Report on Closed system testing

Himadri Sekhar Bandyopadhay(173050004) and Arijit Mukherjee (17305t002)

October 9, 2017

1 Introduction

We conducted a **closed loop** load testing on the server. The request was made to the webserver endpoint that serves the homepage for game creation. The script for testing is written in python, and the request was send in form of get request, with timeout 10 seconds. The code is attached in the uploaded folder.

2 The way the test was conducted

The test was conducted by varying the number of users(realised with varying number of threads) from 1 to 401 , with intervals of 10. Each test was conducted for a duration of 60 seconds.For the graph of Turnaround vs number of users, **Average Response time** was plotted against **Number of users**.

$Average\ Response\ time = \frac{Total\ \ response\ \ time}{Total\ \ number\ \ of\ \ requests}$ For the purpose of obtaining througput, we first needed to find the number requests that were **succesffuly responded** to. This was done by seeing the return value of the get request.

The test was conducted by terminating the server, after every set of users flood the server with requests. This was done so that the observation for the present request was independent of the previous one. If this was not done, then the server was filled with the request of the previous set of user requests, which led to false readings. $Throughput = \frac{Number\ \ succesfull\ \ request}{Total\ \ test\ \ time}$

3 Observation Table

	N	success_count	avg_resp_time	total_time	throughput
0	1	573	0.004762	2.728835	9.550000
1	11	5261	0.018963	99.785664	87.683333
2	21	8124	0.042150	342.426423	135.400000
3	31	10163	0.062961	639.933563	169.383333
4	41	11003	0.096242	1058.953823	183.383333
5	51	11299	0.140371	1586.327189	188.316667
6	61	11488	0.178039	2046.554384	191.466667
7	71	11609	0.232204	2695.886790	193.483333
8	81	11645	0.274924	3201.494587	194.083333
9	91	11480	0.329907	3787.989536	191.333333
10	101	11244	0.381880	4295.387171	187.400000
11	111	11386	0.435881	4963.379746	189.766667
12	121	11086	0.504209	5593.698519	184.766667
13	131	11274	0.542385	6117.557819	187.900000
14	141	10787	0.608382	6574.177828	179.783333
15	151	10888	0.605368	6610.013728	181.466667
16	161	10711	0.662288	7113.635634	178.516667
17	171	10492	0.754772	7940.202775	174.866667
18	181	10380	0.797159	8304.004318	173.000000
19	191	10504	0.757547	8001.213104	175.066667
20	201	9990	0.862330	8691.429109	166.500000

Table 1: Observation table

4 Graphs

4.1 Turn around time vs Number of users

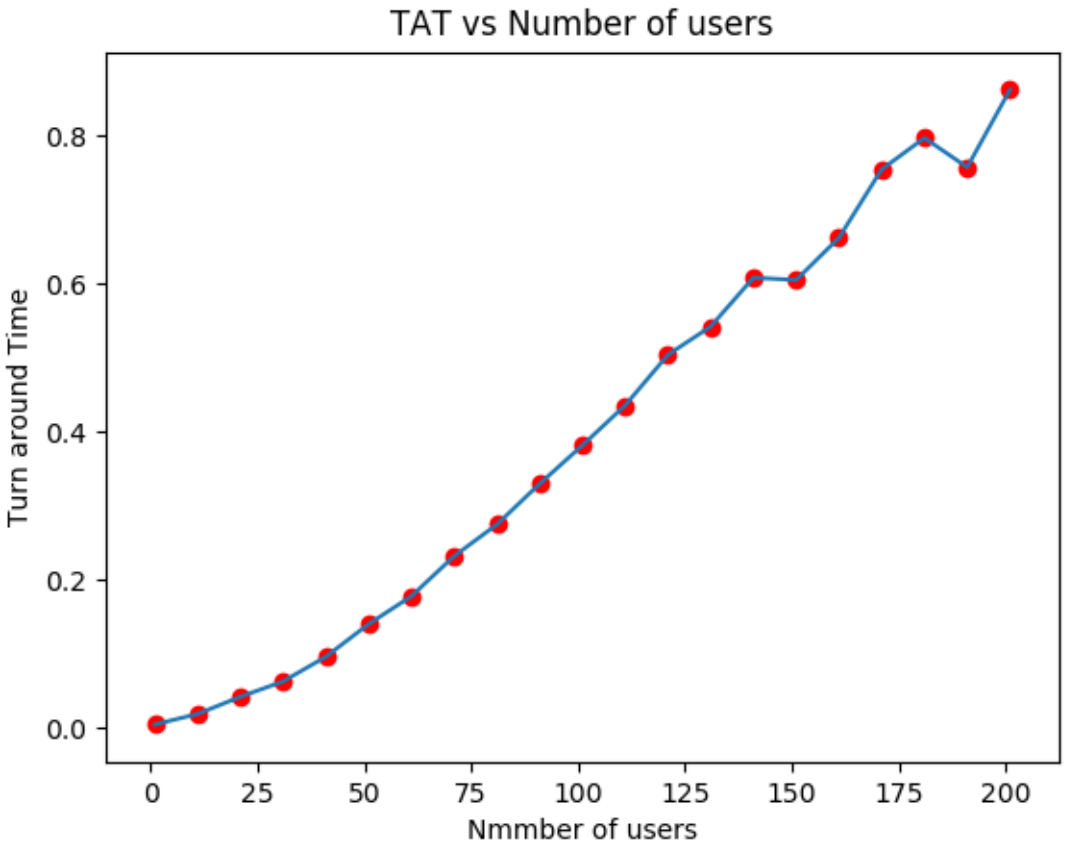

Figure 1: Turn around time vs Number of users

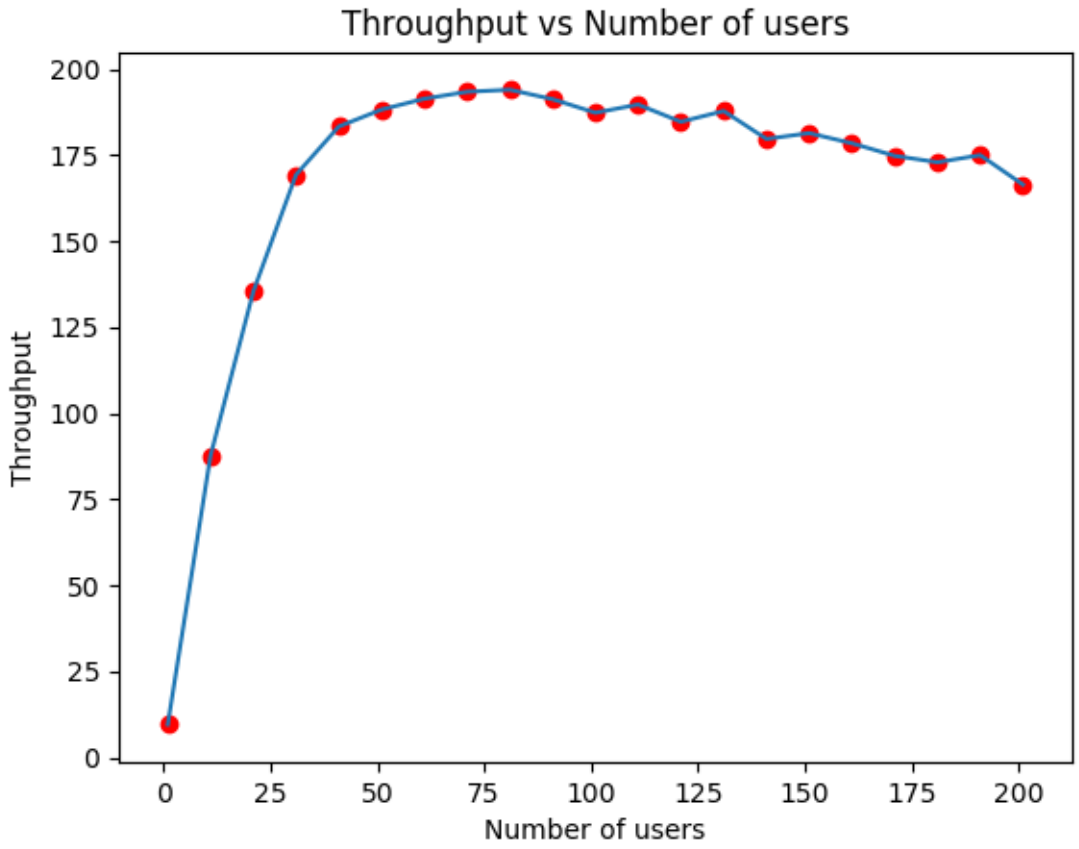

Figure 2: Throughput vs Number of users

5 Observations

The number of users(threads) that saturates the systems was found to be somwhere around 50 to 75. When the number of threads was restricted below 75, the Turn around time was increasing almost linearly, but as it crosses 75, the Turn around time approximates an exponential growth, as is expected ideally.

Similarly, in the graph for throughput, we observe that when the number of threads(users) are less than 50, the throughput grows linearly. After the range of 75 the throughput almost becomes constant, and then it falls when the number of users exceed 200. In practical casses the the throughput is expected to fall , rather than remaining constant because of lot of requests getting denied, due to timeout, as the number of requests to the servers are very high, leading to a huge overhead in terms of reading from disk.

6 Conclusion

The throughput saturated due do CPU bottleneck, as one of the cores hit 100% utilisation, hence the threads on that core had a longer response time in the server. Also, while doing the test, each thread after sending was made to sleep for half a second(this was the way think time was implemented), and this was the reason for getting linearity of the graph in the region from 1 to 75 users.