O.S Lab 2 Report

Shouvanik Chakrabarti (130050072)
Krishna Harsha Reddy (130050076)

1)

Setup:

We are using two personal laptops connected over the institute LAN using 100 Mbps Ethernet cables.

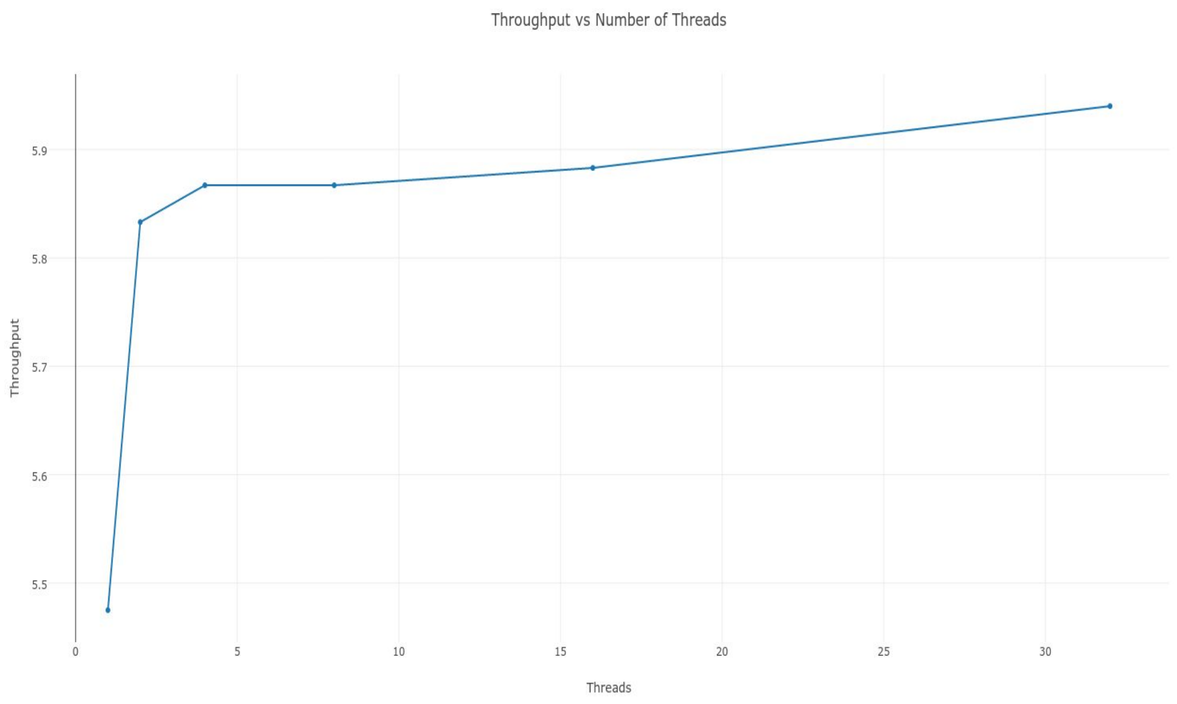
Client: Memory - 3.8 GiB Processor: 2.30 GHz x 4

