

# Отчёт по лабораторной работе № 3


## Жуков Вадим, ИВТ-12М

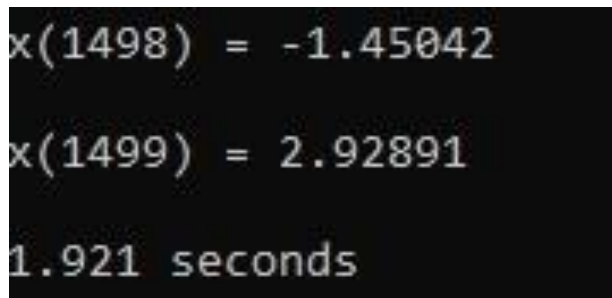
Используемое железо: 2 физических ядра, 4 логических, 3.4ГГц

ОС: Windows 10

VS: Visual Studio 2017 v15.9.18

IPS: v2019

1. В файле [task\\_for\\_lecture3.cpp](#)  приведен код, реализующий последовательную версию метода Гаусса для решения СЛАУ. Проанализируйте представленную программу.
2. Запустите первоначальную версию программы и получите решение для тестовой матрицы `test_matrix`, убедитесь в правильности приведенного алгоритма. Добавьте строки кода для измерения времени (см. задание к занятию 2) выполнения прямого хода метода Гаусса в функцию `SerialGaussMethod()`. Заполните матрицу с количеством строк `MATRIX_SIZE` случайными значениями, используя функцию `InitMatrix()`. Найдите решение СЛАУ для этой матрицы (закомментируйте строки кода, где используется тестовая матрица `test_matrix`).

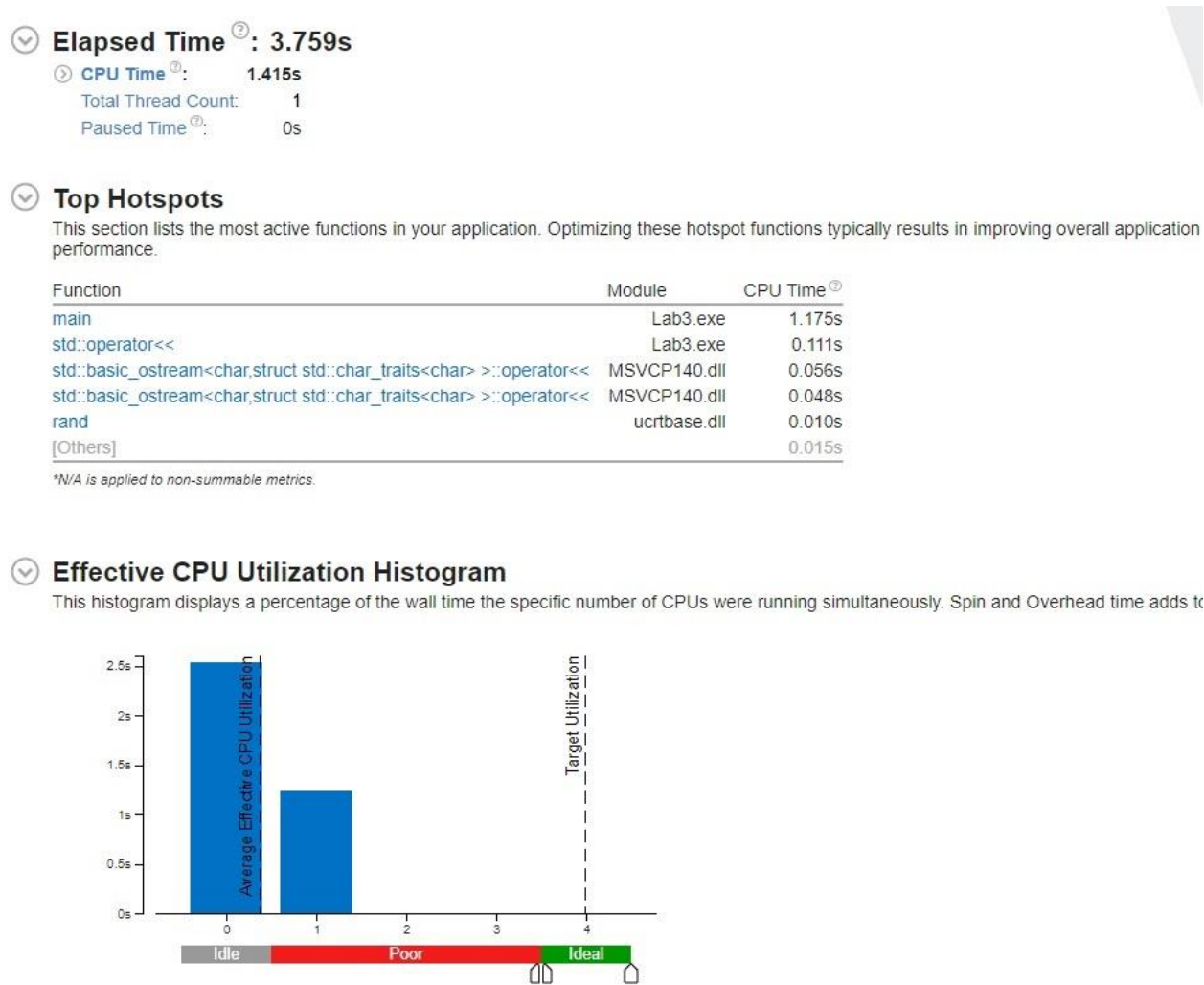


```
x(1498) = -1.45042
x(1499) = 2.92891
1.921 seconds
```

