Отчёт о выполнении нагрузочных тестов метода /user/search/

БД предварительно наполнена данными (999932 пользователей)

Выполнение запроса GET /user/search для поиска существующих анкет в БД с

параметрами firstName/secondName. Параметры динамически рандомно выбираются для каждого запроса.

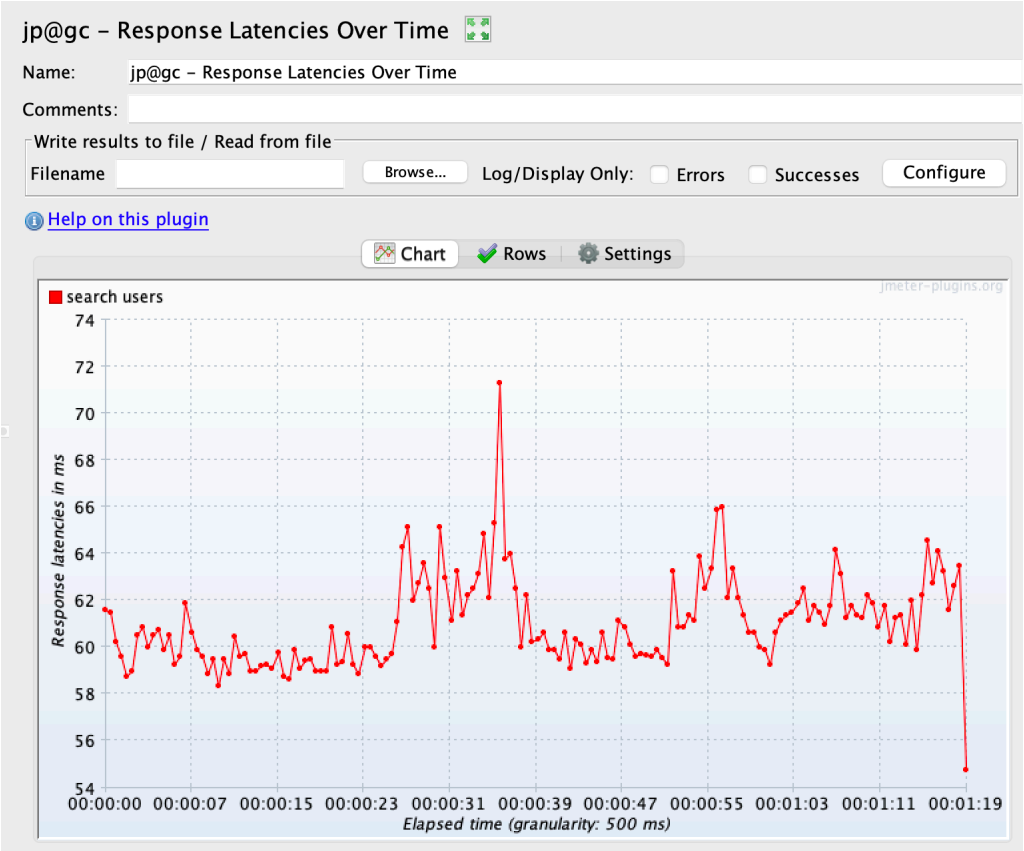
1. Нагрузка без индексов

---

1 user

График throughput

График latency

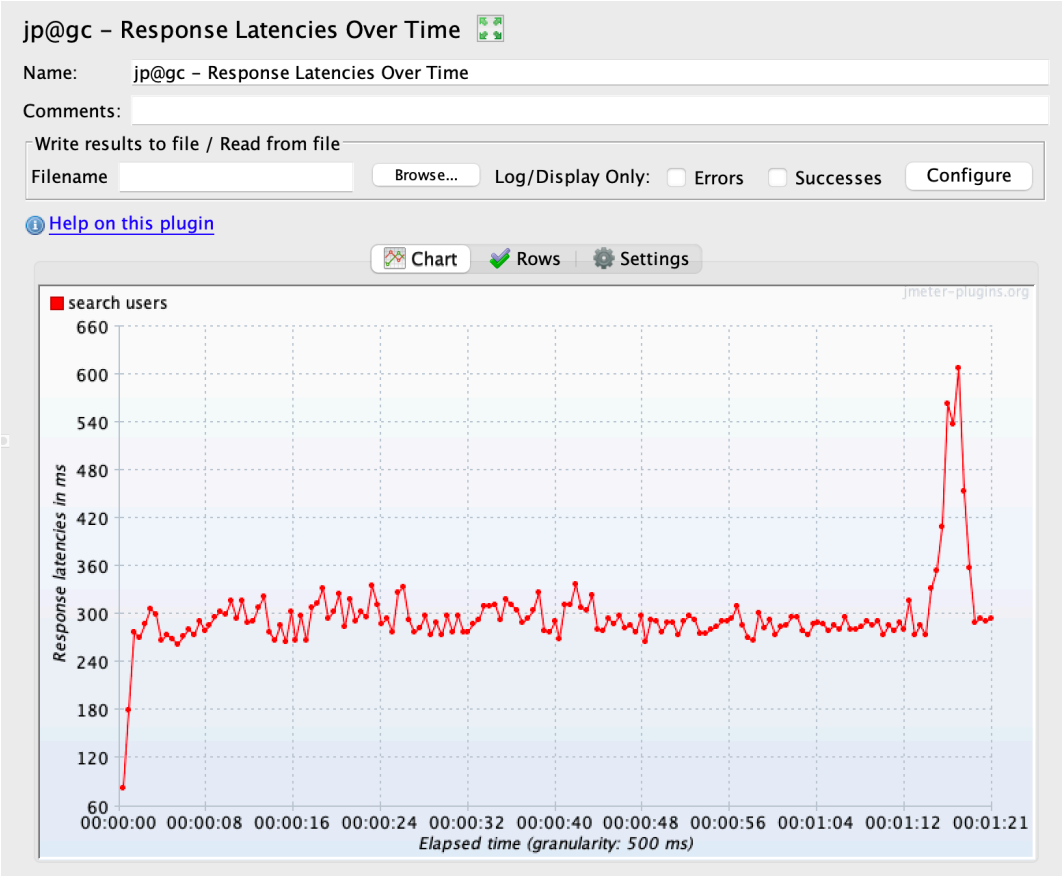


---

10 users

График throughput

График latency



100 users

График throughput

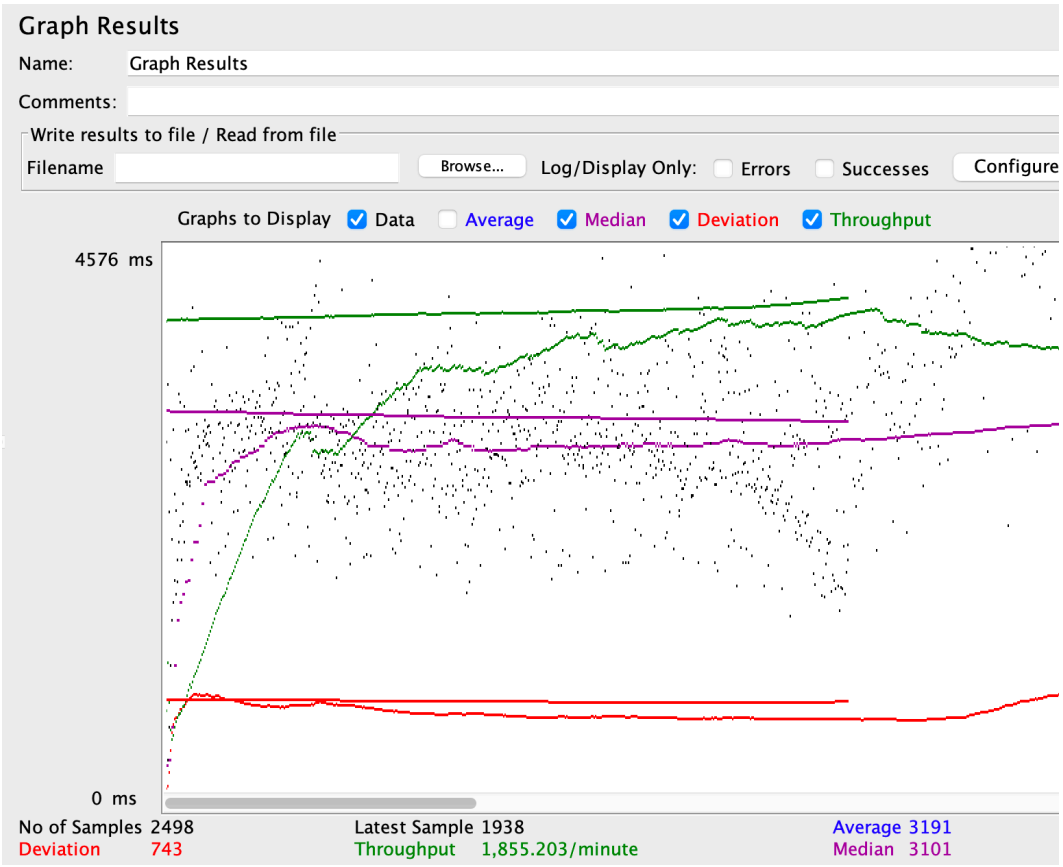
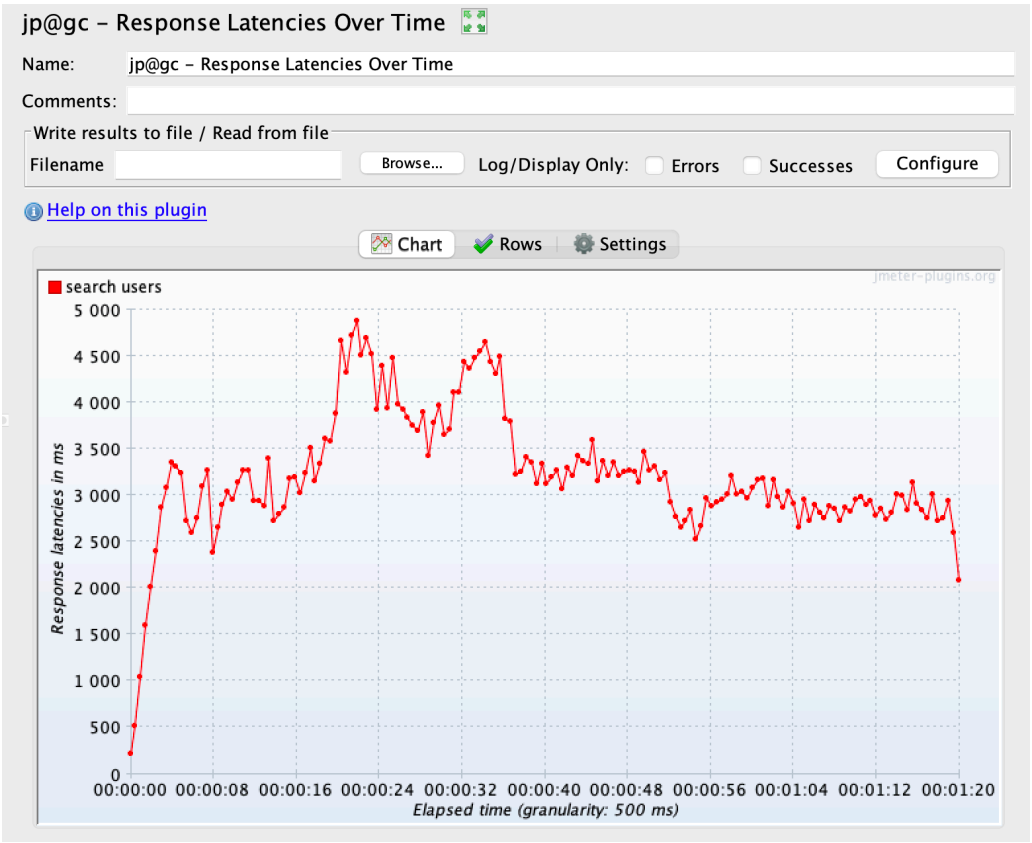


