**Отчёт по лабораторной работе № 3**

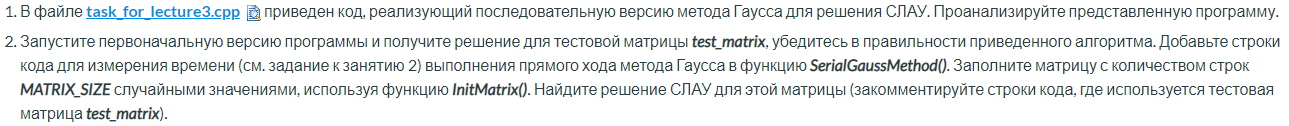
**Жуков Вадим, ИВТ-12М**

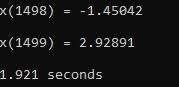
Используемое железо: 2 физических ядра, 4 логических, 3.4ГГц

ОС: Windows 10

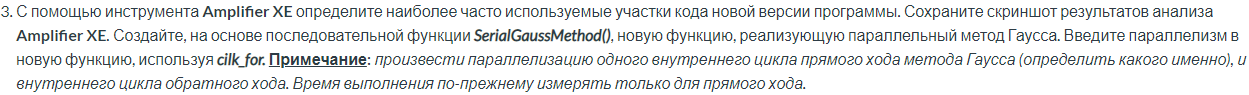
VS: Visual Studio 2017 v15.9.18

IPS: v2019

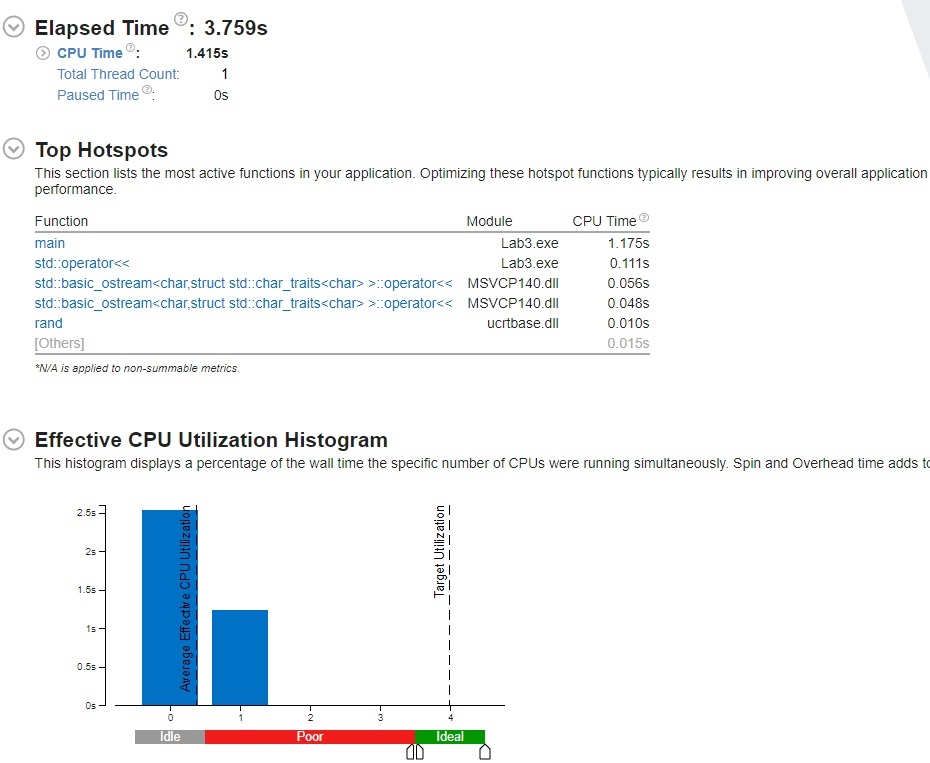




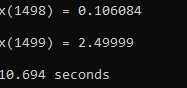
Поиск решения СЛАУ прямым методом Гаусса составил 1.921 секунд.



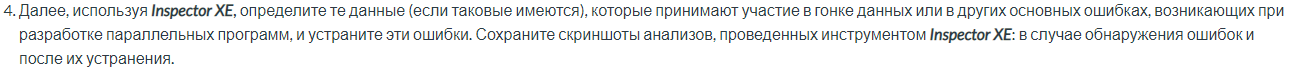
Результат работы Amplifier XE для последовательного выполнения метода Гаусса:

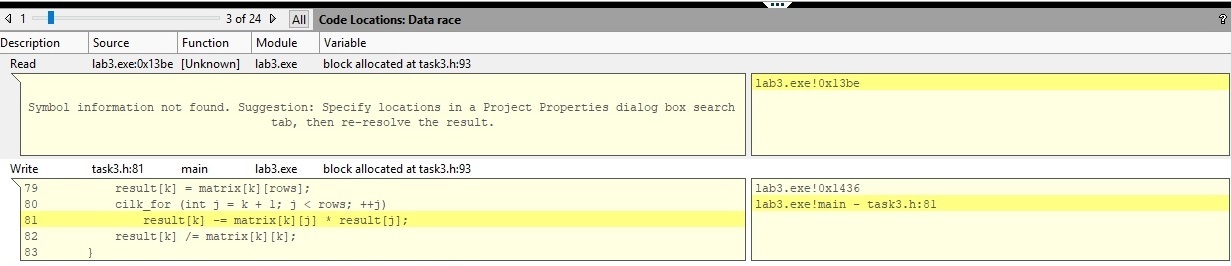


Вводим параллельное выполнение с помощью cilk\_for:



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

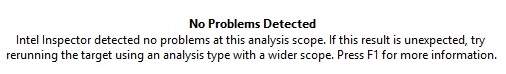




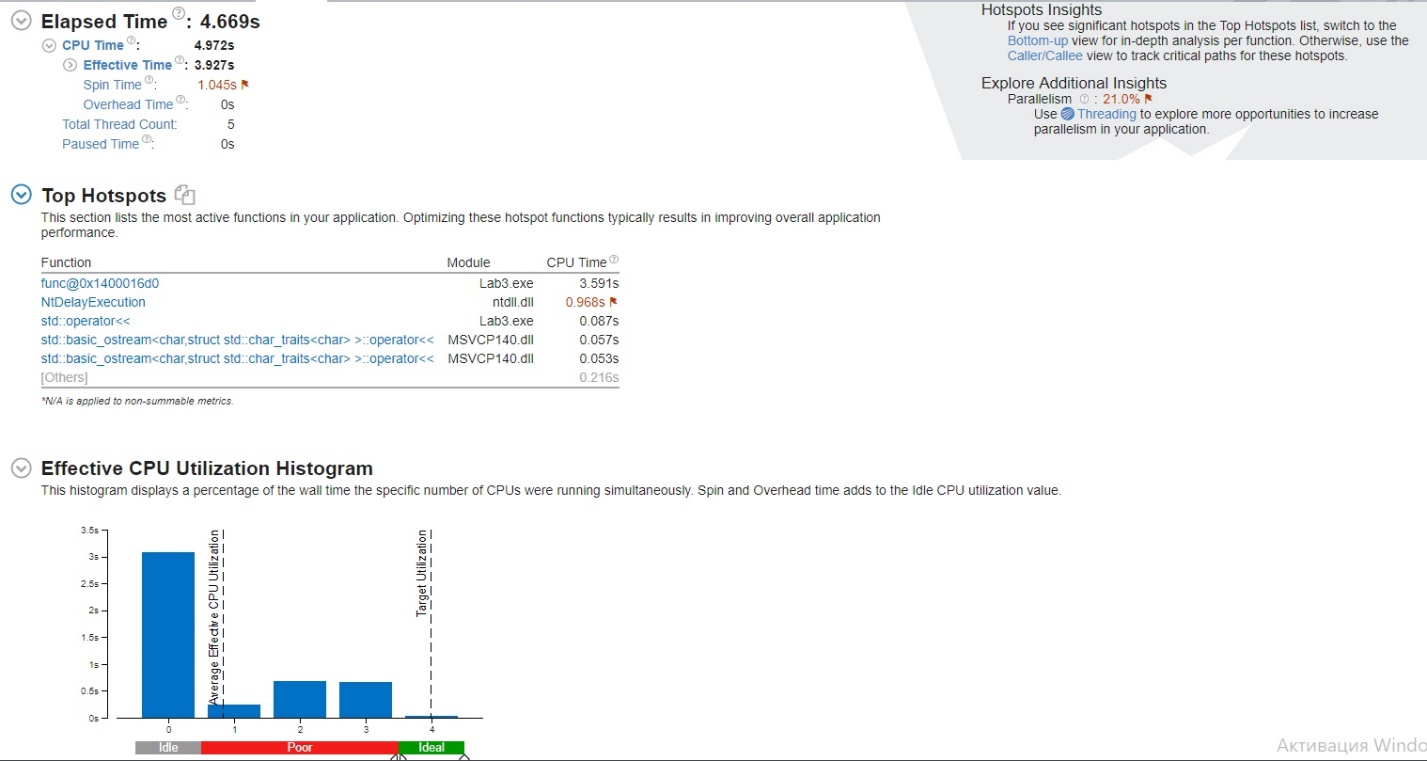
В результате работы Inspector XE видим, что на 81 строке происходит гонка данных, поэтому время работы увеличилось в 5 раз. Для того, чтобы исправить эту ситуацию, следует использовать reducer\_opadd.

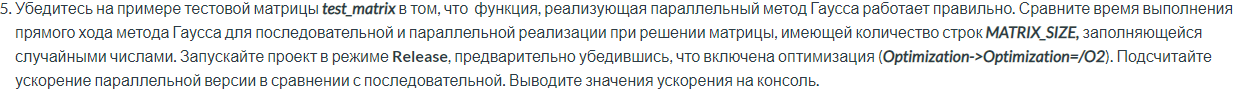
C:\Users\zhuko\OneDrive\Рабочий стол\4_1.jpg

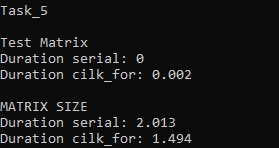
Теперь время выполнения стало приемлемым. Прогнав программу ещё раз через Inspector XE видим, что проблема гонки данных больше не наблюдается.



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







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.