МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
"САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО"
Институт компьютерных наук и технологий
Направление **02.03.01** : Математика и компьютерные науки

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

«Нагрузочное тестирование»
дисциплина: «Программирование на Java»

Исполнитель: _____ Яшнова Дарья Михайловна
группа 5130201/20002

Руководитель: _____ Лукашин Антон Андреевич

«___»_____ 2025г

Санкт-Петербург, 2025

1 Постановка задачи

- Create a separate project to measure performance of your HTTP server and JSON parser
- You can use load testing frameworks (like JMeter for example) or write your own scripts
- HTTP server from `lab-2` should be configured to:
  - `Request 1`: Accept request, parse JSON, store something in file (or in database, you could use something like SQLite, but **NOT** in-memory DB), retrieve something from file
  - `Request 2`: Accept request, parse JSON, get something from memory (or perform calculations), create and return JSON
- Combine different variants: Virtual/Classic Threads, your JSON library (`lab-1`) / Jackson or Gson
- It will be good to run load tests on separate machine (one for HTTP server and another for tests)
- The report must contain:
  - How to configure and launch (README)
  - Experiment description
  - Hardware description
  - Experiment parameters (number of threads, number of requests, amount of data etc)
  - resulting table
- The table with results must contains:

req	Virtual + own parser	Virtual + GSON	Classic + own parser	Classic + GSON
Request-1	avg time per request	...	...	...
Request-2	avg time per request	...	...	...

2 Описание эксперимента

2.1 Цель эксперимента

Сравнить производительность HTTP-сервера при использовании:

- Виртуальных потоков (Loom Project)
- Классических потоков (ThreadPoolExecutor)

2.2 Параметры тестирования

Таблица 1: Параметры эксперимента

Сервер	Java HTTP-сервер (на базе NIO)
Обработчики запросов	`Request1Handler`, `Request2Handler`
Виды потоков	Виртуальный, классический
Количество запросов	1000 на каждый режим
Количество потоков	20 потоков

Внутри Request1:
1. Парсинг JSON
2. Сохранение в БД
3. Получение данных из БД
4. Формирование ответа

Внутри Request2:
1.Извлечение числовых данных
2. Расширенный статистический анализ Вычисляет:
- Базовые метрики: сумма, среднее, медиана
- Показатели распределения: стандартное отклонение, диапазон, межквартильный размах (IQR)
3. Формирование ответа
Возвращает JSON с рассчитанной статистикой

2.2.1 Использование JMeter

Настройки Thread Group в Jmeter представлены на рис.1. Настройки запроса 1 и данные представлены на рис. 2. Настройки запроса 2 и данные представлены на рис. 3. Для данного запроса была сгененирована случайная последовательность 1000 чисел.

