Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»
Институт компьютерных наук и кибербезопасности
Высшая школа технологий искусственного интеллекта
Направление: 02.03.01 Математика и компьютерные науки

Отчёт по дисциплине «Программирование на языке JAVA»
Лабораторная работа №3
«Load testing»

Студент: _____ Богдан А. В.
группы 5130201/20002
Преподаватель: _____ Лукашин А. А.

«_____»_____ 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

Эксперимент проводился на одной вычислительной машине, на ней запускался и JMeter и сервер.

2.1 Цель

Оценить производительность HTTP-сервера при различных конфигурациях:

- Виртуальные или классические потоки.
- Собственный парсер или GSON.

2.2 Описание аппаратного обеспечения

Серверная часть:

- Тип сервера: Физический сервер.
- Процессор: Intel Core Ultra 7 155H, 16 ядер, 22 потока 3.8-4.8 GHz.
- Оперативная память: 16 ГБ LPDDR5.
- Хранилище: SSD NVMe, 1 ТБ.
- Хранилище: Операционная система: Windows 11 Домашняя.

2.3 Описание параметров эксперимента

Для проведения нагрузочного тестирования была использована программа Apache Jmeter. Для всех восьми экспериментов были выставлены следующие параметры тестового плана:

- Количество потоков (пользователей): 1000.
- Время, в течение которого будут прибавляться пользователи: 1.
- Количество раз выполнения тестирования: 100.
- Количество запросов: 100000.
- Объём данных: JSON-запрос для 1 запроса - 91 байт, JSON-запрос для 2 запроса - 55 байт. JSON-ответ для 1 запроса - 9 байт, JSON-ответ для 2 запроса - 4.75 Кбайта.

Серверное приложение было запущено с 10 потоками на localhost и порте 30001. А после перезапускалось для изменения используемого парсера и вида потоков.

На рисунке 1 показаны общие параметры запросов - хост и порт, а на рисунке 2 и 3 отображены параметры и тела запросов 1 и 2.

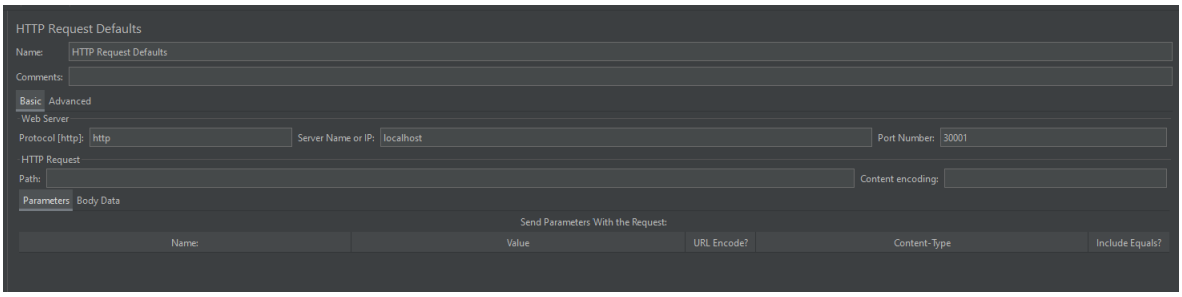

Рис. 1. Общие параметры запросов.

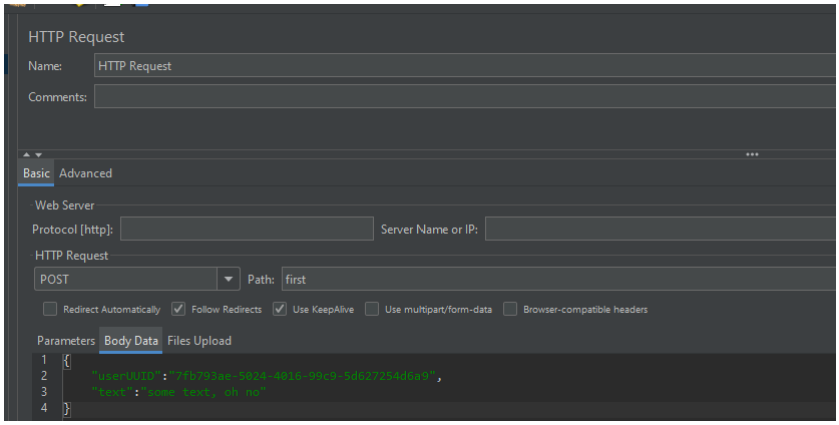

Рис. 2. Параметры первого запроса.

