# ДЗ «Производительность индексов»

1. Сгенерировано 1000000 анкет, данные взяты из подготовленного списка и дополнительно дополнены данными по полу (в зависимости от окончания фамилии) и дате рождения (текущая дата – кол-во полных лет – random(0,364)).
2. Реализован метод поиска анкет по префиксу имени и фамилии, поисковый запрос

"select userid, name, surname, sex, birthdate, biography, city from usersdata where name LIKE ‘<префикс имени>%’ and surname LIKE ‘<префикс фамилии>%’ order by userid;"

1. При помощи jmeter проведено нагрузочное тестирование метода поиска анкет и построены графики (в качестве параметров поиска указаны:

Префикс имени “Ива

Префикс фамилии “Ан”

)

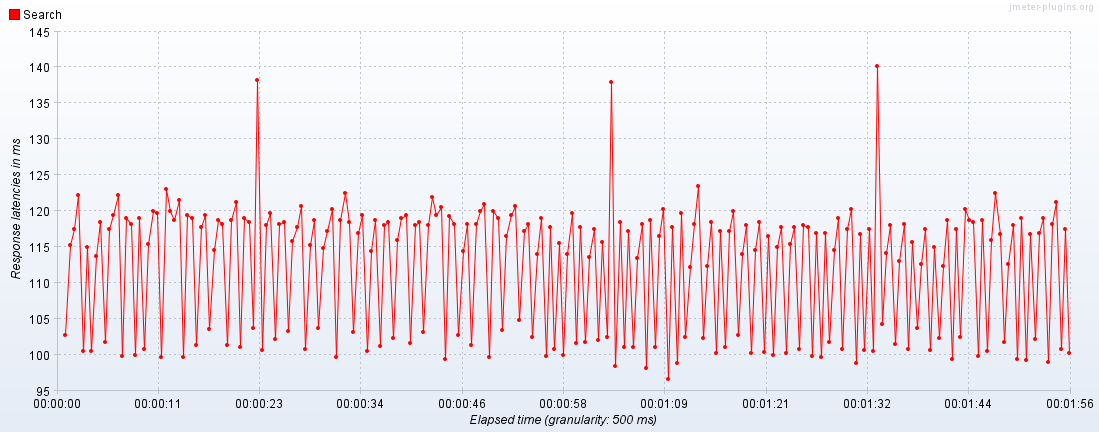
Графики строились:  
 Latency – дополнительный плагин «jp@gc - Response Latencies Over Time»

Throughput – дополнительный плагин «jp@gc - Bytes Throughput Over Time»

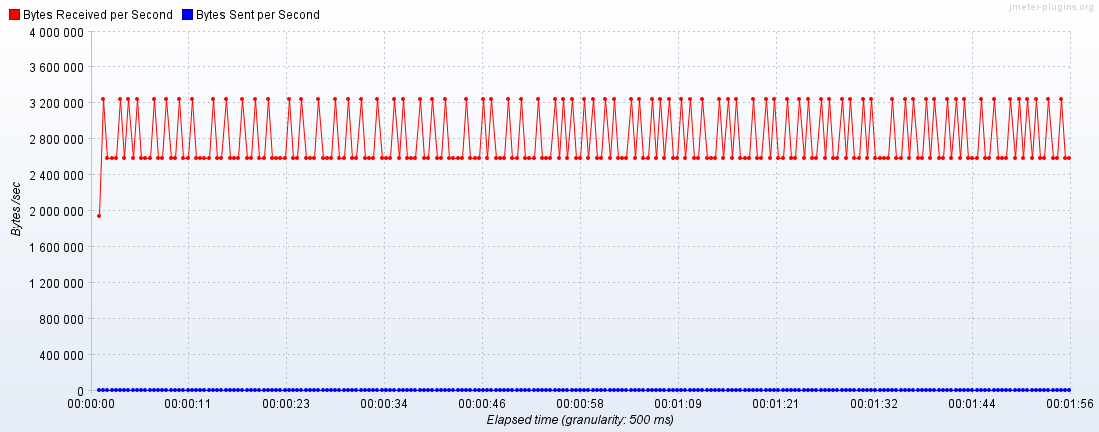
Нагрузка:

1 rps – 1000 повторов:

Latency

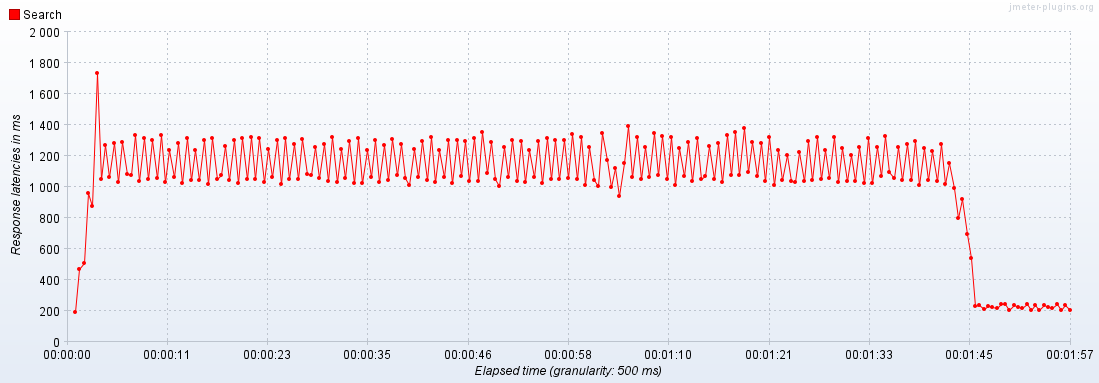


Throughput bytes

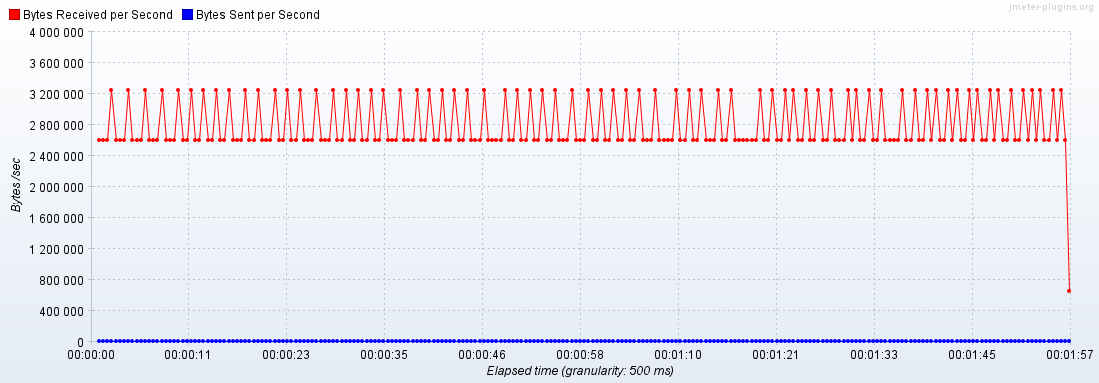


10 rps – 100 повторов:

Latency

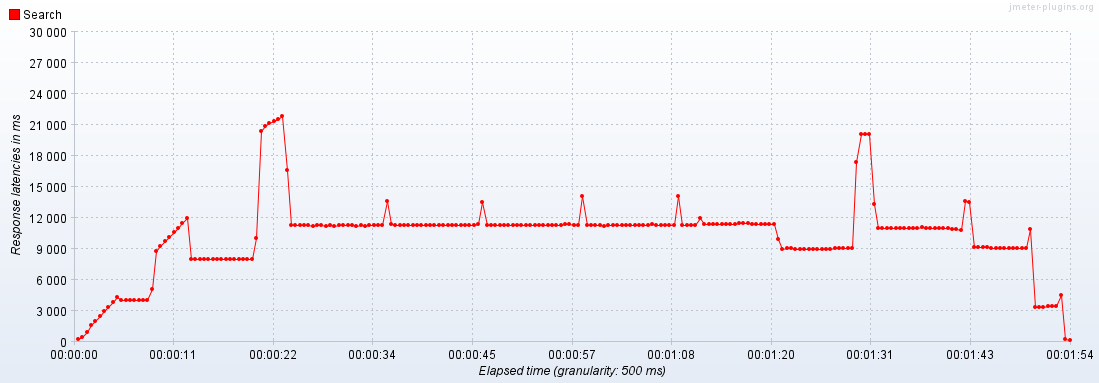


Throughput bytes

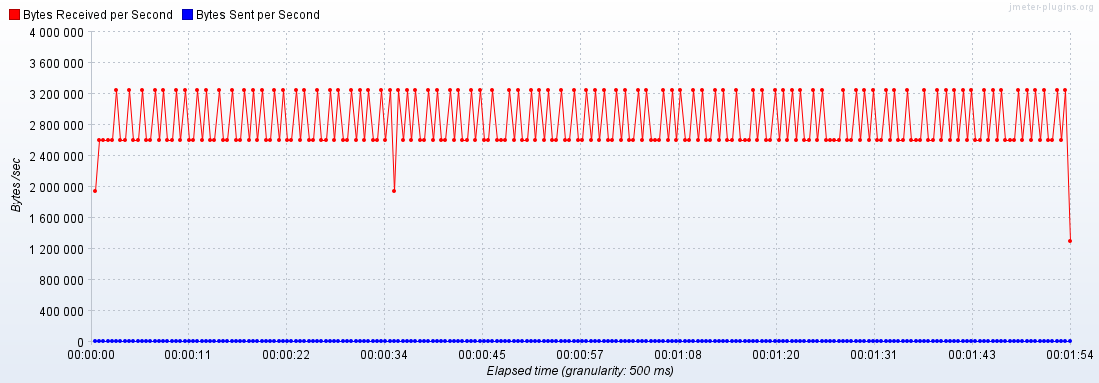


100 rps – 10 повторов:

Latency

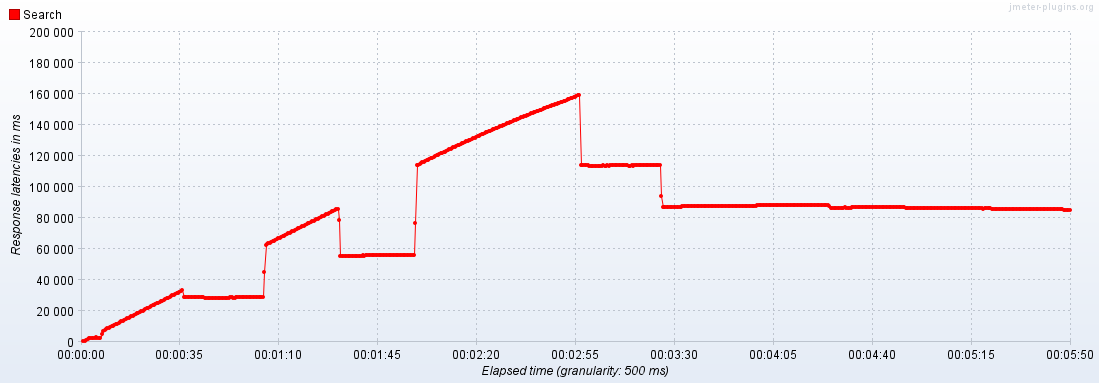


Throughput bytes

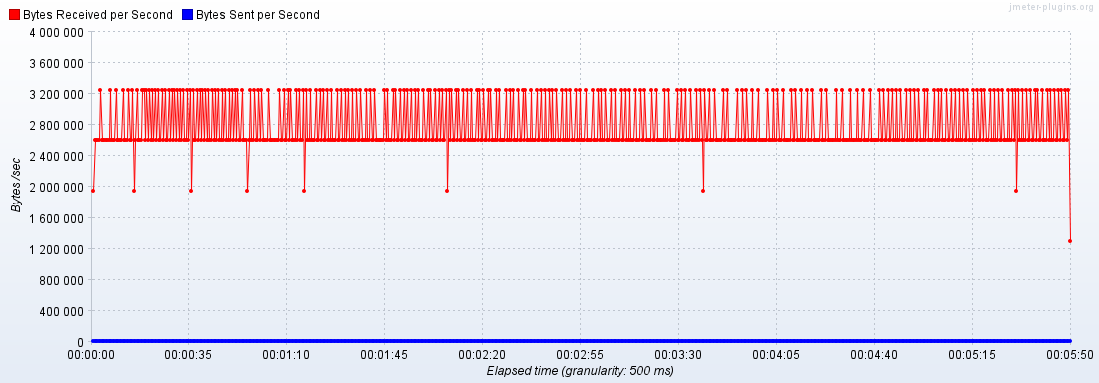


1000 rps – 3 повтора:

Latency



Throughput bytes



1. Для оптимизации поиска создан индексы на поля name и surname с указанием колонок в том же порядке, что и в запросе поиска анкет.

Поле user\_id (id анкеты), по которому выполняется сортировка результата, в индекс не добавлялось, т.к. это primary key

Индекс добавлен командой (через UI pgAdmin):

CREATE INDEX IF NOT EXISTS search\_user

ON public.usersdata USING btree

(name COLLATE pg\_catalog."default" ASC NULLS LAST, surname COLLATE pg\_catalog."default" ASC NULLS LAST)

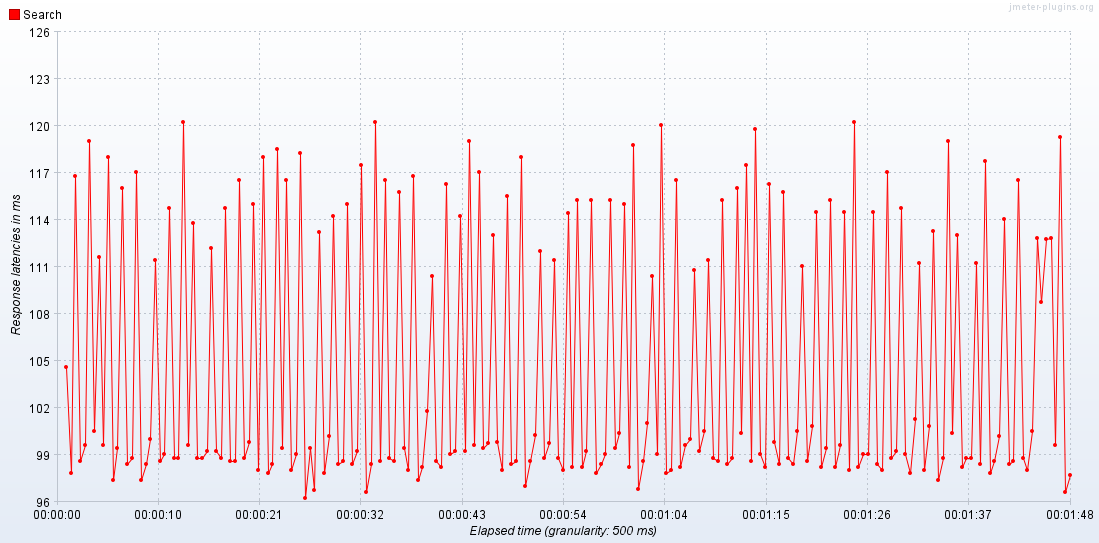
TABLESPACE pg\_default;

1. При помощи jmeter проведено повторное тестирование метода поиска анкет и построены графики с теми же параметрами, построены графики:

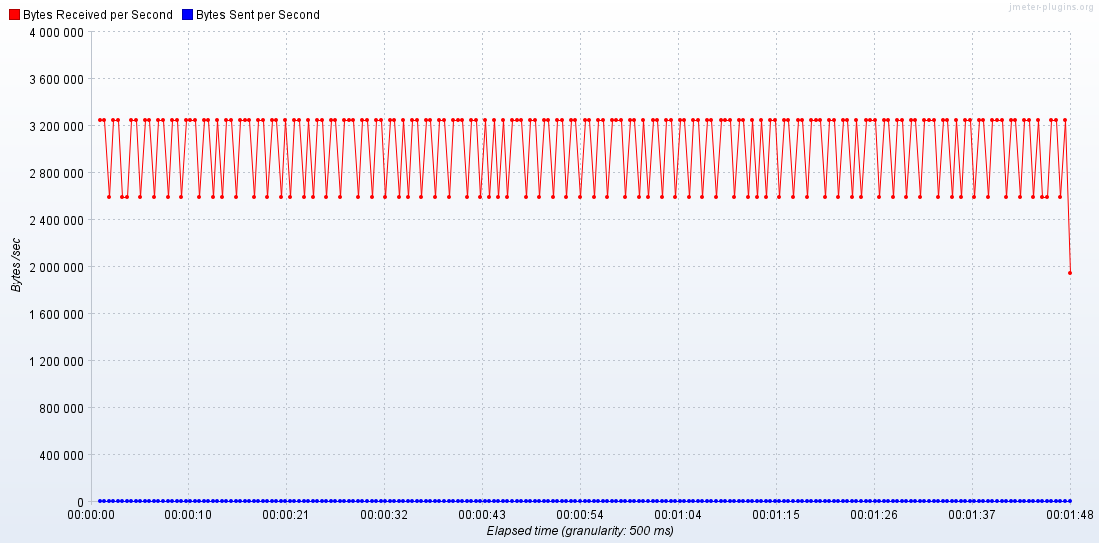
Нагрузка:

1 rps – 1000 повторов:

Latency

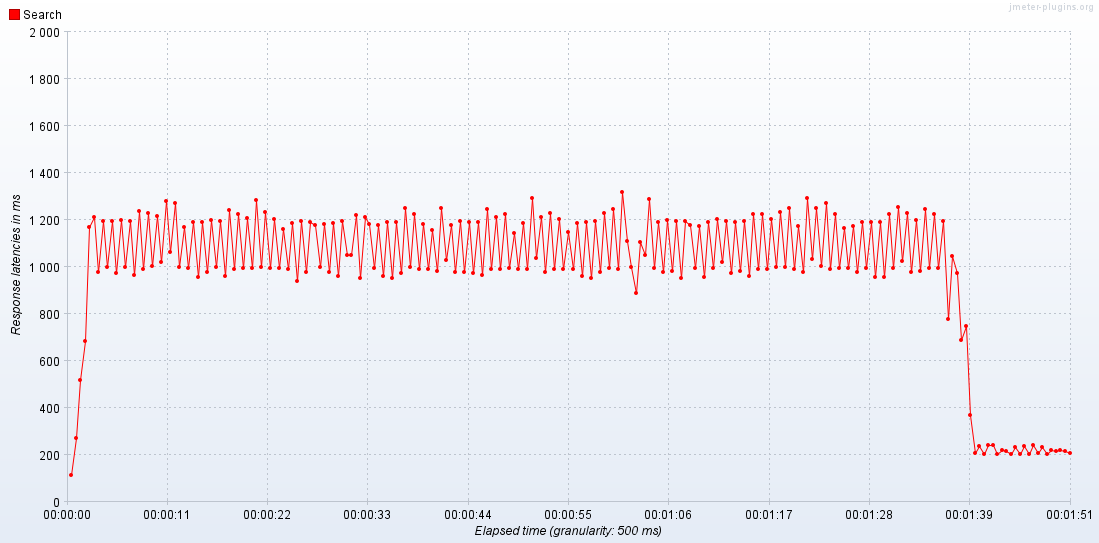


Throughput bytes

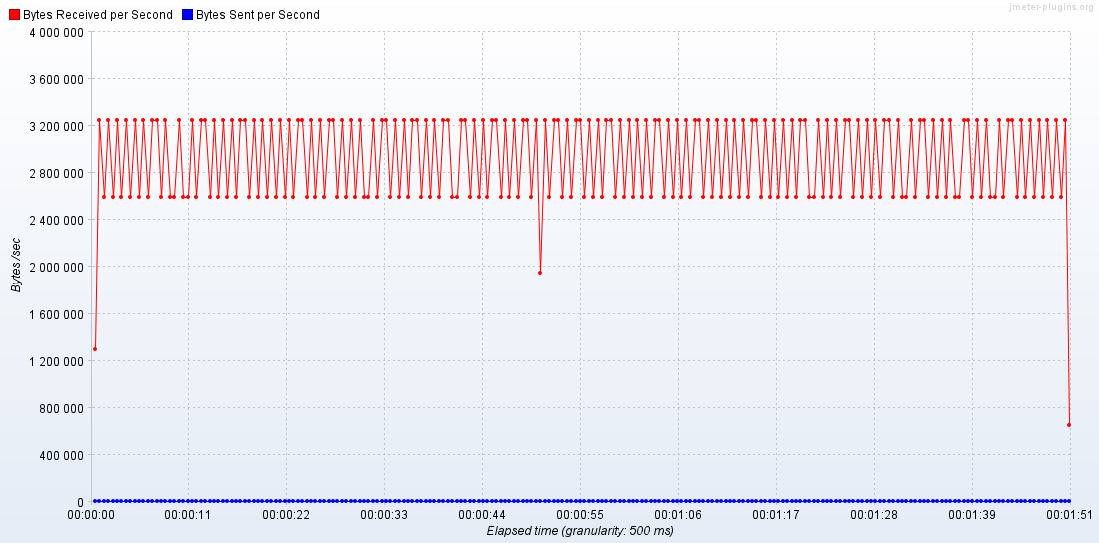


10 rps – 100 повторов:

Latency

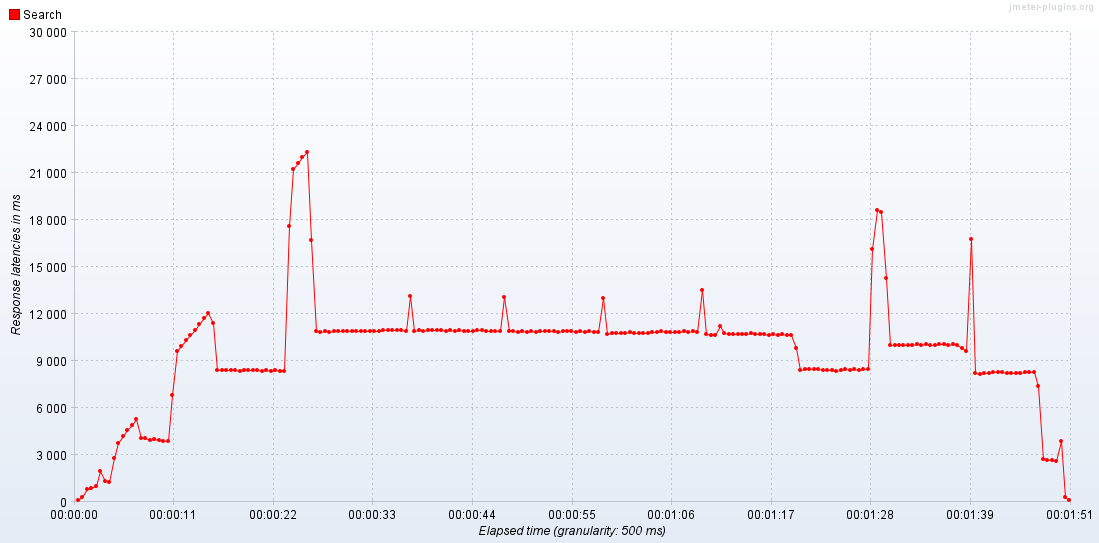


Throughput bytes

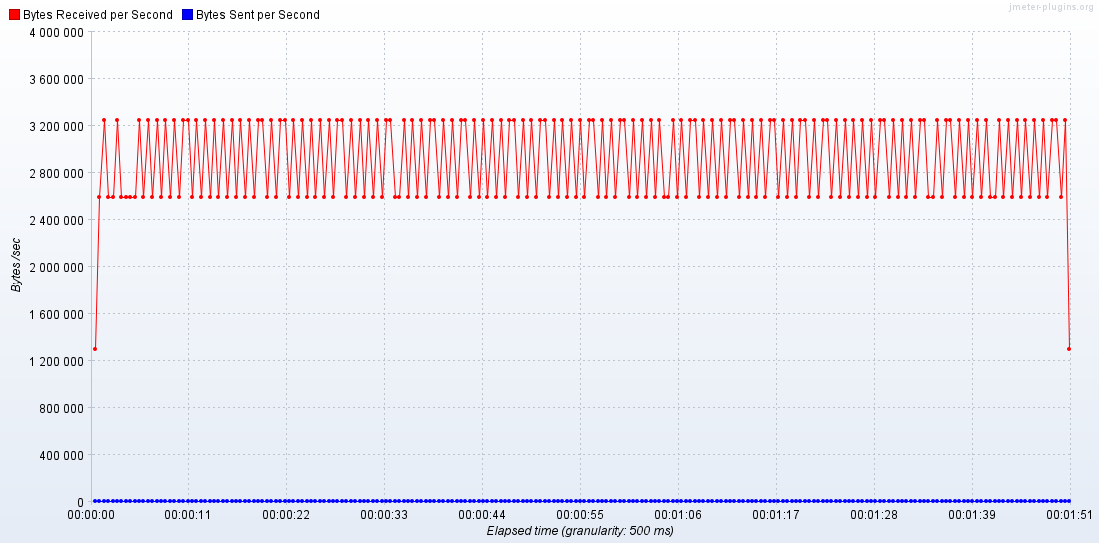


100 rps – 10 повторов:

Latency

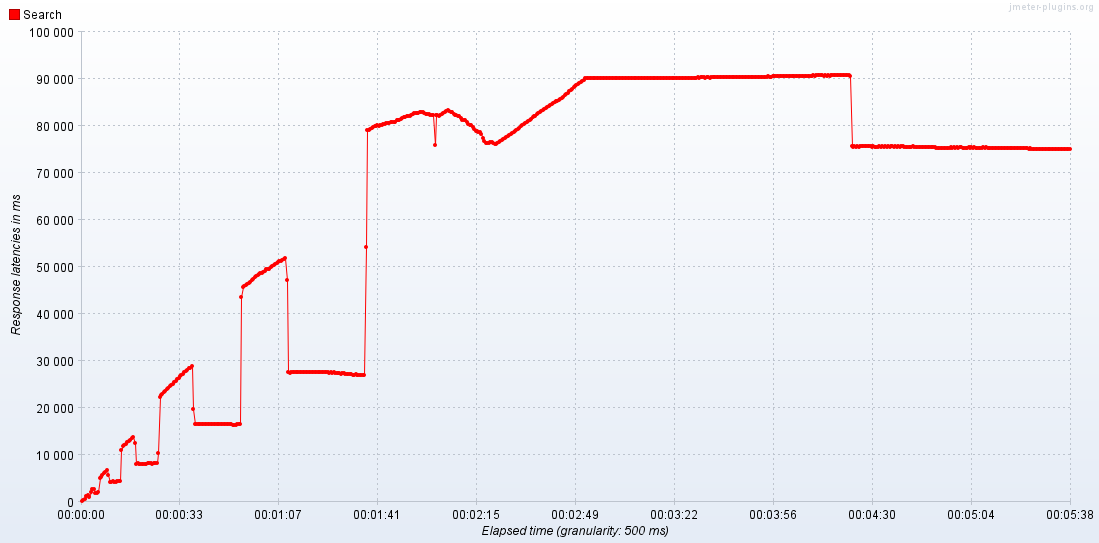


Throughput bytes

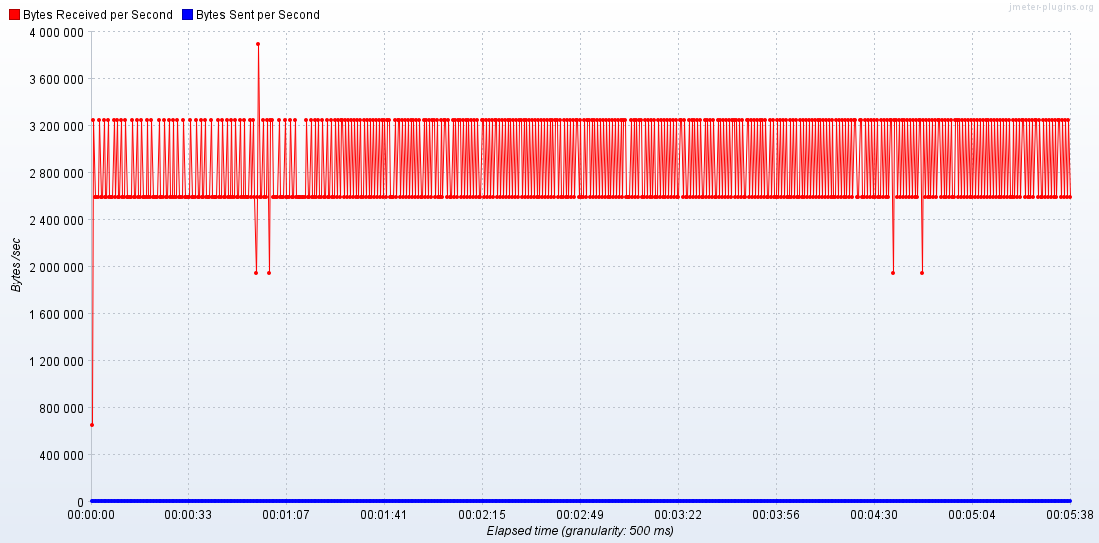


1000 rps – 3 повтора:

Latency



Throughput bytes



1. Explain запроса поиска анкет:

Для индекса выбраны поля имя (name) и фамилия (surname) по которым происходит поиск, индекс выбран не уникальным т.к. пара Фамилия и Имя могут совпадать, порядок полей в индексе соответствует порядку в запросе, чтобы индекс использовался корректно.

Explane подтверждает, что после добавления индекса поиск выполняется по нем.

Выполнен explain для запроса для понимания что индекс помог ускорить обращения к БД.

**До индекса:**

SocialOtus=# explain select userid, name, surname, sex, birthdate, biography, city from usersdata where name LIKE 'Ива' and surname LIKE 'Ан' order by userid;

QUERY PLAN

-----------------------------------------------------------------------------------

Sort (cost=21411.11..21411.11 rows=1 width=81)

Sort Key: userid

-> Gather (cost=1000.00..21411.10 rows=1 width=81)

Workers Planned: 2

-> Parallel Seq Scan on usersdata (cost=0.00..20411.00 rows=1 width=81)

Filter: ((name ~~ 'Ива'::text) AND (surname ~~ 'Ан'::text))

**После добавления индекса:**

SocialOtus=# explain select userid, name, surname, sex, birthdate, biography, city from usersdata where name LIKE 'Ива' and surname LIKE 'Ан' order by userid;

QUERY PLAN

------------------------------------------------------------------------------------

Sort (cost=8.46..8.47 rows=1 width=81)

Sort Key: userid

-> Index Scan using search\_user on usersdata (cost=0.42..8.45 rows=1 width=81)

Index Cond: ((name = 'Ива'::text) AND (surname = 'Ан'::text))

Filter: ((name ~~ 'Ива'::text) AND (surname ~~ 'Ан'::text))

После добавления индекса стоимость запроса уменьшилась на несколько порядков.

Выводы:

После добавления индекса все тесты с теми же параметрами завершились быстрее, т.к. индекс ускорил отклик сервиса.

Малое изменение latency после добавления индекса может говорить о том, что скорость работы запроса СУБД не самое критичное место в работе сервиса. Вся инфраструктура докер запускалась на конфигурации core i7 16Gb Ram и в процессе работы тестов потребления RAM доходило до 20Gb, т.е. использовался файл подкачки, который значительно влияет на скорость работы.

Самое заметное изменение по latency для теста 1000 запросов в секунду и 3 повторения, пиковые значение уменьшились почти в 2 раза со 160 тыс до 90 тыс.