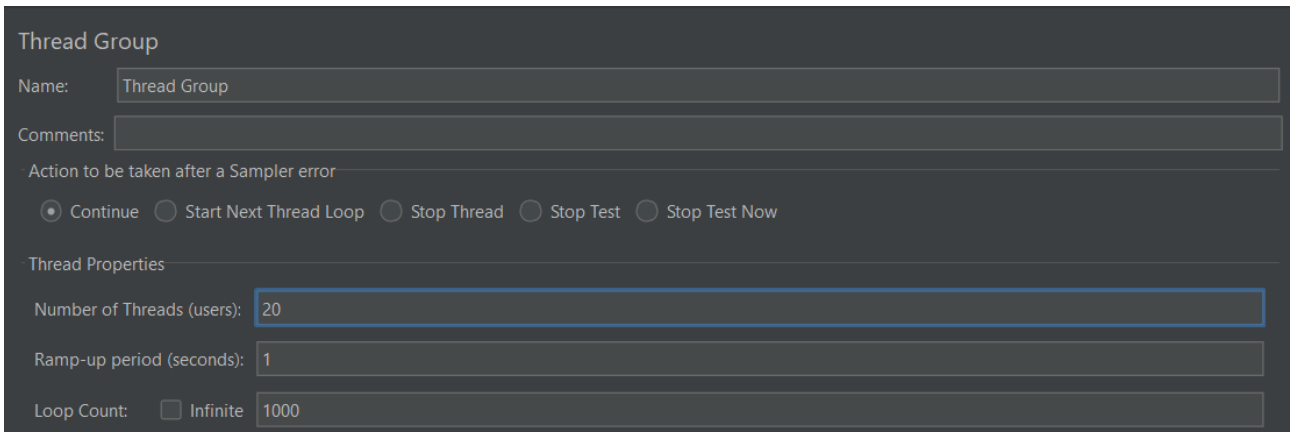

Рис. 1: Настройки Thread Group

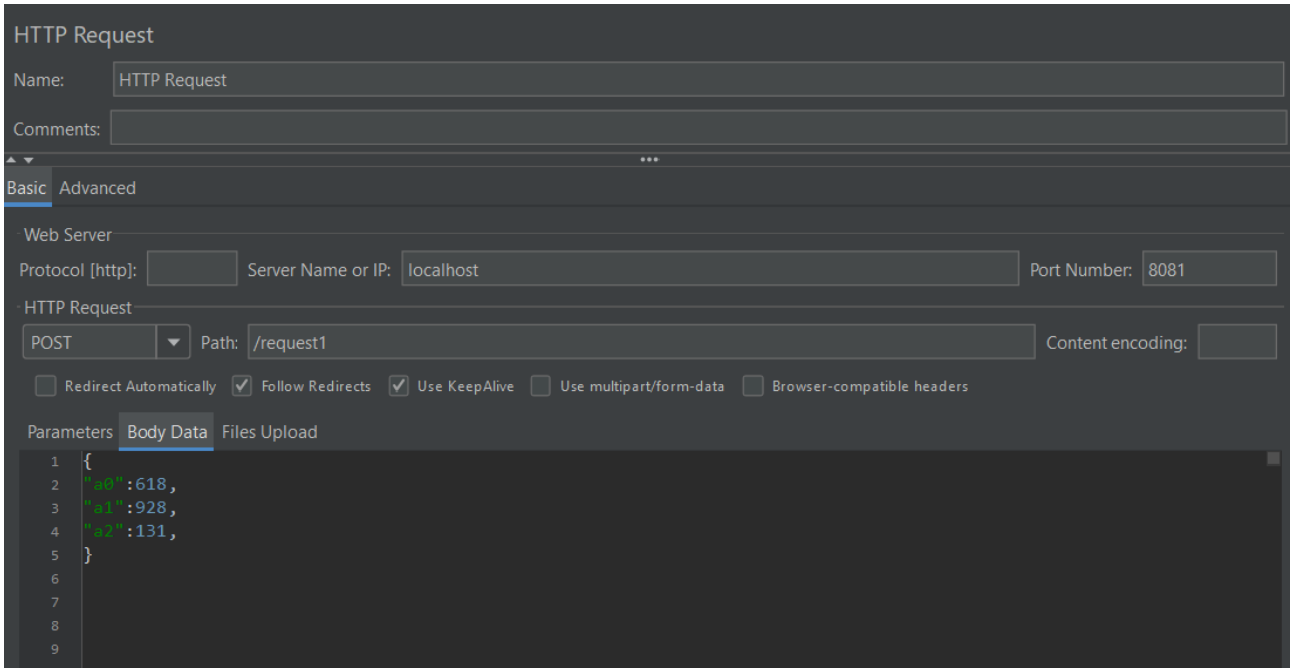

Рис. 2: Настройки запроса 1

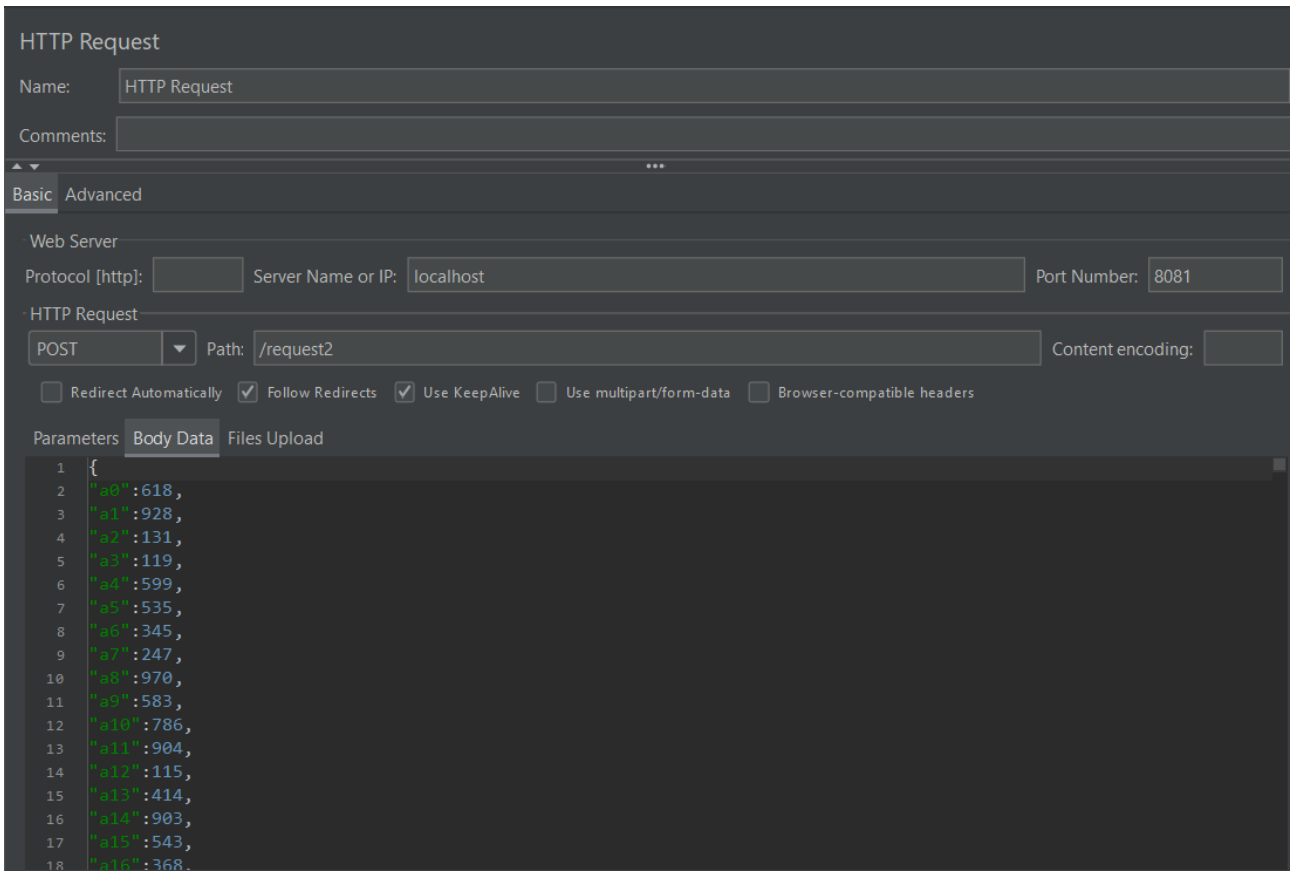

Рис. 3: Настройки запроса 2

2.3 Аппаратная часть

Процессор: Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80 GHz
Оперативная память: 8,00 ГБ
Тип системы: 64-разрядная операционная система, процессор x64
ОС: Windows 10 Pro
SSD: 237 гб

