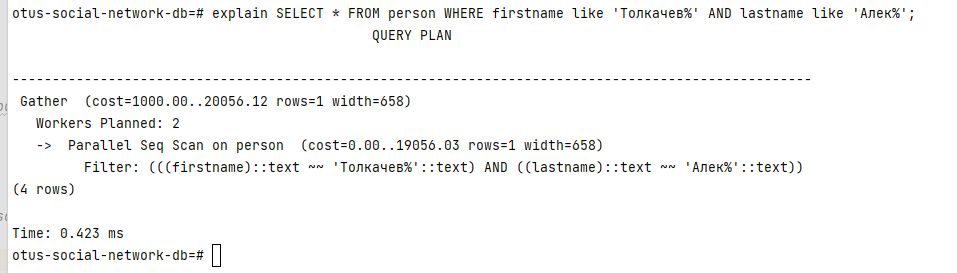
OTUS: Производительность индексов

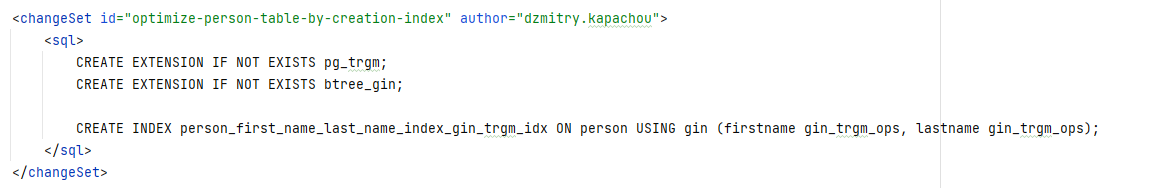
В данных замерах была использована база данных`postgres` (docker-image: 10.05) с выборкой в 1\_000\_000+ записей в таблице `person`.



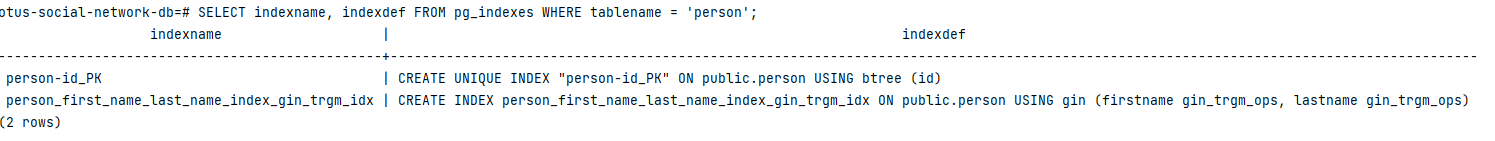
Тестирование производилось на запросе /user/search/\*\*\*:  
SELECT \*   
FROM person  
WHERE firstname like %Толкачев% AND lastname like %Алек%  
ORDER BY id ASC

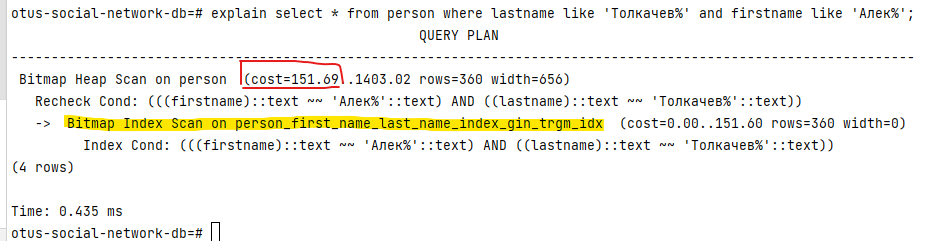
*SQL Plan на запрос в базу без индексов:*

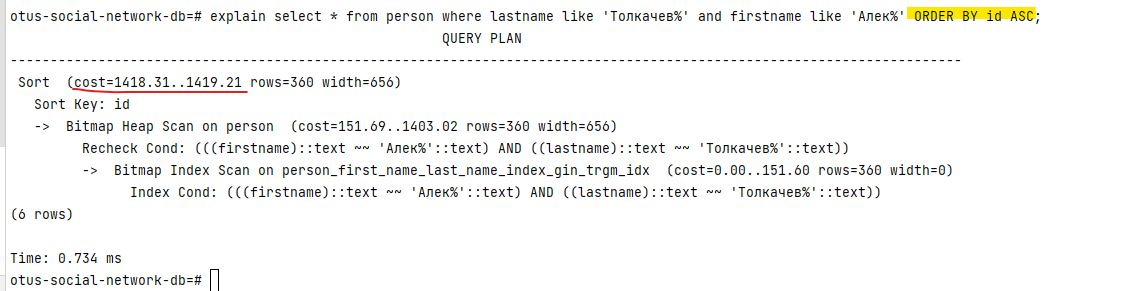
**

Скрипт Liquibase на создание индекса:  


