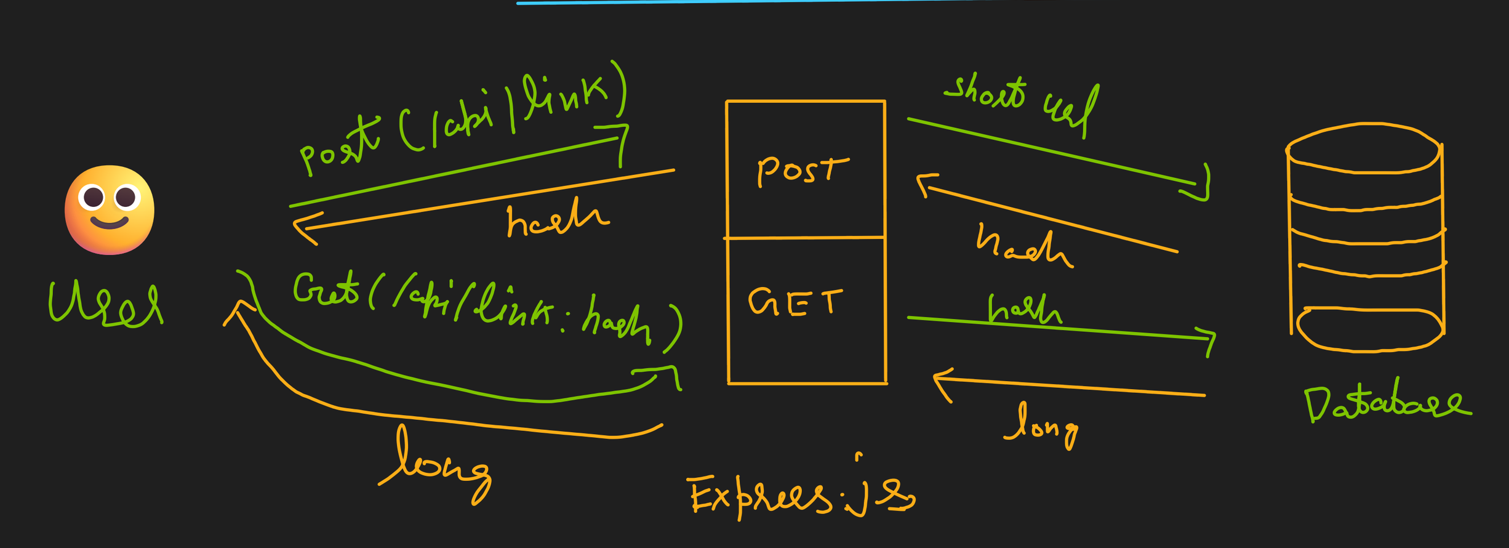
Simplest url shortner



hash can be done using UUID module & taking First 7 chars of it.

Test setup

Virtual users	Start time	Load profile
100 VU	Mar 1, 22:07:10 (GMT+5:30)	Fixed

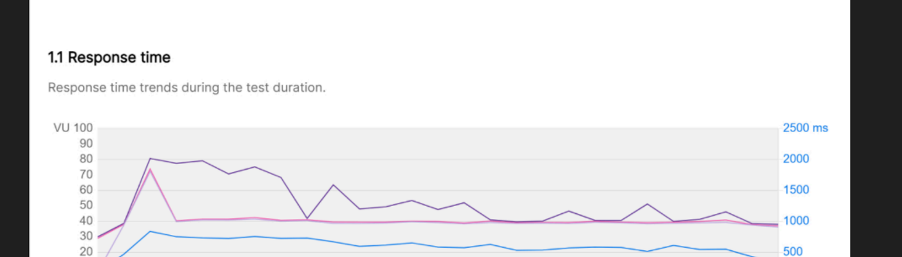
Duration	End time	Environment
1 minute	Mar 1, 22:08:17 (GMT+5:30)	-

1. Summary

Total requests sent	Throughput	Average response time	Error rate
4,419	65.96 requests/second	595 ms	0.00 %

1.1 Response time

Response time trends during the test duration.



1.2 Throughput

Rate of requests sent per second during the test duration.

1.3 Requests with slowest response times

Top 5 slowest requests based on their average response times.