2.4 Результаты request1

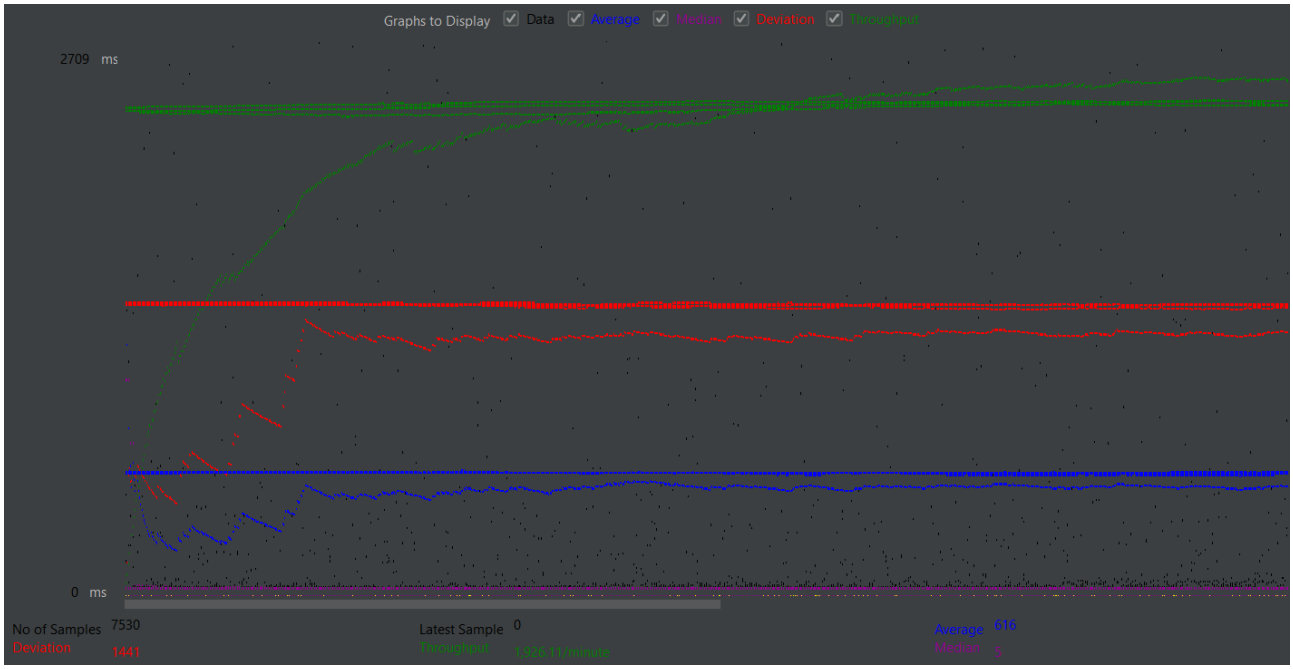

Рис. 4: Классические потоки, GSON

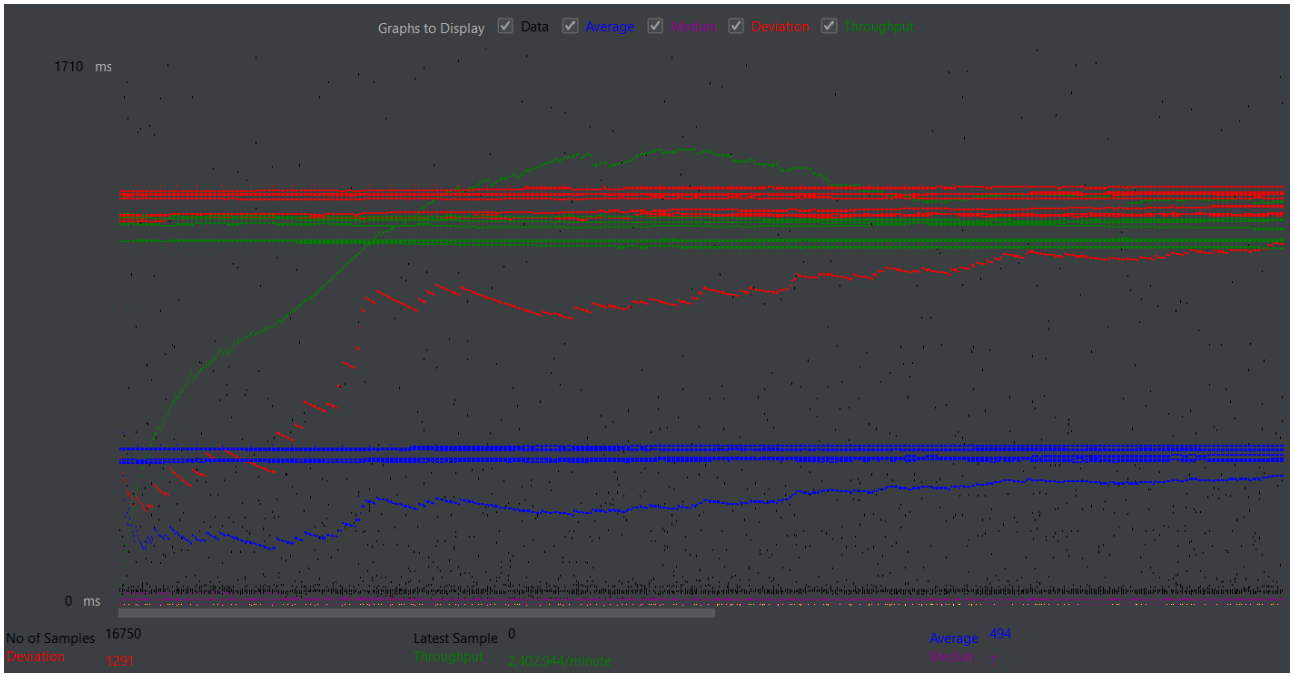

Рис. 5: Классические потоки, собственный парсер

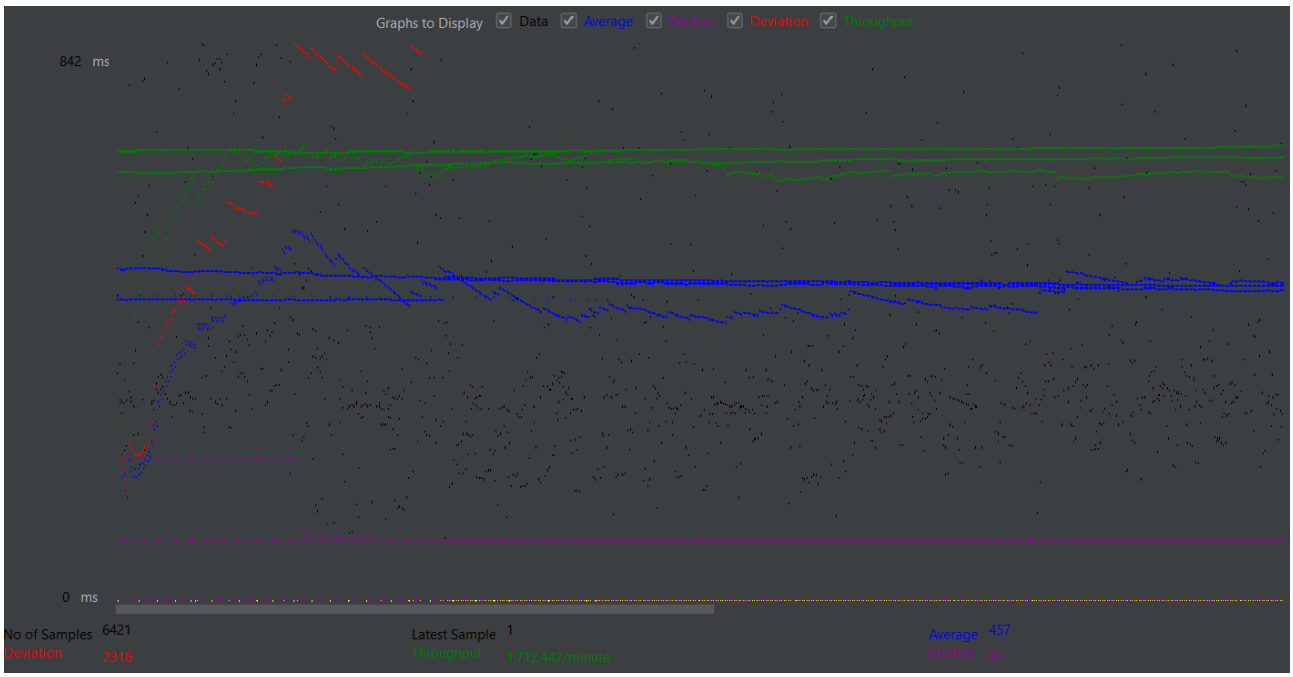

Рис. 6: Виртуальные потоки, собственный парсер