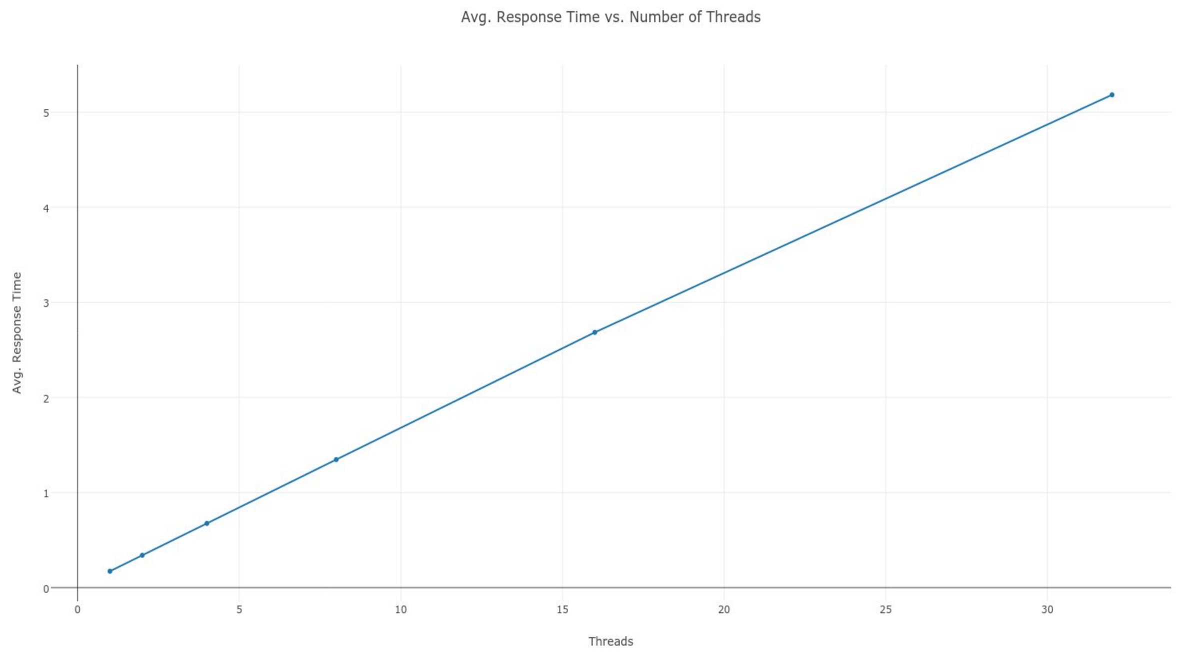
Server: Memory - 7.7 GiB Processor: 2.40GHz × 8

i)The maximum read bandwidth of the disk on the server is 45 MB/s. This translates into ~22.5 req/s throughput at full utilization.

ii)The maximum network bandwidth you can get between the client and server machines is around 95 Mbps ie. ~12 MB/s. This corresponds to a throughput of ~6 req/s at full utilization.

2)





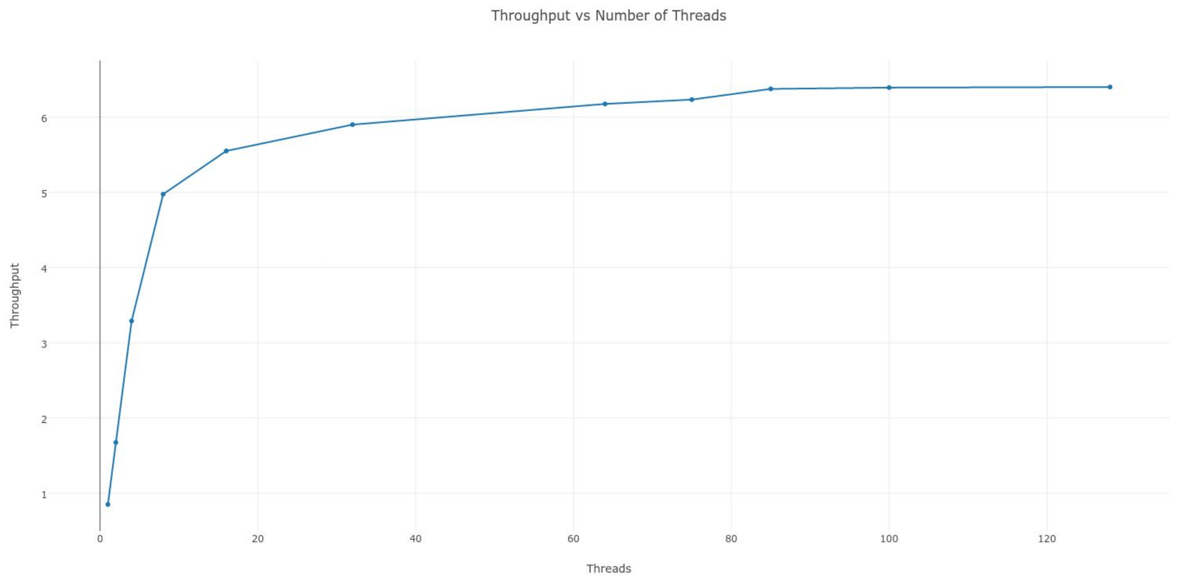
a) The optimal value of n in this case is 4. We can observe that the graph practically flattens out after n=4. At very large values of n the actual experiment time significantly overshoots the actual experiment time, since we have to wait for all the started downloads to complete. This results in a slight increase in perceived throughput due to this underestimation of the run time.