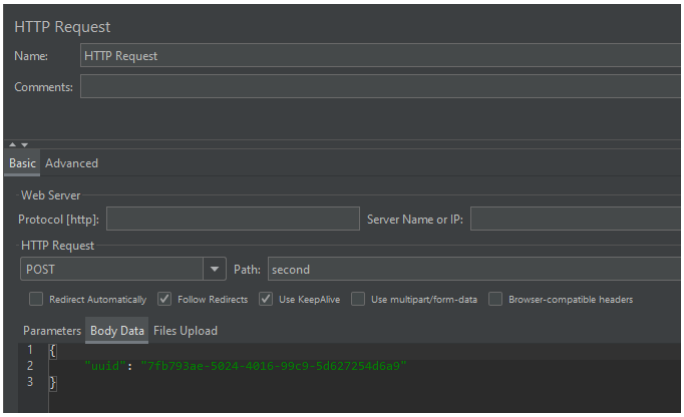

Рис. 3. Параметры второго запроса.

3 Результаты

Все результаты проведения нагрузочного тестирования представлены на рисунках 4-11.

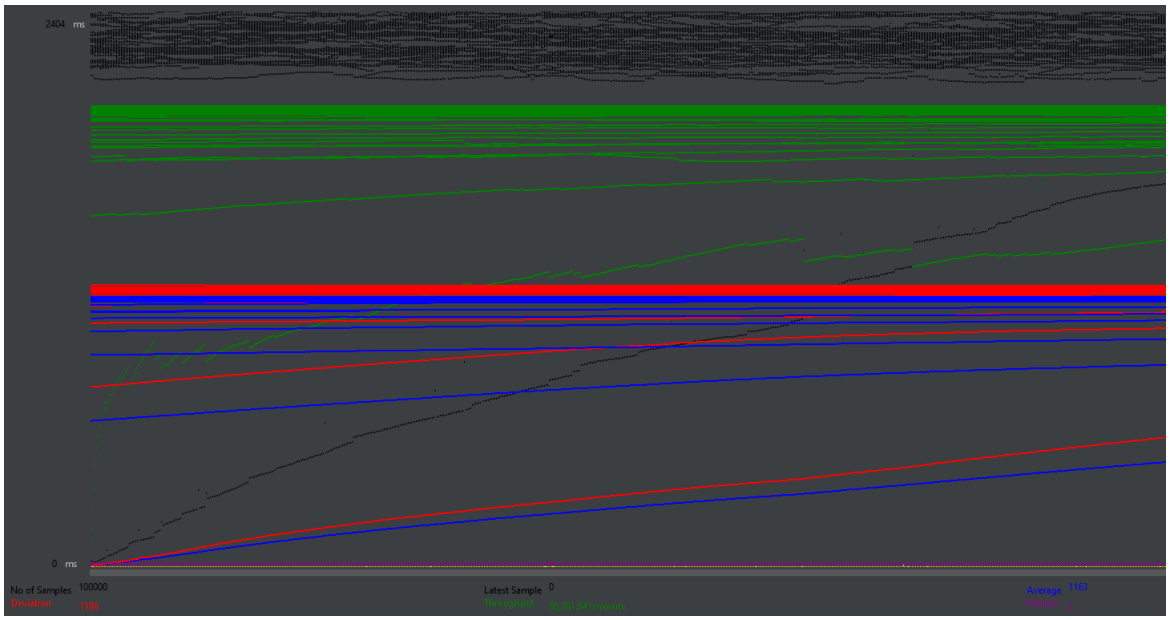

Рис. 4 Результаты выполнения 1 запроса с Virtual + own parser конфигурацией.