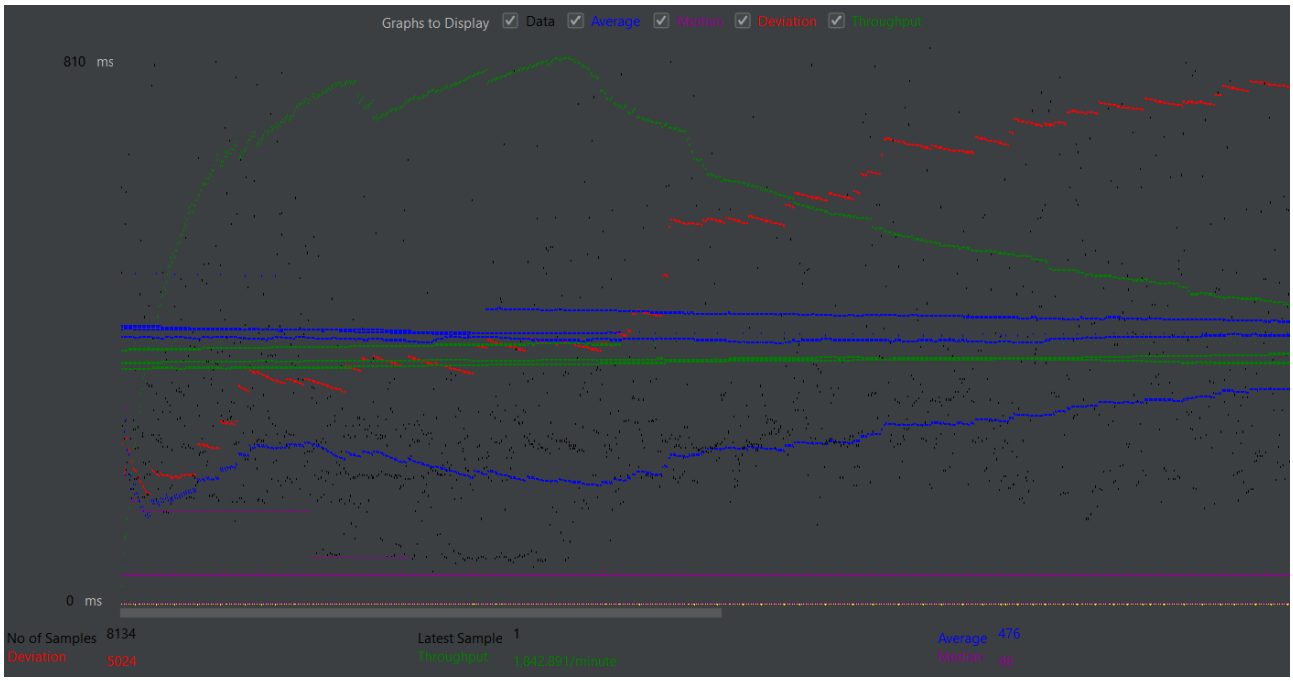

Рис. 7: Виртуальные потоки, GSON

2.5 Результаты request2

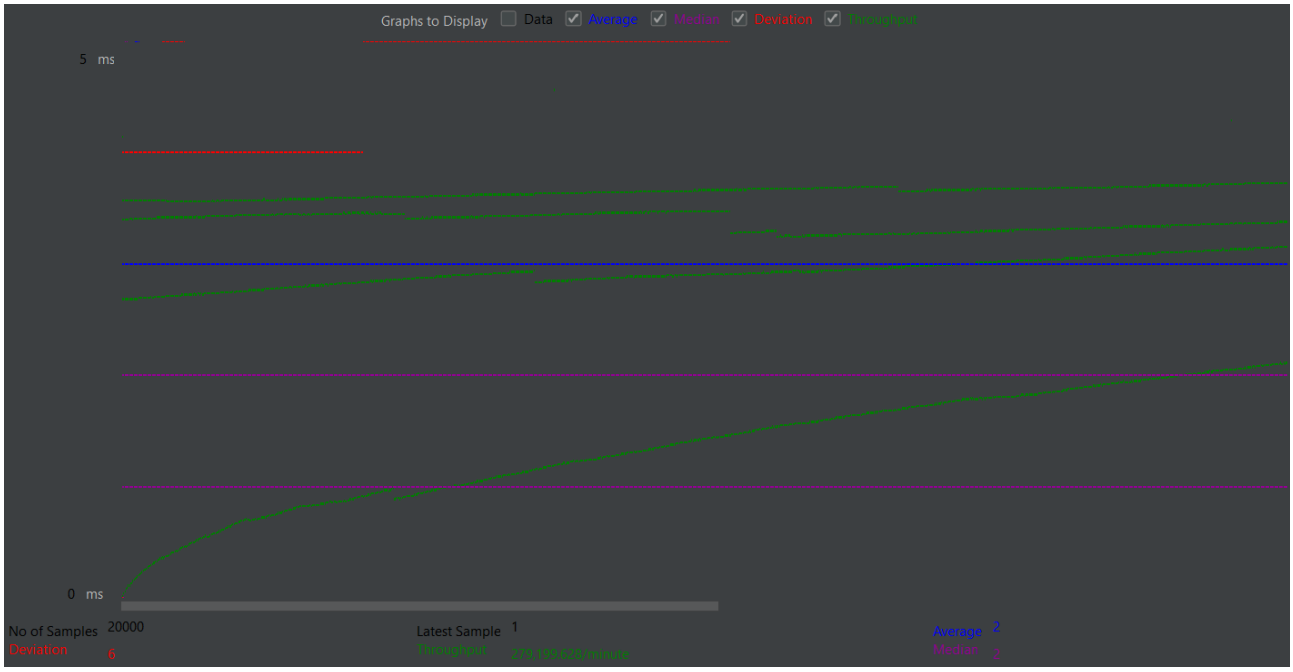

Рис. 8: Классические потоки, собственный парсер

Рис. 9: Классические потоки, GSON

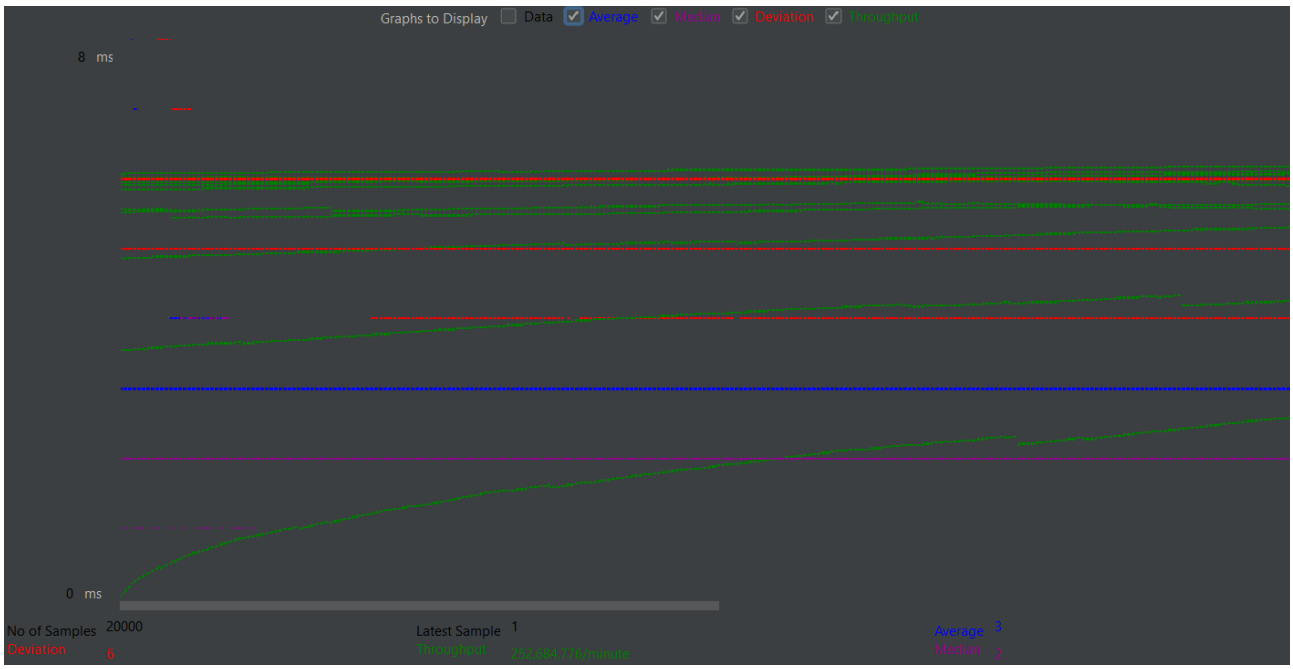