Request	Resp. time (Avg ms)	90th (ms)	95th (ms)	99th (ms)	Min (ms)	Max (ms)
POST Post Link http://localhost:5000/api/link	760	1,003	1,019	1,817	68	2,057
GET Get Link http://localhost:5000/api/link/f5236ec	430	809	876	1,202	33	1,791

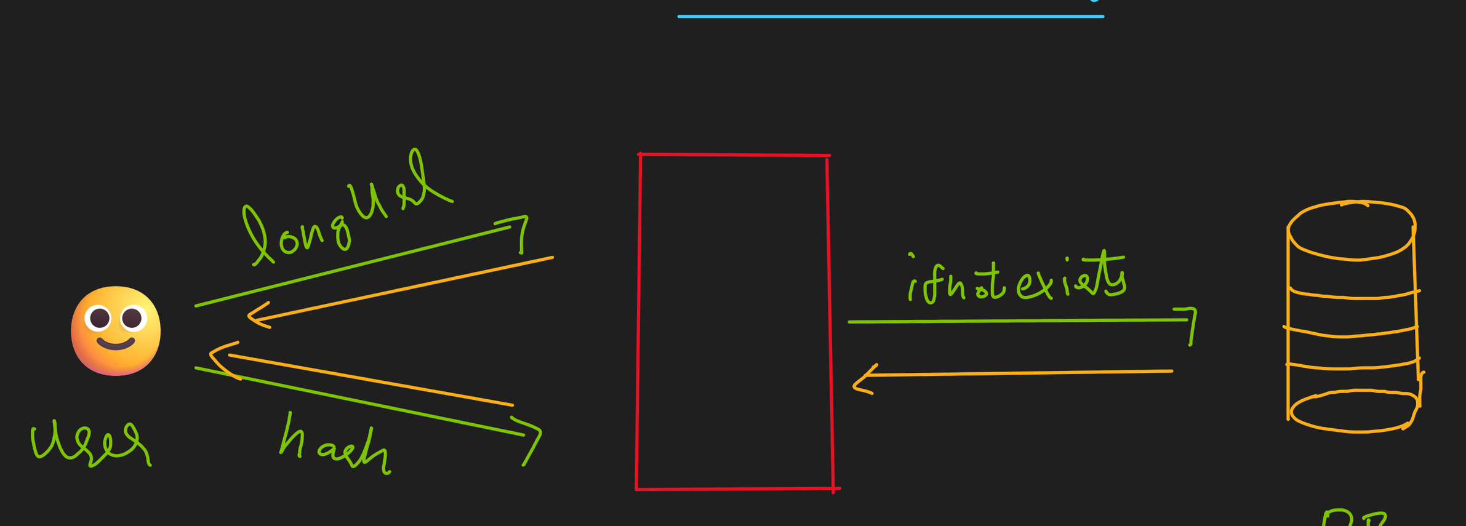
2. Metrics for each request

The requests are shown in the order they were sent by virtual users.

Request	Total requests	Requests/s	Min (ms)	Avg (ms)	90th (ms)	Max (ms)	Error %
GET Get Link http://localhost:5000/api/link/f5236ec	2,213	33.03	33	430	809	1,791	0
POST Post Link http://localhost:5000/api/link	2,206	32.93	68	760	1,003	2,057	0

All this data is for read requests to mongodb.

Let's Cash it! 🤑



Single requests

GET

Before

EVENT	TIME
Prepare	3.95 ms
Socket Initialization	2.01 ms
DNS Lookup	0.94 ms
TCP Handshake	1.39 ms
Transfer Start	38 ms
Download	1.53 ms
Process	0.15 ms
Total	47.94 ms

After

EVENT	TIME
Prepare	2.75 ms
Socket Initialization	0.24 ms
DNS Lookup	Cache
TCP Handshake	Cache
Transfer Start	2.69 ms
Download	0.76 ms
Process	0.04 ms
Total	6.46 ms

Post

Before

EVENT	TIME
Prepare	5.32 ms
Socket Initialization	1.77 ms
DNS Lookup	0.8 ms
TCP Handshake	1.66 ms
Transfer Start	153.08 ms
Download	1.72 ms
Process	0.14 ms
Total	164.47 ms

After

EVENT	TIME
Prepare	2.38 ms
Socket Initialization	0.2 ms
DNS Lookup	Cache
TCP Handshake	Cache
Transfer Start	39.01 ms
Download	0.88 ms
Process	0.04 ms
Total	42.48 ms