График latency



1000 users

График throughput

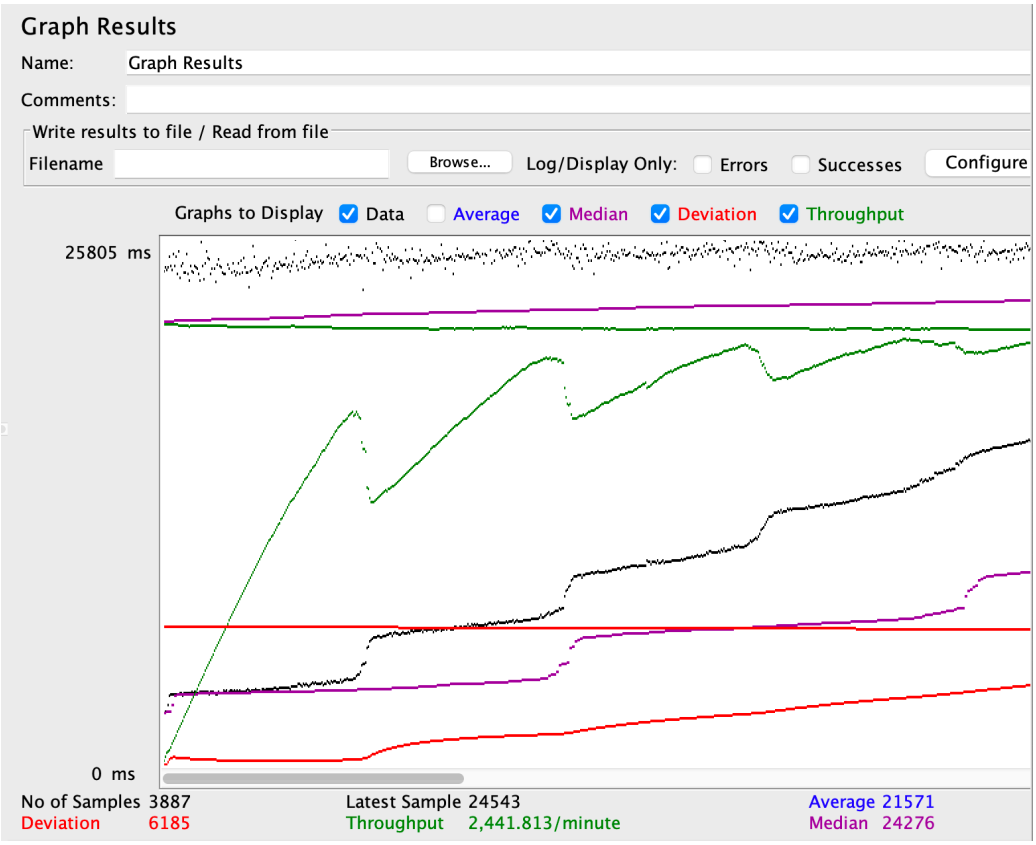
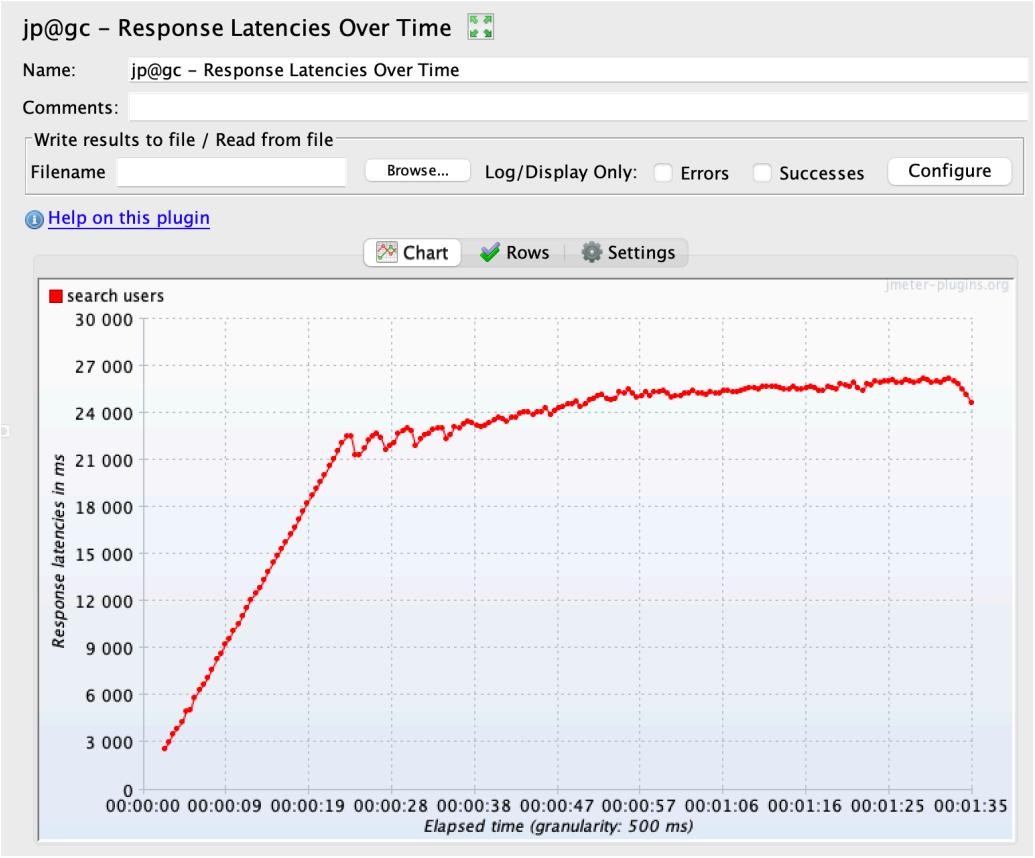


