Отчет об оптимизации

Настоящий документ описывает результаты оптимизации производительности системы поиска профилей участников социальной сети.

Исходный код https://github.com/VladNF/otus-highload/tree/master/py3

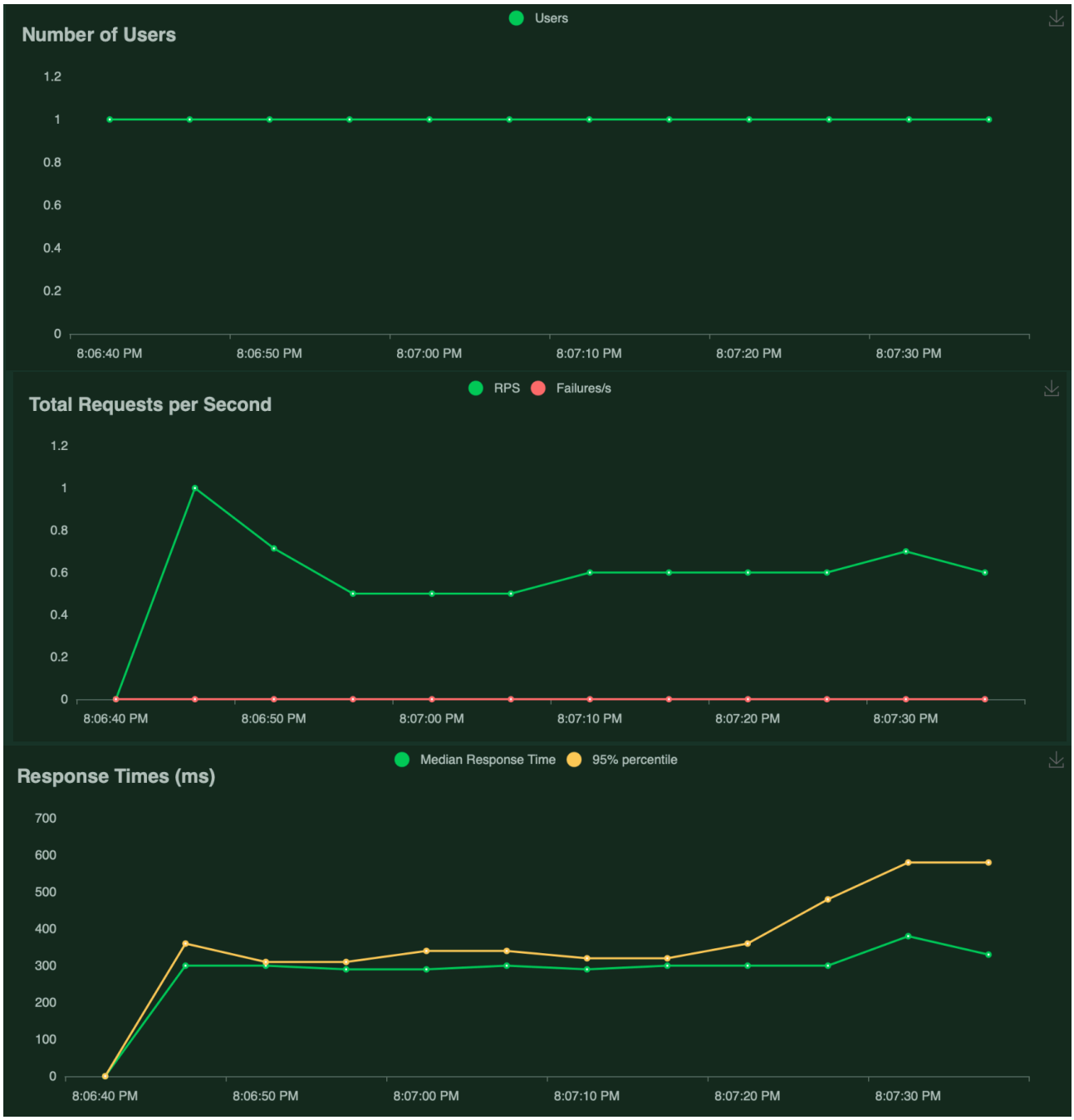
1 Состояние до оптимизации

Поиск профилей выполнялся следующим запросом

