Индексы: оптимизация запросов

Запрос

```
EXPLAIN SELECT u.id, u.name, surname, age, sex, info, c.name
FROM public.users u
LEFT OUTER JOIN public.cities c on c.ID = u.city_id
WHERE u.name LIKE 'Юрий' and surname LIKE 'Воробьев'
ORDER BY id
```

1. EXPLAIN до индексов

```
"Gather Merge  (cost=37343.13..37343.83 rows=6 width=93)"
"  Workers Planned: 2"
"  ->  Sort  (cost=36343.11..36343.12 rows=3 width=93)"
"        Sort Key: u.id"
"        ->  Nested Loop Left Join  (cost=0.15..36343.08 rows=3 width=93)"
"              ->  Parallel Seq Scan on users u  (cost=0.00..36334.00 rows=3 width=79)"
"                    Filter: (((name)::text = 'Юрий'::text) AND ((surname)::text = 'Воробьев'::text))"
"              ->  Index Scan using cities_pkey on cities c  (cost=0.15..3.02 rows=1 width=22)"
"                    Index Cond: (id = u.city_id)"
```

2. Btree Индексы

Раздельные индексы

```
CREATE INDEX IF NOT EXISTS idx_name
    ON public.users USING btree
    (name COLLATE pg_catalog."default" ASC NULLS LAST)
    WITH (deduplicate_items=True)
    TABLESPACE pg_default;

CREATE INDEX IF NOT EXISTS idx_surname
    ON public.users USING btree
    (surname COLLATE pg_catalog."default" ASC NULLS LAST)
    WITH (deduplicate_items=True)
    TABLESPACE pg_default;
```

```
"Sort  (cost=103.75..103.77 rows=7 width=93)"
"  Sort Key: u.id"
"  ->  Hash Left Join  (cost=75.85..103.65 rows=7 width=93)"
"        Hash Cond: (u.city_id = c.id)"
"        ->  Bitmap Heap Scan on users u  (cost=65.60..93.39 rows=7 width=79)"
"              Recheck Cond: (((name)::text = 'Юрий'::text) AND ((surname)::text = 'Воробьев'::text))"
"              ->  BitmapAnd  (cost=65.60..65.60 rows=7 width=0)"
"                    ->  Bitmap Index Scan on idx_name  (cost=0.00..31.42 rows=2533 width=0)"
"                          Index Cond: ((name)::text = 'Юрий'::text)"
"                    ->  Bitmap Index Scan on idx_surname  (cost=0.00..33.93 rows=2867 width=0)"
"                          Index Cond: ((surname)::text = 'Воробьев'::text)"
"        ->  Hash  (cost=6.22..6.22 rows=322 width=22)"
"              ->  Seq Scan on cities c  (cost=0.00..6.22 rows=322 width=22)"
```

Сложный индекс

```
CREATE INDEX IF NOT EXISTS idx_names
    ON public.users USING btree
```

```
   (surname COLLATE pg_catalog."default" ASC NULLS LAST, name COLLATE
pg_catalog."default" ASC NULLS LAST)
   WITH (deduplicate_items=True)
   TABLESPACE pg_default;

"Sort  (cost=29.43..29.44 rows=7 width=93)"
"  Sort Key: u.id"
"  ->  Hash Left Join  (cost=10.67..29.33 rows=7 width=93)"
"        Hash Cond: (u.city_id = c.id)"
"        ->  Index Scan using idx_names on users u  (cost=0.42..19.07 rows=7 width=79)"
"              Index Cond: (((surname)::text = 'Воробьев'::text) AND ((name)::text = 'Юрий'::text))"
"        ->  Hash  (cost=6.22..6.22 rows=322 width=22)"
"              ->  Seq Scan on cities c  (cost=0.00..6.22 rows=322 width=22)"
```

2. Gin Индексы
Раздельные индексы

```
CREATE INDEX IF NOT EXISTS trgm_idx_name
   ON public.users USING gin
   (name COLLATE pg_catalog."default" gin_trgm_ops)
   TABLESPACE pg_default;

CREATE INDEX IF NOT EXISTS trgm_idx_surname
   ON public.users USING gin
   (surname COLLATE pg_catalog."default" gin_trgm_ops)
   TABLESPACE pg_default;

"Sort  (cost=290.90..290.92 rows=7 width=93)"
"  Sort Key: u.id"
"  ->  Hash Left Join  (cost=263.00..290.80 rows=7 width=93)"
"        Hash Cond: (u.city_id = c.id)"
"        ->  Bitmap Heap Scan on users u  (cost=252.75..280.54 rows=7 width=79)"
"              Recheck Cond: (((name)::text = 'Юрий'::text) AND ((surname)::text = 'Воробьев'::text))"
"              ->  BitmapAnd  (cost=252.75..252.75 rows=7 width=0)"
"                    ->  Bitmap Index Scan on trgm_idx_name  (cost=0.00..119.00 rows=2533 width=0)"
"                          Index Cond: ((name)::text = 'Юрий'::text)"
"                    ->  Bitmap Index Scan on trgm_idx_surname  (cost=0.00..133.50 rows=2867
width=0)"
"                          Index Cond: ((surname)::text = 'Воробьев'::text)"
"        ->  Hash  (cost=6.22..6.22 rows=322 width=22)"
"              ->  Seq Scan on cities c  (cost=0.00..6.22 rows=322 width=22)"
```