График latency



2. Нагрузка с индексами

---

1 user

График throughout

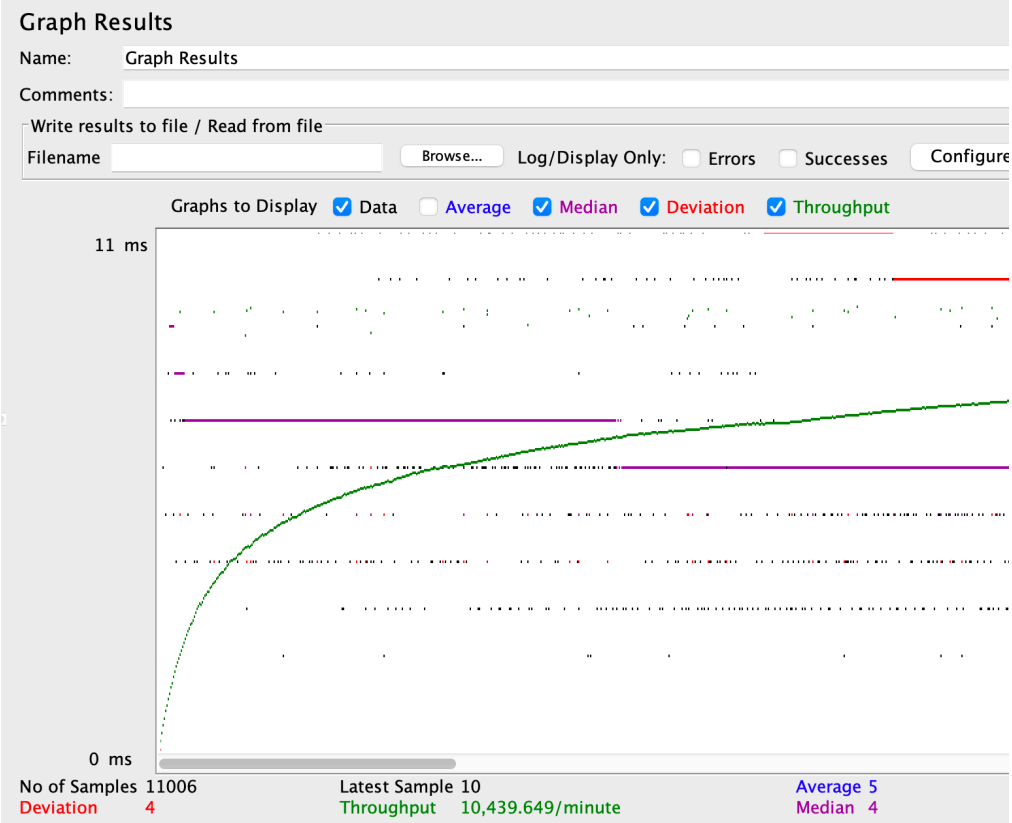
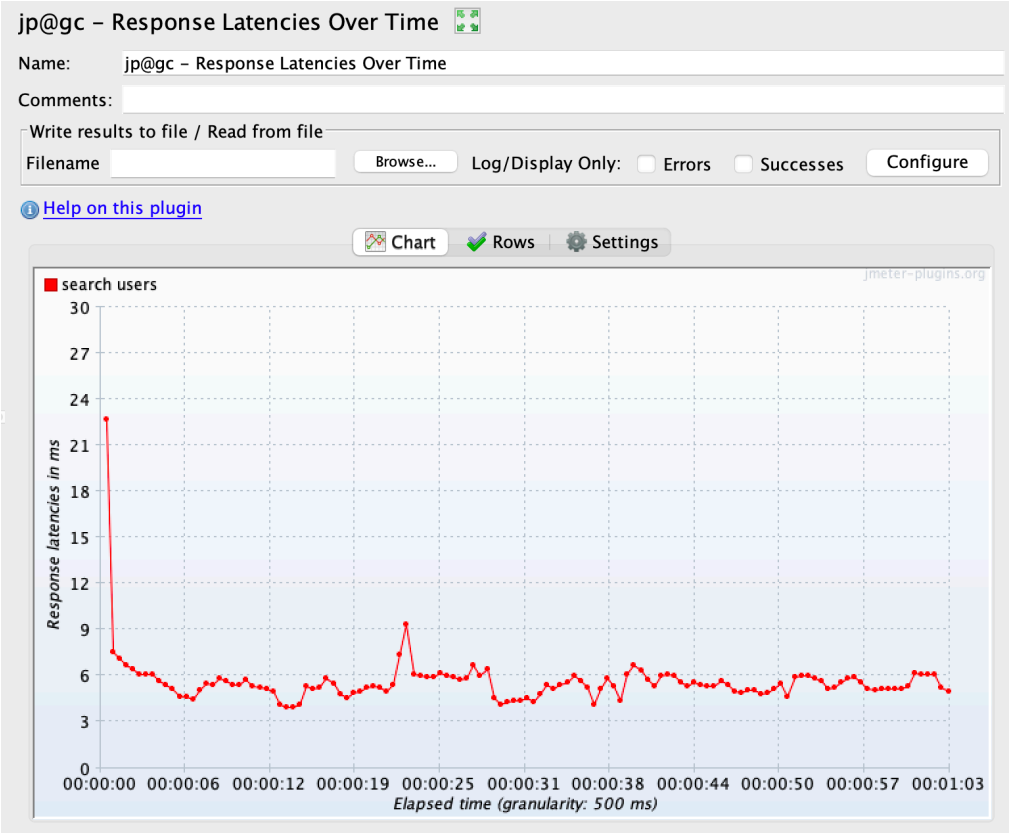


