Problem 1

MAC Addresses

We get the interface config of the pod3-3 machine:

```
pod3-3 51 $ ifconfig -a
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 128.10.25.213  netmask 255.255.255.0  broadcast 128.10.25.255
        inet6 fe80::16b3:1fff:fe02:3e08  prefixlen 64  scopeid 0x20<link>
        ether 14:b3:1f:02:3e:08  txqueuelen 1000  (Ethernet)
        RX packets 893518410  bytes 1018870968114 (1.0 TB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 363304891  bytes 127058012388 (127.0 GB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 16  memory 0xf7200000-f7220000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 3279754  bytes 727661013 (727.6 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3279754  bytes 727661013 (727.6 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

veth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.1  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::60b5:6dff:fea1:dbb  prefixlen 64  scopeid 0x20<link>
        ether 62:b5:6d:a1:0d:bb  txqueuelen 1000  (Ethernet)
        RX packets 1814  bytes 120758 (120.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3478  bytes 461386 (461.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Thus, the MAC address of *veth0* used by `mypingsrv` is `62:b5:6d:a1:0d:bb`.

We get the virtual interface config of the pod3-3 machine:

```
pod3-3 52 $ veth 'ifconfig -a'
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

veth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
```

```
        inet 192.168.1.2  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::4c2:5fff:fe25:ea64  prefixlen 64  scopeid 0x20<link>
        ether 06:c2:5f:25:ea:64  txqueuelen 1000  (Ethernet)
        RX packets 3482  bytes 462238 (462.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1814  bytes 120758 (120.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Thus, the MAC address of *veth0* used by `mypingcli` is `06:c2:5f:25:ea:64`.

Traffic Logging

We use `mypingcli` to interact with `mypingsrv`. The contents of `pingparam` are:

- N: 5
- T: 2
- D: 1
- S: 0

We start the server with port number `22222` first with the following command.

```
./mypingsrv 192.168.1.1 22222
```

Then, we start to ping the server with the following command.

```
veth 'mypingcli 192.168.1.2 192.168.1.1 22222'
```