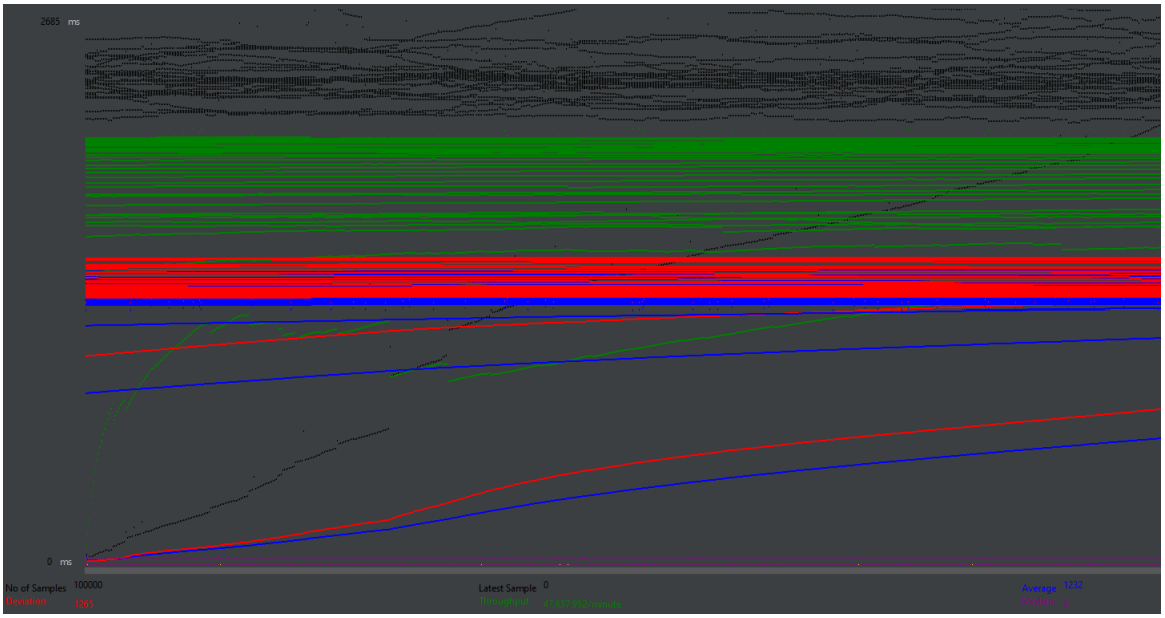

Рис. 5. Результаты выполнения 1 запроса с Virtual + GSON конфигурацией.