Сложный индекс

```
CREATE INDEX IF NOT EXISTS trgm_idx_names
   ON public.users USING gin
   (surname COLLATE pg_catalog."default" gin_trgm_ops, name COLLATE pg_catalog."default"
gin_trgm_ops)
   WITH (fastupdate=True)
   TABLESPACE pg_default;

"Sort  (cost=254.22..254.24 rows=7 width=93)"
```

```
"  Sort Key: u.id"
"  ->  Hash Left Join  (cost=226.32..254.12 rows=7 width=93)"
"        Hash Cond: (u.city_id = c.id)"
"        ->  Bitmap Heap Scan on users u  (cost=216.07..243.86 rows=7 width=79)"
"              Recheck Cond: (((surname)::text = 'Воробьев'::text) AND ((name)::text = 'Юрий'::text))"
"              ->  Bitmap Index Scan on trgm_idx_names  (cost=0.00..216.07 rows=7 width=0)"
"                    Index Cond: (((surname)::text = 'Воробьев'::text) AND ((name)::text =
'Юрий'::text))"
"        ->  Hash  (cost=6.22..6.22 rows=322 width=22)"
"              ->  Seq Scan on cities c  (cost=0.00..6.22 rows=322 width=22)"
```

Вывод: Для запросов с точным совпадением наилучшим является **btree сложный индекс**

Запрос

```
EXPLAIN SELECT u.id, u.name, surname, age, sex, info, c.name
FROM public.users u
LEFT OUTER JOIN public.cities c on c.ID = u.city_id
WHERE u.name LIKE '%Юри%' and surname LIKE '%Воробье%'
ORDER BY id
```

2. Btree Индексы

Раздельные индексы

```
CREATE INDEX IF NOT EXISTS idx_name
    ON public.users USING btree
    (name COLLATE pg_catalog."default" ASC NULLS LAST)
    WITH (deduplicate_items=True)
    TABLESPACE pg_default;

CREATE INDEX IF NOT EXISTS idx_surname
    ON public.users USING btree
    (surname COLLATE pg_catalog."default" ASC NULLS LAST)
    WITH (deduplicate_items=True)
    TABLESPACE pg_default;
```

```
"Gather Merge  (cost=37344.03..37345.67 rows=14 width=93)"
"  Workers Planned: 2"
"  ->  Sort  (cost=36344.01..36344.03 rows=7 width=93)"
"        Sort Key: u.id"
"        ->  Nested Loop Left Join  (cost=0.15..36343.91 rows=7 width=93)"
"              ->  Parallel Seq Scan on users u  (cost=0.00..36334.00 rows=7 width=79)"
"                    Filter: (((name)::text ~~ '%Юри%'::text) AND ((surname)::text ~~ '%Воробье
%'::text))"
"              ->  Index Scan using cities_pkey on cities c  (cost=0.15..1.41 rows=1 width=22)"
"                    Index Cond: (id = u.city_id)"
```

Сложный индекс

```
CREATE INDEX IF NOT EXISTS idx_names
    ON public.users USING btree
    (surname COLLATE pg_catalog."default" ASC NULLS LAST, name COLLATE
pg_catalog."default" ASC NULLS LAST)
    WITH (deduplicate_items=True)
```

```
    TABLESPACE pg_default;

"Gather Merge  (cost=37344.03..37345.67 rows=14 width=93)"
"  Workers Planned: 2"
"  ->  Sort  (cost=36344.01..36344.03 rows=7 width=93)"
"        Sort Key: u.id"
"        ->  Nested Loop Left Join  (cost=0.15..36343.91 rows=7 width=93)"
"              ->  Parallel Seq Scan on users u  (cost=0.00..36334.00 rows=7 width=79)"
"                    Filter: (((name)::text ~~ '%Юри%'::text) AND ((surname)::text ~~ '%Воробье
%'::text))"
"              ->  Index Scan using cities_pkey on cities c  (cost=0.15..1.41 rows=1 width=22)"
"                    Index Cond: (id = u.city_id)"
```

2. Gin Индексы
Раздельные индексы

```
CREATE INDEX IF NOT EXISTS trgm_idx_name
    ON public.users USING gin
    (name COLLATE pg_catalog."default" gin_trgm_ops)
    TABLESPACE pg_default;

CREATE INDEX IF NOT EXISTS trgm_idx_surname
    ON public.users USING gin
    (surname COLLATE pg_catalog."default" gin_trgm_ops)
    TABLESPACE pg_default;

"Sort  (cost=224.79..224.83 rows=16 width=93)"
"  Sort Key: u.id"
"  ->  Hash Left Join  (cost=161.29..224.47 rows=16 width=93)"
"        Hash Cond: (u.city_id = c.id)"
"        ->  Bitmap Heap Scan on users u  (cost=151.05..214.18 rows=16 width=79)"
"              Recheck Cond: (((name)::text ~~ '%Юри%'::text) AND ((surname)::text ~~ '%Воробье
%'::text))"
"              ->  BitmapAnd  (cost=151.05..151.05 rows=16 width=0)"
"                    ->  Bitmap Index Scan on trgm_idx_name  (cost=0.00..39.07 rows=2543 width=0)"
"                          Index Cond: ((name)::text ~~ '%Юри%'::text)"
"                    ->  Bitmap Index Scan on trgm_idx_surname  (cost=0.00..111.72 rows=6362
width=0)"
"                          Index Cond: ((surname)::text ~~ '%Воробье%'::text)"
"        ->  Hash  (cost=6.22..6.22 rows=322 width=22)"
"              ->  Seq Scan on cities c  (cost=0.00..6.22 rows=322 width=22)"
```