The following is the contents of `testlogfile` collected by `tcpdump`:

```
pod3-3 55 $ tcpdump -XX -n -r - < testlogfile
reading from file -, link-type EN10MB (Ethernet)
10:35:57.121253 IP 192.168.1.2.59727 > 192.168.1.1.22222: UDP, length 5
        0x0000:  62b5 6da1 0dbb 06c2 5f25 ea64 0800 4500  b.m....._%.d..E.
        0x0010:  0021 3ad5 4000 4011 7ca3 c0a8 0102 c0a8  .!:.@.@.|.......
        0x0020:  0101 e94f 56ce 000d 8372 0000 0000 01    ...OV....r.....
10:35:58.121470 IP 192.168.1.1.22222 > 192.168.1.2.59727: UDP, length 5
        0x0000:  06c2 5f25 ea64 62b5 6da1 0dbb 0800 4500  .._%.db.m.....E.
        0x0010:  0021 d963 4000 4011 de14 c0a8 0101 c0a8  .!.c@.@.........
        0x0020:  0102 56ce e94f 000d 8372 0000 0000 01    ..V..O...r.....
10:35:59.121321 IP 192.168.1.2.59727 > 192.168.1.1.22222: UDP, length 5
        0x0000:  62b5 6da1 0dbb 06c2 5f25 ea64 0800 4500  b.m....._%.d..E.
        0x0010:  0021 3b5b 4000 4011 7c1d c0a8 0102 c0a8  .!;[@.@.|.......
        0x0020:  0101 e94f 56ce 000d 8372 0100 0000 01    ...OV....r.....
10:36:00.121569 IP 192.168.1.1.22222 > 192.168.1.2.59727: UDP, length 5
        0x0000:  06c2 5f25 ea64 62b5 6da1 0dbb 0800 4500  .._%.db.m.....E.
        0x0010:  0021 db2e 4000 4011 dc49 c0a8 0101 c0a8  .!..@.@..I......
        0x0020:  0102 56ce e94f 000d 8372 0100 0000 01    ..V..O...r.....
10:36:01.121417 IP 192.168.1.2.59727 > 192.168.1.1.22222: UDP, length 5
        0x0000:  62b5 6da1 0dbb 06c2 5f25 ea64 0800 4500  b.m....._%.d..E.
        0x0010:  0021 3cb5 4000 4011 7ac3 c0a8 0102 c0a8  .!<.@.@.z.......
```

```
        0x0020:  0101 e94f 56ce 000d 8372 0200 0000 01    ...OV....r.....
10:36:02.121717 IP 192.168.1.1.22222 > 192.168.1.2.59727: UDP, length 5
        0x0000:  06c2 5f25 ea64 62b5 6da1 0dbb 0800 4500  .._%.db.m.....E.
        0x0010:  0021 dc8d 4000 4011 daea c0a8 0101 c0a8  .!..@.@.........
        0x0020:  0102 56ce e94f 000d 8372 0200 0000 01    ..V..O...r.....
10:36:02.230216 ARP, Request who-has 192.168.1.1 tell 192.168.1.2, length 28
        0x0000:  62b5 6da1 0dbb 06c2 5f25 ea64 0806 0001  b.m....._%.d....
        0x0010:  0800 0604 0001 06c2 5f25 ea64 c0a8 0102  ........_%.d....
        0x0020:  0000 0000 0000 c0a8 0101                 ..........
10:36:02.230267 ARP, Reply 192.168.1.1 is-at 62:b5:6d:a1:0d:bb, length 28
        0x0000:  06c2 5f25 ea64 62b5 6da1 0dbb 0806 0001  .._%.db.m.......
        0x0010:  0800 0604 0002 62b5 6da1 0dbb c0a8 0101  ......b.m.......
        0x0020:  06c2 5f25 ea64 c0a8 0102                 .._%.d....
10:36:03.121535 IP 192.168.1.2.59727 > 192.168.1.1.22222: UDP, length 5
        0x0000:  62b5 6da1 0dbb 06c2 5f25 ea64 0800 4500  b.m....._%.d..E.
        0x0010:  0021 3d1e 4000 4011 7a5a c0a8 0102 c0a8  .!=.@.@.zZ......
        0x0020:  0101 e94f 56ce 000d 8372 0300 0000 01    ...OV....r.....
10:36:03.254177 ARP, Request who-has 192.168.1.2 tell 192.168.1.1, length 28
        0x0000:  06c2 5f25 ea64 62b5 6da1 0dbb 0806 0001  .._%.db.m.......
        0x0010:  0800 0604 0001 62b5 6da1 0dbb c0a8 0101  ......b.m.......
        0x0020:  0000 0000 0000 c0a8 0102                 ..........
10:36:03.254200 ARP, Reply 192.168.1.2 is-at 06:c2:5f:25:ea:64, length 28
        0x0000:  62b5 6da1 0dbb 06c2 5f25 ea64 0806 0001  b.m....._%.d....
        0x0010:  0800 0604 0002 06c2 5f25 ea64 c0a8 0102  ........_%.d....
        0x0020:  62b5 6da1 0dbb c0a8 0101                 b.m.......
10:36:04.121810 IP 192.168.1.1.22222 > 192.168.1.2.59727: UDP, length 5
        0x0000:  06c2 5f25 ea64 62b5 6da1 0dbb 0800 4500  .._%.db.m.....E.
        0x0010:  0021 de50 4000 4011 d927 c0a8 0101 c0a8  .!.P@.@..'......
        0x0020:  0102 56ce e94f 000d 8372 0300 0000 01    ..V..O...r.....
```

We ignore the ARP frames, focusing only on the IP packages. The first packets has the following contents:

```
62b5 6da1 0dbb 06c2 5f25 ea64 0800 4500
0021 3ad5 4000 4011 7ca3 c0a8 0102 c0a8
0101 e94f 56ce 000d 8372 0000 0000 01
```

Ethernet Frame

In the first packet, the first 14 bytes are the header of the Ethernet frame. Inside the header, the first 6 bytes represent the destination MAC address. In this pinging case, the target MAC address is server's address, `62:b5:6d:a1:0d:bb`.

```
62 B5 6D A1 0D BB
```

The next 6 bytes represent the source MAC address, which is the client's address, `06:c2:5f:25:ea:64`.

```
06 C2 5F 25 EA 64
```

