130050027 (Anmol Arora), 130050041 (Aman Goel)
Lab 8 report

Part 1:

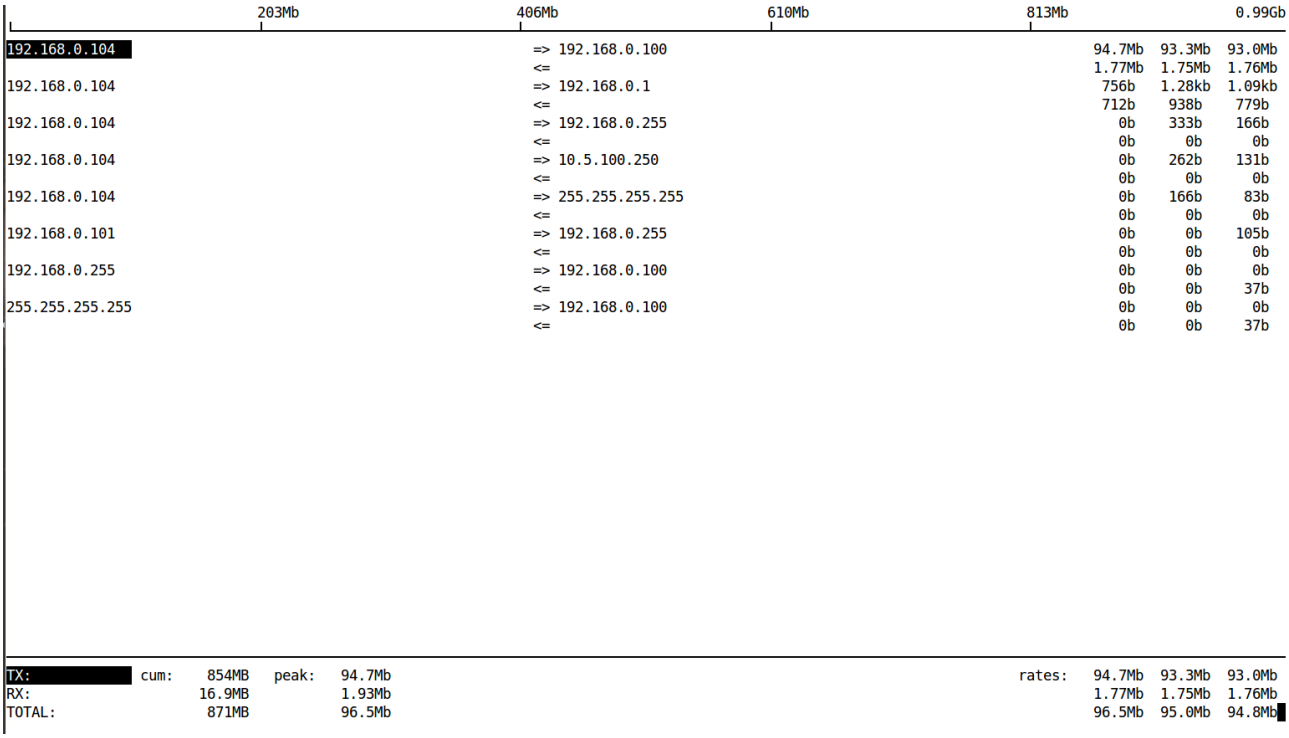
The maximum throughput observed is **5.879** files/sec, which is observed for the first time at **N=2**. Thus the number of worker threads at the server to fully saturate the server is **2** {The difference in throughput at N=1 and N=2 is insignificant, owing due to small calculation errors, hence for practical purposes the saturation is reached at N=1 itself}.
The load created by 20 users is small enough to be handled by a couple of server threads, hence the saturation is reached pretty quickly.The bottleneck resource here is the network bandwidth as apprent on running iftop, top and iostat:

```
top - 11:24:27 up 3 days, 19:23,  3 users,  load average: 0.80, 0.61, 0.50
Tasks: 256 total,   2 running, 254 sleeping,   0 stopped,   0 zombie
%Cpu(s): 13.6 us,  0.8 sy,  0.0 ni, 83.1 id,  1.4 wa,  0.0 hi,  1.1 si,  0.0 st
KiB Mem:   8054896 total,  7868552 used,   186344 free,   119396 buffers
KiB Swap:        0 total,        0 used,        0 free.  3204744 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
30163 root      20   0  457288   1172    964 S  59.5  0.0   0:56.29 server-mt
 2406 mewtomat  20   0 2276544 443064  73140 S   0.7  5.5  61:18.53 chrome
    8 root      20   0       0      0      0 S   0.3  0.0   0:45.15 rcuos/0
 1396 root      20   0  523180 176576 160940 S   0.3  2.2  39:39.32 Xorg
```

This shows CPU is not the bottleneck.

```
mewtomat@mewtomat-Inspiron-5737:~$ iostat -d 1 -m
Linux 3.13.0-68-generic (mewtomat-Inspiron-5737)        Thursday 10 March 2016

Device:            tps    MB_read/s    MB_wrtn/s    MB_read    MB_wrtn
sda               3.75         0.03         0.05      10433      16206

Device:            tps    MB_read/s    MB_wrtn/s    MB_read    MB_wrtn
sda               0.00         0.00         0.00          0          0

Device:            tps    MB_read/s    MB_wrtn/s    MB_read    MB_wrtn
sda               0.00         0.00         0.00          0          0

Device:            tps    MB_read/s    MB_wrtn/s    MB_read    MB_wrtn
sda               0.00         0.00         0.00          0          0

Device:            tps    MB_read/s    MB_wrtn/s    MB_read    MB_wrtn
sda               0.00         0.00         0.00          0          0
```

This shows disk is not the bottleneck.



This shows the network is the bottleneck (The average total rate, on lower right corner{96.5Mbps} is close to the peak value of link bandwidth).

The throughput measurements for part 1 are as follows:

N(#worker threads)	Throughput(Files served/sec)
1	5.878
2	5.879
3	5.879
4	5.879
5	5.879
6	5.879
7	5.879
8	5.879
9	5.879
10	5.879

Part 2:

When the experiment is run for large number of users, it is observed that many clients are denied service. This is reflected by the error message “Error: Connection Timed Out” being printed multiple times at the client. This is happening because the client program is sending a large number of requests at a rate which is greater than the rate of servicing at the server. As a result the request queue at server gets full and it drops any more requests from the client. The clients correstponding to these dropped requests wait for some time, hoping to get a reply from server, but evenutally get timed out and print the above mentioned error. These are the client threads which are being denied service from server.

On measuring the throughput values for each N from 1 to 10, it is observed that initially the throughput increases but then it decreases as N is increased further.

The increase in throughput can be explained from the fact that with greater number of worker threads, the requests in queue get served quickly, enabling more and more pending requests to get accepted and subsequently getting served, giving higher throughput.

The decrease in throughput for further higher values of N is because of contention among threads for the lock. The threads acquire lock at various points of time in their runtime. At a time only one of the threads can hold a lock. But when there are many threads waiting for the same lock, the whole program starts incurring significant amounts of overheads in lock acquiring. Thus the latency of serving the requests starts increasing at a higher rate than the gain in throughput through increased successful requests, resulting in overall throughput decrease for large N.

Measurements for part 2 are as follows:

n = 1
Total Successful Requests: 620
Throughput: 2.846

n = 2
Total Successful Requests: 691
Throughput: 3.509

n = 3
Total Successful Requests: 747
Throughput: 4.086

n = 4
Total Successful Requests: 816
Throughput: 4.282

n = 5
Total Successful Requests: 812
Throughput: 4.364

n = 6
Total Successful Requests: 812
Throughput: 4.533

n = 7
Total Successful Requests: 803
Throughput: 4.435

n = 8
Total Successful Requests: 838

Throughput: 4.191

n = 9
Total Successful Requests: 811
Throughput: 4.004

n = 10
Total Successful Requests: 836
Throughput: 3.674

Plot of Throughput vs N for Part 2