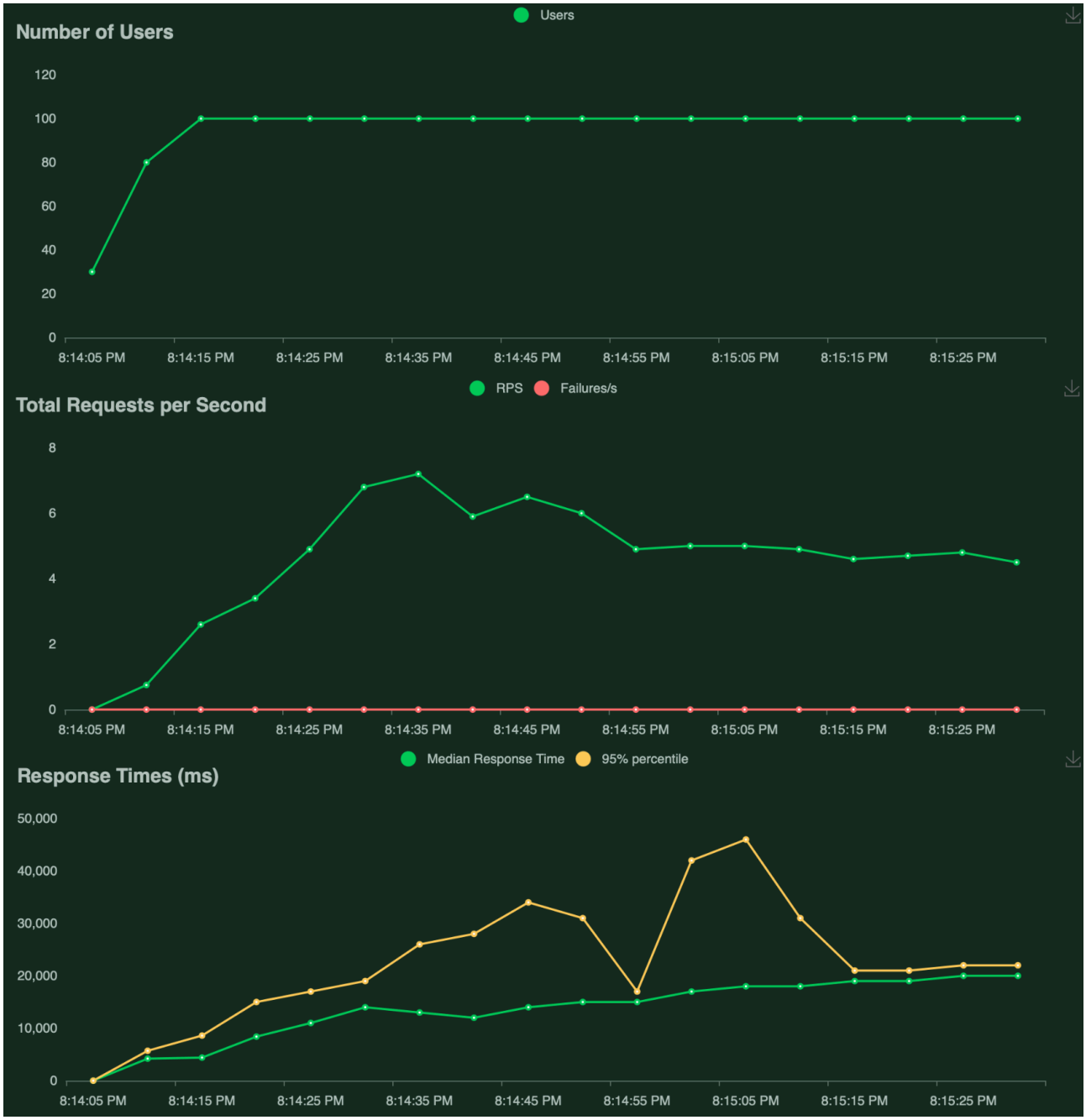
```
select * from users where first_name ilike 'влад%' and
last_name ilike 'п%' offset 0 limit 25;
```

Индексов на таблице не было, запросы выполнялись полным сканированием. Нагрузочное тестирование показало следующие результаты для 1, 100 и 100 пользователей соответственно.