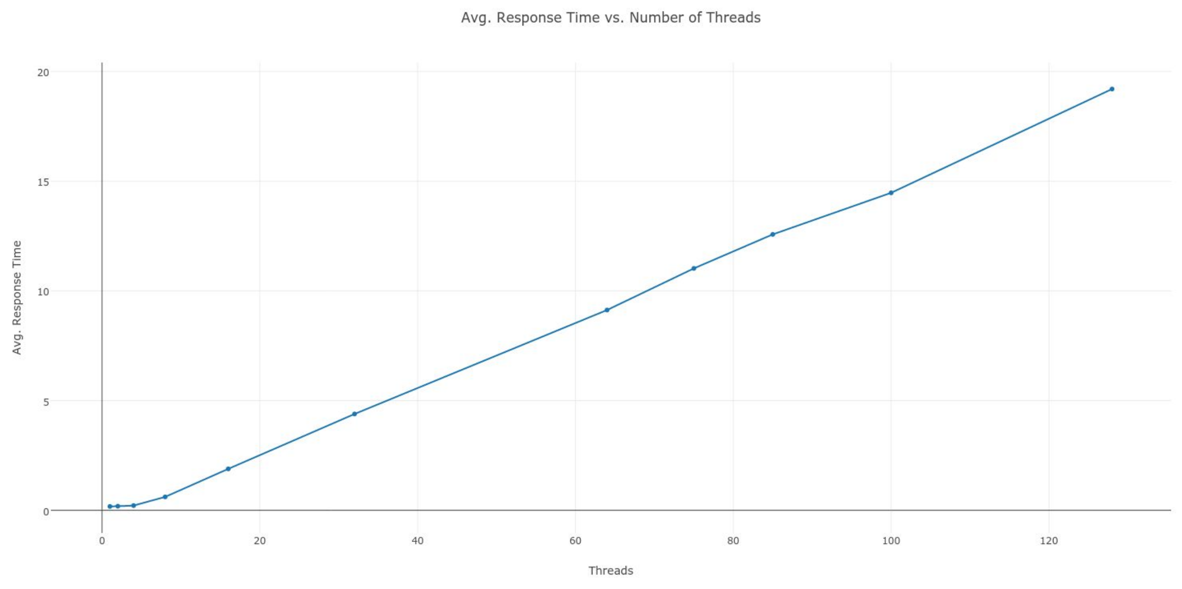
b) The graphs are attached above. The first graph is Throughput versus #Threads.
The second graph is Average Response Time versus #Threads.
The graph flattens out due to two conflicting factors. As the number of clients increases the server spawns a larger number of processes, and thus a larger fraction of the server's CPU time is used in servicing clients. However, after a point the waiting time for each request becomes very large as a each process must wait for disk reads and to use the network. Thus once the network (or disk) bottleneck is reached, the throughput does not perceptibly increase further.

c) The bottleneck resource in this case was the network. To know for sure that the network was the bottleneck resource, we observe that the bandwidth obtained from the transfer is almost the same as the maximum bandwidth as measured by iperf. We ran iostat -dx on the server as well to confirm that the disk was not the bottleneck.

d)The server throughput at saturation is 5.867 req/s. This corresponds to a network load of 93.8 Mbps. It is thus consistent with the ~6 req/s predicted by our measured bandwidth of 95 Mbps.