According to the DIX (Ethernet II) spec, the next 4 bytes represent the type of the payload (EtherType). In our ping application, we only use IPv4. The EtherType of IPv4 is `0x0800`, which matches our experiment.

```
08 00
```

IPv4 Packet

The payload of the Ethernet frame is the IPv4 packet. The last 8 bytes are the source and destination IP addresses. Take the first Ethernet packet from the log for example,

```
C0 A8 01 02 # source IP address: 192.168.1.2 (client)
C0 A8 01 01 # destination IP address: 192.168.1.1 (server)
```

We can see the IP addresses in the packet match the IP addresses we assigned to the `mypingsrv` and `mypingcli`.

UDP Packet

The payload of IPv4 packet is the UDP packet. The type of the payload can be identified from the 10th byte in IPv4 header. The value of the byte is `0x11` representing the payload uses UDP. The first 4 bytes in UDP is the source and destination port numbers. Take the first Ethernet packet from the log for example,

```
E9 4F # source port number: 59727 (client automatically assigned by OS)
56 CE # destination port number: 22222 (server)
```

We can see the destination port number in the packet matches the port number we assigned to the server with `mypingsrv`.

The last 5 bytes store the content of our message. Take the first Ethernet packet from the log for example,

```
00      00      00      00      01
```

That matches our configuration from `pingparam.dat`. In the first ping, the first 4 bytes of the data is the message number, which is `0` in this case. The last byte of the data is the delay, which is `1` in this case. We can see that for each pinging (client to server and back), the message number increases 1 in big-endian and delay remain the same, which is the same as our implementation.

```
10:35:57.121253 IP 192.168.1.2.59727 > 192.168.1.1.22222:
        0x002A:  0000 0000 01
10:35:58.121470 IP 192.168.1.1.22222 > 192.168.1.2.59727:
        0x002A:  0000 0000 01
10:35:59.121321 IP 192.168.1.2.59727 > 192.168.1.1.22222:
```

```
        0x002A:  0100 0000 01
10:36:00.121569 IP 192.168.1.1.22222 > 192.168.1.2.59727:
        0x002A:  0100 0000 01
10:36:01.121417 IP 192.168.1.2.59727 > 192.168.1.1.22222:
        0x002A:  0200 0000 01
10:36:02.121717 IP 192.168.1.1.22222 > 192.168.1.2.59727:
        0x002A:  0200 0000 01
10:36:03.121535 IP 192.168.1.2.59727 > 192.168.1.1.22222:
        0x002A:  0300 0000 01
10:36:04.121810 IP 192.168.1.1.22222 > 192.168.1.2.59727:
        0x002A:  0300 0000 01
```

Problem 2

Same Lab vs Different Labs

The following is the statistics of file transmission between 2 machines in same lab.

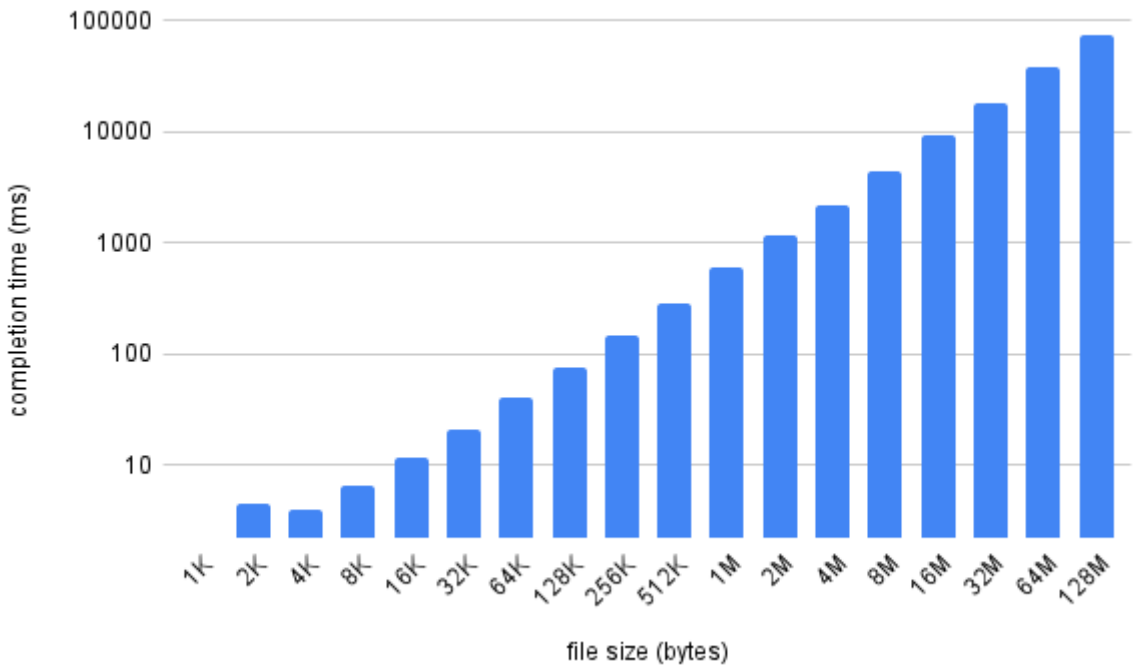
Environment:

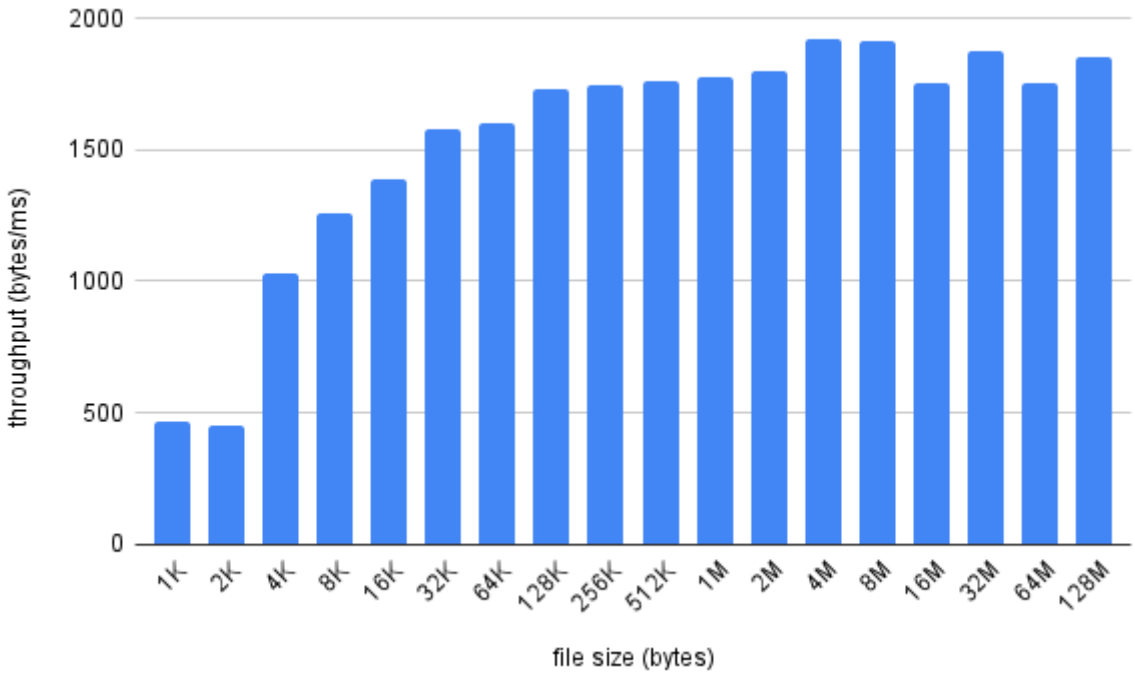
- Blocksize: 1024 bytes
- Same lab
  - Server on: `amber05.cs.purdue.edu` (`128.10.12.135`)
  - Client on: `amber06.cs.purdue.edu` (`128.10.112.136`)
- Different labs
  - Server on: `amber05.cs.purdue.edu` (`128.10.12.135`)
  - Client on: `pod3-3.cs.purdue.edu` (`128.10.25.213`)

file size (bytes)	same-lab completion time (ms)	same-lab throughput (bytes/ms)	different-labs completion time (ms)	different-labs throughput (bytes/ms)
1K	2.195	466.515	4.235	241.795
2K	4.519	453.198	4.927	415.669
4K	3.983	1028.371	7.366	556.068
8K	6.53	1254.518	11.95	685.523
16K	11.836	1384.251	20.284	807.73
32K	20.75	1579.181	36.328	902.004
64K	40.93	1601.173	72.201	907.688
128K	75.714	1731.146	131.6	995.988
256K	149.943	1748.291	258.006	1016.038
512K	291.836	1760.324	564.554	928.676
1M	591.683	1772.192	961.443	1090.627

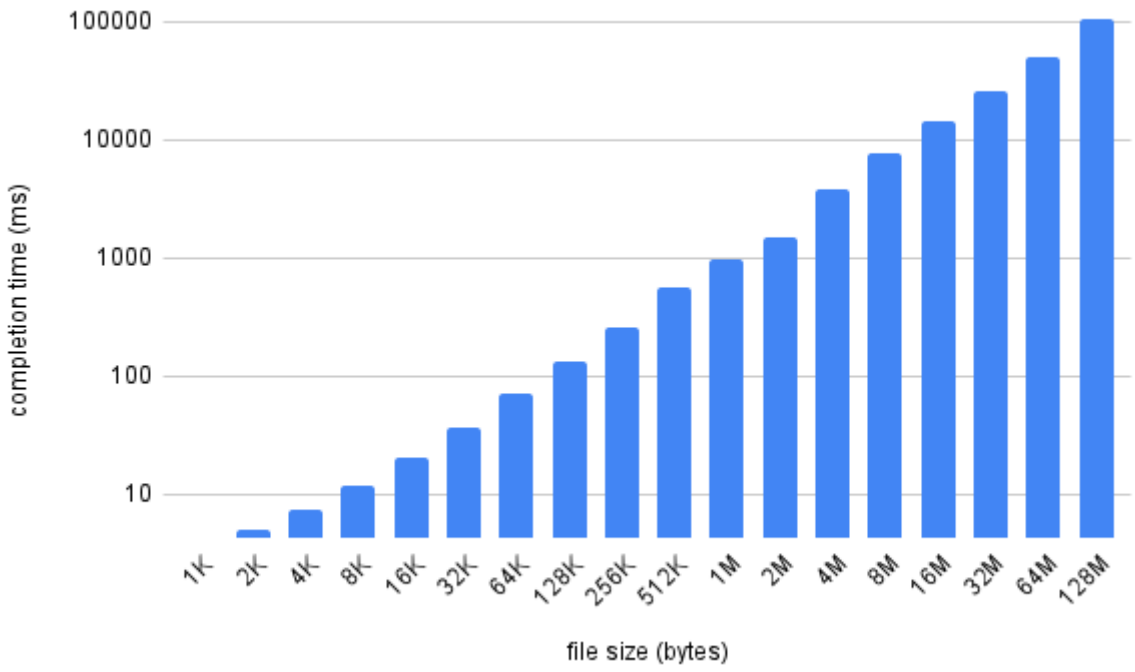
file size (bytes)	same-lab completion time (ms)	same-lab throughput (bytes/ms)	different-labs completion time (ms)	different-labs throughput (bytes/ms)
2M	1164.455	1800.973	1489.877	1407.601
4M	2182.411	1921.867	3794.393	1105.395
8M	4378.048	1916.061	7578.114	1106.952
16M	9550.879	1756.615	14313.703	1172.109
32M	17888.158	1875.79	26028.311	1289.151
64M	38215.547	1756.062	50110.191	1339.226
128M	73504.717	1852.974	106147.926	1264.44

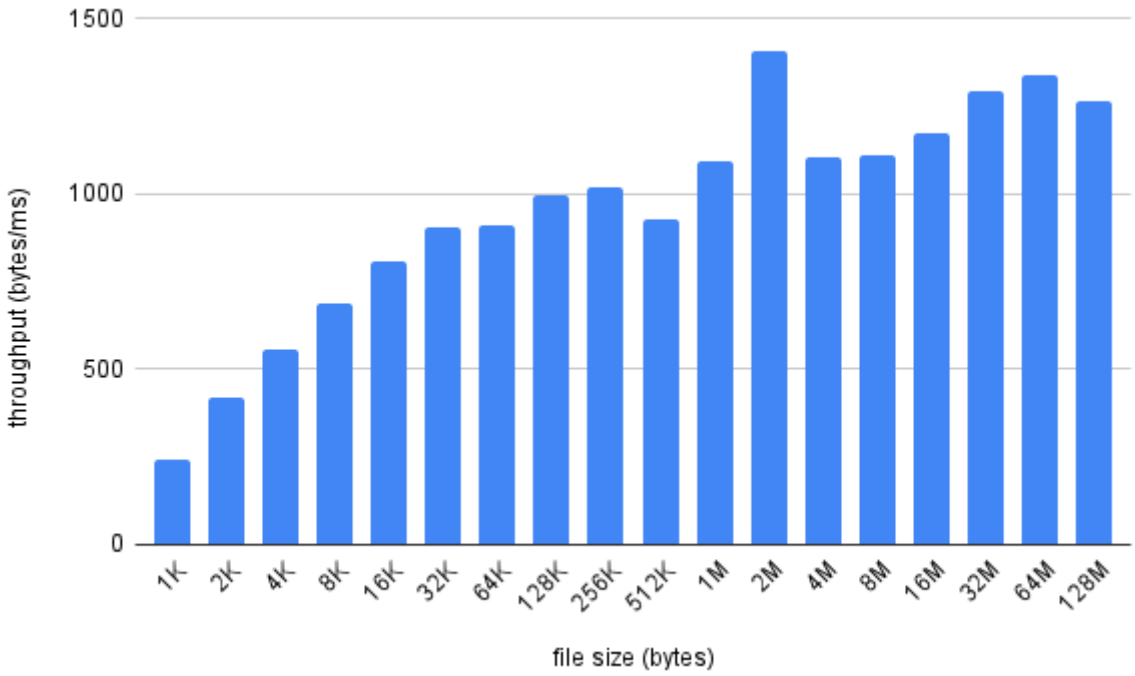
The following is the visualization of the "same-lab" statistics. Note that the completion time chart is in log scale.





The following is the visualization of the "different-lab" statistics. Note that the completion time chart is in log scale.





From the statistics, we can see that the overall completion time of the "same-lab" is smaller than the one of "different-lab". This is expected as the physical distance is longer for 2 machines in different labs than in the same lab. The throughput is also higher in the "same-lab" case as less router or other core infrastructures is needed to transfer the data between 2 machines in same lab in comparison to 2 machines in 2 different labs. The same reason causes the fluctuation of throughput in "different-labs" case as more infrastructures involved, which causes higher nondeterministic overhead. Finally, the throughput for small files is relatively low as the overtime dominate the completion time for these cases.

There is another big factor for the completion time and throughput: network file system. All lab machines use network file system to access the user data. Thus, if the file system is busy or the connection is heavily loaded, the throughput in the statistics above will decreases significantly.