Поиск решения СЛАУ прямым методом Гаусса составил 1.921 секунд.

3. С помощью инструмента **Amplifier XE** определите наиболее часто используемые участки кода новой версии программы. Сохраните скриншот результатов анализа **Amplifier XE**. Создайте, на основе последовательной функции **SerialGaussMethod()**, новую функцию, реализующую параллельный метод Гаусса. Введите параллелизм в новую функцию, используя **cilk\_for**. **Примечание:** произвести параллелизацию одного внутреннего цикла прямого хода метода Гаусса (определить какого именно), и внутреннего цикла обратного хода. Время выполнения по-прежнему измерять только для прямого хода.

Результат работы Amplifier XE для последовательного выполнения метода Гаусса:



Вводим параллельное выполнение с помощью **cilk\_for**:

```
x(1498) = 0.106084
x(1499) = 2.49999
10.694 seconds
```

Как видим, время выполнения сильно ухудшилось, примерно в 5 раз. Объяснение этому дано в 4 задании.

4. Далее, используя **Inspector XE**, определите те данные (если таковые имеются), которые принимают участие в гонке данных или в других основных ошибках, возникающих при разработке параллельных программ, и устраните эти ошибки. Сохраните скриншоты анализов, проведенных инструментом **Inspector XE**: в случае обнаружения ошибок и после их устранения.

Code Locations: Data race				
Description	Source	Function	Module	Variable
Read	lab3.exe!0x13be	[Unknown]	lab3.exe	block allocated at task3.h:93
Symbol information not found. Suggestion: Specify locations in a Project Properties dialog box search tab, then re-resolve the result.				
Write	task3.h:81	main	lab3.exe	block allocated at task3.h:93
79 result[k] = matrix[k][rows]; 80 cilk_for (int j = k + 1; j < rows; ++j) 81 result[k] -= matrix[k][j] * result[j]; 82 result[k] /= matrix[k][k]; 83 }				
lab3.exe!0x1436 lab3.exe!main - task3.h:81				

В результате работы Inspector XE видим, что на 81 строке происходит гонка данных, поэтому время работы увеличилось в 5 раз. Для того, чтобы исправить эту ситуацию, следует использовать `reducer_opadd`.

