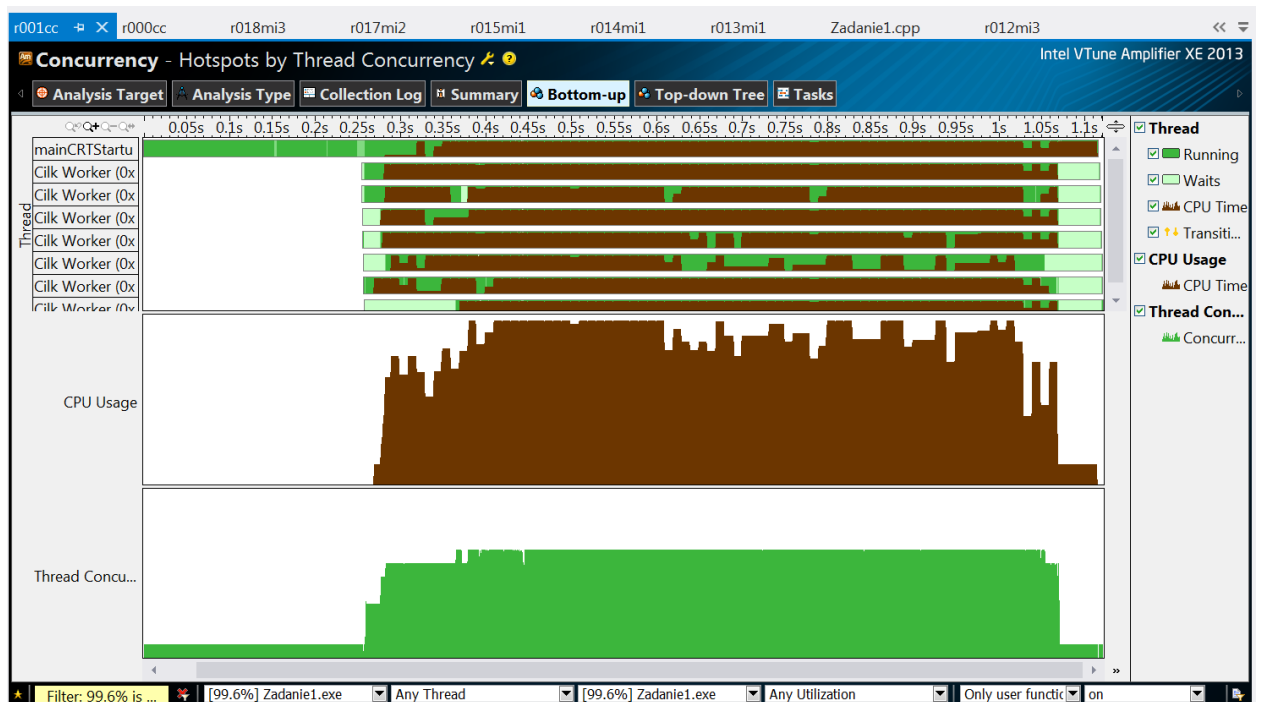
This histogram represents a breakdown of the Elapsed Time. It visualizes the percentage of the wall time the specific number of threads were running simultaneously. Threads are considered running if they are either actually running on a CPU or are in the runnable state in the OS scheduler. Essentially, Thread Concurrency is a measurement of the number of threads that were not waiting. Thread Concurrency may be higher than CPU usage if threads are in the runnable state and not consuming CPU time.



CPU Usage Histogram

This histogram represents a breakdown of the Elapsed Time. It visualizes what percentage of the wall time the specific number of CPUs were running simultaneously. CPU Usage may be higher than the thread concurrency if a thread is executing code on a CPU while it is logically waiting.





- Сведения о зависимости времени выполнения от заданных параметров алгоритма.

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

Число процессов равно 1
Количество точек = 10000000
Величина шага = 0,000000
Значение интеграла : 6.28319
Время выполнения : 0.0468 сек

Количество точек = 80000000
Величина шага = 0,000000
Значение интеграла : 6.28319
Время выполнения : 0.39 сек

Насколько увеличилось время выполнения программы с увеличением разбиения : 8.333
33

Количество точек = 130000000
Величина шага = 0,000000
Значение интеграла : 6.28319
Время выполнения : 0.4836 сек

Насколько увеличилось время выполнения программы с увеличением разбиения : 1.24
Для продолжения нажмите любую клавишу . . .
```

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

Число процессов равно 2
Количество точек = 10000000
Величина шага = 0,000000
Значение интеграла : 6.28319
Время выполнения : 0.0312 сек

Количество точек = 80000000
Величина шага = 0,000000
Значение интеграла : 6.28319
Время выполнения : 0.2184 сек

Насколько увеличилось время выполнения программы с увеличением разбиения : 7

Количество точек = 130000000
Величина шага = 0,000000
Значение интеграла : 6.28319
Время выполнения : 0.312 сек

Насколько увеличилось время выполнения программы с увеличением разбиения : 1.428
57
Для продолжения нажмите любую клавишу . . .
```

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

Число процессов равно 4
Количество точек = 10000000
Величина шага = 0,000000
Значение интеграла : 6.28319
Время выполнения : 0.024 сек

Количество точек = 80000000
Величина шага = 0,000000
Значение интеграла : 6.28319
Время выполнения : 0.172 сек

Насколько увеличилось время выполнения программы с увеличением разбиения : 7.166
67

Количество точек = 130000000
Величина шага = 0,000000
Значение интеграла : 6.28319
Время выполнения : 0.3 сек

Насколько увеличилось время выполнения программы с увеличением разбиения : 1.744
19
Для продолжения нажмите любую клавишу . . .
```

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

Число процессов равно 8
Количество точек = 10000000
Величина шага = 0,000000
Значение интеграла : 6.28319
Время выполнения : 0.024 сек

Количество точек = 80000000
Величина шага = 0,000000
Значение интеграла : 6.28319
Время выполнения : 0.164 сек

Насколько увеличилось время выполнения программы с увеличением разбиения : 6.833
33

Количество точек = 130000000
Величина шага = 0,000000
Значение интеграла : 6.28319
Время выполнения : 0.262 сек

Насколько увеличилось время выполнения программы с увеличением разбиения : 1.597
56
Для продолжения нажмите любую клавишу . . .
```

	$N=10^7$	$N=8 \cdot 10^7$	$N=13 \cdot 10^7$
$P=1$	0.0468	0.39	0.4836
$P=2$	0.0312	0.2184	0.312
$P=4$	0.024	0.172	0.3
$P=8$	0.024	0.164	0.262

Здесь P – количество потоков , N – число разбиений

в цикле `cilk_for` итерации распределяются между потоками . Сначала каждому потоку “раздают” итерации цикла , затем каждый поток выполняет свои итерации , а потом происходит синхронизация между потоками . Раздача итераций и синхронизация между потоками занимают некоторое небольшое время . Однако при небольшом количестве итераций из-за этого времени последовательная реализация (или реализация с меньшим количеством потоков) будет быстрее или примерно таким же (время синхронизации будет велико по сравнению с временем вычислений) , что мы и видим при 4 и 8 потоках и $N=10^7$. Но при больших количествах итераций целесообразнее использовать большее число потоков (время синхронизации будет очень мало по сравнению с временем вычислений) .