Сложный индекс

```
CREATE INDEX IF NOT EXISTS trgm_idx_names
    ON public.users USING gin
    (surname COLLATE pg_catalog."default" gin_trgm_ops, name COLLATE pg_catalog."default"
gin_trgm_ops)
    WITH (fastupdate=True)
    TABLESPACE pg_default;

"Sort  (cost=165.91..165.95 rows=16 width=93)"
```

" Sort Key: u.id"
" -> Hash Left Join (cost=102.41..165.59 rows=16 width=93)"
"   Hash Cond: (u.city_id = c.id)"
"   -> Bitmap Heap Scan on users u (cost=92.17..155.30 rows=16 width=79)"
"   Recheck Cond: (((surname)::text ~~ '%Воробье%'::text) AND ((name)::text ~~ '%Юри
%'::text))"
"   -> Bitmap Index Scan on trgm_idx_names (cost=0.00..92.16 rows=16 width=0)"
"   Index Cond: (((surname)::text ~~ '%Воробье%'::text) AND ((name)::text ~~ '%Юри
%'::text))"
"   -> Hash (cost=6.22..6.22 rows=322 width=22)"
"   -> Seq Scan on cities c (cost=0.00..6.22 rows=322 width=22)"

Вывод: Для запросов с поиском подстрок наилучшим является **gin сложный индекс**

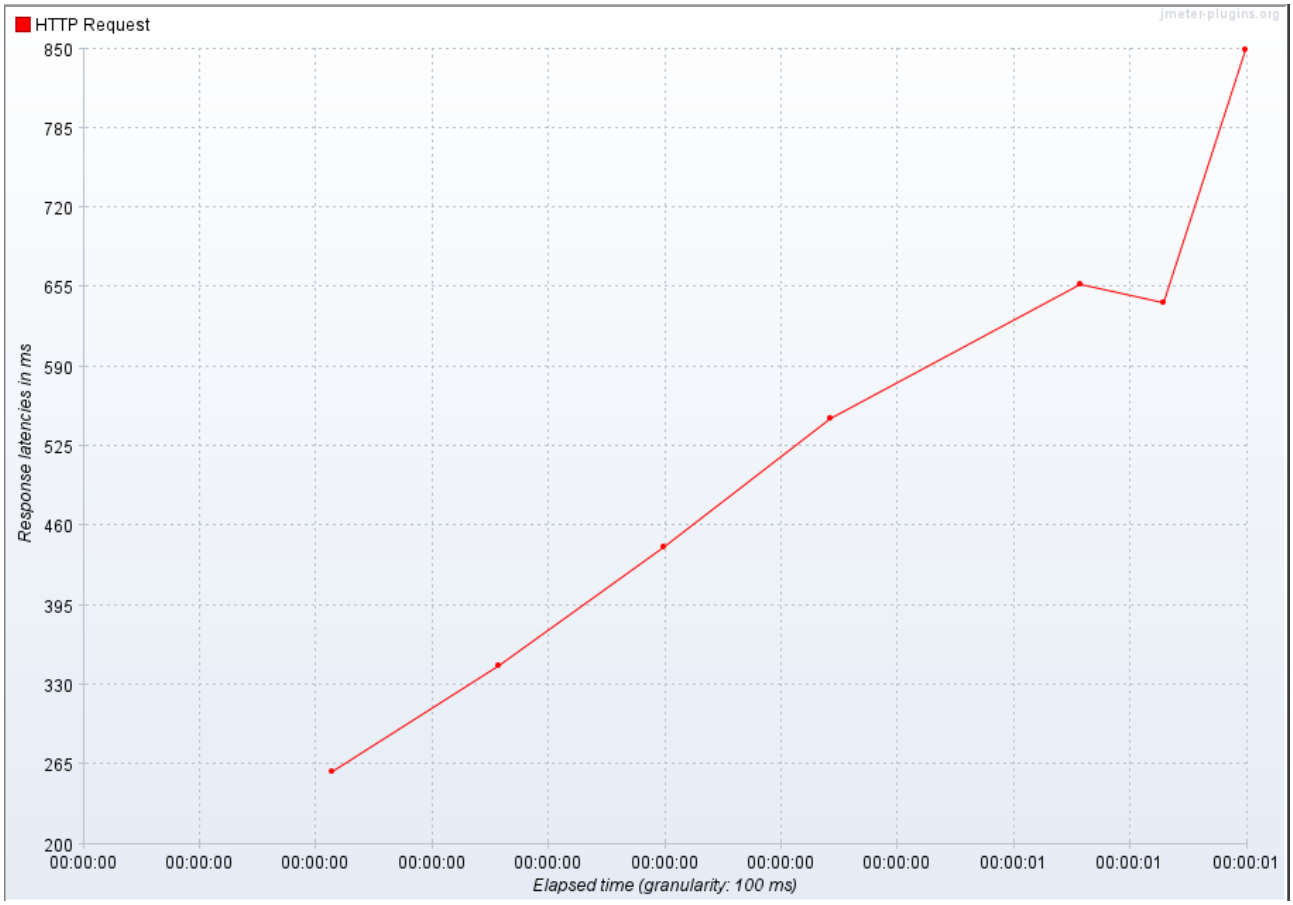
ВЫВОД: Учитывая что нам неизвестно будет ли использоваться поиск по подстрокам, оптимальным индексом является gin сложный индекс

