All the data and graphs are in the Excel files.

1. For the first part of the homework, we have set a fixed number of threads for the wrk stress test @ 20 threads. A 7th node has been created from which all the benchmarks were made so that there would be no added latency or CPU bottlenecks on the nodes running the webservers or the node running the load balancer (node0). We can see that only the first scenario was able to finish the benchmark and handle 1000 connections. When we started increasing the Webservers, the load balancer crushed at 400~500 connections and the benchmark stopped working. Although we have observed a significant increase in throughput when we increased the number of web servers, the successful requests were fewer than the actual requests since the load balancer was easily overloaded (~80% CPU usage @ 200 connections) and started dropping requests. We can also see that, in terms of latency, running multiple web servers does decrease the 50th percentile latency experienced by the users but the 99th percentile is not affected by much, except the case where we had 1 webserver. In that case we can see a sudden increase in latency which means that the load balancer was under heavy load and the requests were queued and then served. Another notable observation is the number of bad requests. While the number of web servers was increasing, the number of total requests was also increasing but also the number of bad requests; the requests that could not be served properly and returned HTTP error code 5xx or 4xx (HTTP Code 4xx means the site was not found, HTTP code 5xx means bad gateway). This is probably due to the high load on the load balancer and its inability to redirect the requests to the appropriate web servers. With the increase of web servers, the load balancer was easily overloaded and was hitting CPU utilization close to 100% at a low number of connections. In conclusion, having a load balancer can result at an increase in throughput, as we’ve seen, and in the decrease of latency but with a limited number of connections. But since the throughput is increased, then there will be fewer simultaneous connections on the load balancer since the requests will be served quicker meaning that in a real-world situation a load balancer would perform a lot better than in a synthetic benchmark. It should also be noted that the web servers were running under limited number of connections to better simulate queuing and request latency under heavy load.

[Static Website](https://ucy-my.sharepoint.com/:x:/g/personal/ssofok02_ucy_ac_cy/EcmQ7bkeP0FHjCzEYv3bcXMBHeuhpvQmMEra28gIF2vW9w)

1.2) For this part of the homework we’ve ran the hotel app monolith implementation. The scenarios stayed the same as the previous experiment with the only change in the number of connections each web server accepts. At 16 connections it was impossible to run the benchmark for more than 100 connections from wrk. The number of connections was increased to 500. The biggest improvement is seen in the throughput. As we increase the number of web servers the throughput is also increased. Latency is also improved but not by much, as connections increase the latency reaches a ceiling at around 1.5 Sec. Compared to the static site, the observed latency is larger due to the heavier website that must be loaded. Due to the increase of allowed connections to the web servers we have not observed any dropped or unserved requests.

[HotelApp](https://ucy-my.sharepoint.com/:x:/g/personal/ssofok02_ucy_ac_cy/ETAadGURrURBq6S3hXF38eEBgTfND1dtPUO-GrbEyD2qrg)

1. For this part of the homework, we had to run the web search benchmark. There were 3 phases in this experiment, running a single index server, then 2 and lastly 4 Index servers. Running just 1 index server, we tested the web search instance using a high core count client. For each iteration of the test, the thread count of the client was doubled. We have observed an increase of the operations/sec as the threads were increasing but also an increase in latency. We can explain the increase of latency since the system (Index Server) was overloaded. As we have observed, when the clients (number of threads of the client) increased at around 16, the index server’s CPU utilization jumped @ around 60% and kept climbing as the thread count were increasing. Since the system was overloaded, the increase latency was expected.

2.2) For the second part, we have configured 2 index servers so that we could evenly distribute the indexes to these 2 servers. In theory, this should give us better throughput and lower latency since each index server would have half the usage compared to one index server. Running the same client as before resulted in some un-expected results. The latency did not get improved, but we have seen some slight improvement over the throughput. Stepping up to 4 index servers, the results got much better. The latency dropped to half and the throughput hit its maximum very soon. The overall CPU usage was reduced since the index got distributed to 4 servers and each server was receiving less traffic.

[Websearch](https://ucy-my.sharepoint.com/:x:/g/personal/ssofok02_ucy_ac_cy/EWzPWg4N-XhFnrWMIuvI-GsB4835_m-bY78uB4FSKxCdIA)