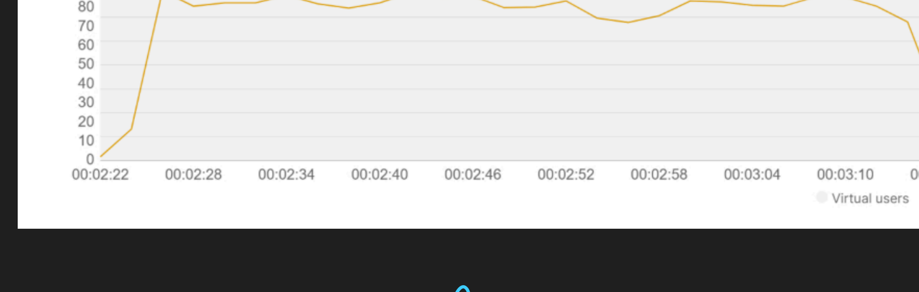
For 100 Users

1. Summary

Total requests sent	Throughput	Average response time	Error rate
8,369	125.11 requests/second	26 ms	0.00 %

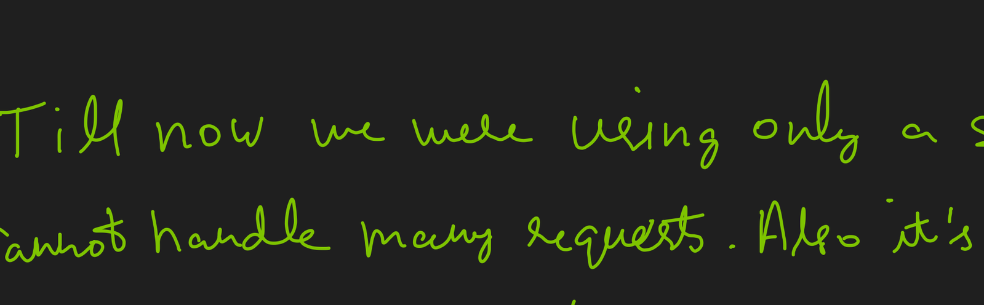
1.1 Response time

Response time trends during the test duration.



1.2 Throughput

Rate of requests sent per second during the test duration.



1.3 Requests with slowest response times

Top 5 slowest requests based on their average response times.

Request	Resp. time (Avg ms)	90th (ms)	95th (ms)	99th (ms)	Min (ms)	Max (ms)
POST Post Link http://localhost:5000/api/link	43	46	51	174	32	512
GET Get Link http://localhost:5000/api/link/6040b1e	9	9	15	101	3	507