Графики

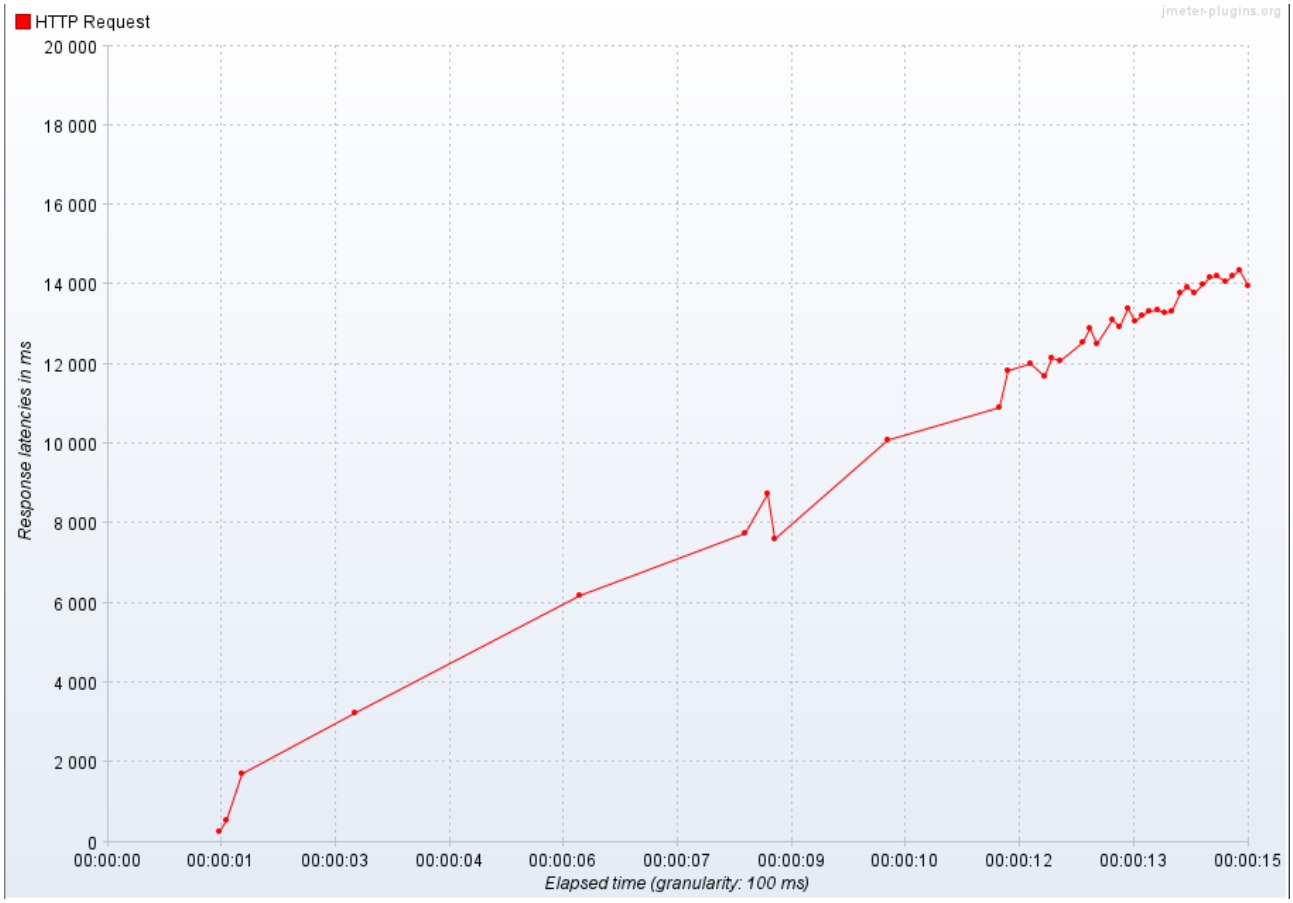
Графики получены из Jmeter

графики latency до индекса

10 юзеров

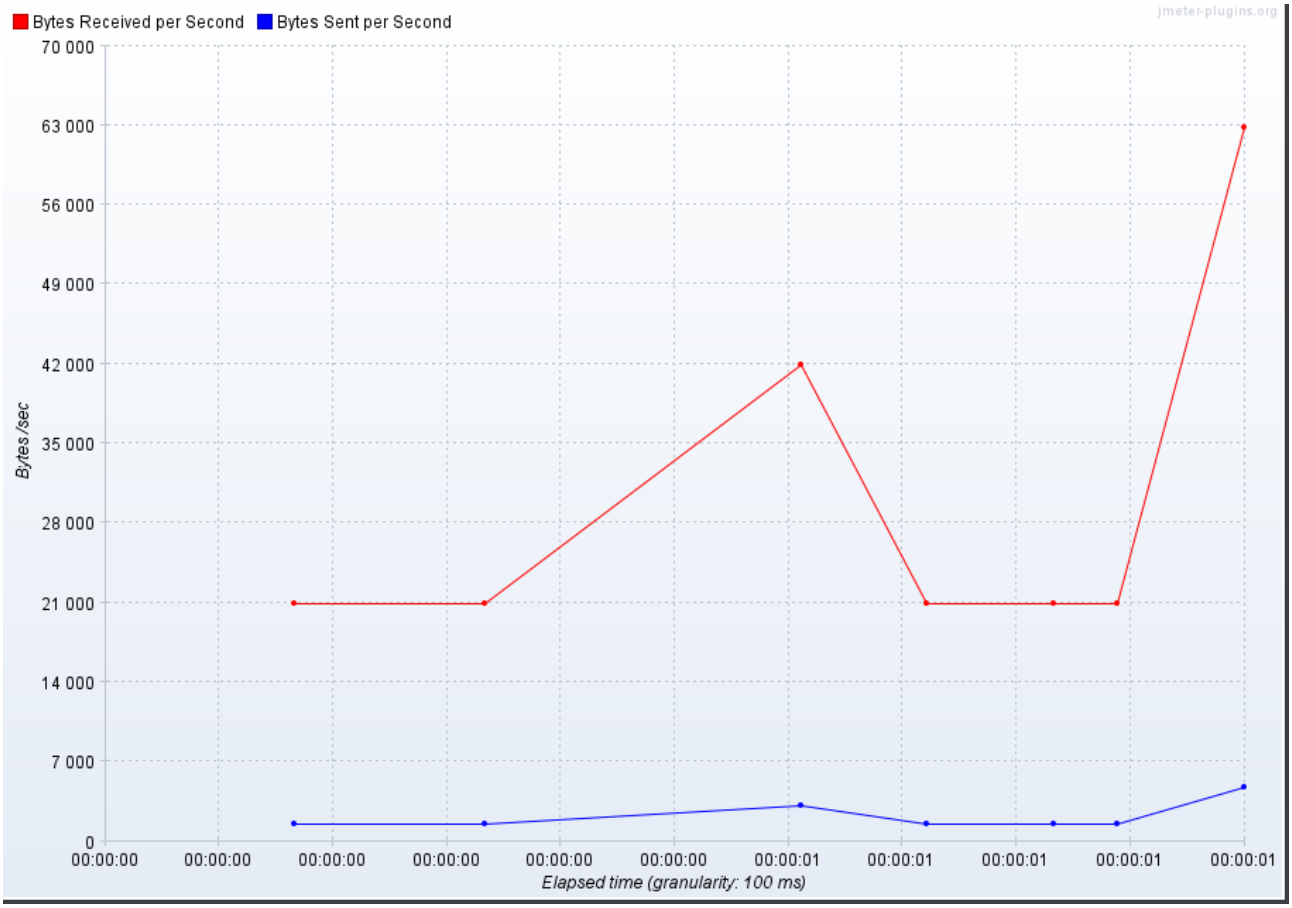


100 юзеров

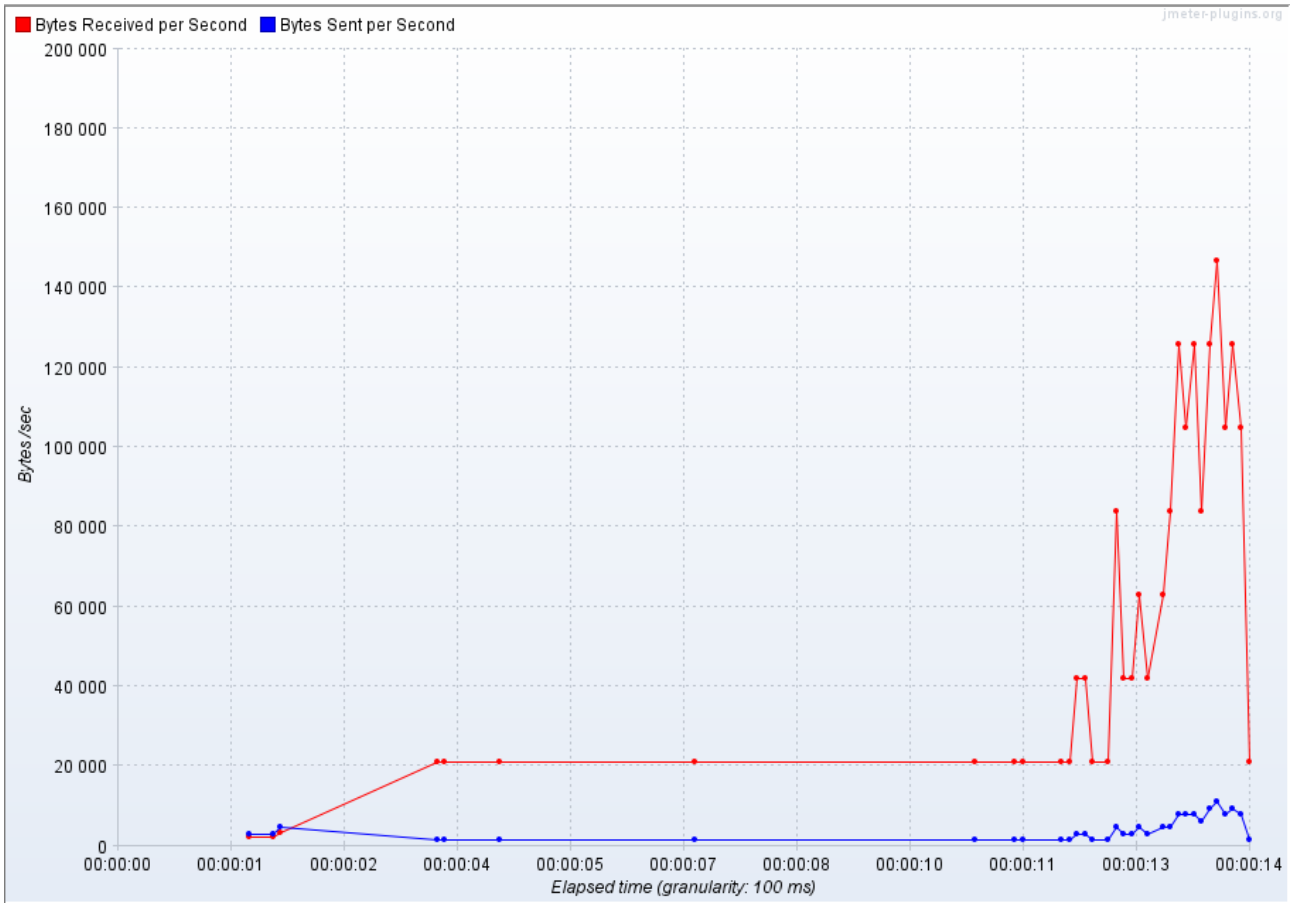

HTTP Request
jmeter-plugins.org
20 000
18 000
16 000
14 000
12 000
10 000
8 000
6 000
4 000
2 000
0
Response latencies in ms
00:00:00
00:00:01
00:00:03
00:00:04
00:00:06
00:00:07
00:00:09
00:00:10
00:00:12
00:00:13
00:00:15
Elapsed time (granularity: 100 ms)