Добавляем индекс GIN (на параметры запроса firstname/lastname), т.к нужен оптимизированный matcher на подстроки в WHERE LIKE конструкций запроса:



План запроса будет выглядеть следующим образом:

И в случае добавления сортировки по возрастанию (+ ORDER BY id ASC):  


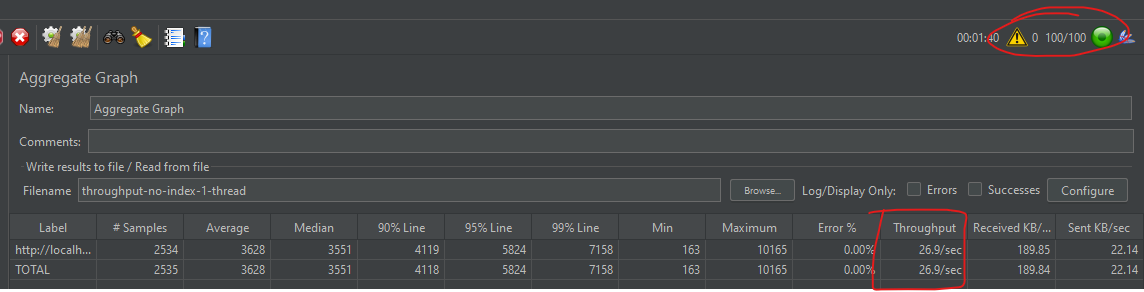
Пропустил диаграммы метрик на 1/10 потоков, т.к вижу не такую значительная разницу на маленьких цифрах и лишние скриншоты диаграмм лишь подтверждающие результаты потоков 100/1000.

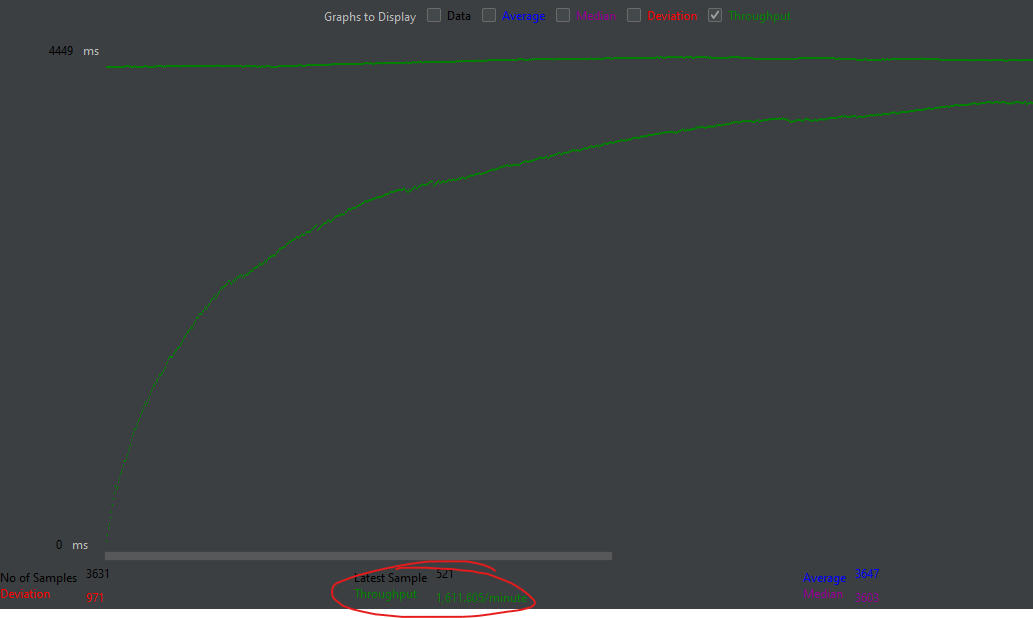
Выводы на основе метрик, представленных ниже:

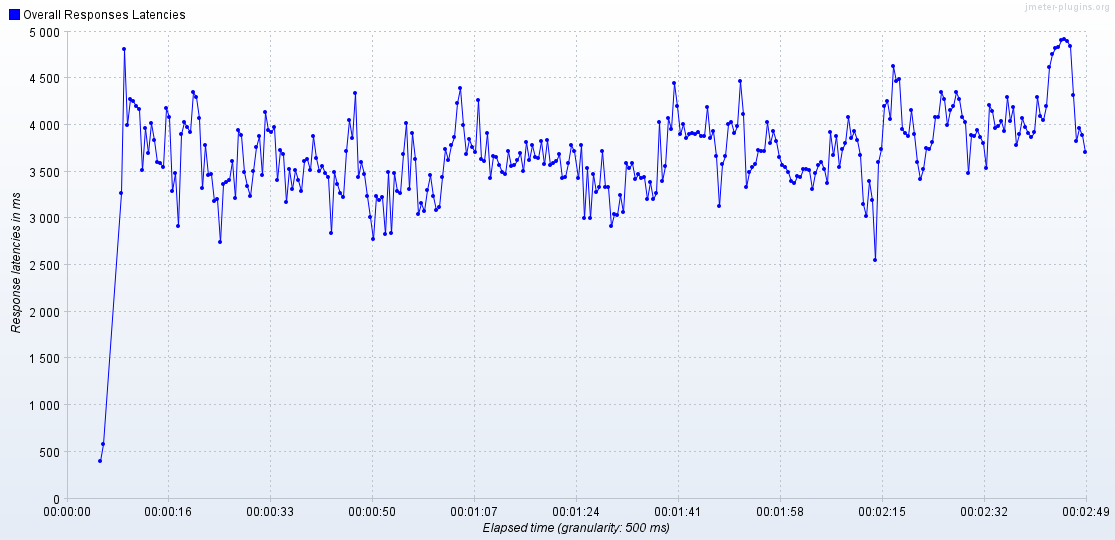
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Threads | latency, ms  (no index) | latency, ms | throughput, rps  (no index) | throughput, rps |
| 100 | ~4k+ | Latency < 200 | 26.9/sec | 712.5/sec |
| 1000 | ~38k+ | Latency < 2k | 30.3/sec | 555. 2/sec |

JMeter latency and throughput metrics diagram with no indexes for – 100 threads:

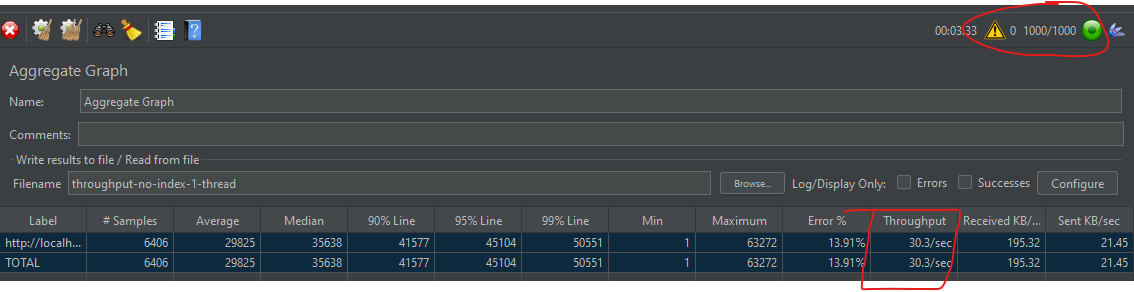
Throughput:

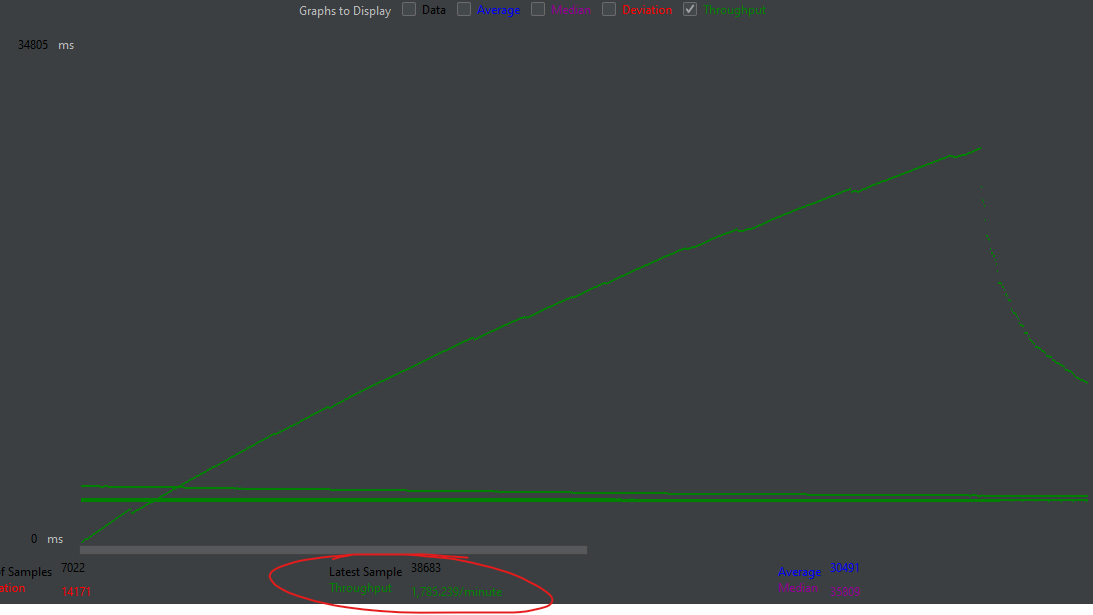




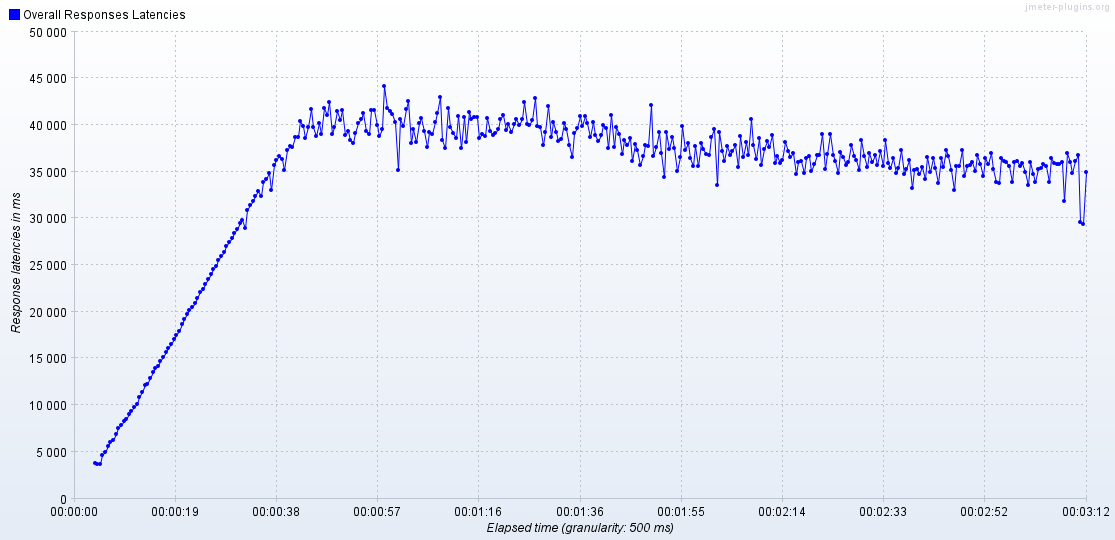
Latency:  


JMeter latency and throughput metrics diagram with no indexes for – 1000 threads:

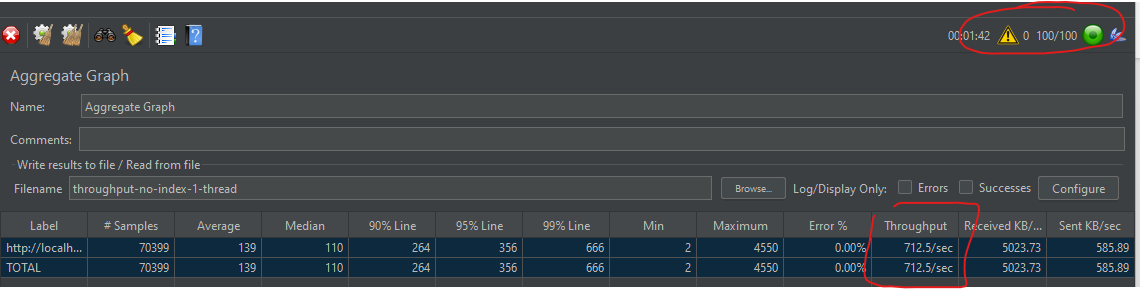
Throughput:  