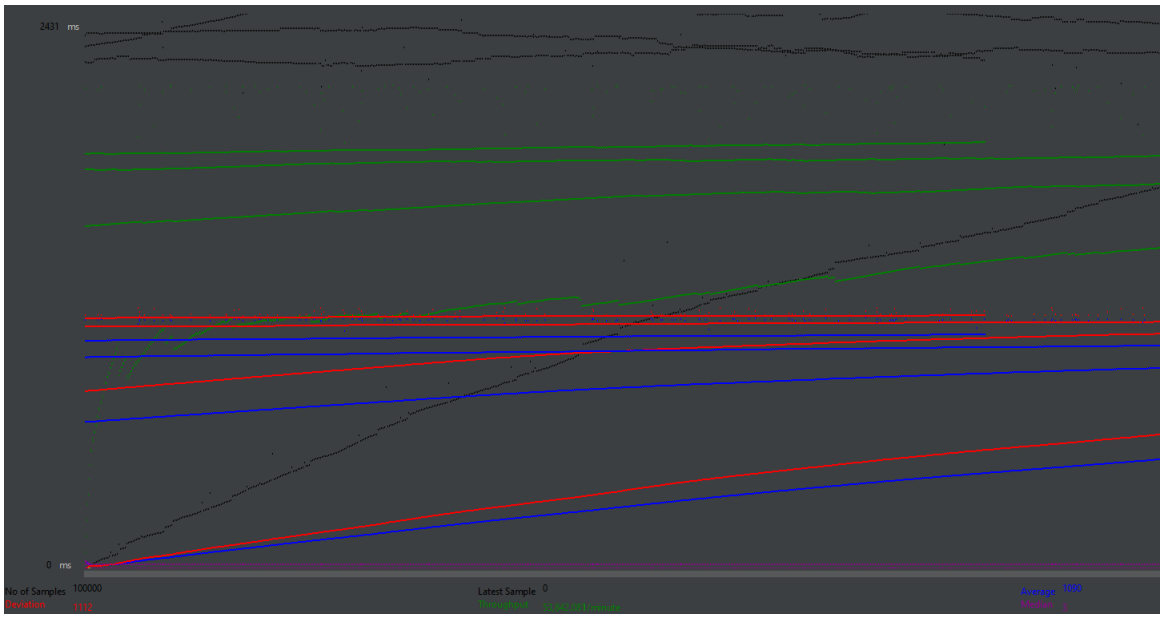

Рис. 6. Результаты выполнения 1 запроса с Classic + own parser конфигурацией.

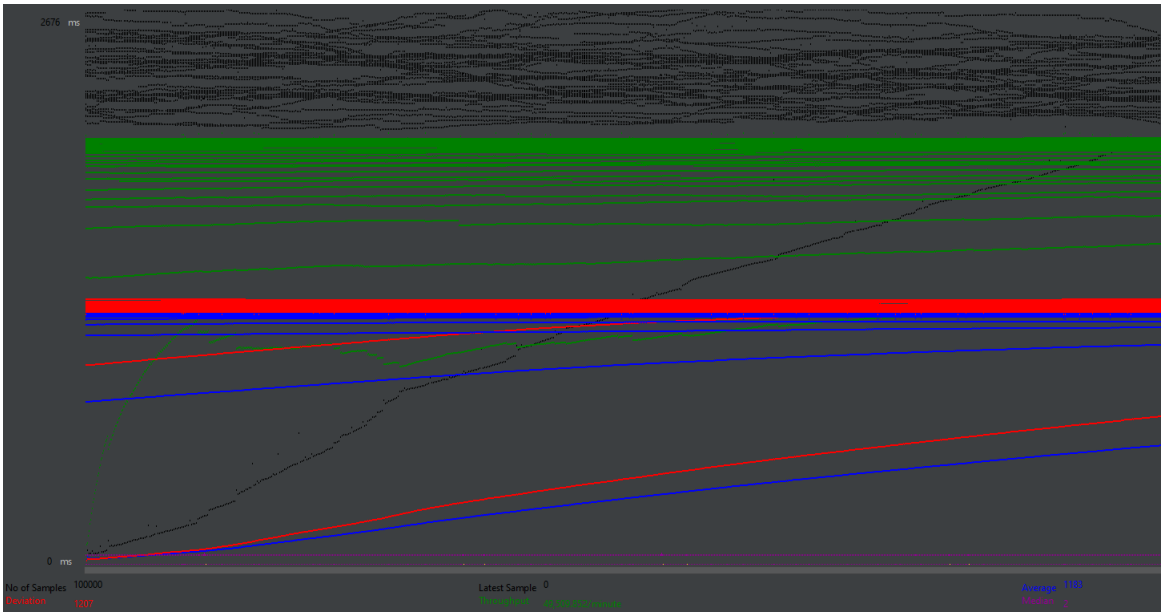

Рис. 7. Результаты выполнения 1 запроса с Classic + GSON конфигурацией.