графики throughput до индекса

10 юзеров

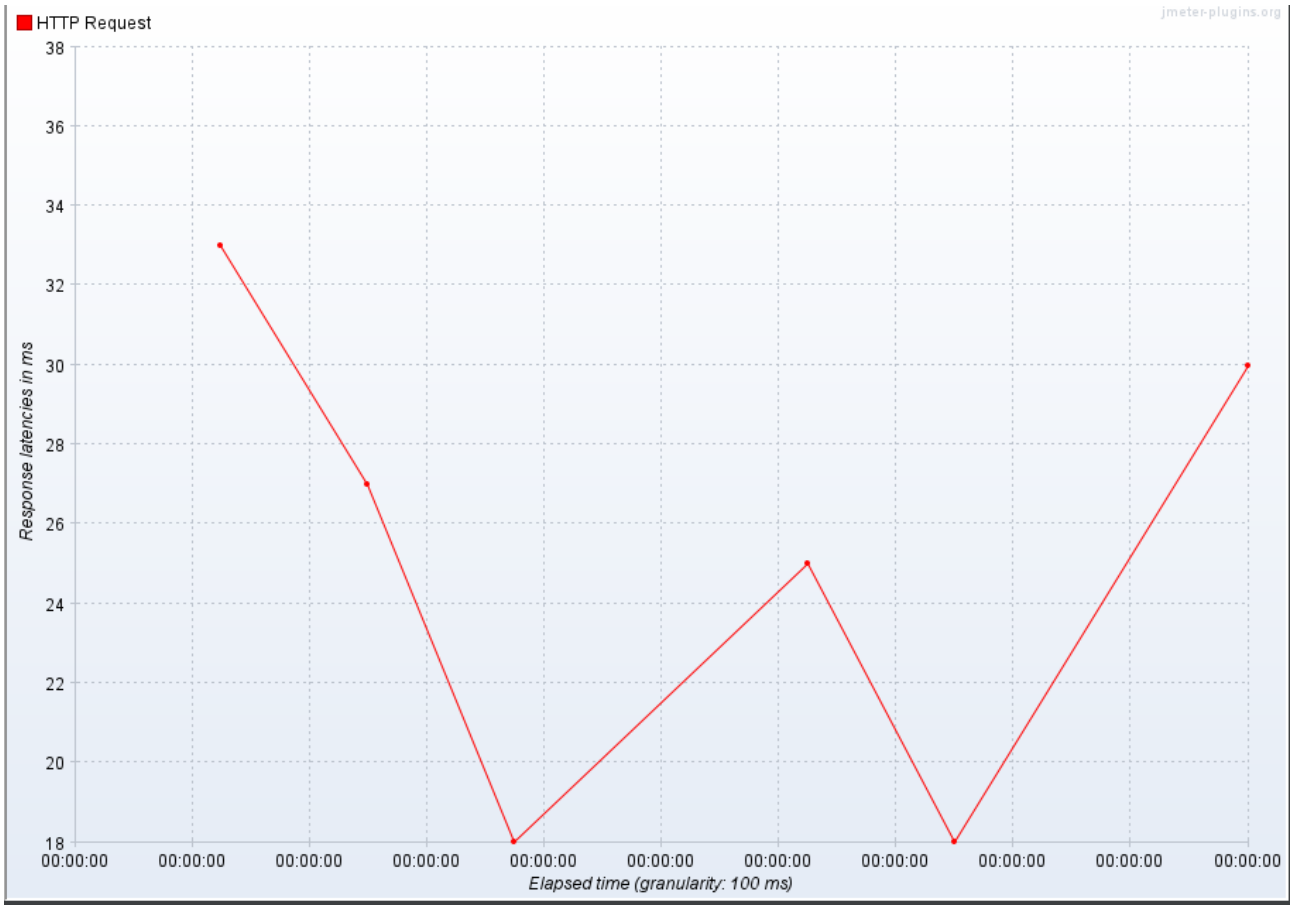


100 юзеров


Bytes Received per Second
Bytes Sent per Second
jmeter-plugins.org
200 000
180 000
160 000
140 000
120 000
100 000
80 000
60 000
40 000
20 000
0
Bytes /sec
00:00:00
00:00:01
00:00:02
00:00:04
00:00:05
00:00:07
00:00:08
00:00:10
00:00:11
00:00:13
00:00:14
Elapsed time (granularity: 100 ms)

графики latency после индекса gin

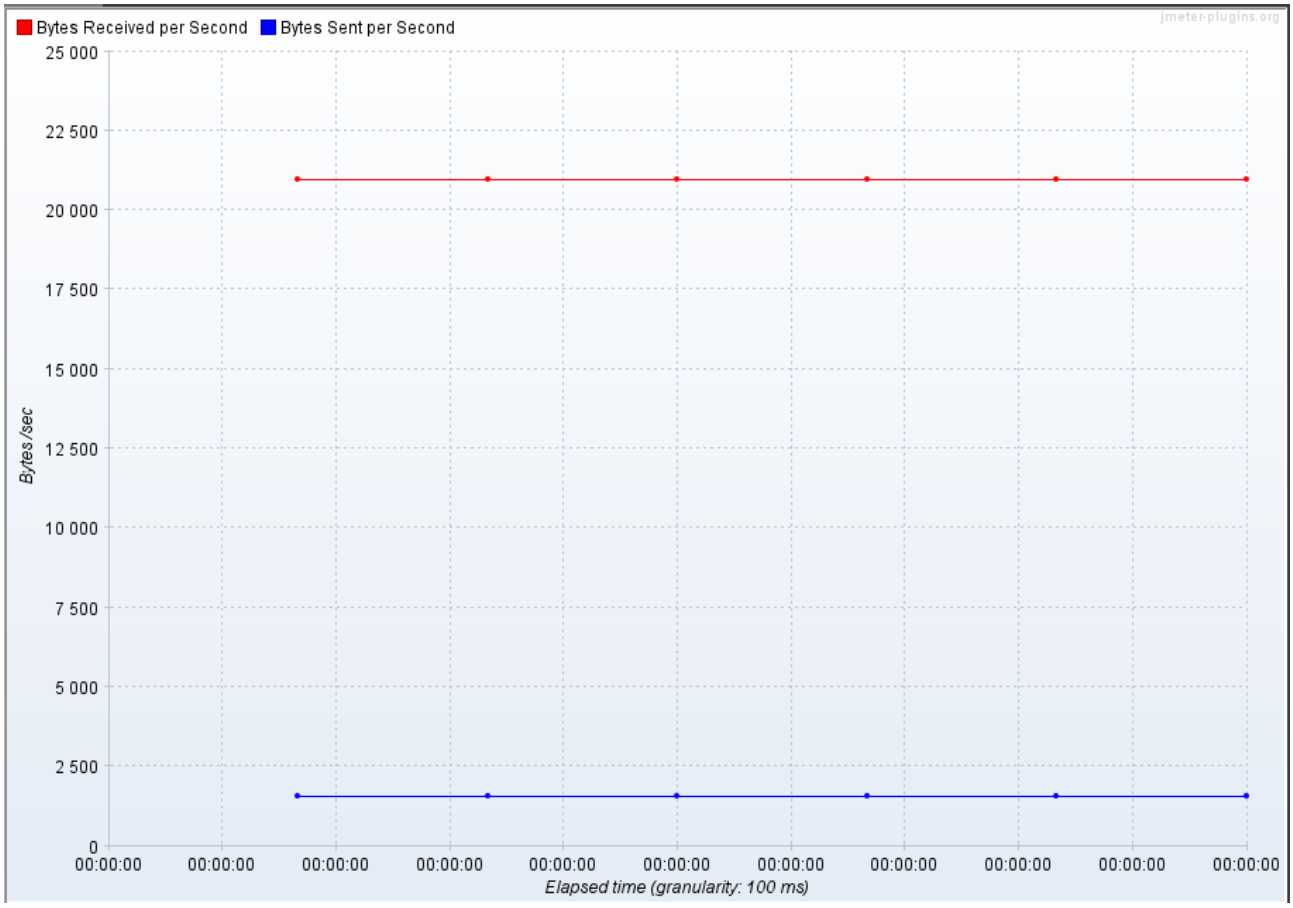
10 юзеров



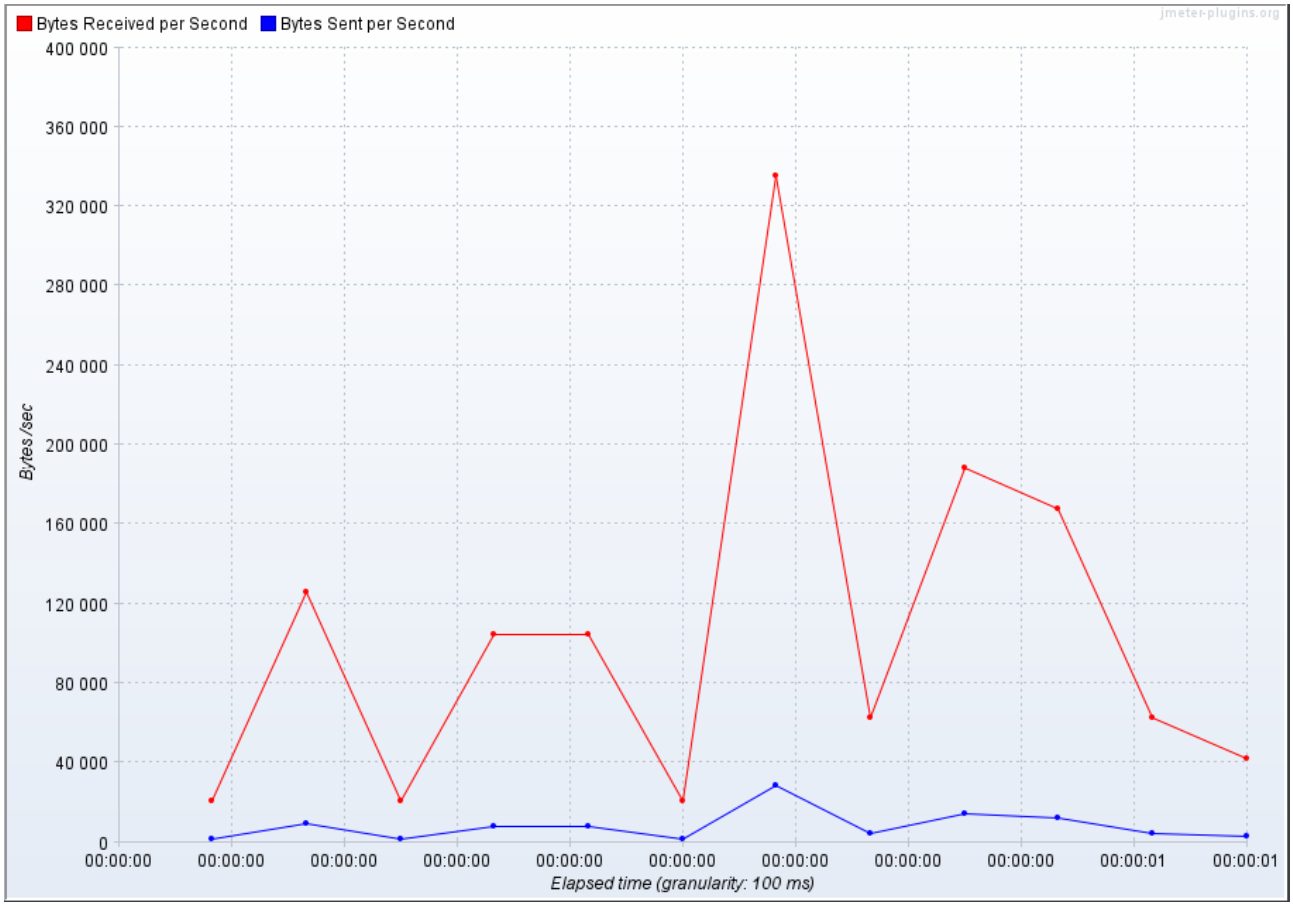
100 юзеров


HTTP Request
jmeter-plugins.org
300
270
240
210
180
150
120
90
60
30
0
Response latencies in ms
00:00:00
00:00:00
00:00:00
00:00:00
00:00:00
00:00:00
00:00:00
00:00:00
00:00:00
00:00:00
00:00:01
Elapsed time (granularity: 100 ms)

графики throughput после индекса gin
10 юзеров



100 юзеров


Bytes Received per Second
Bytes Sent per Second
jmeter-plugins.org
400 000
360 000
320 000
280 000
240 000
200 000
160 000
120 000
80 000
40 000
0
Bytes /sec
00:00:00
00:00:00
00:00:00
00:00:00
00:00:00
00:00:00
00:00:00
00:00:00
00:00:00
00:00:01
00:00:01
Elapsed time (granularity: 100 ms)