График latency



10 users

График throughput

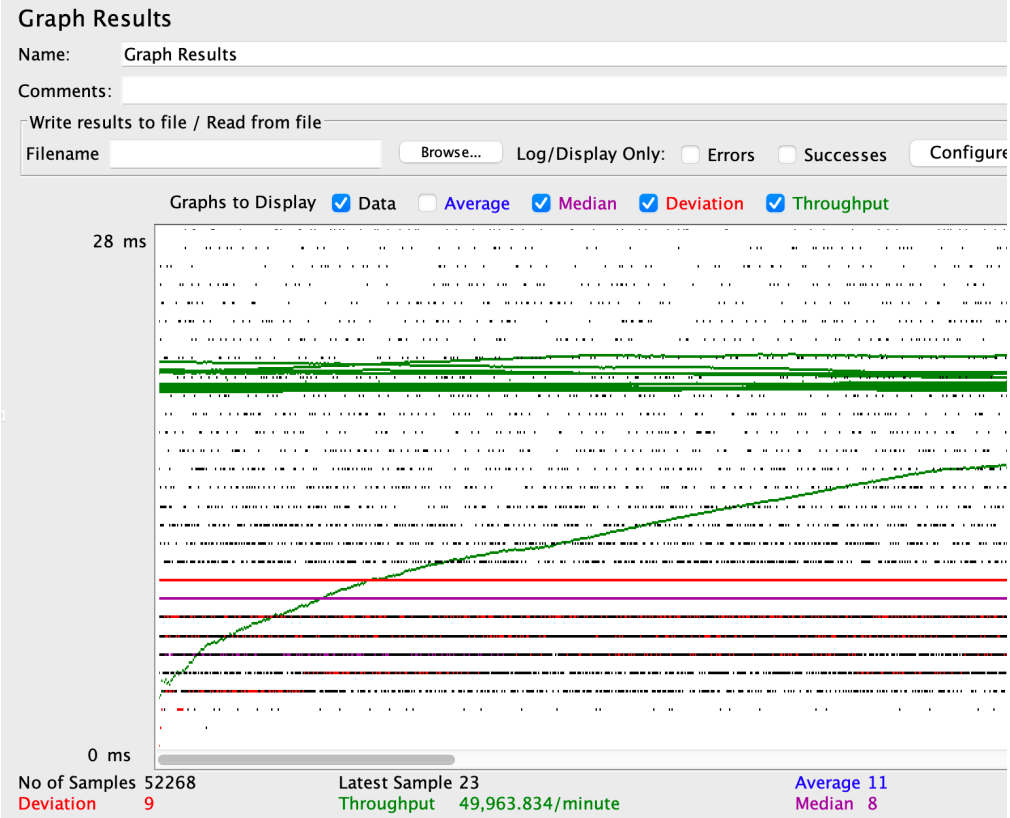
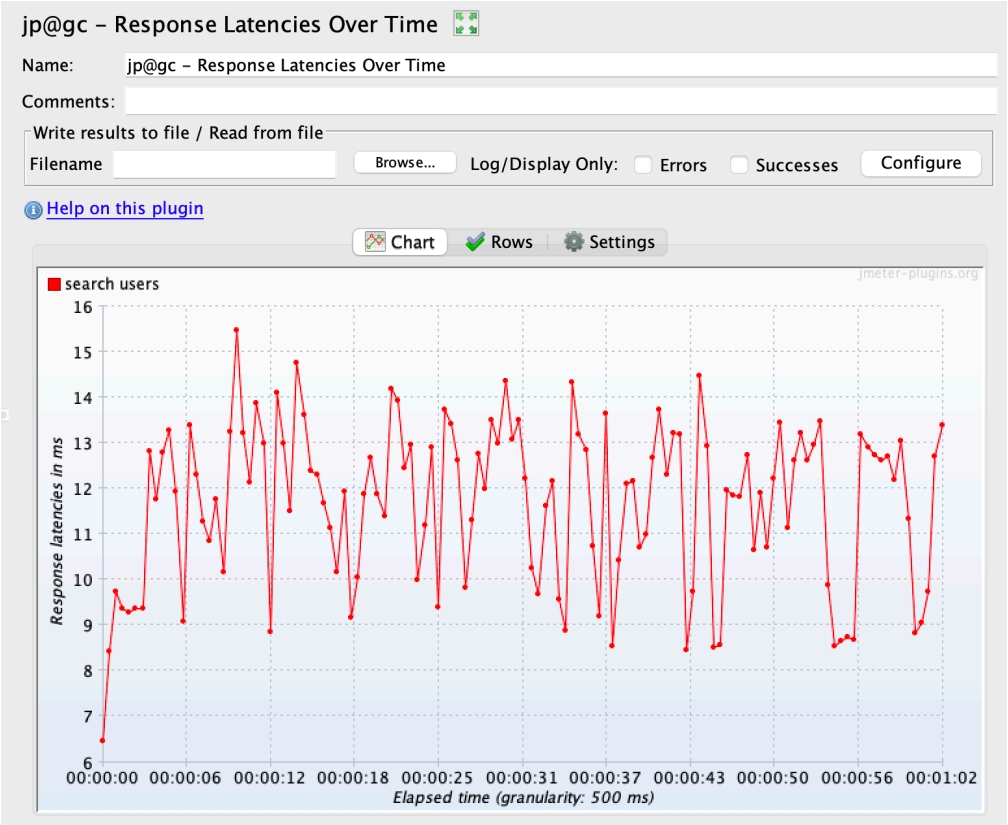