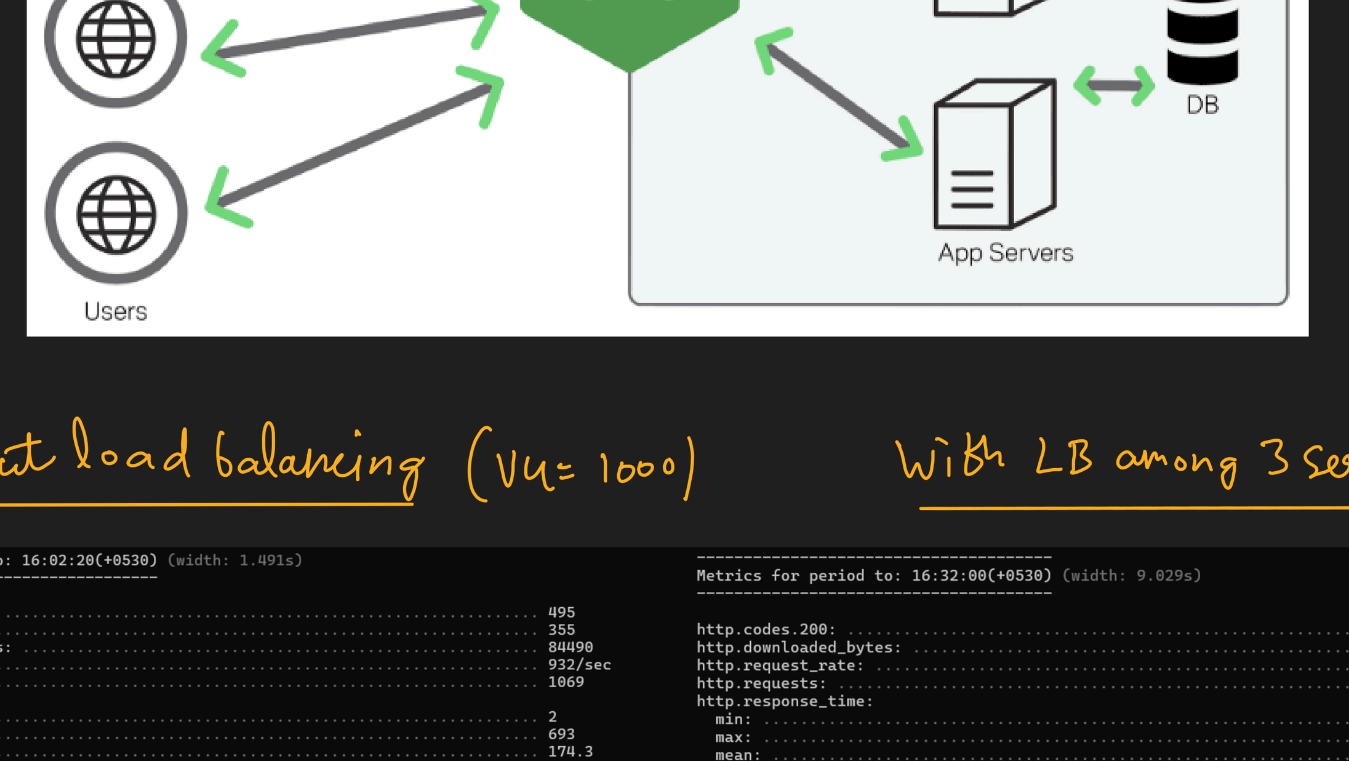
2. Metrics for each request

The requests are shown in the order they were sent by virtual users.

Request	Total requests	Requests/s	Min (ms)	Avg (ms)	90th (ms)	Max (ms)	Error %
GET Get Link http://localhost:5000/api/link/6040b1e	4,188	62.61	3	9	9	507	0
POST Post Link http://localhost:5000/api/link	4,181	62.50	32	43	46	512	0

And to go even further beyond! 😱

Till now we were using only a single server & at peak times it cannot handle many requests. Also it's a single point of failure. So we can use 'n' servers to distribute the load.



Without load balancing (VU= 1000)

```
Metrics for period to: 16:02:20(+0530) (width: 1.491s)
--------------------------------------------
errors.ECONNREFUSED: ........................ 495
http.codes.200: ............................. 355
http.downloaded_bytes: ...................... 84490
http.request_rate: .......................... 932/sec
http.requests: .............................. 1069
http.response_time:
  min: ...................................... 2
  max: ...................................... 693
  mean: ..................................... 174.3
  median: ................................... 64.7
  p95: ...................................... 685.5
  p99: ...................................... 685.5
http.responses: ............................. 355
vusers.completed: ........................... 355
vusers.created: ............................. 1071
vusers.created_by_name.0: ................... 1071
vusers.failed: .............................. 495
vusers.session_length:
  min: ...................................... 5.3
  max: ...................................... 705.3
  mean: ..................................... 182.6
  median: ................................... 70.1
  p95: ...................................... 685.5
  p99: ...................................... 699.4
```

With LB among 3 Servers (VU=1000)

```
--------------------------------------------
Metrics for period to: 16:32:00(+0530) (width: 9.029s)
--------------------------------------------
http.codes.200: ............................. 868
http.downloaded_bytes: ...................... 206584
http.request_rate: .......................... 100/sec
http.requests: .............................. 868
http.response_time:
  min: ...................................... 2
  max: ...................................... 14
  mean: ..................................... 3.8
  median: ................................... 3
  p95: ...................................... 6
  p99: ...................................... 7
http.responses: ............................. 868
vusers.completed: ........................... 868
vusers.created: ............................. 868
vusers.created_by_name.0: ................... 868
vusers.failed: .............................. 0
vusers.session_length:
  min: ...................................... 4.5
  max: ...................................... 34.5
  mean: ..................................... 7.4
  median: ................................... 6.8
  p95: ...................................... 10.9
  p99: ...................................... 27.9
```

These load tests are done by artillery as postman only supports max VU = 100.

We have done what we could to optimize this web app but As the users will ↑ probability of collision may also ↑ as we are taking uuid[:8] only.

Using a token to generate Base62 encoding of length 7 which gives us 3.5 trillion possible combinations. We can allot a token range to specific servers & all the synchronization will be done by ZooKeeper.