3)





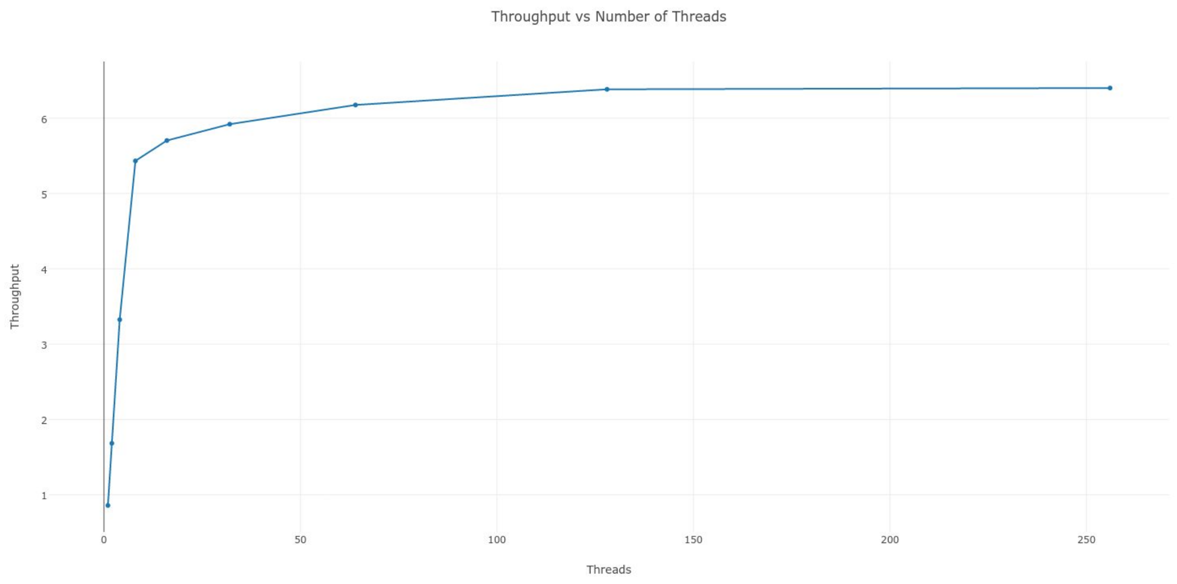
a) The optimal value of n in this case is 64. We can observe that the graph practically flattens out after n=64. Here the sleep time of 1 results in less load on the server for the same n, and thus the optimal n to push the server to its limit is higher.
At very large values of n the actual experiment time significantly overshoots the actual experiment time, since we have to wait for all the started downloads to complete. This results in a slight increase in perceived throughput due to this underestimation of the run time.

