Лабораторная работа 6.

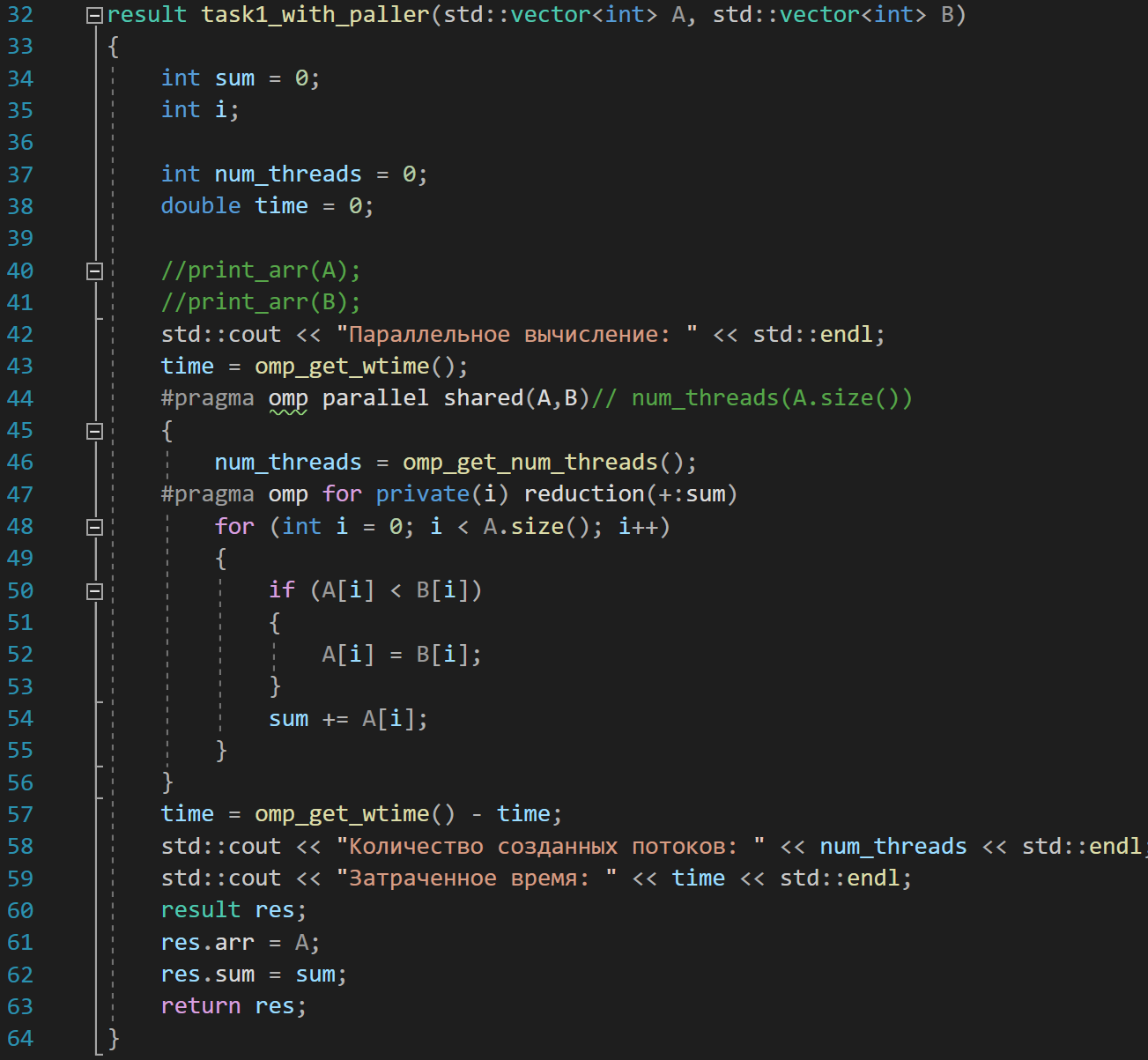
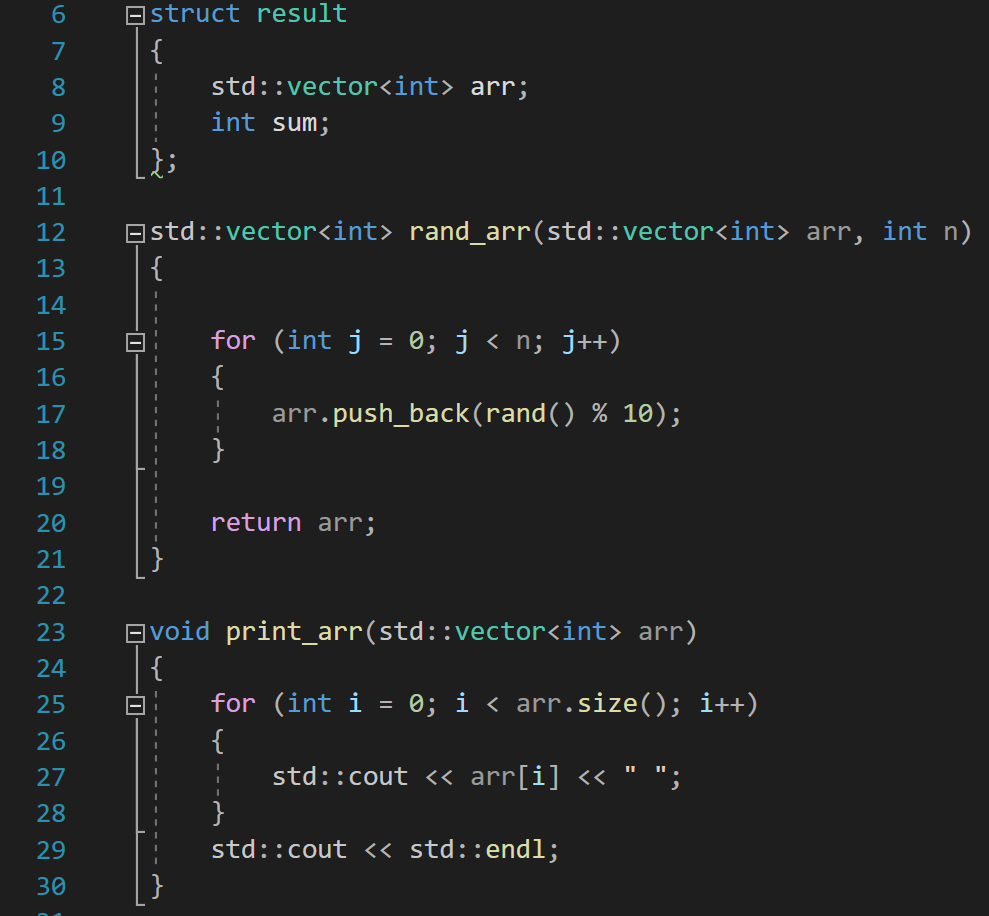
Отчет

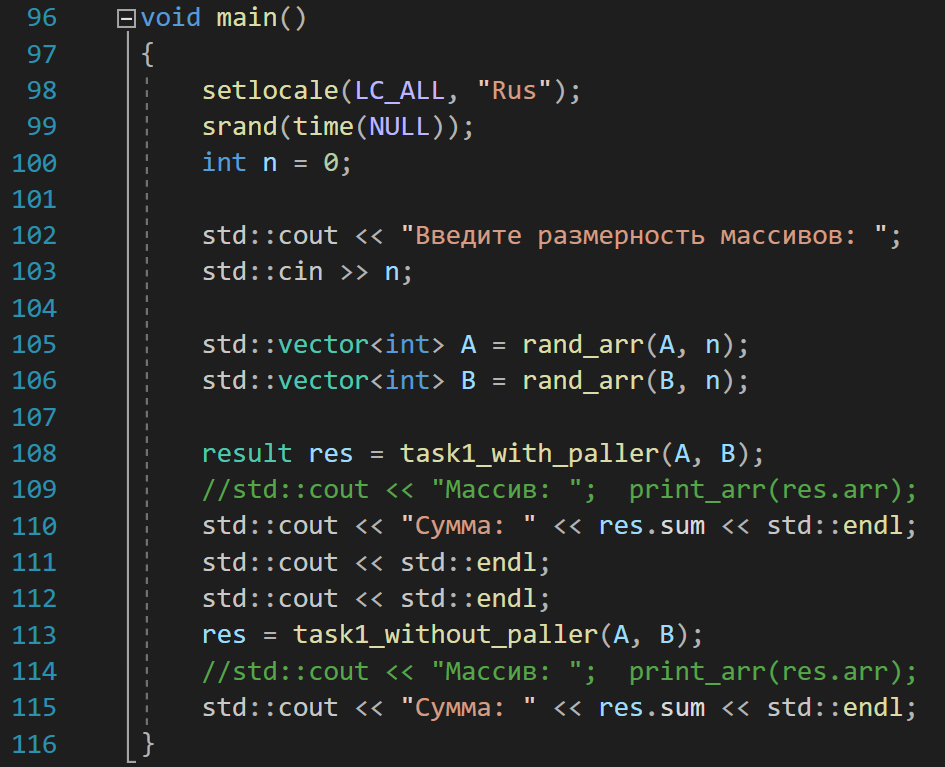
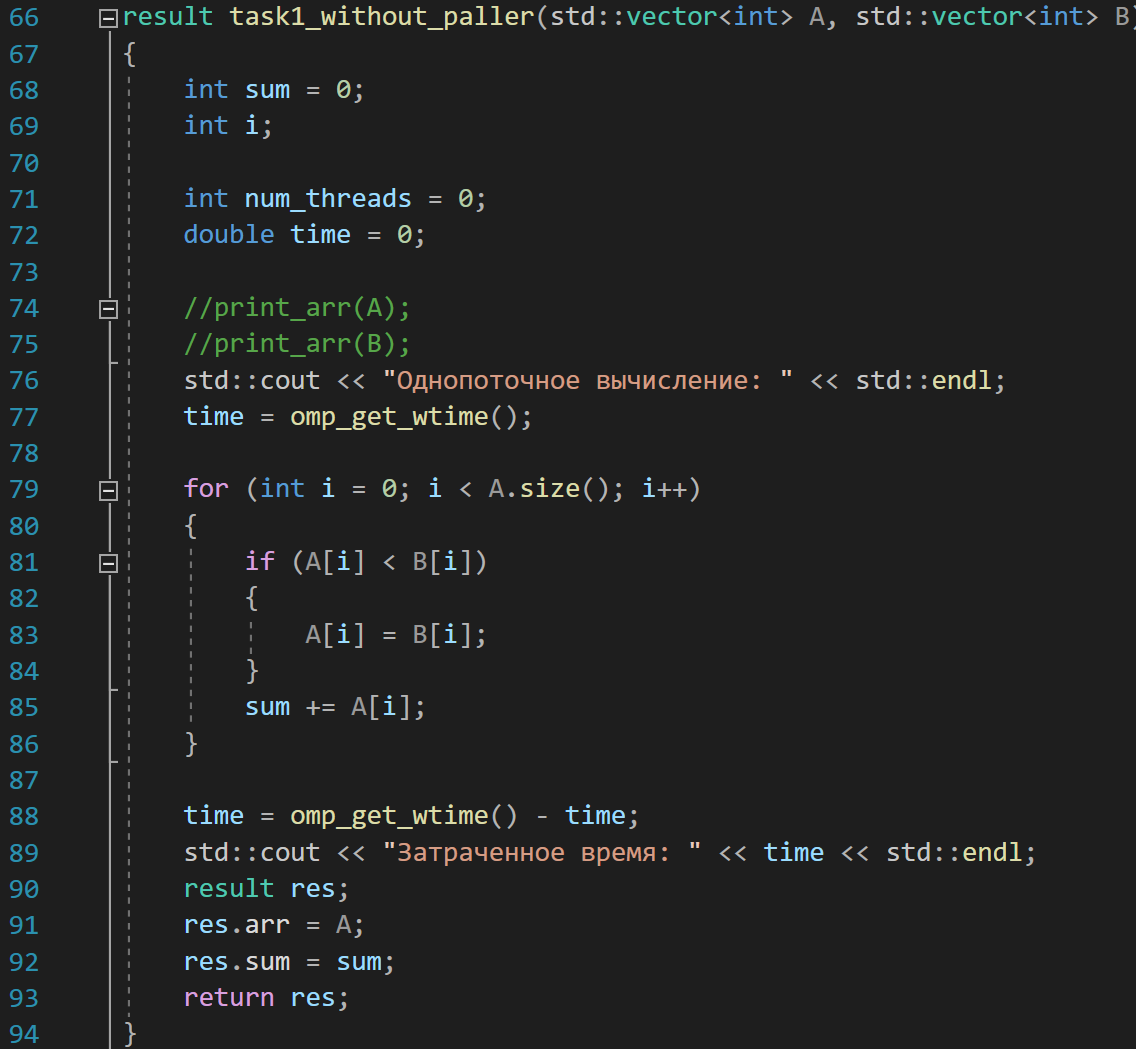
Выполнил: Гусаров Андрей ПИН-44

# Написать многопоточную программу.

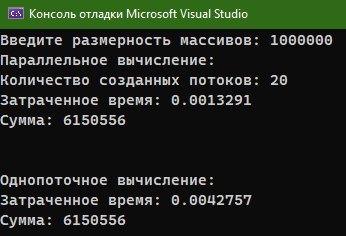
На основе двух равно размерных массивов A и B (длины N) функция формирует массив C, таким образом, что: , и возвращает сумму значений элементов массива C.

Код программы:

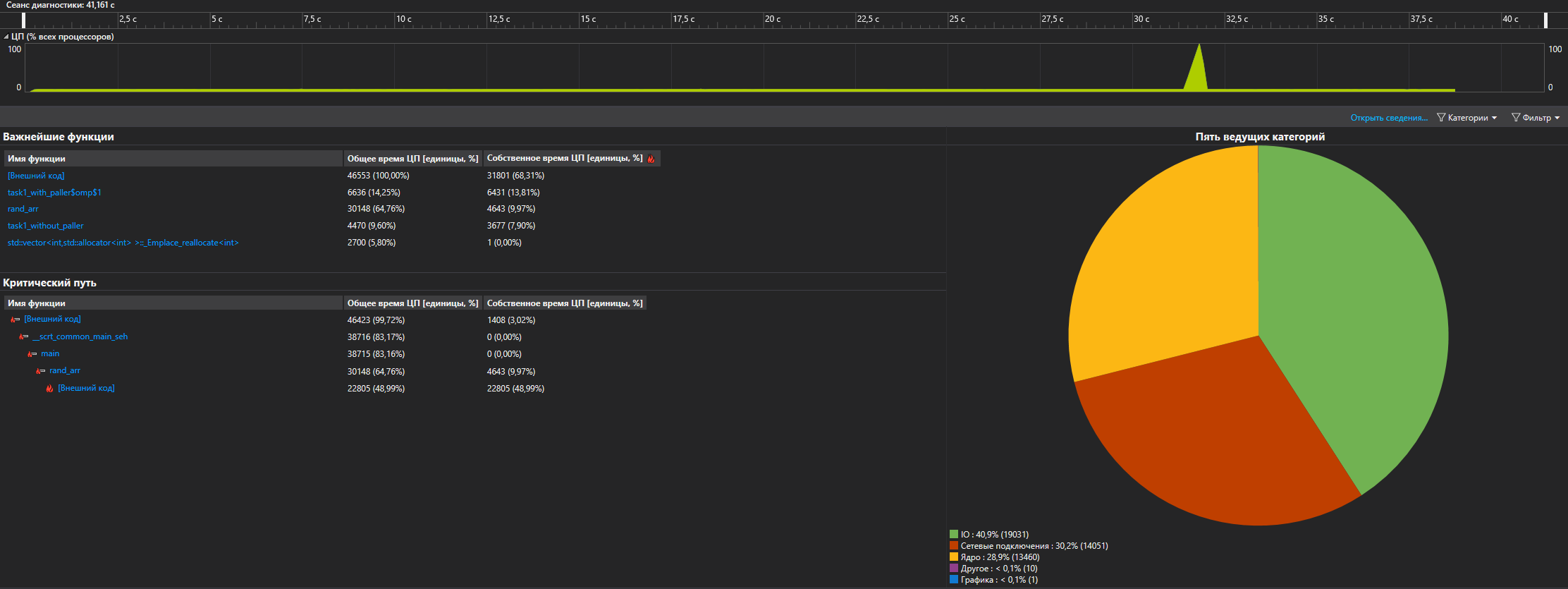




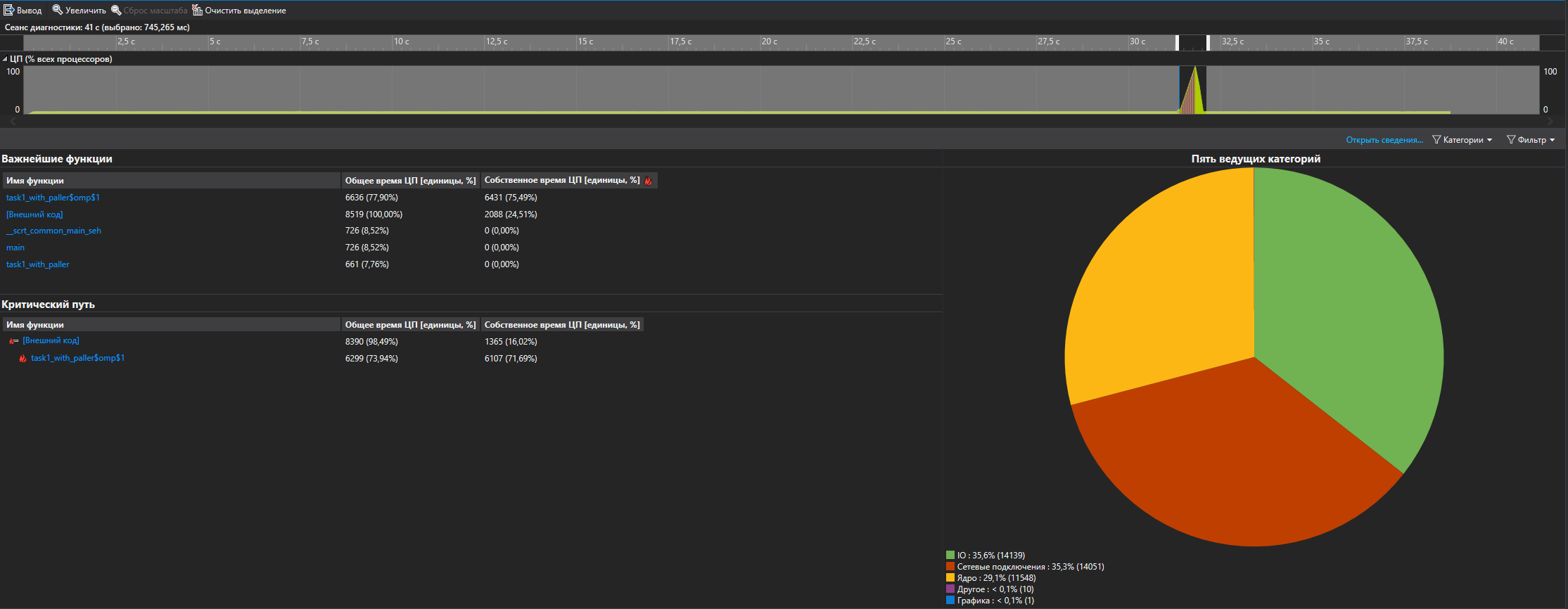
Результат:



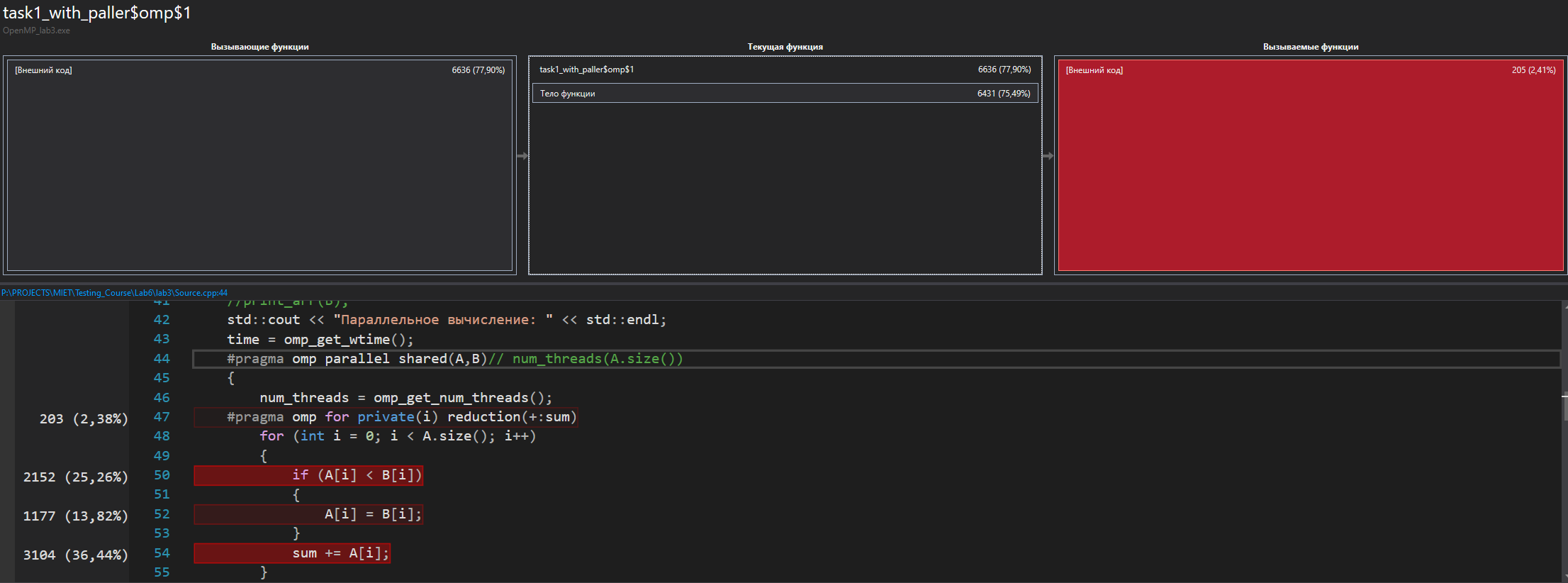
# Провести профилирование с помощью дискретного профилировщика.

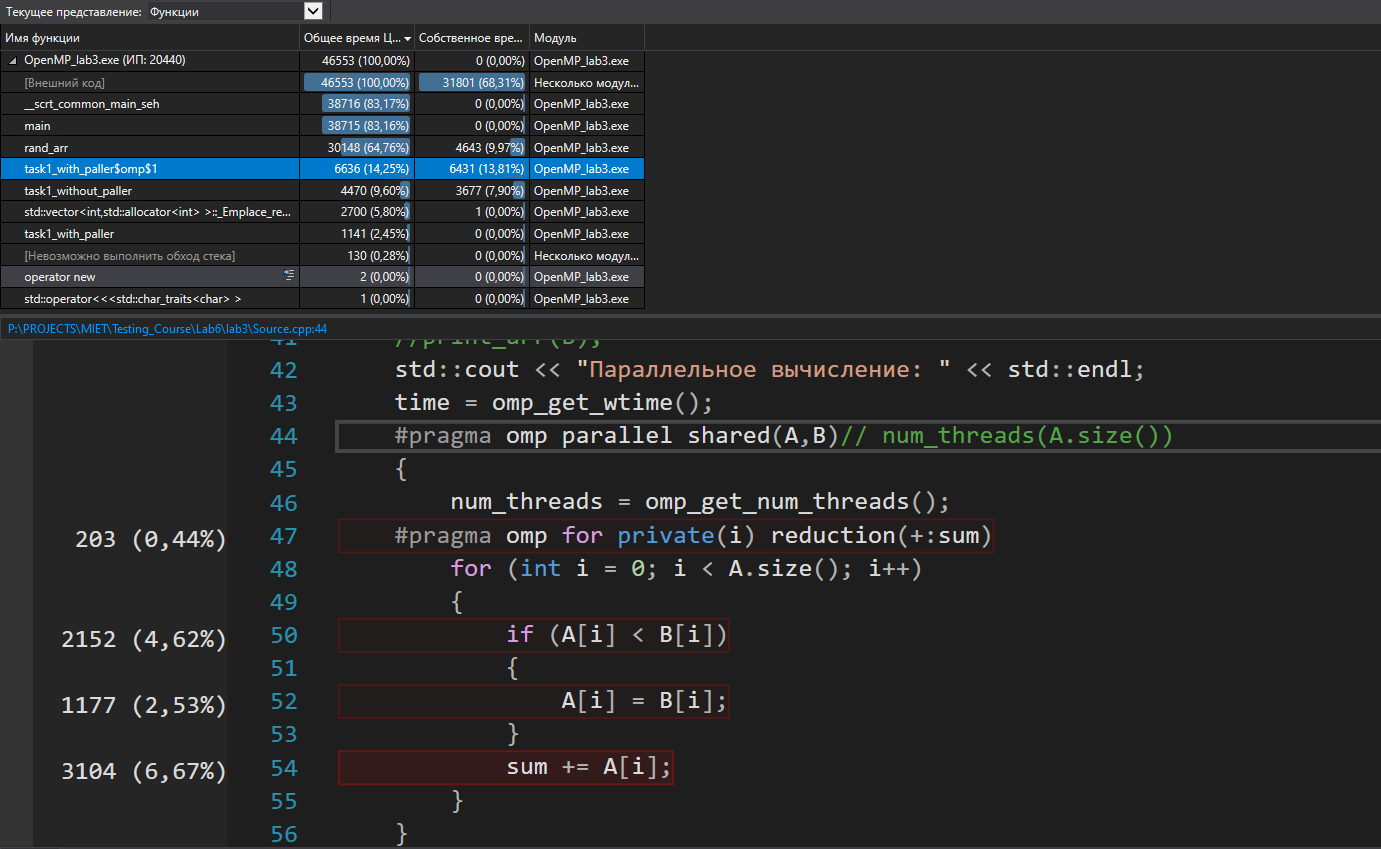


По графику видно, что большую часть времени программа занимает 5-10% ресурсов ЦП.

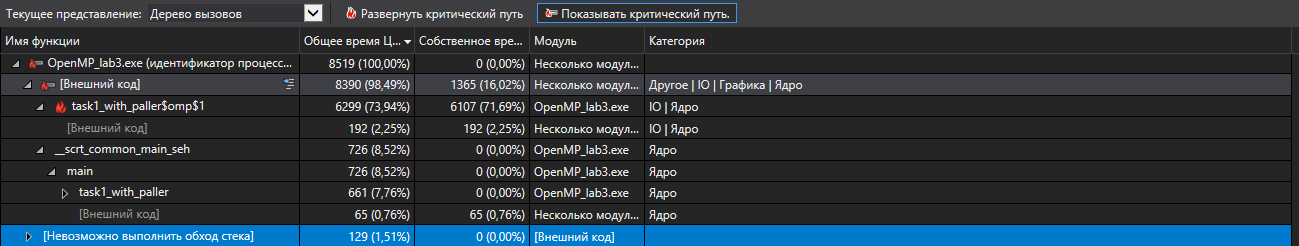


Однако в момент многопоточного вычисления нагрузка на ЦП поднимается до 100%. Максимальная по нагрузке функция – параллельное вычисление.



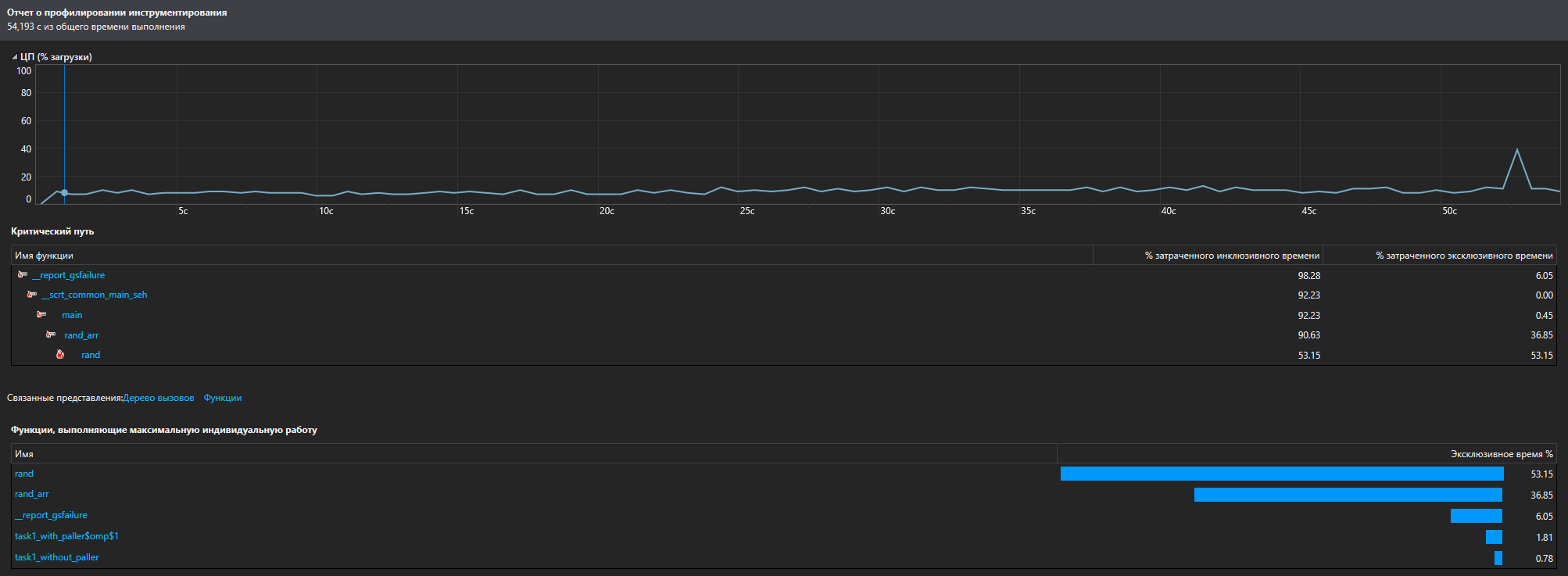


Наибольшее собственное время занимает task1\_with\_paller$omp$1 (сумма времени параллельных вычислений). По общему времени rand\_arr(подфункция main) лидирует среди вызываемых функций в main.



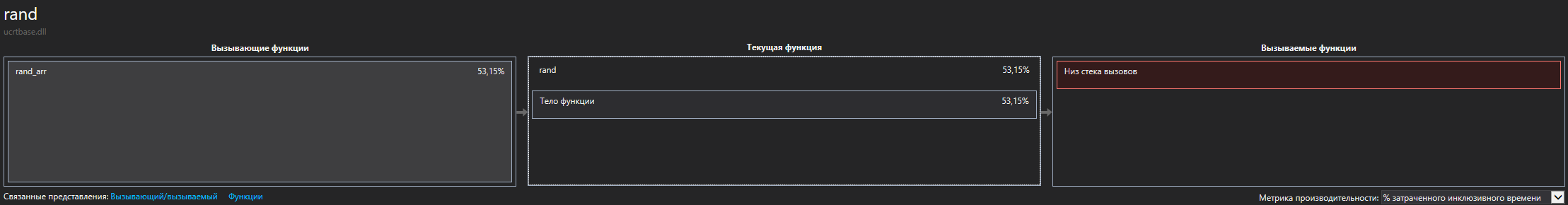
В дереве вызов можно увидеть какие функции откуда были вызваны.

# Провести профилирование с помощью инструментированного профилировщика.



На графике мы можем видеть процент загрузки процессора. Под ним можно посмотреть затраченное функциями время в секции критического пути, а также в секции функций, выполняющих максимальную индивидуальную работу.

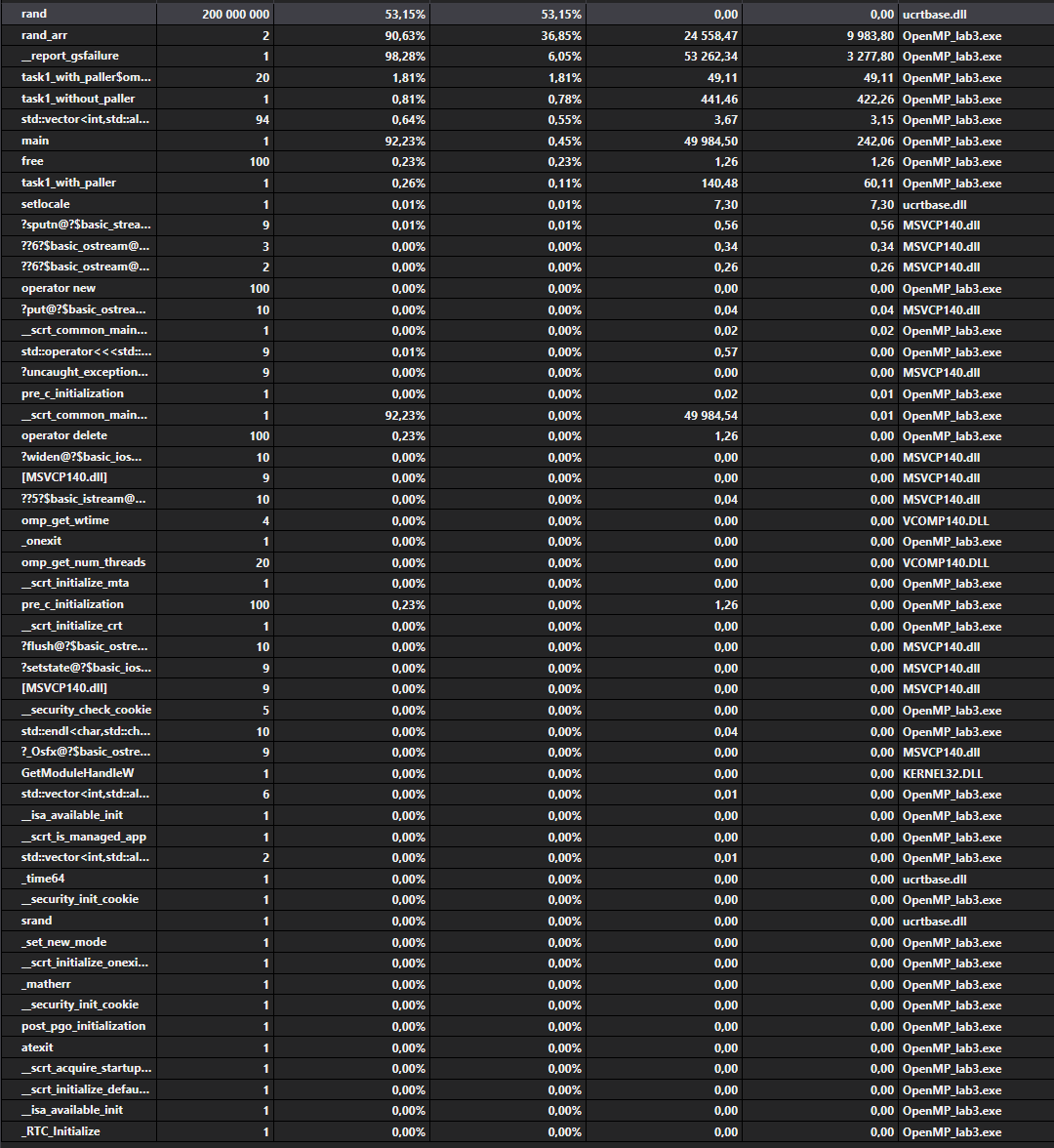
Функции с наибольшим исключительным временем:



На этом рисунке мы можем заметить, что функция rand вызывается внутри функции rand\_arr, представленной ниже:



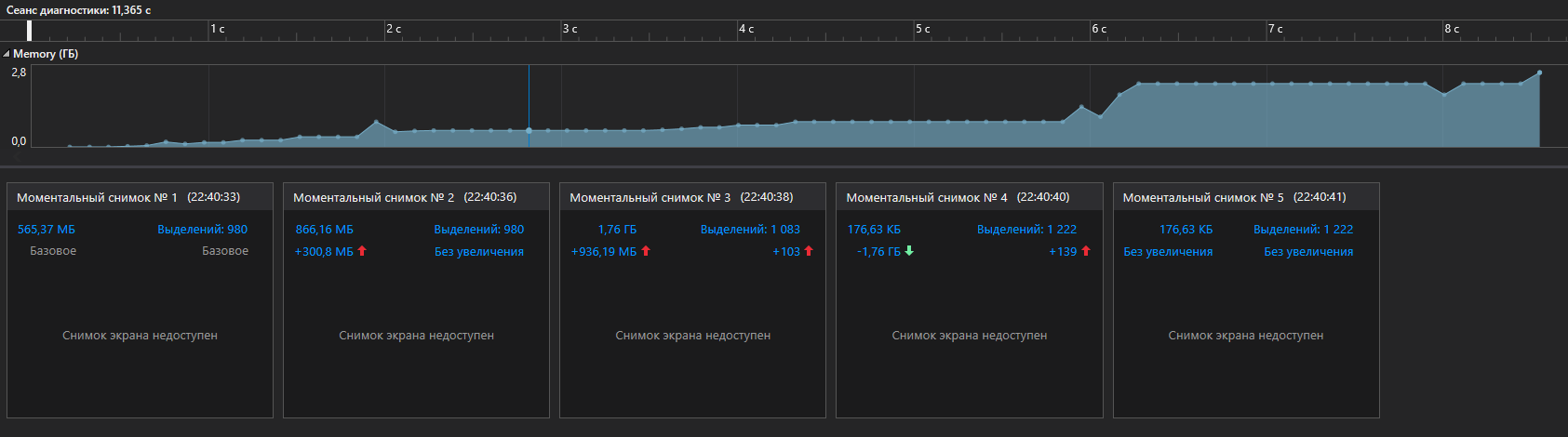
Функция rand\_arr в свое время вызывается из main.

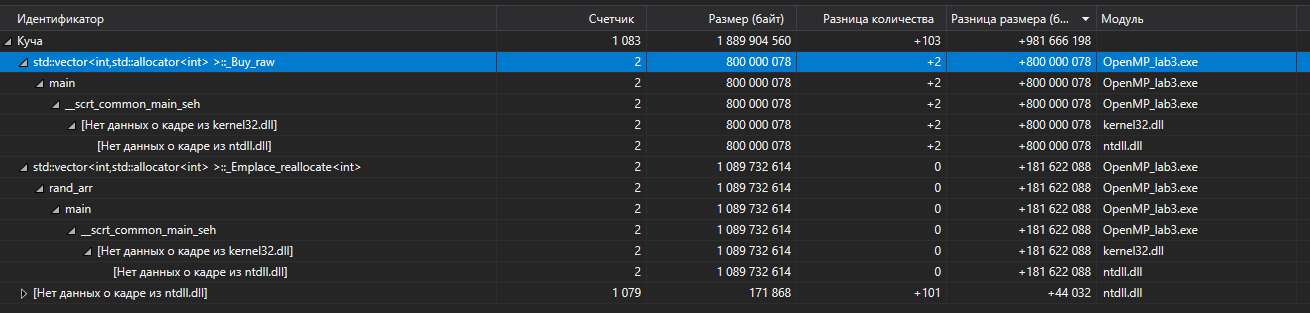


Можем заметить, что максимальное количество вызовов у функции rand.

Также функция параллельного вычисления вызывалась 20 раз, что соответствует количеству логический ядер процессора.

# Провести профилирование с помощью профилировщика выделения памяти.



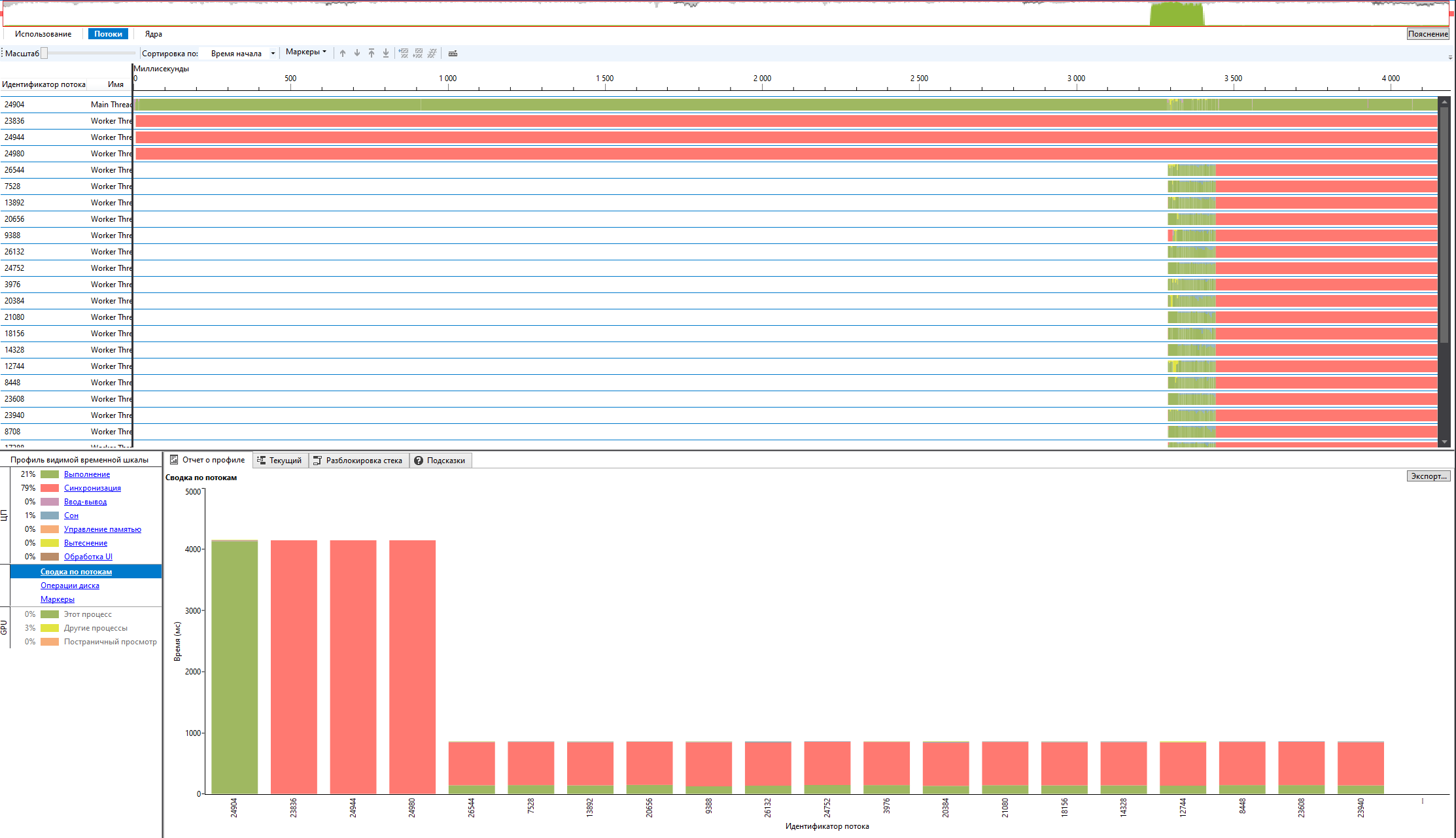


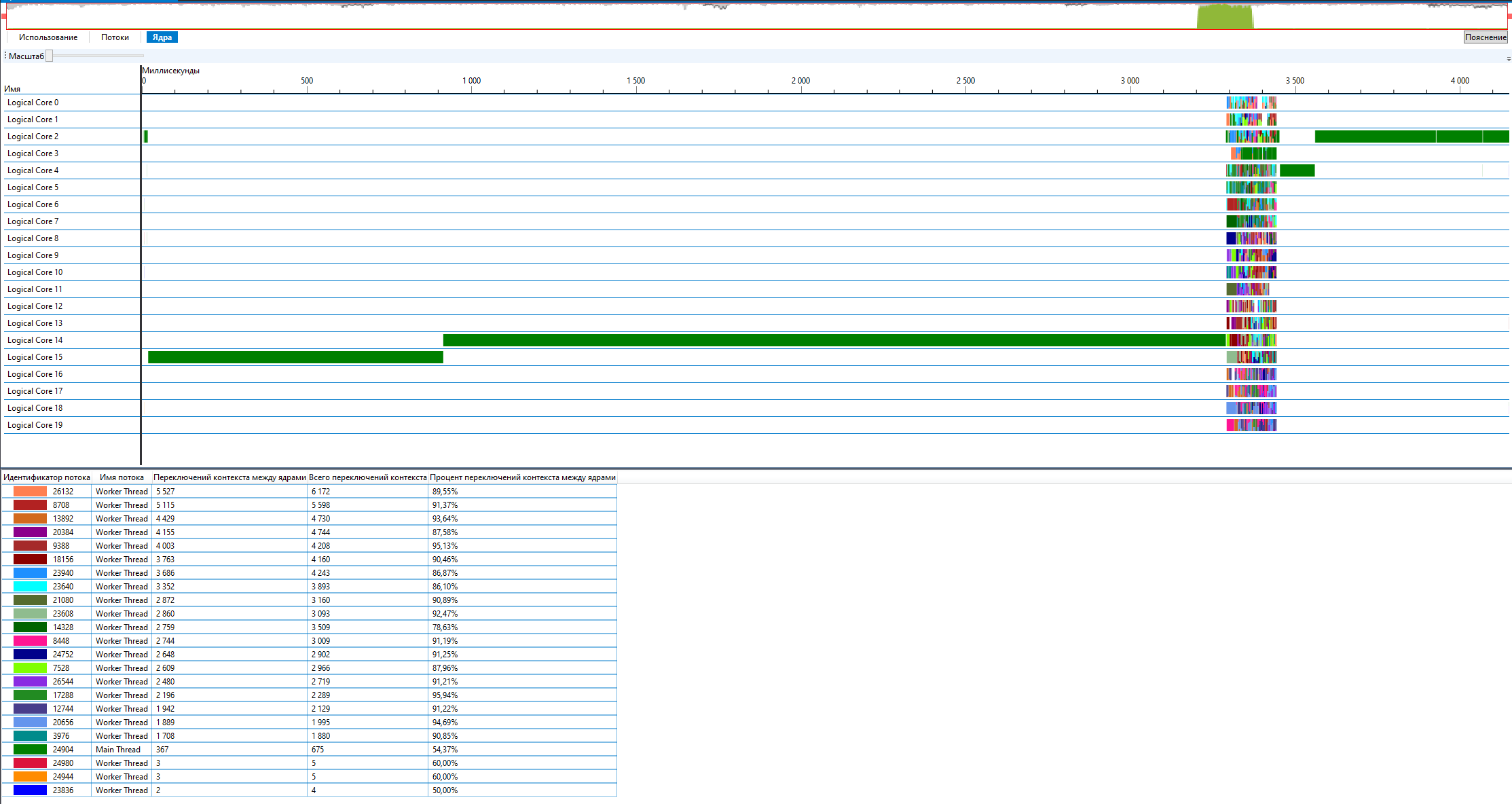
С помощью моментальных снимков мы можем наблюдать изменение памяти во времени, ее освобождение и выделение. Это позволяет оценить расход памяти во времени и может помочь в поиске утечек памяти.

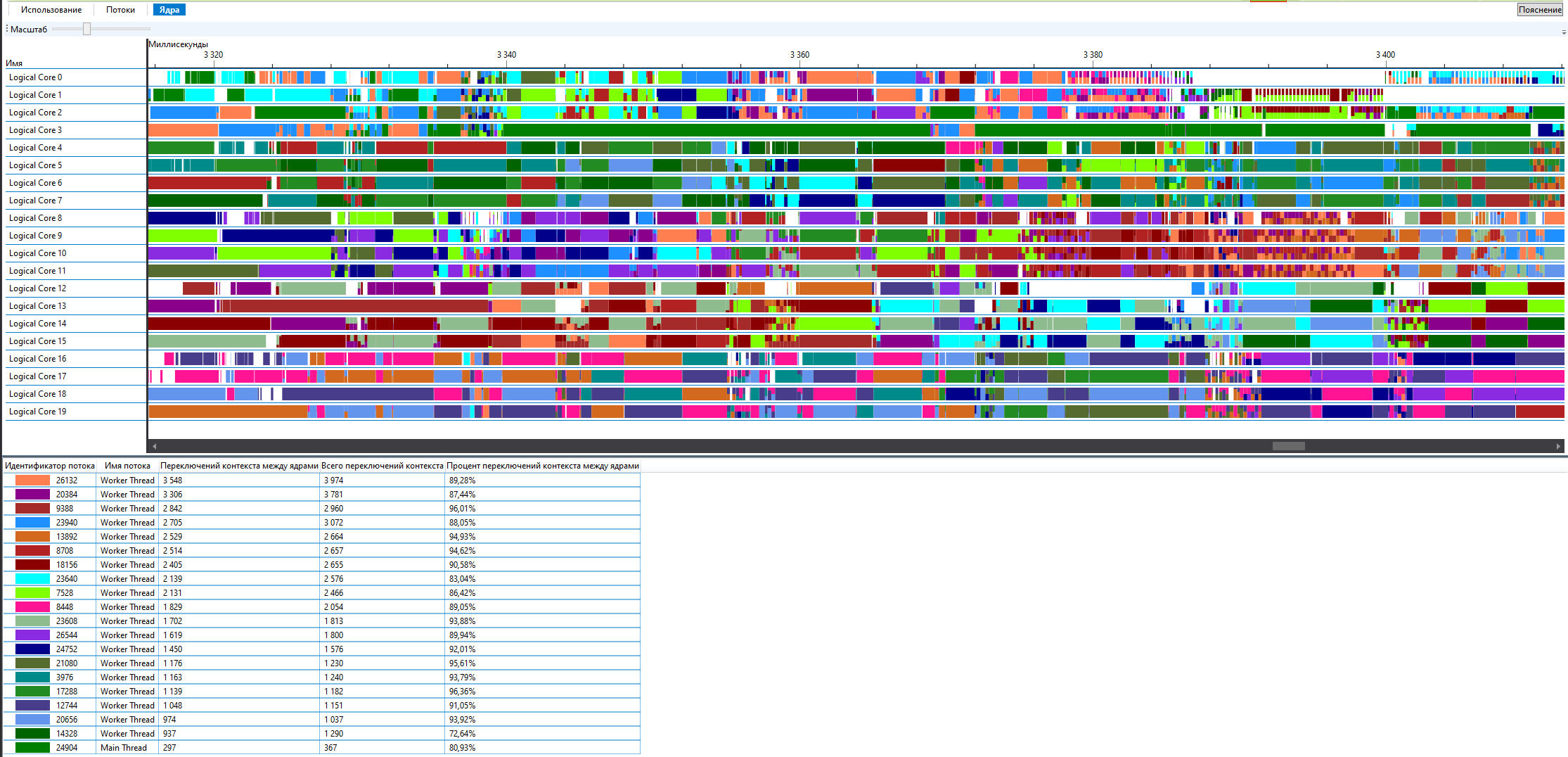
# Провести профилирование с помощью профилировщика конкуренции.



График нагрузки ЦП и GPU.









# Вывод.

На основании полученной информации можно сделать выводы:

* Максимальное время выполнения самописной функции отличной от main – rand\_arr
* Максимальный объем занимаемой оперативной памяти приложением – 2.8 гб
* При параллельном выполнение задействуются все 20 потоков/логический ядер, и нагрузка ЦП возрастает почти до 100 процентов, в остальное же время нагрузка 10% и задействуется одно логическое ядро.
* В программе отсутствует утечка памяти