Для теста генерируется 150000 чисел, из которых отбираются ~7500 простых. Так как мы фиксируем seed генератора, то числа при разных запусках будут одинаковые.
Для теста мы берем только простые числа, так как именно такой набор входных данных позволяет увидеть прирост производительности. Дополнительно приведен test2 и test3, в которых преимущество имеет последовательное решение. Объясняется это тем, что в случайном наборе чисел вероятность встретить простое число не такая уж и маленькая, поэтому последовательно решение находит такое число быстрее, так как для многопоточного исполнения требуется много промежуточных операций. Также и в примере test3, где в списке на первом месте стоит непростое число - последовательное решение находит это число сразу же.

Процессор имеет 4 ядра и поддерживает технологию hyper-threading, что позволяет эффективно использовать 8 потоков для многопоточных задач. Как видно на графике, наилучший результат показал запуск программы с использованием 8 потоков. Меньшее число потоков не позволяет так же быстро выполнять вычисления, а при количестве потоков больше 8, переключение потоков занимает значительное количество времени, а это ухудшает производительность.

Также стоит заметить, что ParallelStream использует 4 потока, так как процессор имеет 4 ядра. Это хорошо видно га графике, где значение для 4 потоков приблизительно равно значению ParallelStream