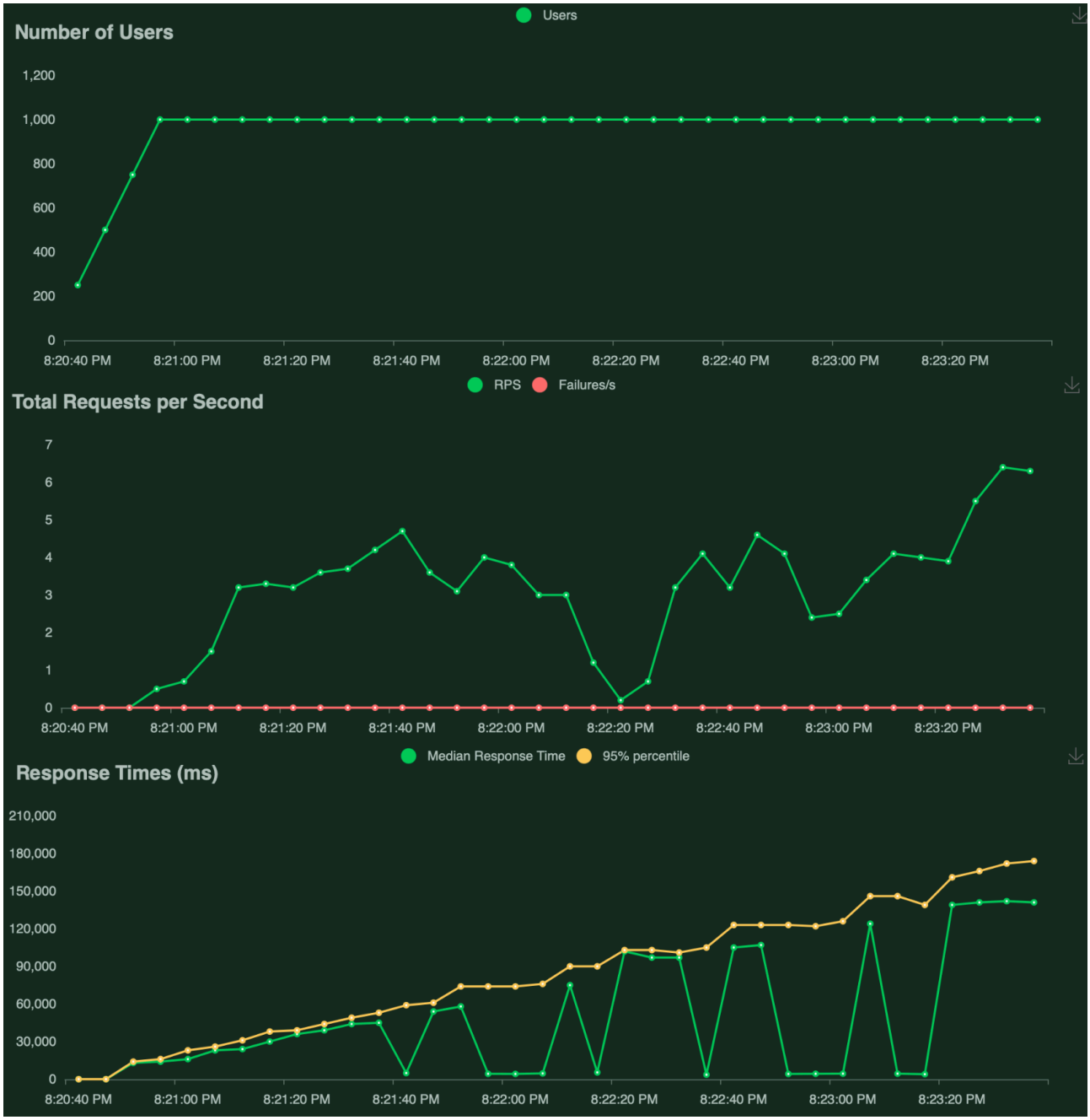
1 пользователь



100 пользователей



1 000 пользователей



2 Выполненная работа

В приложении увеличен пул соединений до 50, сделан таймаут на запрос к серверу БД 5 сек. Дополнительно был добавлен индекс и изменен запрос, для того, чтобы индекс использовался.

```
create index idx_first_last_name on users (lower(first_name) text_pattern_ops, lower(last_name) text_pattern_ops);
```

```
select * from users where lower(first_name) like 'влад%' and lower(last_name) like 'п%' offset 0 limit 25;
```