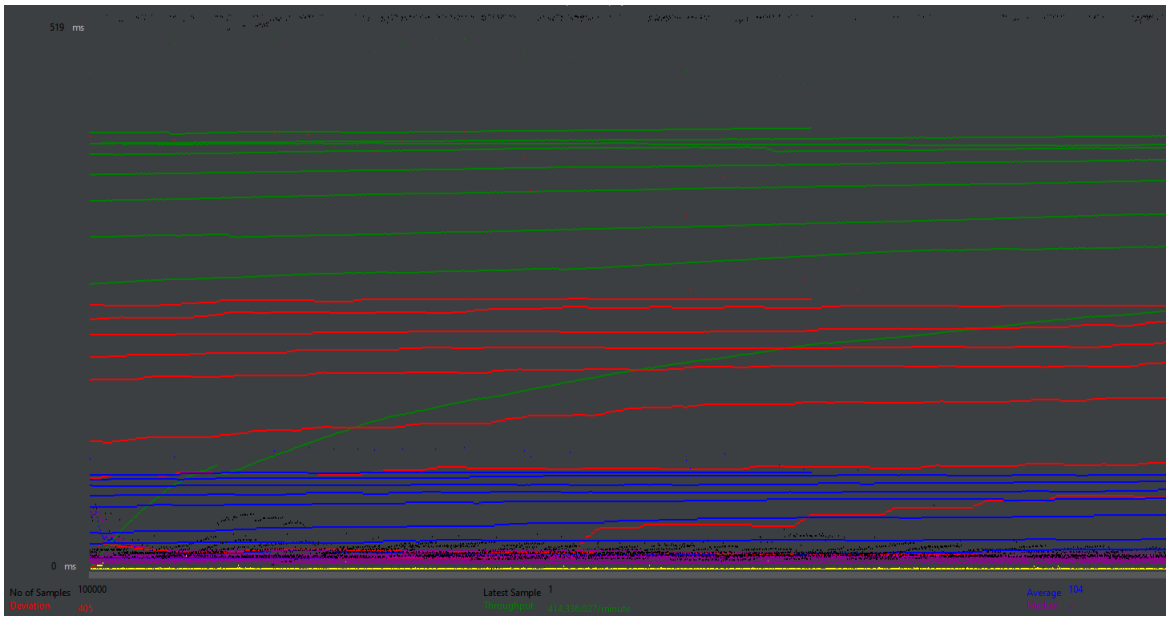

Рис. 8. Результаты выполнения 2 запроса с Virtual + own parser конфигурацией.

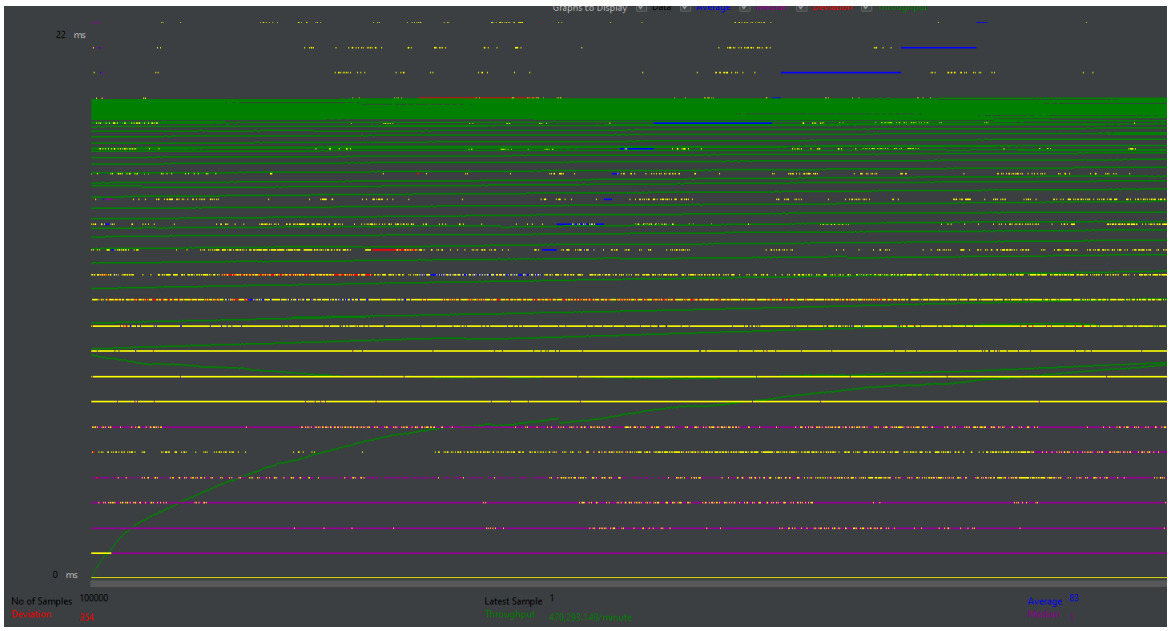

Рис. 9. Результаты выполнения 2 запроса с Virtual + GSON конфигурацией.

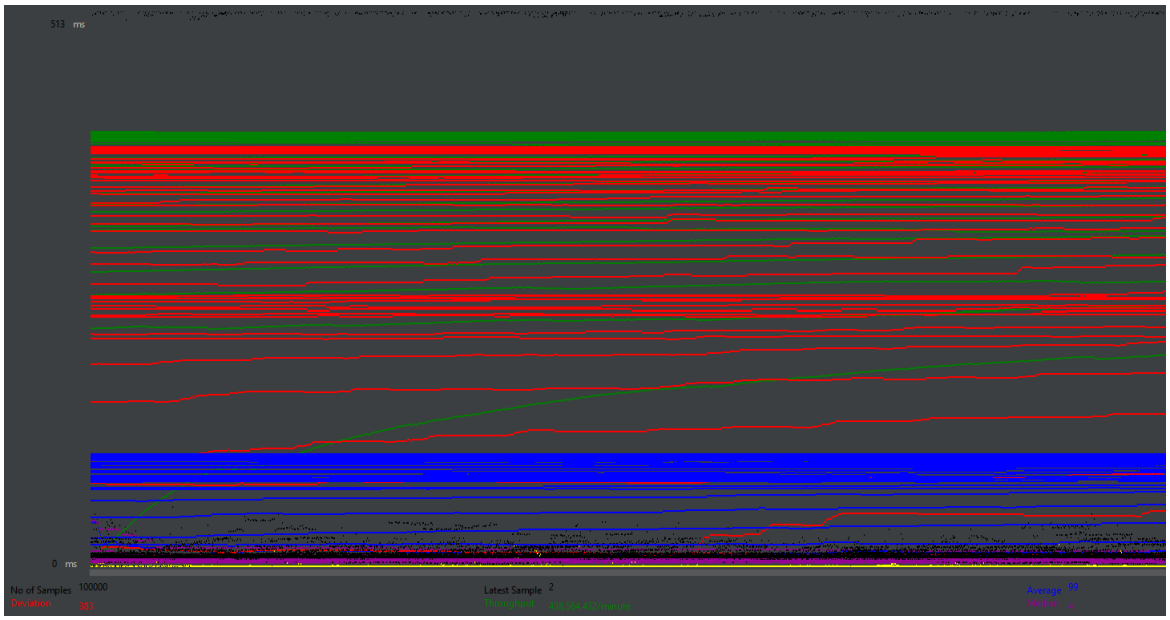

Рис. 10. Результаты выполнения 2 запроса с Classic + own parser конфигурацией.