Рис. 10: Виртуальные потоки, GSON

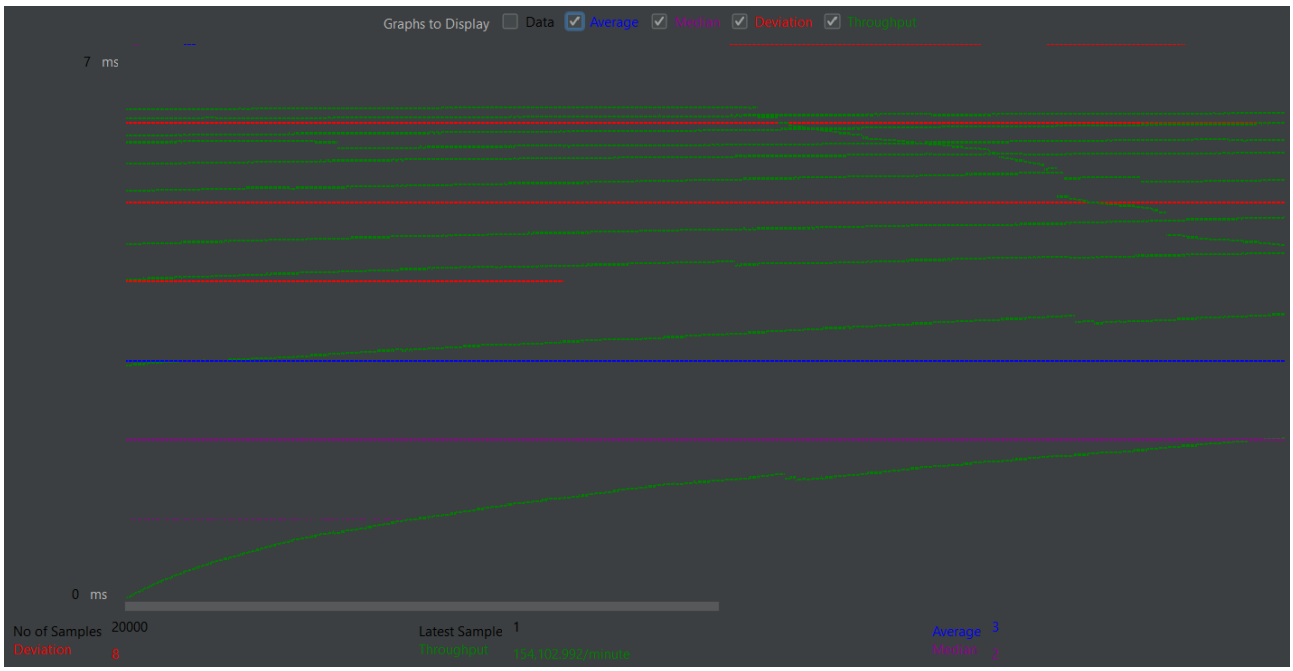

Рис. 11: Виртуальные потоки, собственный парсер

3 Выводы

При тестировании различных конфигураций получились следующие показатели:

req	Virtual + own parser	Virtual + GSON	Classic + own parser	Classic + GSON
Request-1	35.05 ms	32.57 ms	31.15 ms	24.97 ms
Request-2	0.39 ms	0.24 ms	0.21 ms	0.31 ms

- GSON демонстрирует лучшую производительность по сравнению с собственным парсером во всех тестах. Разница особенно заметна для Request-1 (ускорение около 7%).
- Request-1 требует значительно больше времени обработки (24.97-35.05 мс) по сравнению с Request-2 (0.21-0.39 мс), что указывает на более сложную логику обработки.
- Реализация с виртуальными потоками во всех случаях работает медленнее, чем с классическими.