График latency



100 users

График throughput

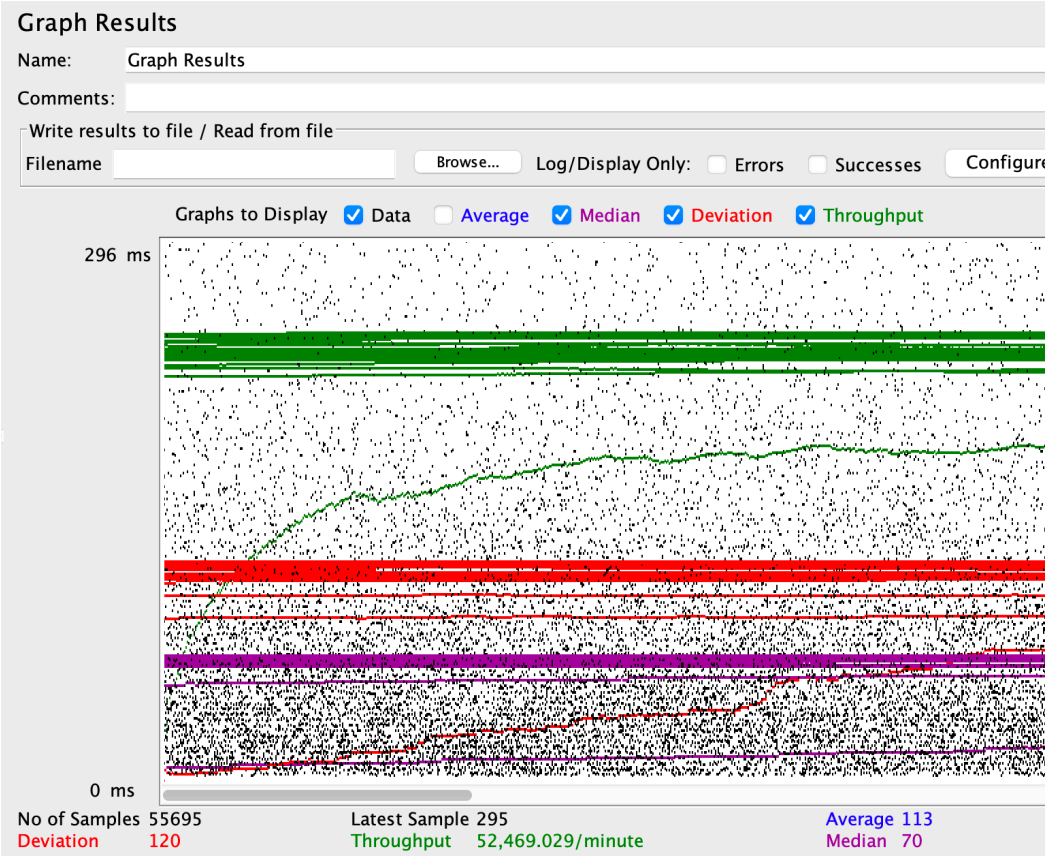
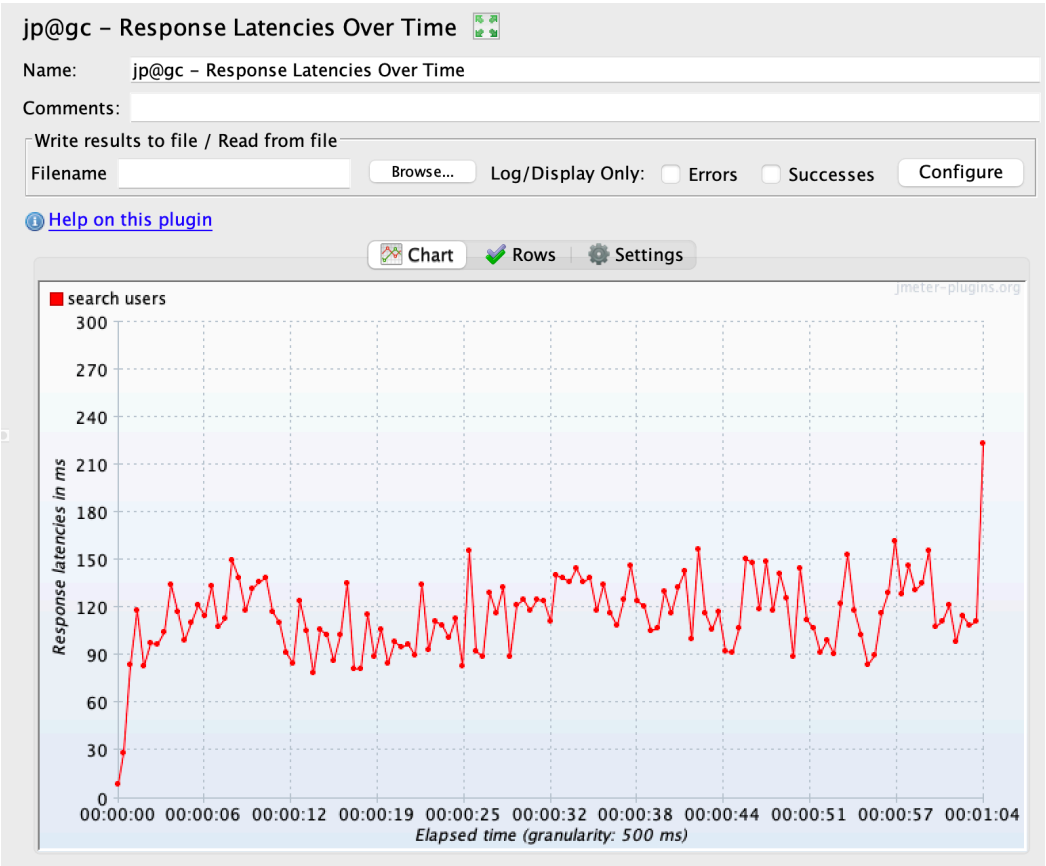