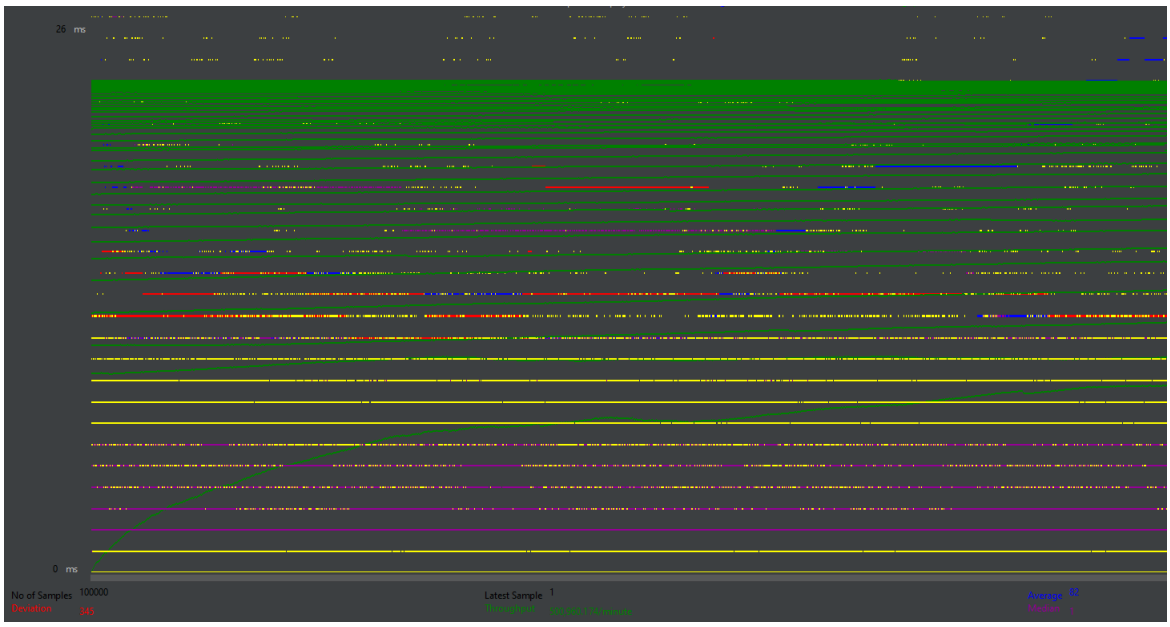

Рис. 11. Результаты выполнения 2 запроса с Classic + GSON конфигурацией.

В таблице 1 представлены результаты эксперимента.

Таблица 1. Результаты эксперимента

req	Virtual + own parser	Virtual + GSON	Classic + own parser	Classic + GSON
Request-1	1.191 мс	1.2595 мс	1.1144	1.212 мс
Request-2	0.1448 мс	0.1276 мс	0.1368 мс	0.1198 мс

На основе проведённого эксперимента можно сделать следующие выводы:

1. Влияние типа потоков на производительность.

- Виртуальные потоки меньшую производительность с классическими потоками для обоих типов запросов.
- Для Request-1 время обработки с собственным парсером составило 1.191 мс (Virtual) и 1.1144 мс (Classic).
- Для Request-1 время обработки с GSON составило 1.2595 мс (Virtual) и 1.212 мс (Classic).
- Для Request-2 время обработки с собственным парсером составило 0.1448 мс (Virtual) и 0.1368 мс (Classic).
- Для Request-2 время обработки с GSON составило 0.1276 мс (Virtual) и 0.1198 мс (Classic).
- Классические потоки демонстрируют преимущество в 5% - 8 % в скорости обработки, что может быть связано с отсутствием накладных расходов на управление виртуальными потоками.

2. Влияние парсера на производительность.

   - Собственный парсер показал прирост в производительности в запросах с простым телом на 5% - 8% и ответом:

```
Тело:
{
    "userUUID":"7fb793ae-5024-4016-99c9-5d627254d6a9",
    "text":"some␣text,␣oh␣no"
}
Ответ:
{
    "userUUID": "7fb793ae-5024-4016-99c9-5d627254d6a9",
    "publicationID": 99786,
    "text": "some␣text,␣oh␣no"
}
```

     Когда как с более сложными ответами, например, где присутствуют JSON-массивы, собственный парсер работает хуже чем GSON на 13.4% - 14.1%.

   - Для Request-1 время обработки с виртуальными потоками составило 1.191 мс (Own Parser) и 1.2595 мс (GSON).
   - Для Request-1 время обработки с классическими потоками составило 1.1144 мс (Own Parser) и 1.212 мс (GSON).
   - Для Request-2 время обработки с виртуальными потоками составило 0.1448 мс (Own Parser) и 0.1276 мс (GSON).
   - Для Request-2 время обработки с классическими потоками составило 0.1368 мс (Own Parser) и 0.1198 мс (GSON).
   - Это может быть связано с тем, что собственный парсер сильно оптимизирован для простых JSON-структур, когда как GSON хорошо оптимизирован для любых JSON-структур.

4 Заключение

Проведённый эксперимент показал, что:

1. Классические потоки демонстрируют лучшую производительность на 5%-8% по сравнению с виртуальными потоками.
2. Собственный парсер показывает лучшие результаты на 5%-8% на простых запросах и ответах по сравнению с GSON, однако, когда используются более сложные JSON структуры по производительности выигрывает GSON, примерно на 13.4% - 14.1%.
3. Классические потоки в сочетании с GSON демонстрируют лучшую производительность в тесте со вторым запросом, но для систем с преобладанием простых запросов и ответов стоит рассмотреть вариант с использованием собственного парсера.