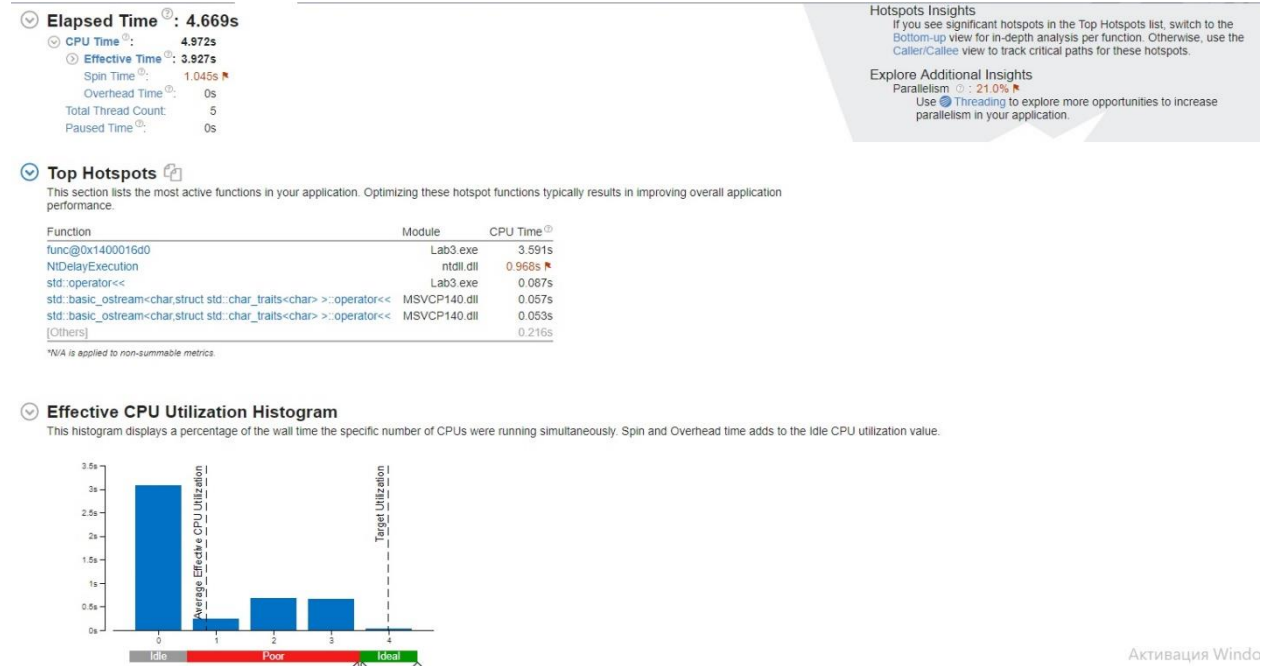
```
x(1499) = -0.315591
2.344 seconds
```

Теперь время выполнения стало приемлемым. Прогнав программу ещё раз через Inspector XE видим, что проблема гонки данных больше не наблюдается.

No Problems Detected

Intel Inspector detected no problems at this analysis scope. If this result is unexpected, try rerunning the target using an analysis type with a wider scope. Press F1 for more information.

Также я ещё раз прогнал программу через Amplifier XE. Программа выполняется в 5 потоках, а процент параллелизма составляет 21%.



5. Убедитесь на примере тестовой матрицы `test_matrix` в том, что функция, реализующая параллельный метод Гаусса работает правильно. Сравните время выполнения прямого хода метода Гаусса для последовательной и параллельной реализации при решении матрицы, имеющей количество строк `MATRIX_SIZE`, заполняющейся случайными числами. Запускайте проект в режиме `Release`, предварительно убедившись, что включена оптимизация (`Optimization->Optimization=/O2`). Подсчитайте ускорение параллельной версии в сравнении с последовательной. Выводите значения ускорения на консоль.

```
Task_5

Test Matrix
Duration serial: 0
Duration cilk_for: 0.002

MATRIX SIZE
Duration serial: 2.013
Duration cilk_for: 1.494
```

В результате поиска решения СЛАУ для тестовой матрицы время параллельного выполнения получилось больше, чем при последовательном. Для матрицы большего размера наоборот. Это связано с тем, что `cilk_for` помимо вычислений делает ещё несколько операций, о чём подробнее написано в отчёте по лабораторной работе №2. В целом, при действительно большом объёме данных, использование `cilk_for` имеет смысл.