График latency



1000 users

График throughput

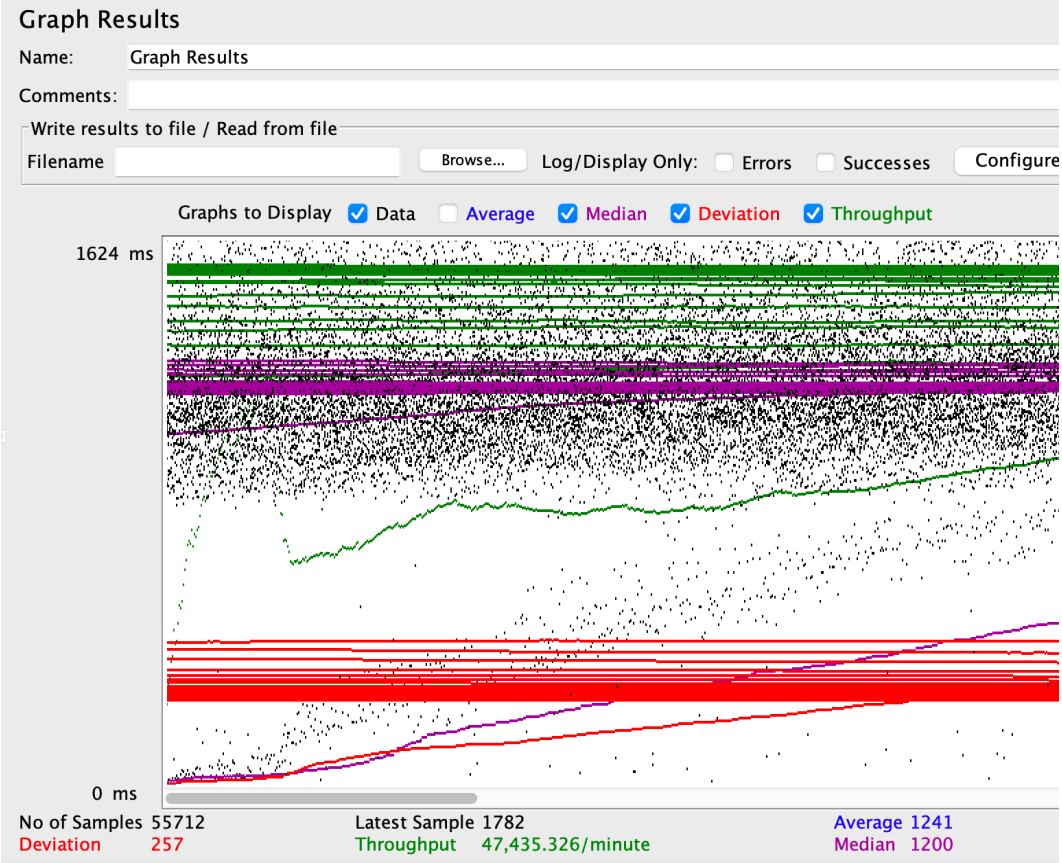
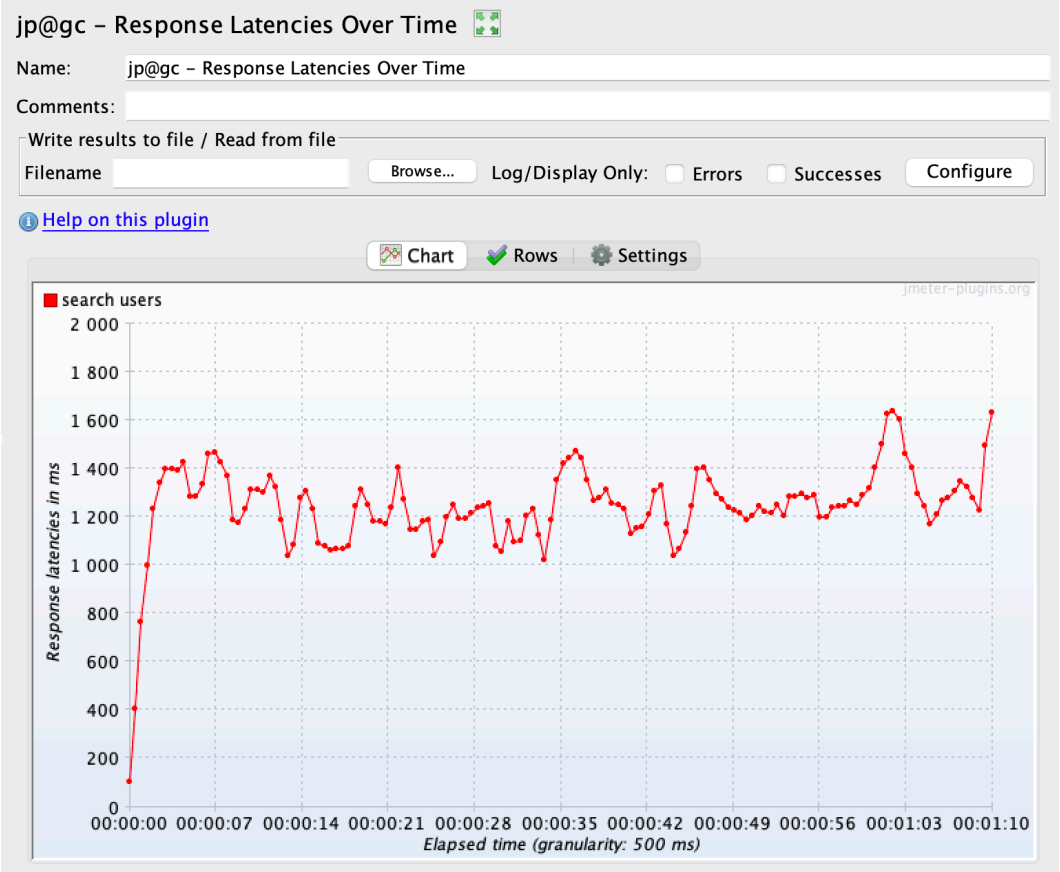