План исполнения запроса показывает, использование индекса

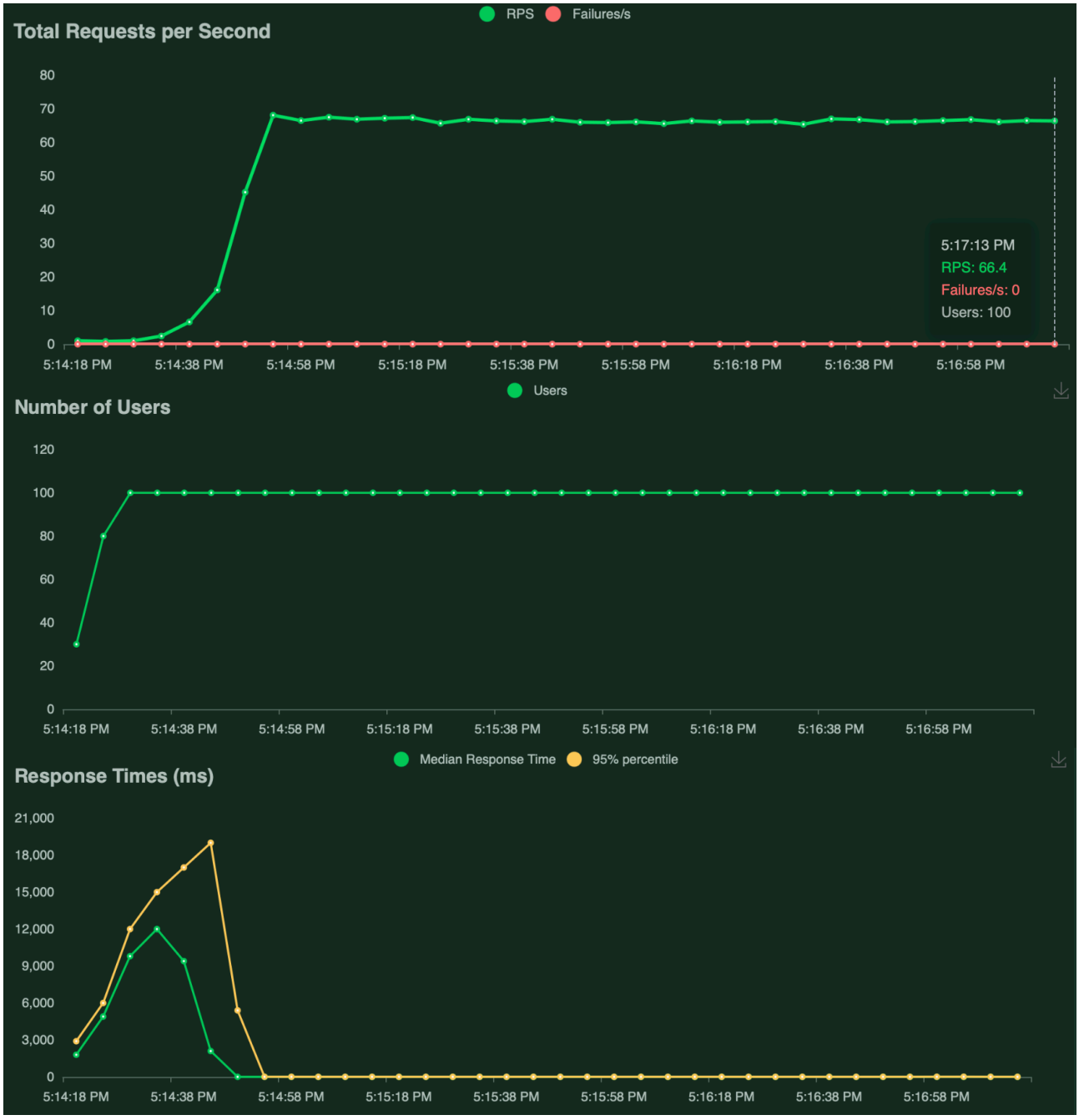
```
Limit  (cost=0.42..460.82 rows=25 width=217) (actual time=0.112..0.512 rows=25 loops=1)
  ->  Index Scan using idx_first_last_name on users  (cost=0.42..19134.63 rows=1039 width=217)
(actual time=0.111..0.507 rows=25 loops=1)
        Index Cond: ((lower(first_name) ~>=~ 'влад'::text) AND (lower(first_name) ~<~ 'влае'::text)
AND (lower(last_name) ~>=~ 'п'::text) AND (lower(last_name) ~<~ 'ѿ'::text))
        Filter: ((lower(first_name) ~~ 'влад%'::text) AND (lower(last_name) ~~ 'п%'::text))
Planning Time: 3.199 ms
Execution Time: 0.625 ms
```

Видно, что используются операции больше/меньше, что поддерживается B-tree индексами.