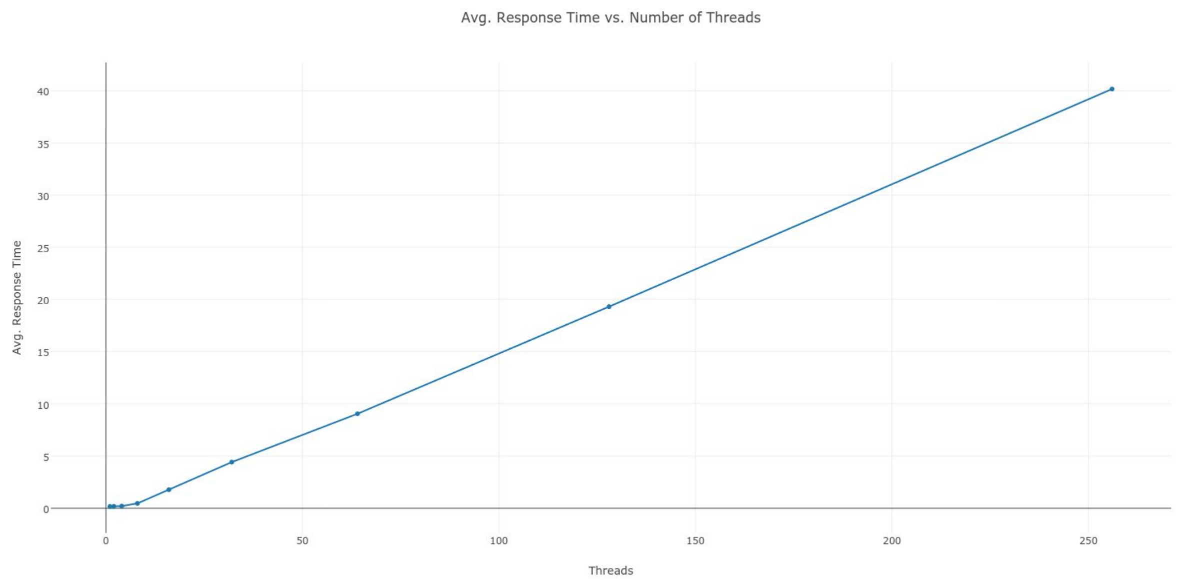
b) The graphs are attached above. The first graph is Throughput versus #Threads.
The second graph is Average Response Time versus #Threads.
The graph flattens out due to two conflicting factors. As the number of clients increases the server spawns a larger number of processes, and thus a larger fraction of the server's CPU time is used in servicing clients. However, after a point the waiting time for each request becomes very large as a each process must wait for disk reads and to use the network. Thus once the network (or disk) bottleneck is reached, the throughput does not perceptibly increase further.

c) The bottleneck resource in this case was the network. To know for sure that the network was the bottleneck resource, we observe that the bandwidth obtained from the transfer is almost the same as the maximum bandwidth as measured by iperf. We ran iostat -dx on the server as well to confirm that the disk was not the bottleneck.

d)The server throughput at saturation is 6.175 req/s. This corresponds to a network load of 98.8 Mbps. It is thus consistent with the ~6 req/s predicted by our measured bandwidth of 95 Mbps. As explained in part a, this value slightly overshoots what is expected, due to our systematic underestimation of the runtime.

4)





a) The optimal value of n in this case is 64. We can observe that the graph practically flattens out after n=64. The difference of this part from 3 is that there is significantly less load on the disk as the fixed file we repeatedly download every experiment gets cached. However this difference is not visible as the network (and not the disk) is the bottleneck in our setup. On a disk imited system, drastic differences may be visible.
At very large values of n the actual experiment time significantly overshoots the actual experiment time, since we have to wait for all the started downloads to complete. This results in a slight increase in perceived throughput due to this underestimation of the run time.

b) The graphs are attached above. The first graph is Throughput versus #Threads.
The second graph is Average Response Time versus #Threads.
The graph flattens out due to two conflicting factors. As the number of clients increases the server spawns a larger number of processes, and thus a larger fraction of the server's CPU time is used in servicing clients. However, after a point the waiting time for each request becomes very large as a each process must wait for disk reads and to use the network. Thus once the network (or disk) bottleneck is reached, the throughput does not perceptibly increase further.

c) The bottleneck resource in this case was the network. To know for sure that the network was the bottleneck resource, we observe that the bandwidth obtained from the transfer is almost the same as the maximum bandwidth as measured by iperf. We ran iostat -dx on the server as well to confirm that the disk was not the bottleneck.

d)The server throughput at saturation is 6.17 req/s. This corresponds to a network load of 98.72 Mbps. It is thus consistent with the ~6 req/s predicted by our measured bandwidth of 95 Mbps. As explained in part a, this value slightly overshoots what is expected, due to our systematic underestimation of the runtime.