График latency



3. Индексы

Запросы добавления индексов

CREATE INDEX users_first_name_btree_idx on users (first_name text_pattern_ops);

CREATE INDEX users_second_name_btree_idx on users (second_name text_pattern_ops);

EXPLAIN PLAN до добавления индексов

```
QUERY PLAN
Gather Merge  (cost=28182.01..28257.15 rows=644 width=133) (actual time=59.846..67.287 rows=788 loops=1)
  Workers Planned: 2
  Workers Launched: 2
  ->  Sort  (cost=27181.99..27182.79 rows=322 width=133) (actual time=55.659..55.677 rows=263 loops=3)
        Sort Key: id
        Sort Method: quicksort  Memory: 78kB
        Worker 0:  Sort Method: quicksort  Memory: 66kB
        Worker 1:  Sort Method: quicksort  Memory: 60kB
        ->  Parallel Seq Scan on users  (cost=0.00..27168.58 rows=322 width=133) (actual time=29.373..55.256 rows=26
              Filter: (((first_name)::text ~~ 'Алекс%'::text) AND ((second_name)::text ~~ 'Фе%'::text))
              Rows Removed by Filter: 333048
Planning Time: 0.285 ms
Execution Time: 67.346 ms
```

EXPLAIN PLAN после добавления индексов

```
QUERY PLAN
Sort  (cost=3565.40..3567.33 rows=773 width=133) (actual time=6.044..6.095 rows=788 loops=1)
  Sort Key: id
  Sort Method: quicksort  Memory: 155kB
  ->  Bitmap Heap Scan on users  (cost=933.42..3528.32 rows=773 width=133) (actual time=4.080..4.976 rows=788 loops=
        Filter: (((first_name)::text ~~ 'Алекс%'::text) AND ((second_name)::text ~~ 'Фе%'::text))
        Heap Blocks: exact=272
        ->  BitmapAnd  (cost=933.42..933.42 rows=766 width=0) (actual time=4.037..4.038 rows=0 loops=1)
              ->  Bitmap Index Scan on users_second_name_btree_idx  (cost=0.00..192.06 rows=13963 width=0) (actual t
                    Index Cond: (((second_name)::text ~>=~ 'Фе'::text) AND ((second_name)::text ~<~ 'Фж'::text))
              ->  Bitmap Index Scan on users_first_name_btree_idx  (cost=0.00..740.73 rows=54830 width=0) (actual ti
                    Index Cond: (((first_name)::text ~>=~ 'Алекс'::text) AND ((first_name)::text ~<~ 'Алект'::text))
Planning Time: 0.446 ms
Execution Time: 6.153 ms
```

Объяснение:

Перед добавлением индексов на плане выполнения запросов видно, что выполняется Чтение всех строк в таблице (Seq scan), время выполнения запроса - 67.346 ms.

После добавления индексов на плане выполнения запросов видим, что выполняется поиск по индексу btree (bitmap index scan), время выполнения запроса - 6.153 ms, в 10 раз быстрее, чем без использования индекса.

Были добавлены 2 btree индекса для полей first_name, second_name с параметром *text_pattern_ops, так как в запросе мы используем поиск по паттерну LIKE '…%'.*