Blocksize

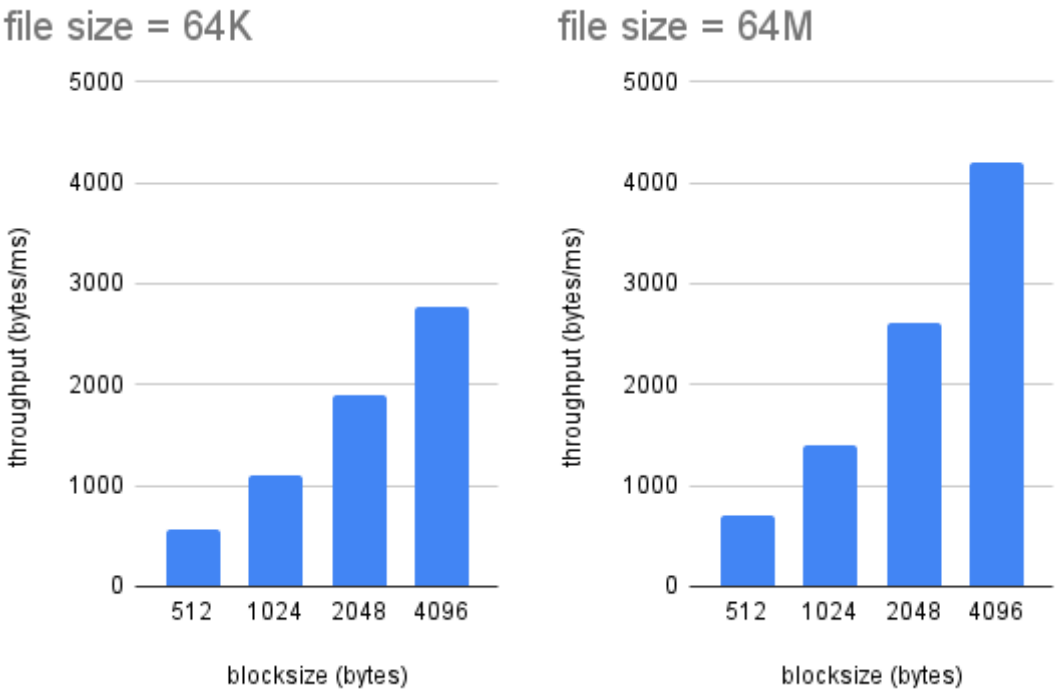
The following is the statistics of file transmission in different blocksizes and file sizes.

Environment:

- Server on: `amber05.cs.purdue.edu` (`128.10.112.135`)
- Client on: `amber06.cs.purdue.edu` (`128.10.112.136`)

blocksize (bytes)	**64K bytes file throughput (bytes/ms)**	**64M bytes file throughput (bytes/ms)**
512	567.063	699.483
1024	1101.390	1403.790
2048	1897.724	2617.399
4096	2763.600	4200.077

The following is the visualization of above data.



We can see that the throughput is heavily affected by the blocksize. The blocksize decides how many times a series of system calls need to be invoked. Thus, high blocksize can greatly reduce the overhead of socket I/O and disk I/O.

The load of network file system is still a big factor for above statistics. Any network congestion would heavily impact the throughput.

Bonus problem

Statistics

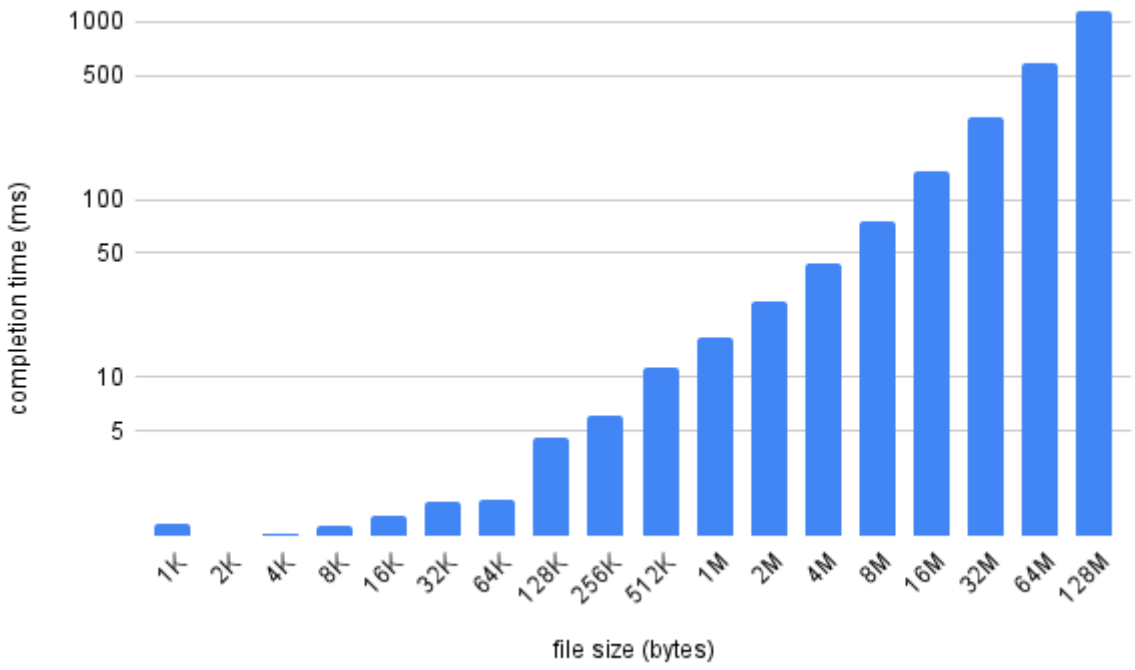
Environment:

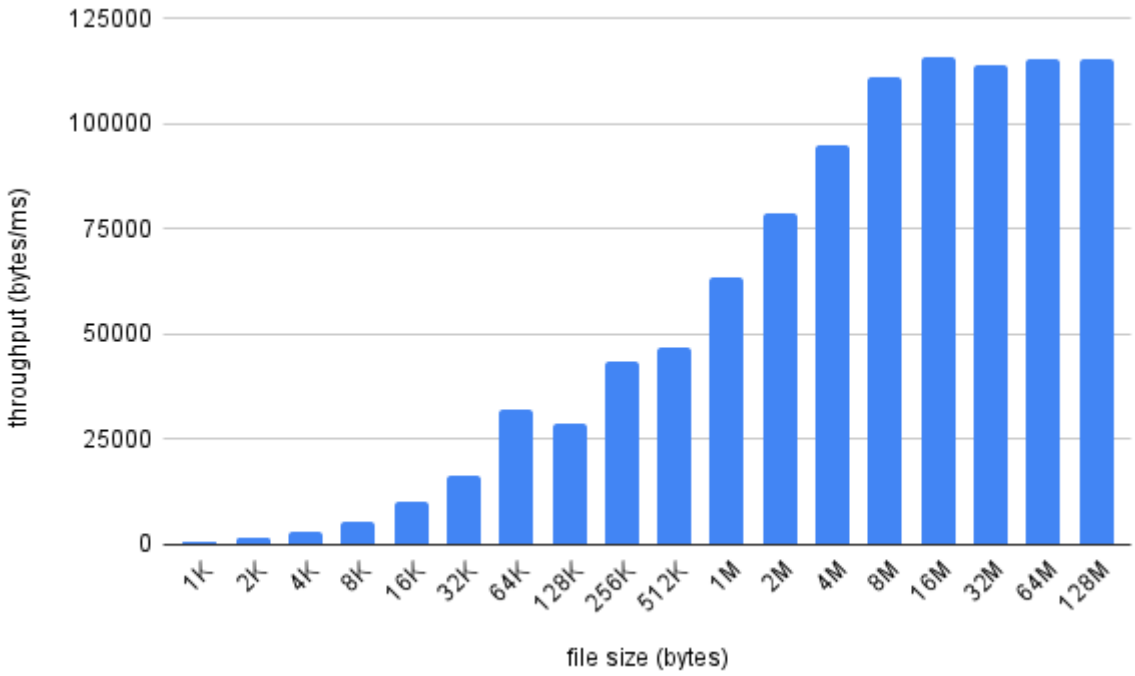
- Blocksize: 1024 bytes
- Server on: `amber05.cs.purdue.edu` (`128.10.112.135`)
- Client on: `amber06.cs.purdue.edu` (`128.10.112.136`)

file size (bytes)	completion time (ms)	throughput (bytes/ms)
1K	1.495	684.95
2K	1.273	1608.798
4K	1.322	3098.336
8K	1.472	5565.217
16K	1.642	9978.076
32K	1.991	16458.061

file size (bytes)	completion time (ms)	throughput (bytes/ms)
64K	2.052	31937.622
128K	4.607	28450.619
256K	6.023	43523.825
512K	11.265	46541.323
1M	16.544	63381.044
2M	26.724	78474.48
4M	44.166	94966.807
8M	75.736	110761.17
16M	145.237	115516.129
32M	294.389	113979.911
64M	581.909	115325.358
128M	1164.106	115296.827

Visualization





Analysis

When the file size reach to 8M bytes, the throughput has approximately reached the upper bound. The upper bound of the throughput increases 63 times compared to the statistics when using network file system. Thus, the completion time decreases significantly. For example, the completion time of a 128M bytes file decreases from 73504.717 ms to 1164.106 ms, which is about 63 times faster. Namely, performance of local file system is about 63 faster than the one of network file system.