3 Состояние после оптимизации

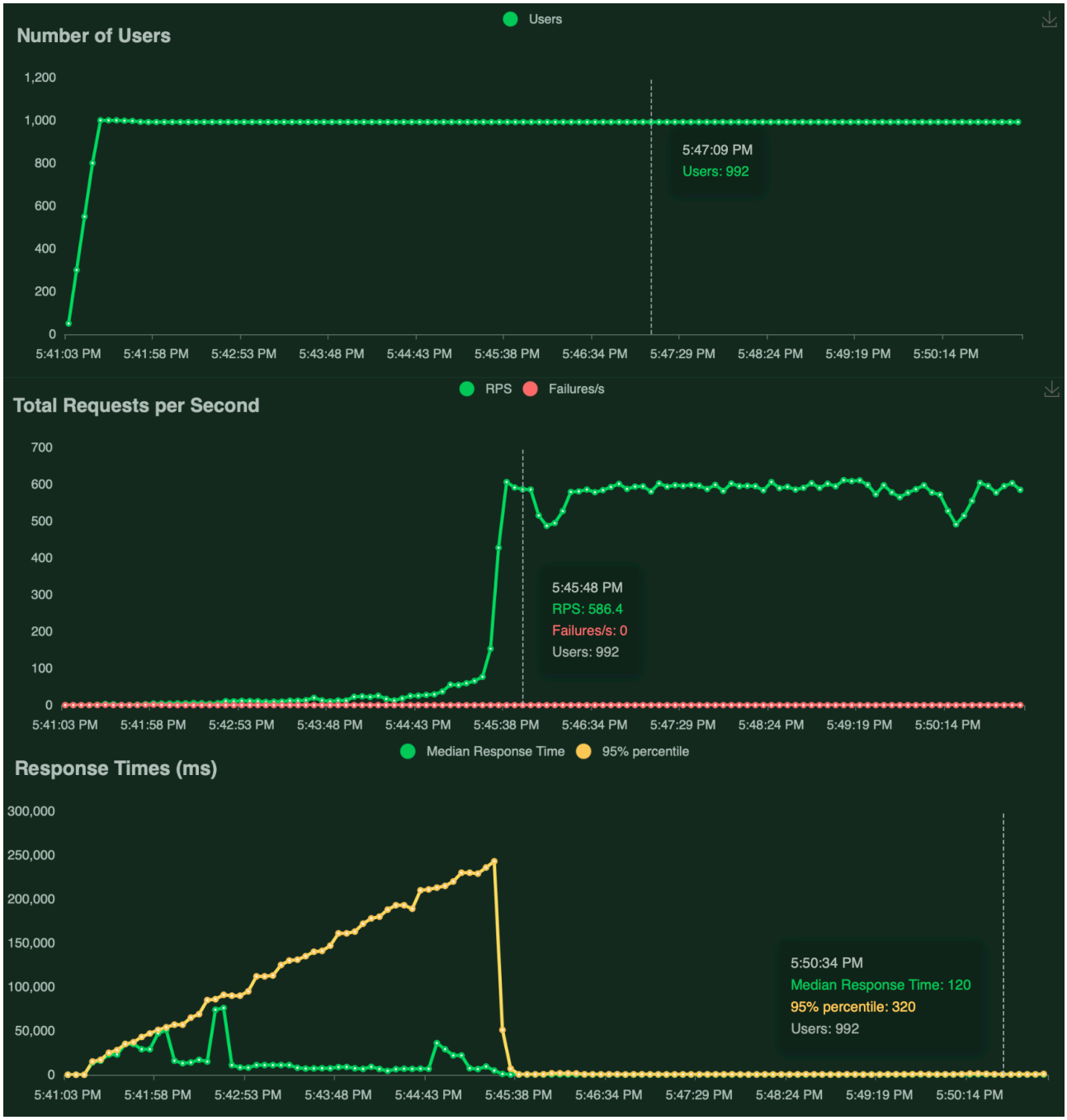
Нагрузочное тестирование показало следующие результаты для 100 и 1 000 пользователей соответственно.

<u>100 пользователей</u>



Сценарий нагрузки подразумевает, что виртуальный пользователь сначала делает операцию sign up, после чего выполняет поиск. Из-за этого начальные показатели можно рассматривать как “разогрев” системы.

1 000 пользователей



Итоги

По итогу оптимизации на конфигурации с 1 тыс пользователей было достигнуто улучшение показателя RPS в 100 раз, время отклика в примерно 500 раз.