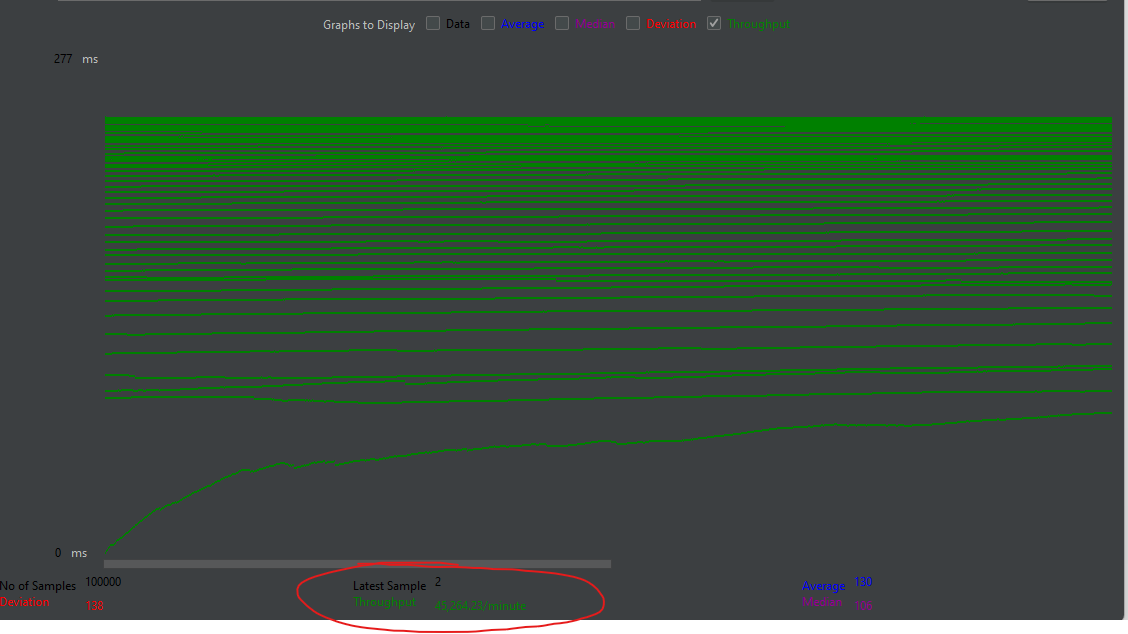


Latency:

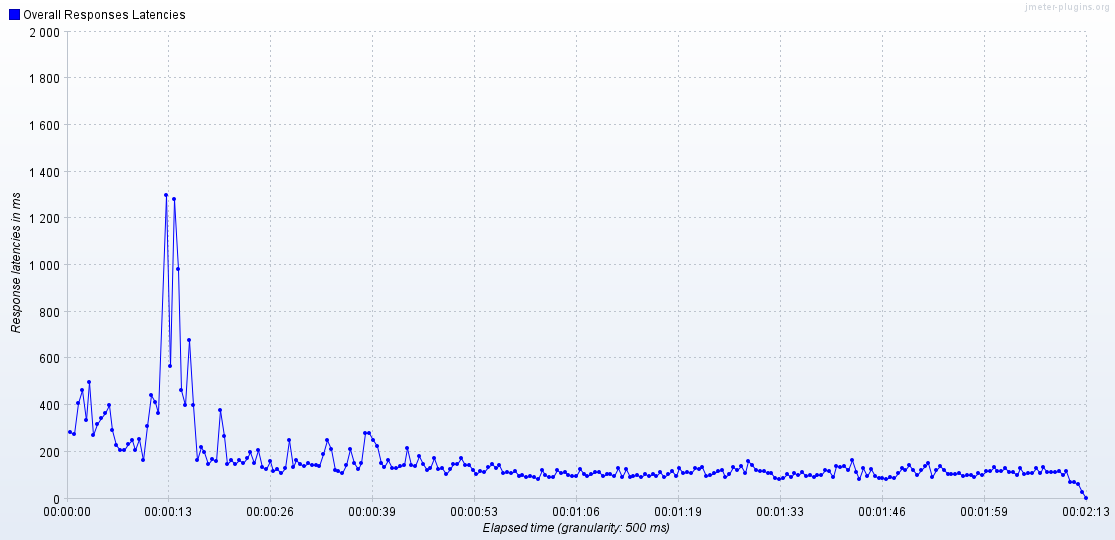


JMeter latency and throughput metrics diagram with built indexes for – 100 threads:

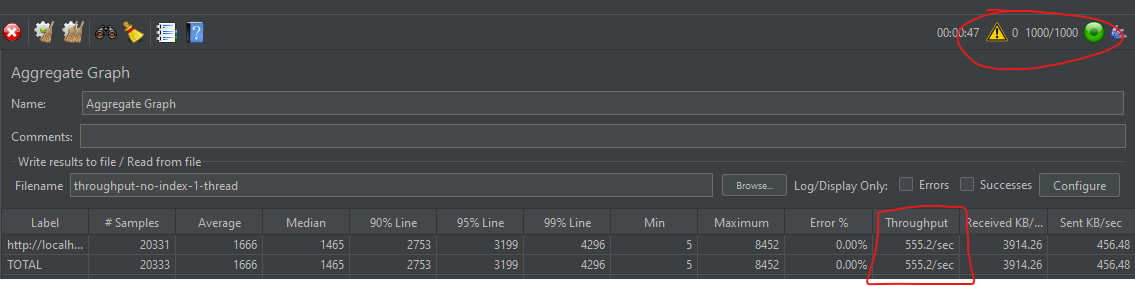
Throughput:  


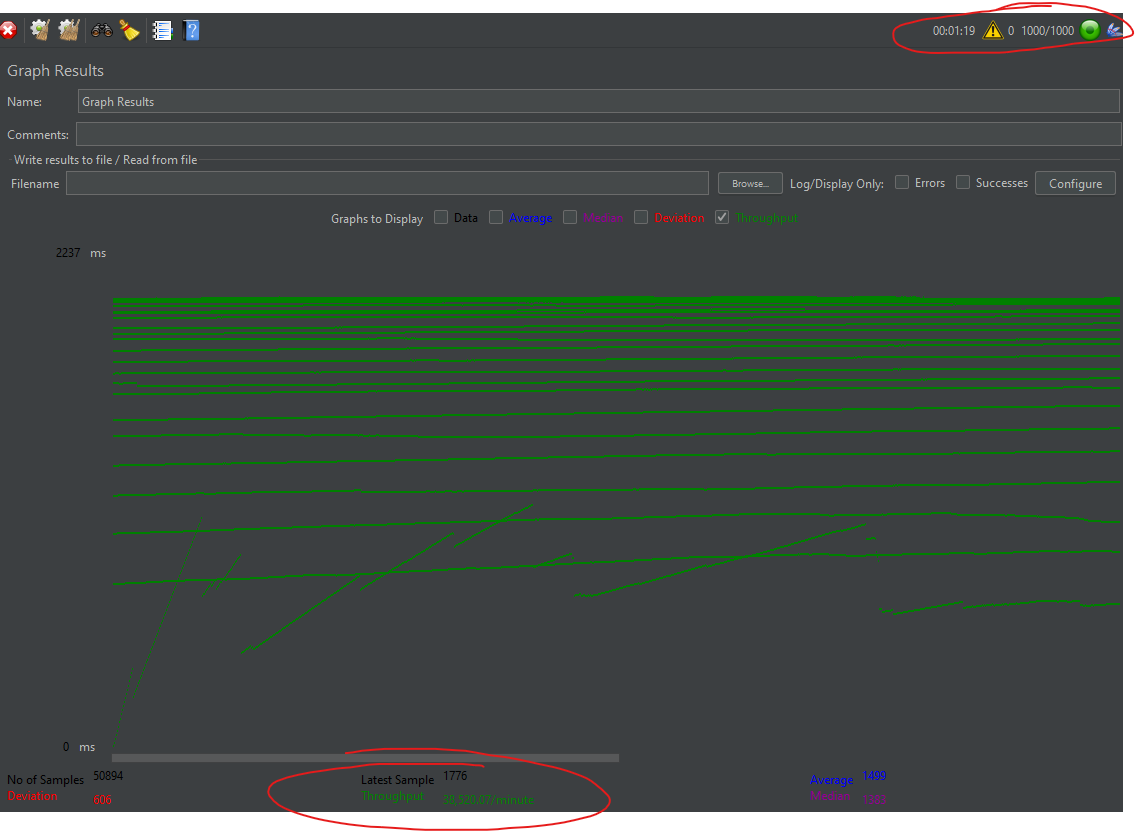


Latency:



JMeter latency and throughput metrics diagram with indexes for – 1000 threads:

Throughput:  




Latency:

