ECE 456
Lab 07

Pablo Corona Alvarez

IOzone:

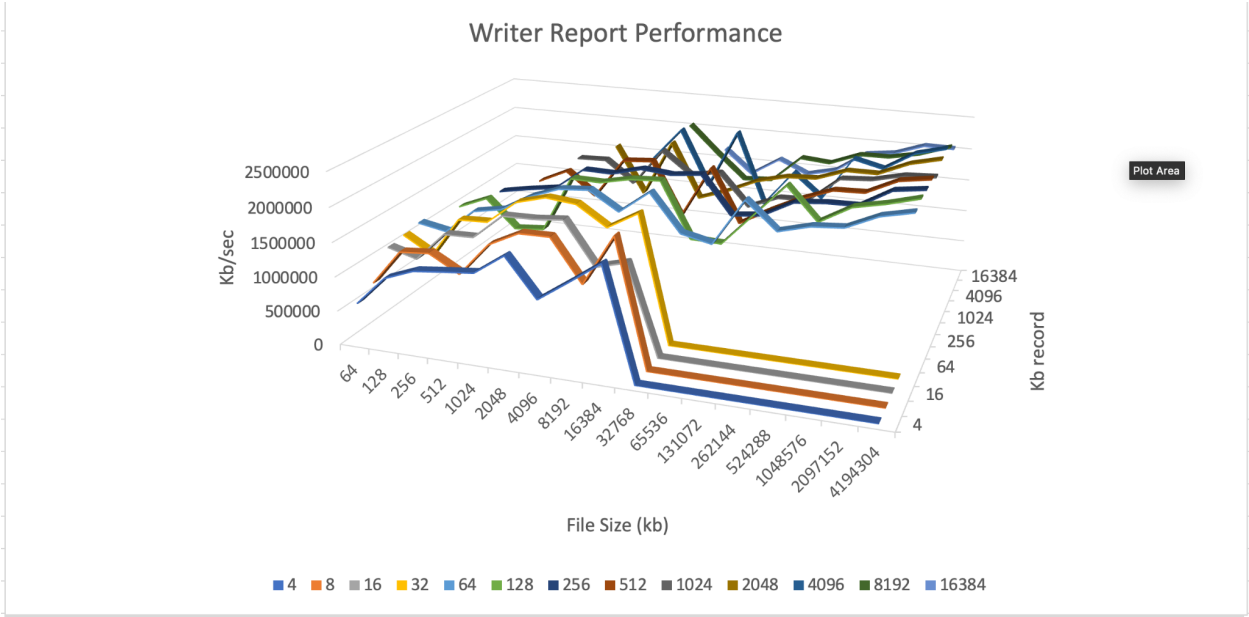

Figure 1: Writer Report Perfomance

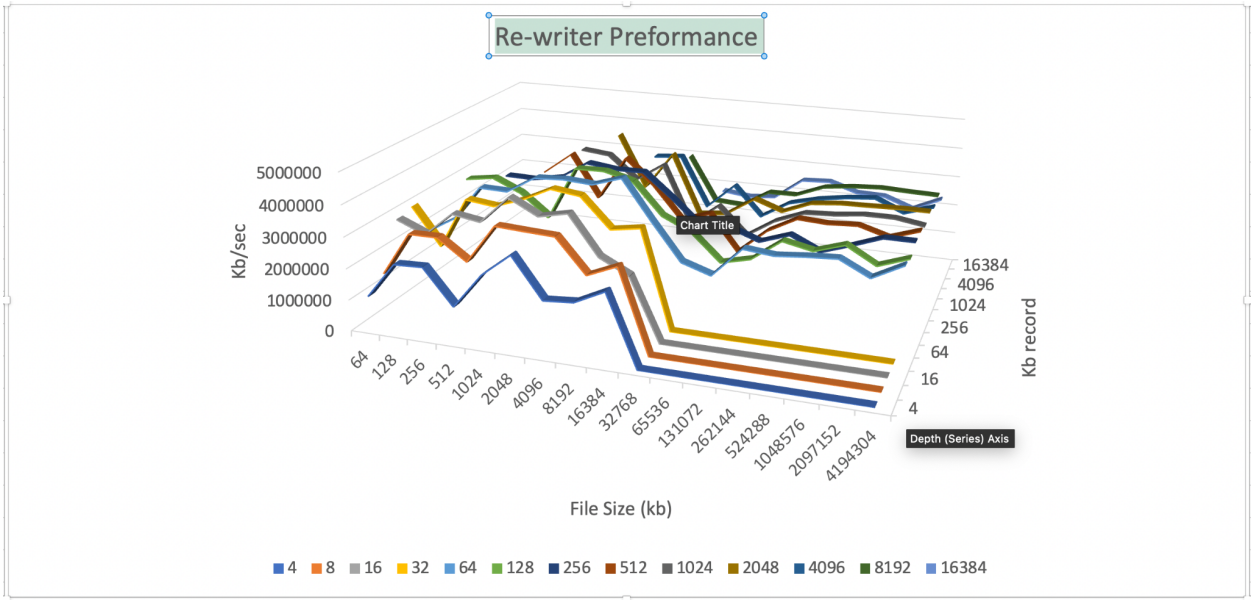

Figure 2: Re-writer Performance

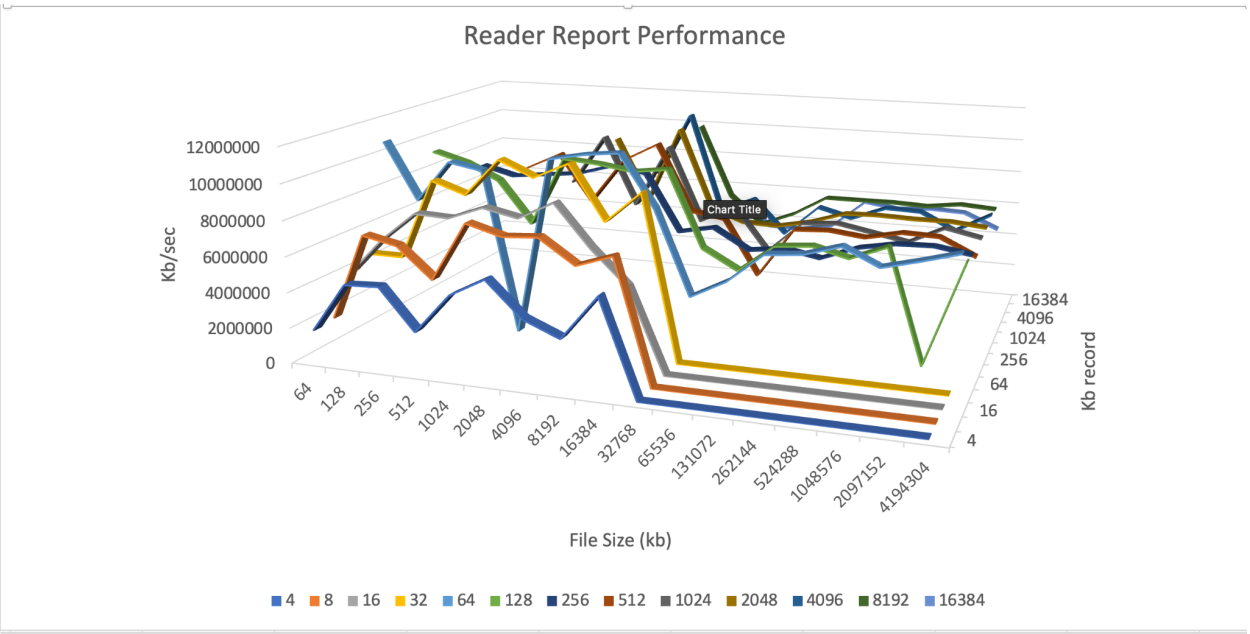

Figure 3: Reader Report Performance

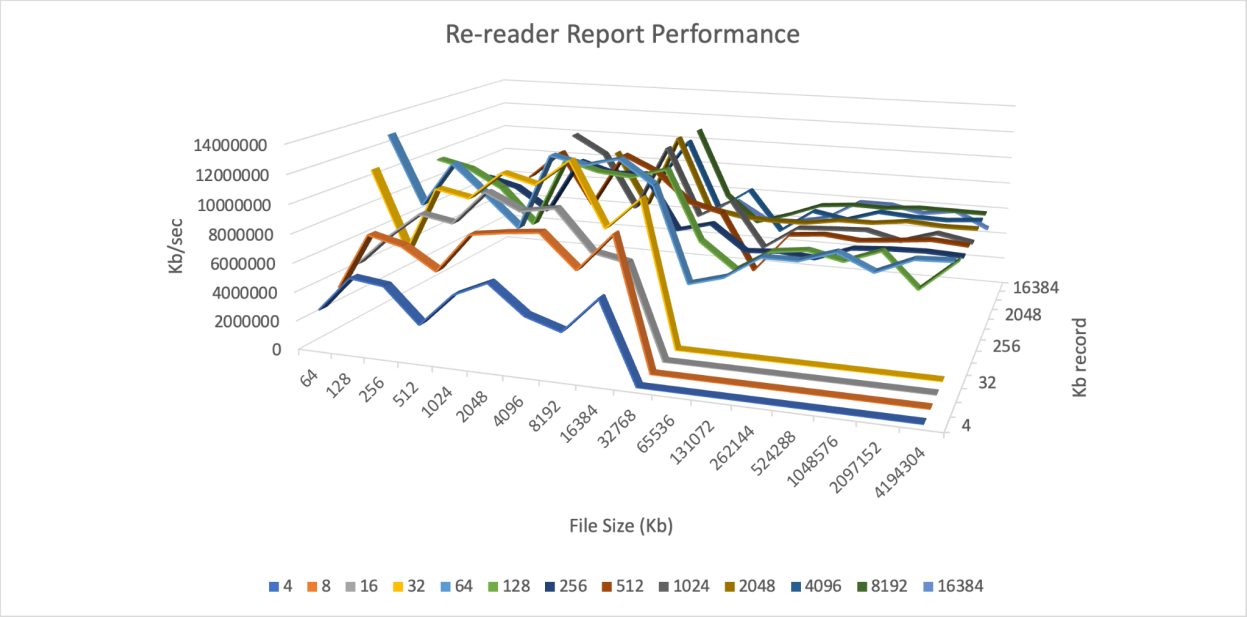

Figure 4: Re-reader Report Performance

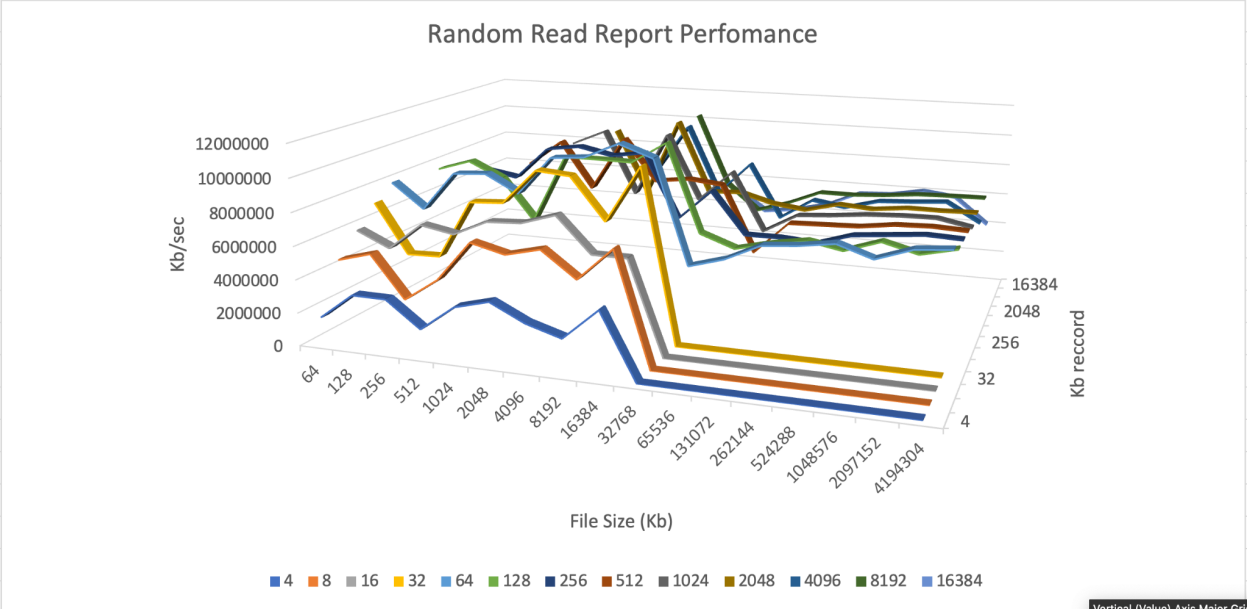

Figure 5: Random Read Report Performance

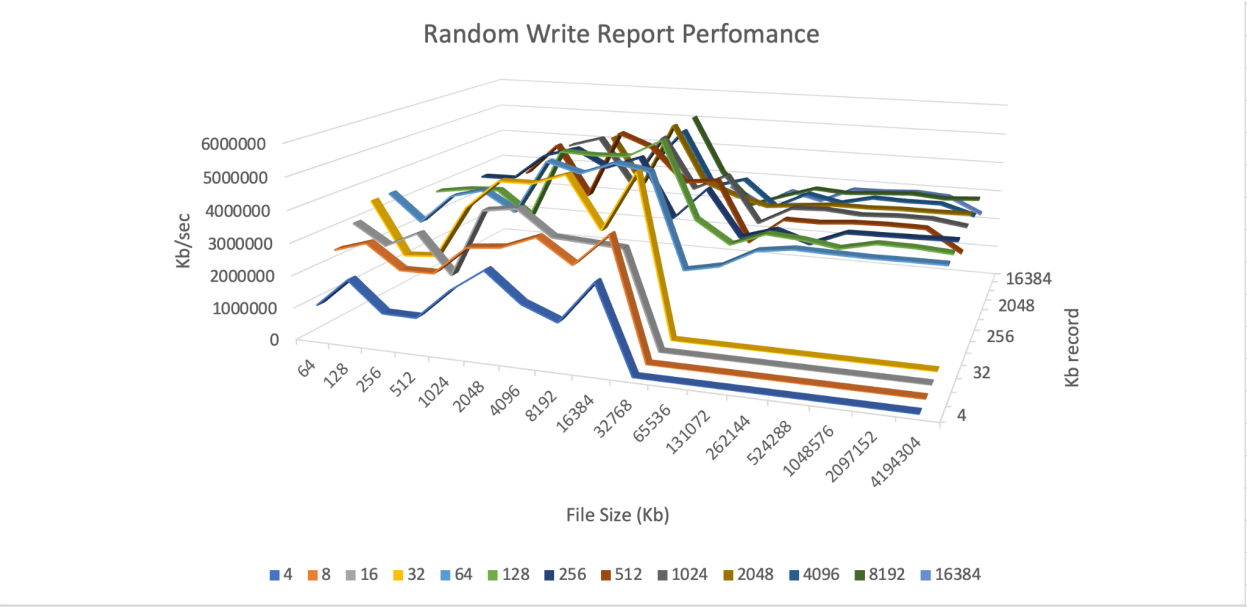

Figure 6: Random Write Report Performance

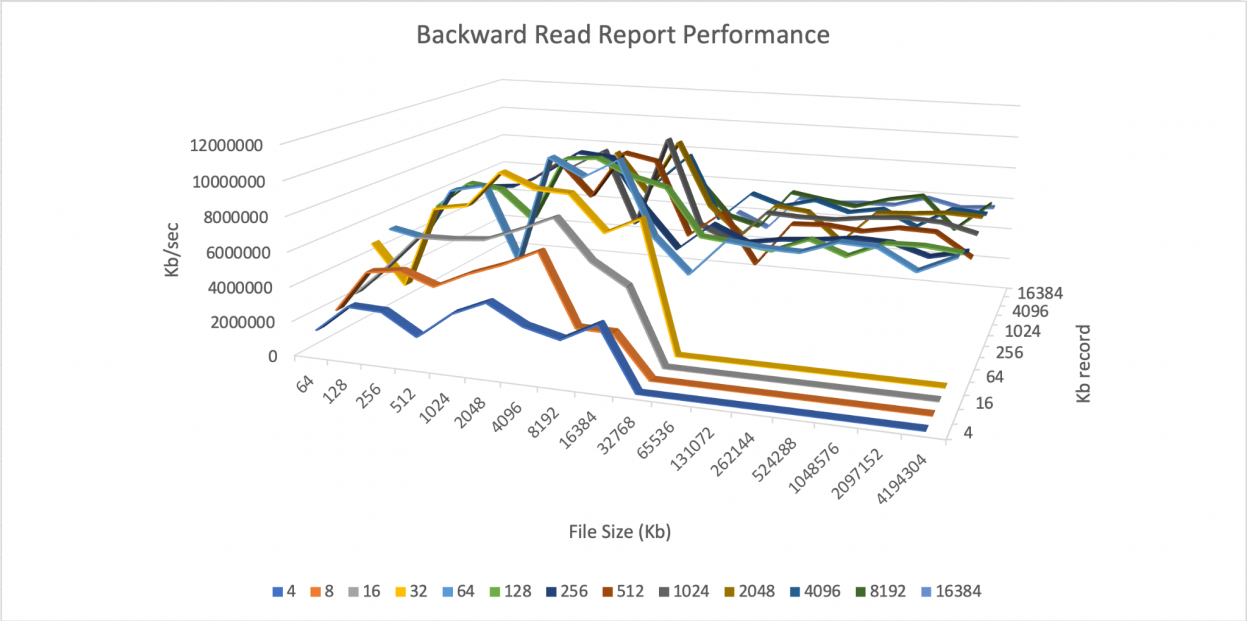

Figure 7: Backward Read Report Performance

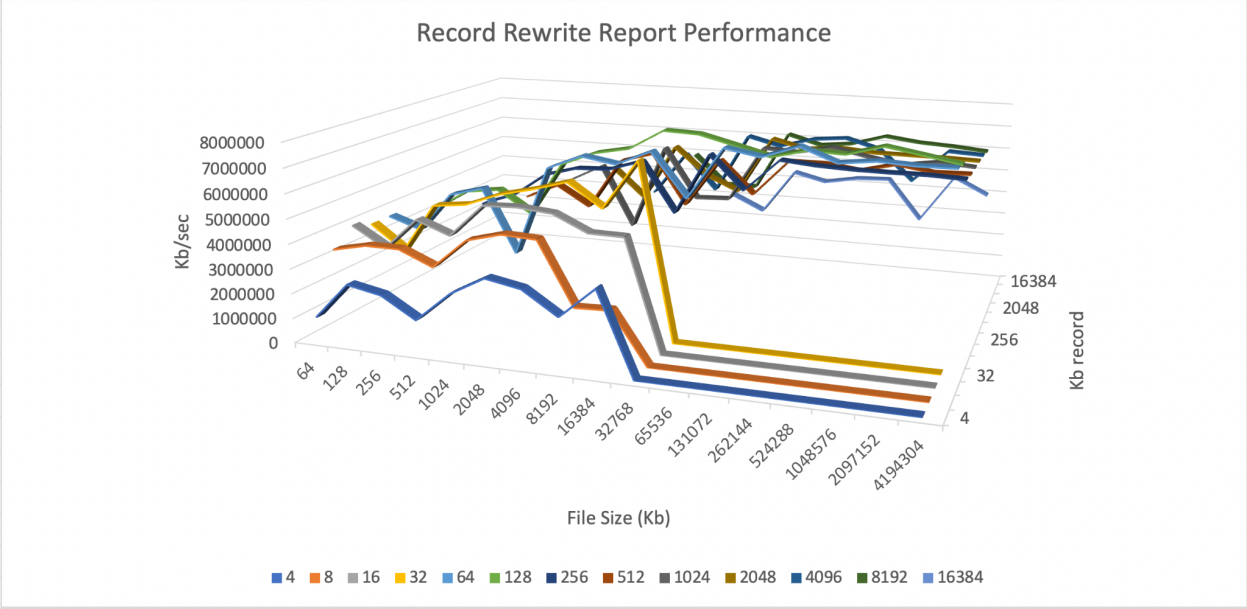

Figure 8: Record Rewrite Report Performance

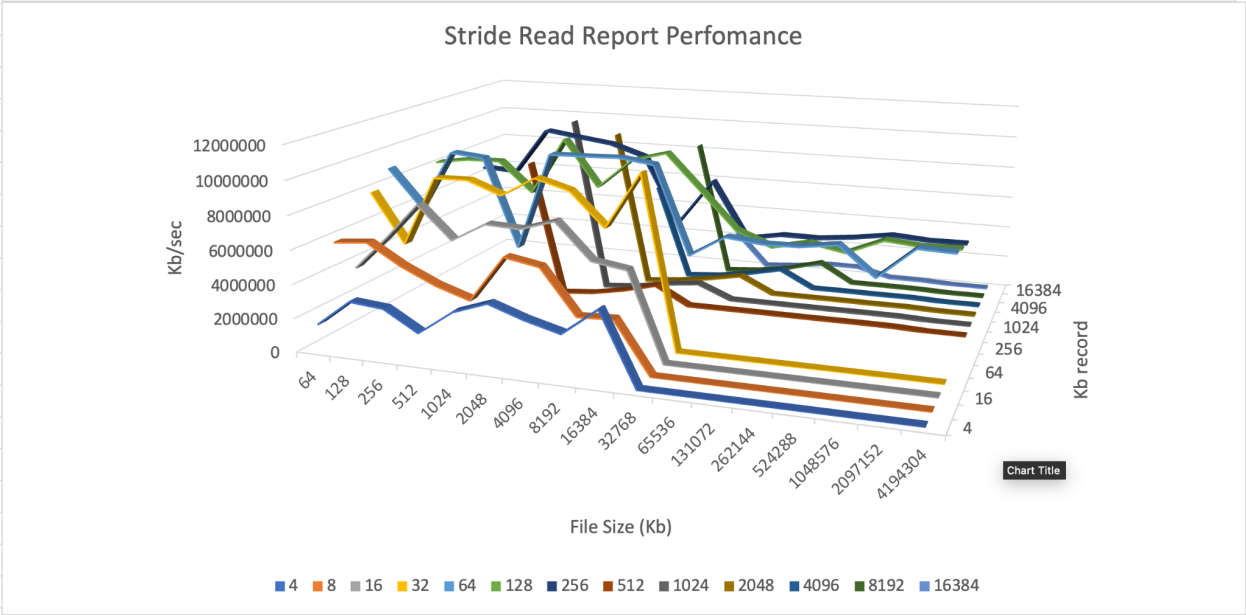

Figure 9: Stride Read Report Performance

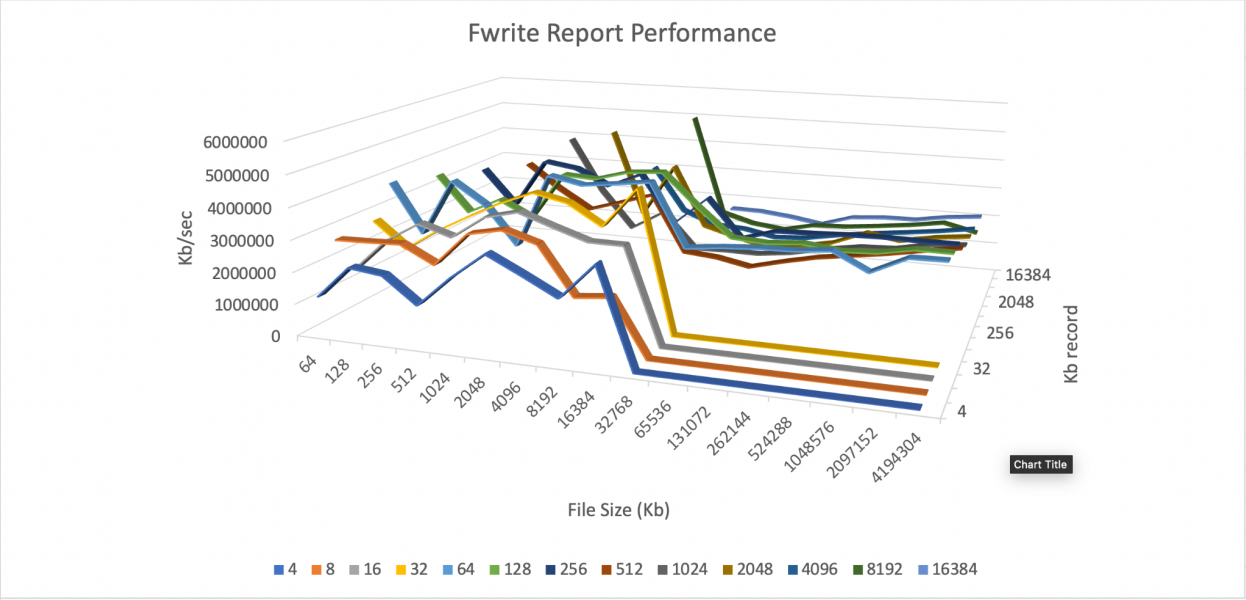

Figure 10: FWrite Report Performance

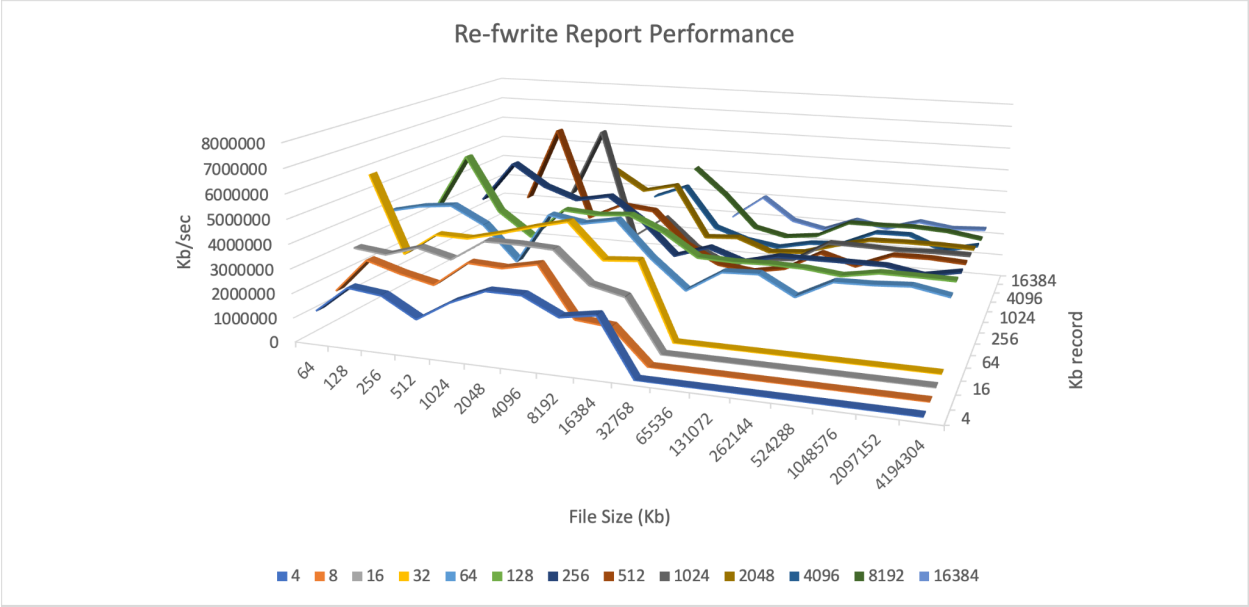

Figure 11: Re-fwrite Report Performance

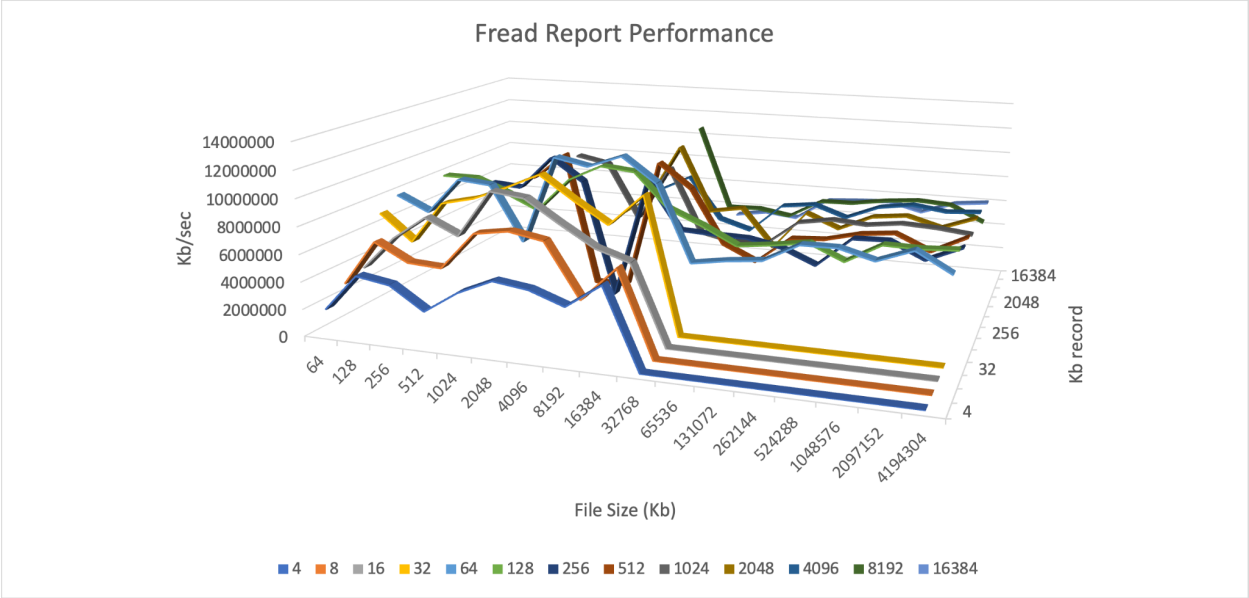

Figure 12: Fread Report Performance

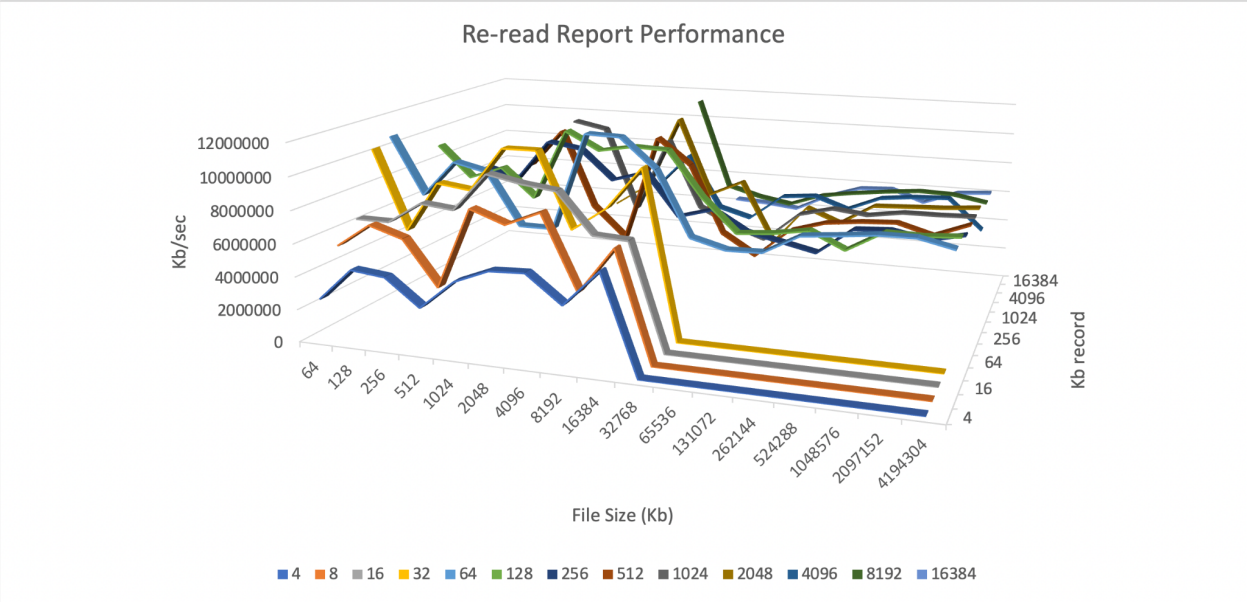

Figure 13: Re-Read Report Performance

Looking at all of the Reports that were generated from IOzone there is one thing that I noticed many had in common. All of the Reports seem to have dips in performance around the same file sizes. Particularly whenever the file size was 64, 512 and 8192. Though this didn't happen in all the scenarios and test that it ran i found it interesting that regardless of the test and Kb record they experience the dip in performance at those file sizes. This makes me believe that certain file size are more optimal for transferring data than others. The other thing that is almost a universal trend among all reports is that the performance is almost the same universally for all tests except for the few outliers Being record rewrite performance and writer performance. We also see that as we add more Kb records that the overall performance decreases in most cases when going to larger file sizes.

Netperf:

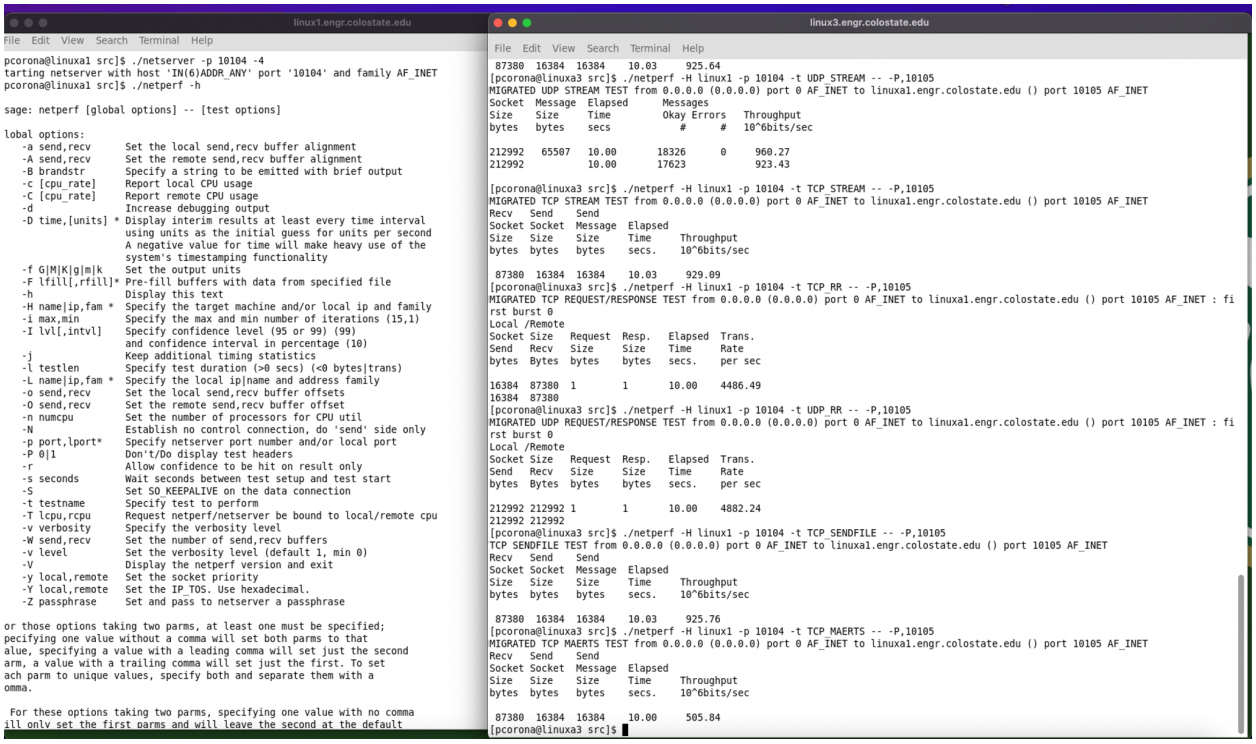

Figure 14: netperf Server

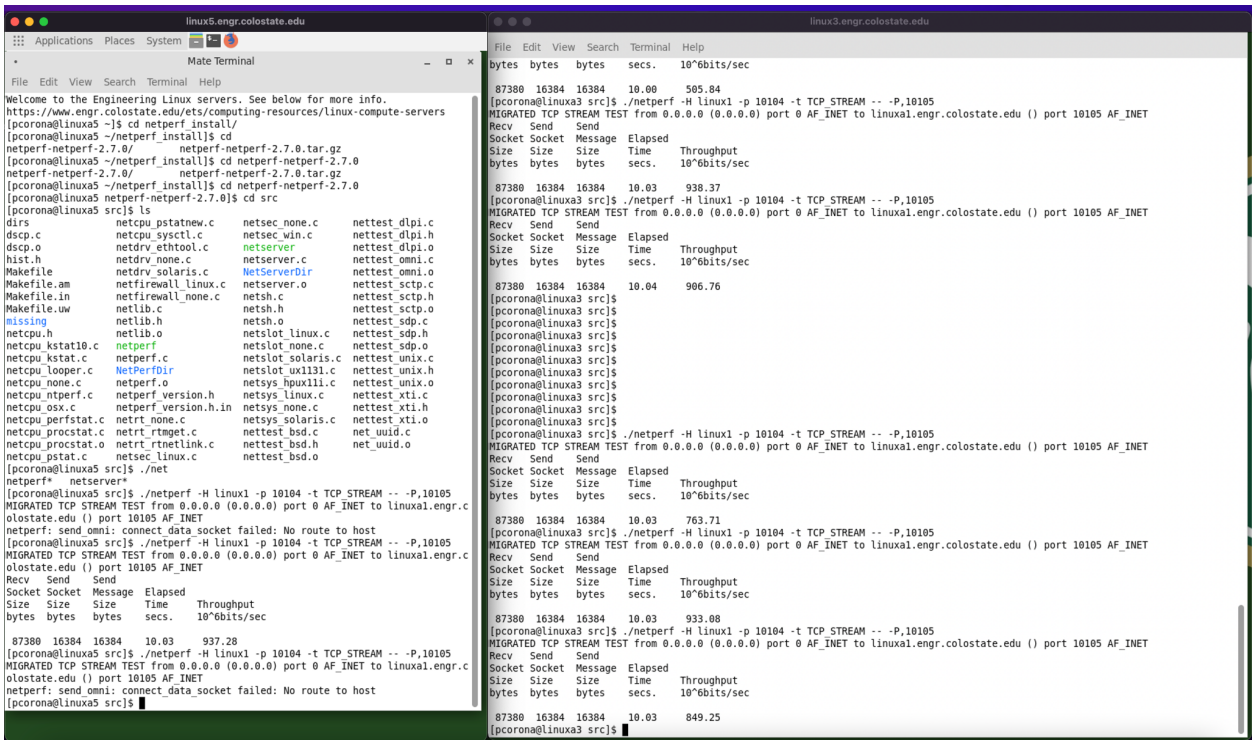

Figure 15: Net perf Client

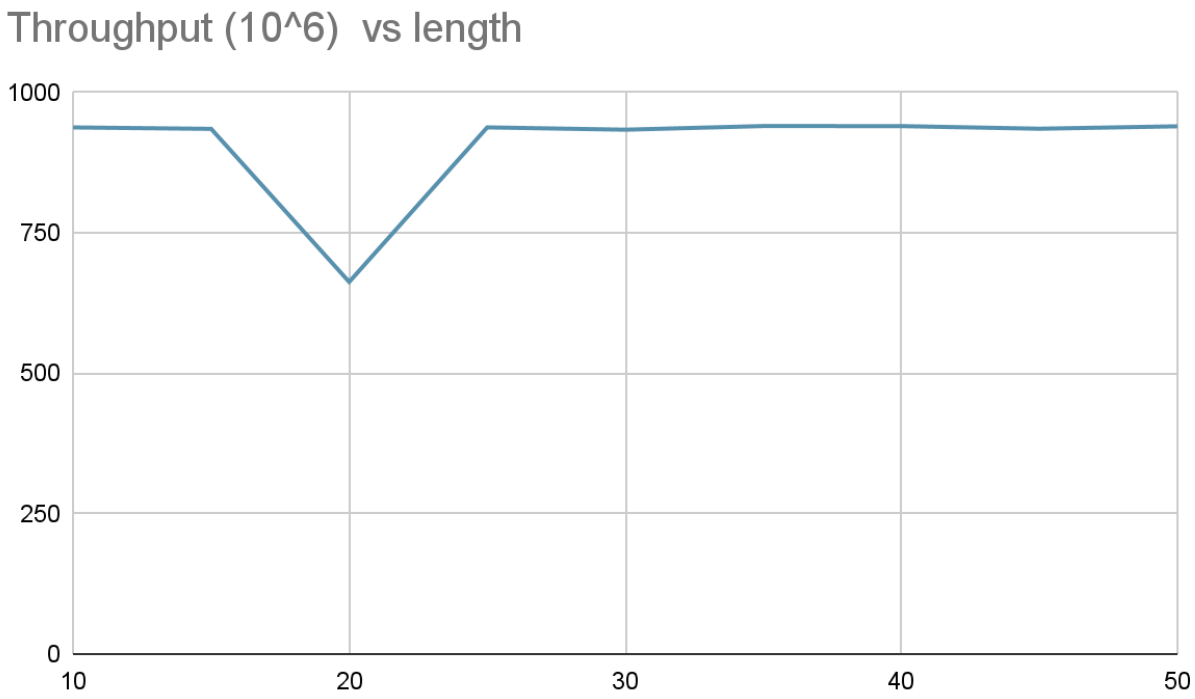

Figure 16: Netperf Throughput vs length

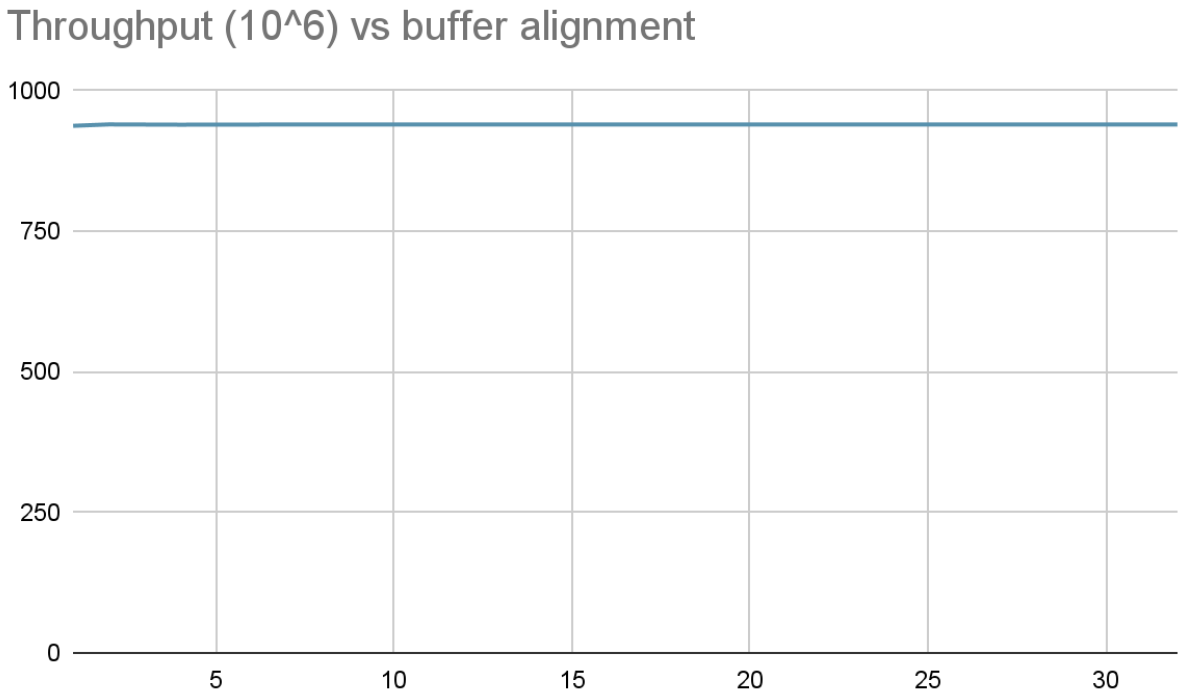

Figure 17: Netperf Throughput vs buffer alignment

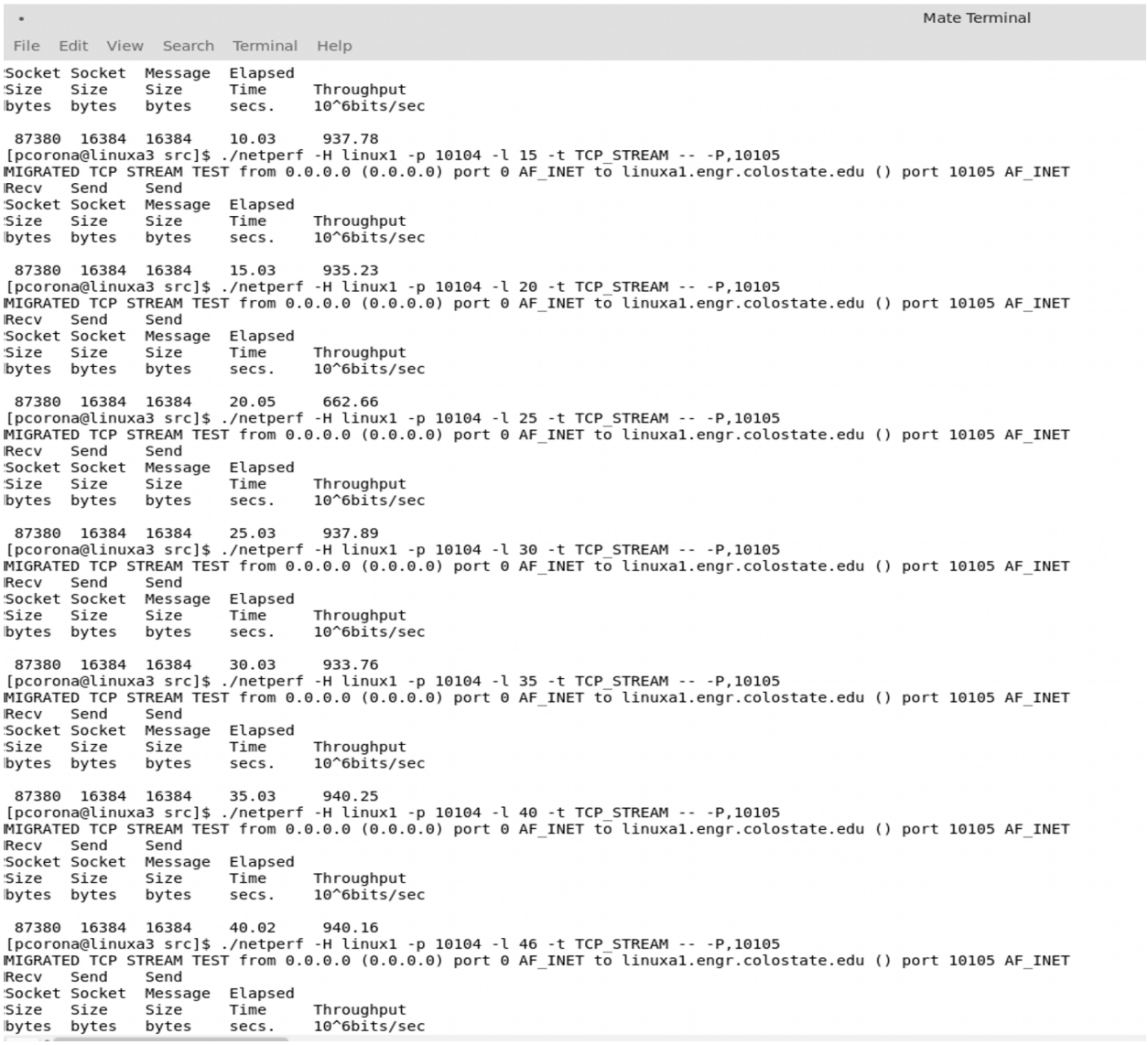

Figure 18: Netperf Throughput by varying -l

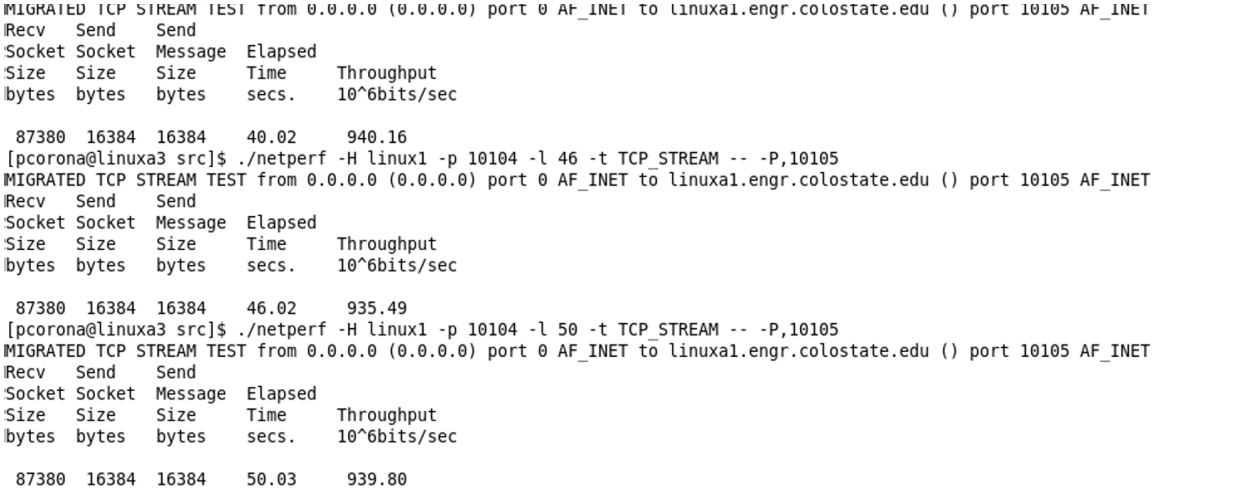

Figure 19:Netperf Throughput by varying -l part 2

```
[pcorona@linuxa3 src]$
[pcorona@linuxa3 src]$
[pcorona@linuxa3 src]$ ./netperf -H linux1 -p 10104 -a 1 -t TCP_STREAM -- -P,10105
MIGRATED TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to linuxa1.engr.colostate.edu () port 10105 AF_INET
Recv   Send    Send
Socket Socket  Message  Elapsed
Size   Size    Size     Time     Throughput
bytes  bytes   bytes    secs.    10^6bits/sec

 87380  16384  16384    10.03     937.50
[pcorona@linuxa3 src]$ ./netperf -H linux1 -p 10104 -a 2 -t TCP_STREAM -- -P,10105
MIGRATED TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to linuxa1.engr.colostate.edu () port 10105 AF_INET
Recv   Send    Send
Socket Socket  Message  Elapsed
Size   Size    Size     Time     Throughput
bytes  bytes   bytes    secs.    10^6bits/sec

 87380  16384  16384    10.02     940.01
[pcorona@linuxa3 src]$ ./netperf -H linux1 -p 10104 -a 4 -t TCP_STREAM -- -P,10105
MIGRATED TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to linuxa1.engr.colostate.edu () port 10105 AF_INET
Recv   Send    Send
Socket Socket  Message  Elapsed
Size   Size    Size     Time     Throughput
bytes  bytes   bytes    secs.    10^6bits/sec

 87380  16384  16384    10.03     939.49
[pcorona@linuxa3 src]$ ./netperf -H linux1 -p 10104 -a 8 -t TCP_STREAM -- -P,10105
MIGRATED TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to linuxa1.engr.colostate.edu () port 10105 AF_INET
Recv   Send    Send
Socket Socket  Message  Elapsed
Size   Size    Size     Time     Throughput
bytes  bytes   bytes    secs.    10^6bits/sec

 87380  16384  16384    10.02     939.94
[pcorona@linuxa3 src]$ ./netperf -H linux1 -p 10104 -a 16 -t TCP_STREAM -- -P,10105
MIGRATED TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to linuxa1.engr.colostate.edu () port 10105 AF_INET
Recv   Send    Send
Socket Socket  Message  Elapsed
Size   Size    Size     Time     Throughput
bytes  bytes   bytes    secs.    10^6bits/sec

 87380  16384  16384    10.03     939.82
[pcorona@linuxa3 src]$ ./netperf -H linux1 -p 10104 -a 32 -t TCP_STREAM -- -P,10105
MIGRATED TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to linuxa1.engr.colostate.edu () port 10105 AF_INET
Recv   Send    Send
Socket Socket  Message  Elapsed
Size   Size    Size     Time     Throughput
bytes  bytes   bytes    secs.    10^6bits/sec

 87380  16384  16384    10.02     939.88
[pcorona@linuxa3 src]$
```

Figure 20: Netperf Throughput by varying -a

One of the things that I came to notice about netperf is that it doesn't really matter which test is being run TCP or UDP, in either case they are about the same in the rate of throughput. The thing that made a difference in the throughput was the test that was being run. Interestingly enough we can see that both the buffer alignment and pause for a certain amount of running options don't affect the throughput of the test being run, interestingly only one of the pauses affected the throughput. The one thing which reliably brought the throughput down was attempting to use the same port and server from another machine. While this would fail on the new machine it reduced the throughput of the test which was running

TTCP:

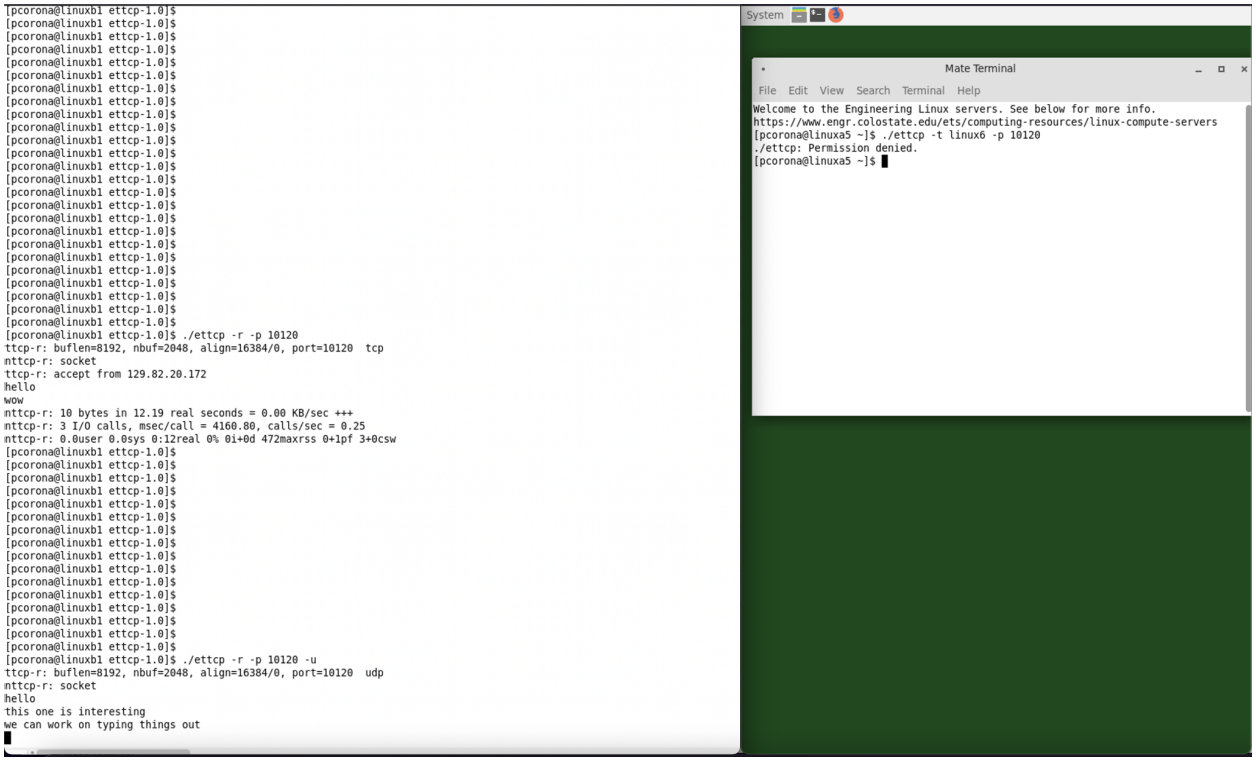

Figure 21: TTCP Connection with TCP and UDP

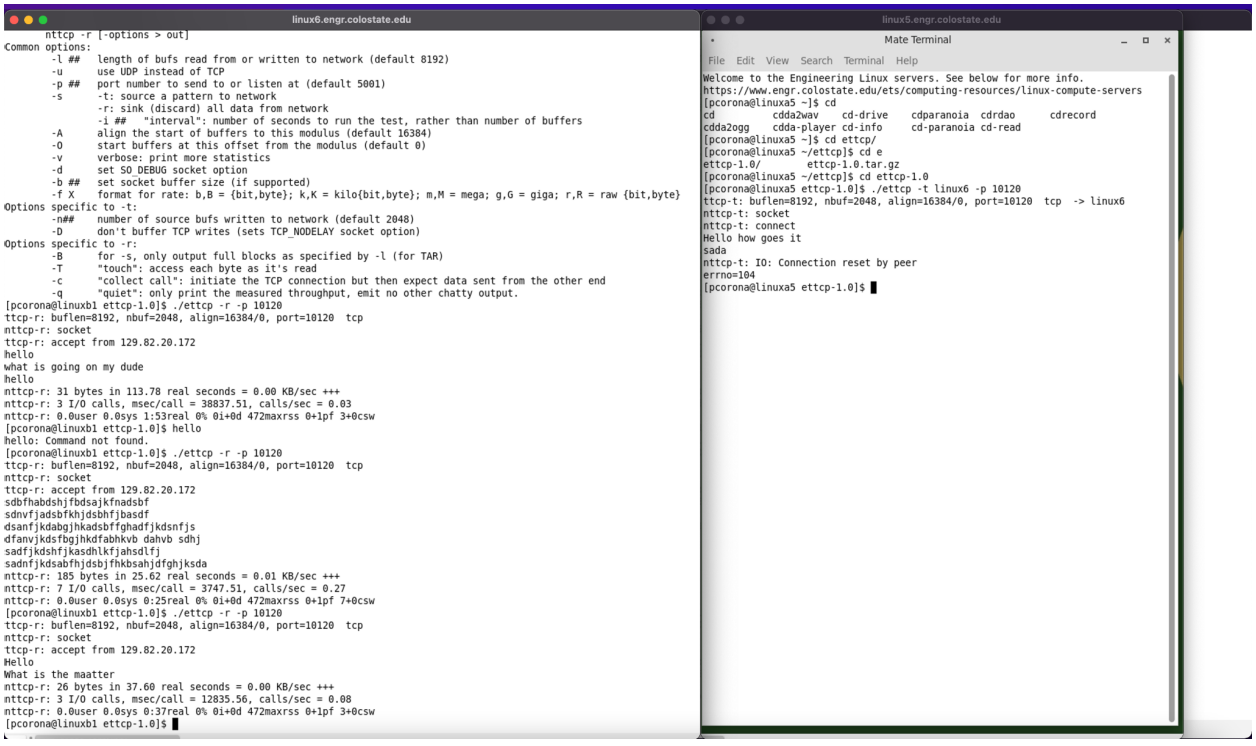

Figure 22: TTCP Connection fails with two TCP connections

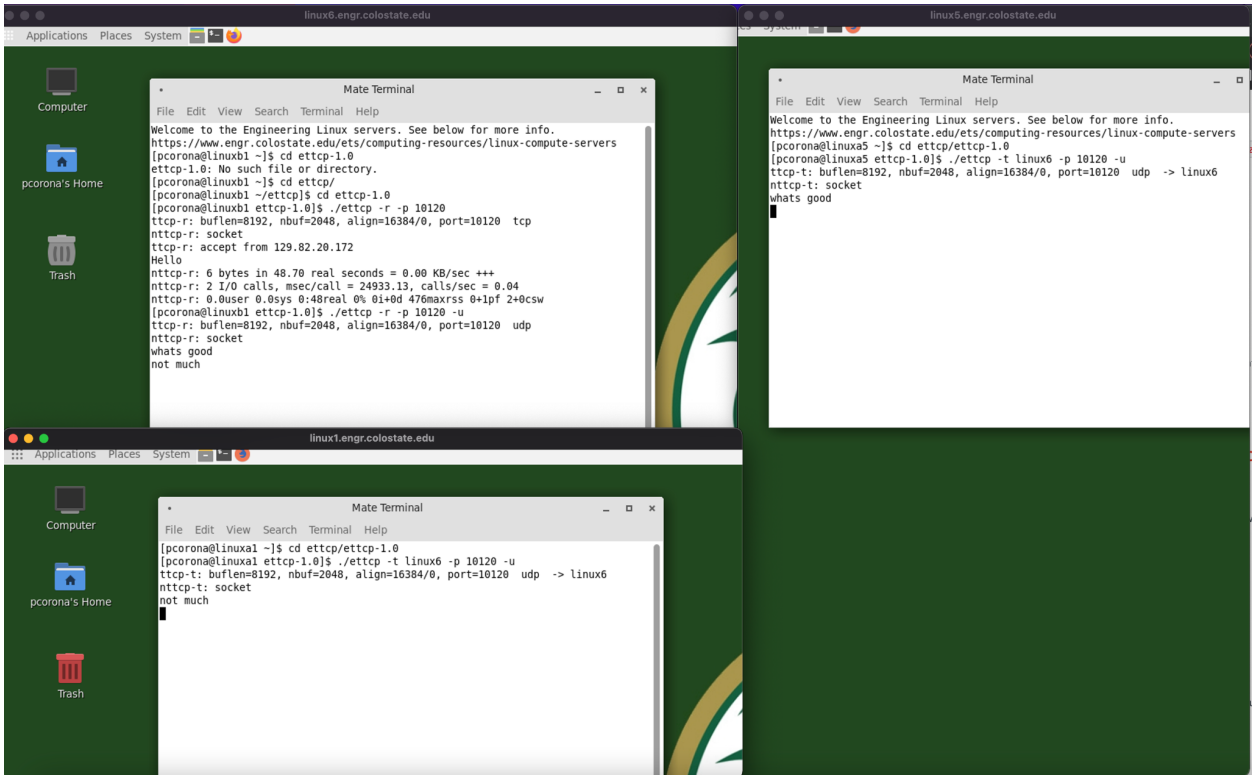

Figure 23: TTCP UDP functioning with UDP two client

The Things that I found most notable about running the tests for TTCP was that it was relatively similar to some of the programs which we have coded. It essentially works as a message board and is sending strings which are then displayed on the server. When this was run the thing in particular that I found interesting was that TCP was incapable of having two connections at the same time, however, attempting to connect from another client did not seem to have any kind of effect on the overall performance of the system. The client which tried to connect to the server while it was busy simply could not connect and would have its connection reset once the other client finished. UDP on the other hand was able to have multiple clients sending information and messages at the same time.

Iperf:

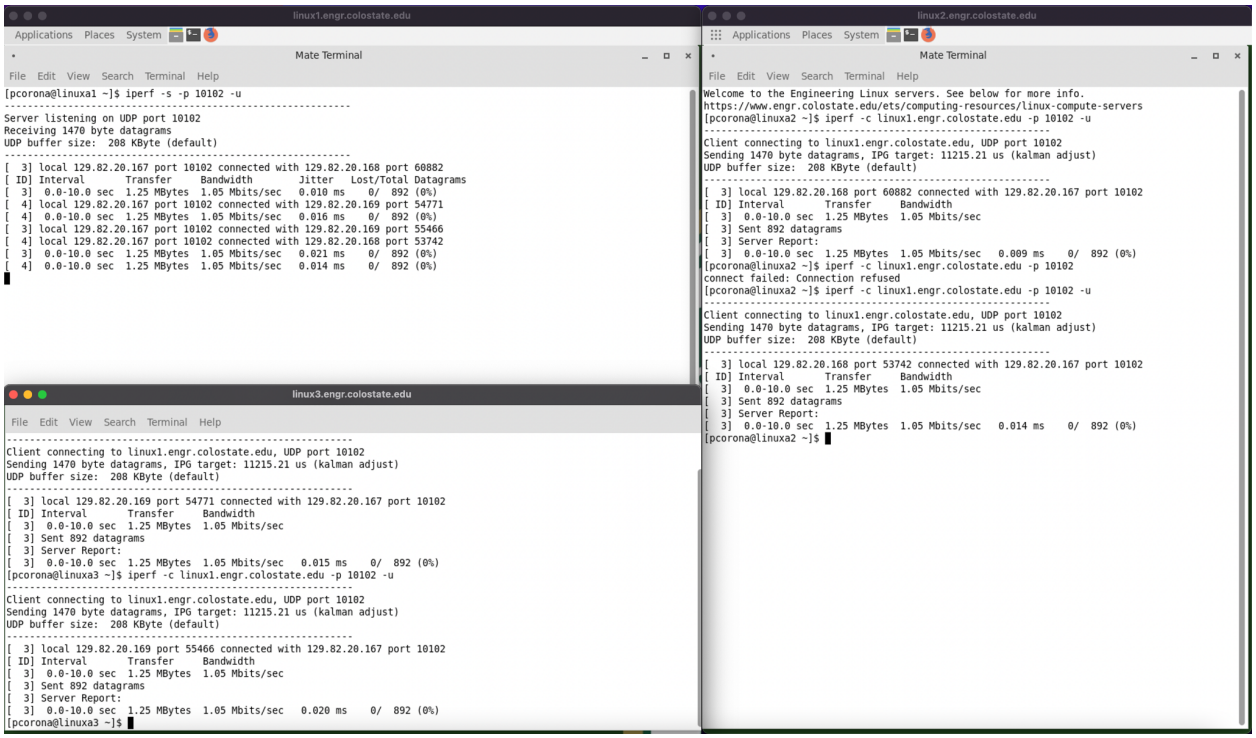

Figure 24: Iperf UDP Server connection with two clients

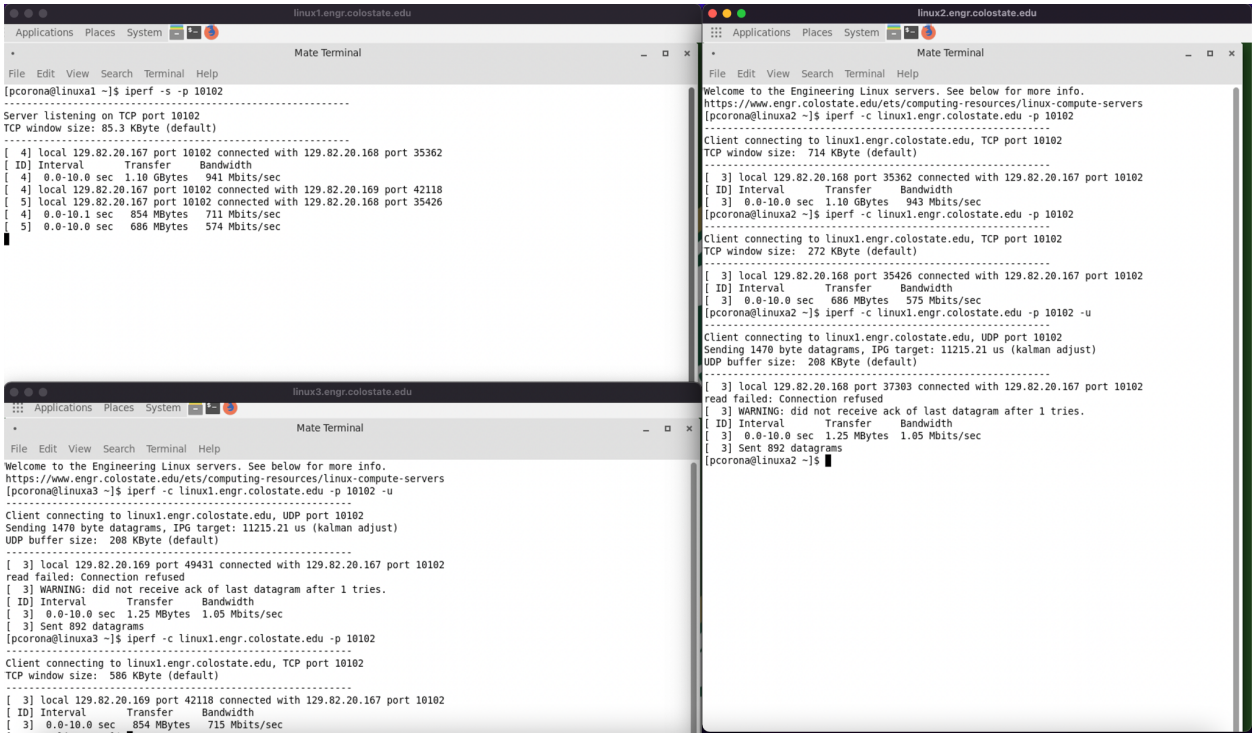

Figure 25: Iperf TCP Server connection with client

One of the more interesting things that I noticed when using Iperf to test the bandwidth of the connection between the server and client was that the UDP connection did not experience much performance degradation when two clients connected at the same time and remained at a relatively similar performance as when only one connection was present. However this is not the case with TCP. While Iperf can handle two TCP connections at the same time it does experience some performance degradation when there were two connections. While the first client experiences a decent performance losing about 20% the second client experiences a drastic performance loss of about 40%.