Report for CSE 124 Project 1

Name: Haitian Jiang, PID: A14283657, GitHub username: bill-ginger.

Server region: Oregon, US; client region: Seoul, South Korea.

Note: in this report, I will use K to represent 1,000, M to represent 1,000,000.

Experiment

The result are as follows:

```
# Serial
➜ ~ for i in {1..3}
for> go run fetchall.go http://54.244.98.195:8000/gendata\?numBytes\=1
0.25s        1  http://54.244.98.195:8000/gendata?numBytes=1
0.25s elapsed
0.25s        1  http://54.244.98.195:8000/gendata?numBytes=1
0.25s elapsed
0.25s        1  http://54.244.98.195:8000/gendata?numBytes=1
0.25s elapsed
➜ ~ for i in {1..3}
for> go run fetchall.go http://54.244.98.195:8000/gendata\?numBytes\=1000
0.25s     1000  http://54.244.98.195:8000/gendata?numBytes=1000
0.25s elapsed
0.25s     1000  http://54.244.98.195:8000/gendata?numBytes=1000
0.25s elapsed
0.25s     1000  http://54.244.98.195:8000/gendata?numBytes=1000
0.25s elapsed
➜ ~ for i in {1..3}
for> go run fetchall.go http://54.244.98.195:8000/gendata\?
numBytes\=1000000
1.02s  1000000  http://54.244.98.195:8000/gendata?numBytes=1000000
1.02s elapsed
1.00s  1000000  http://54.244.98.195:8000/gendata?numBytes=1000000
1.00s elapsed
1.00s  1000000  http://54.244.98.195:8000/gendata?numBytes=1000000
1.00s elapsed
➜ ~ for i in {1..3}
for> go run fetchall.go http://54.244.98.195:8000/gendata\?
numBytes\=100000000
11.89s  100000000  http://54.244.98.195:8000/gendata?numBytes=100000000
11.89s elapsed
12.08s  100000000  http://54.244.98.195:8000/gendata?numBytes=100000000
12.08s elapsed
13.28s  100000000  http://54.244.98.195:8000/gendata?numBytes=100000000
13.28s elapsed

# Concurrent
➜ ~ for i in {1..3}
for> go run fetchall.go http://54.244.98.195:8000/gendata\?
numBytes\=1{,000,000000,00000000}
0.25s     1000  http://54.244.98.195:8000/gendata?numBytes=1000
0.25s        1  http://54.244.98.195:8000/gendata?numBytes=1
0.99s  1000000  http://54.244.98.195:8000/gendata?numBytes=1000000
```

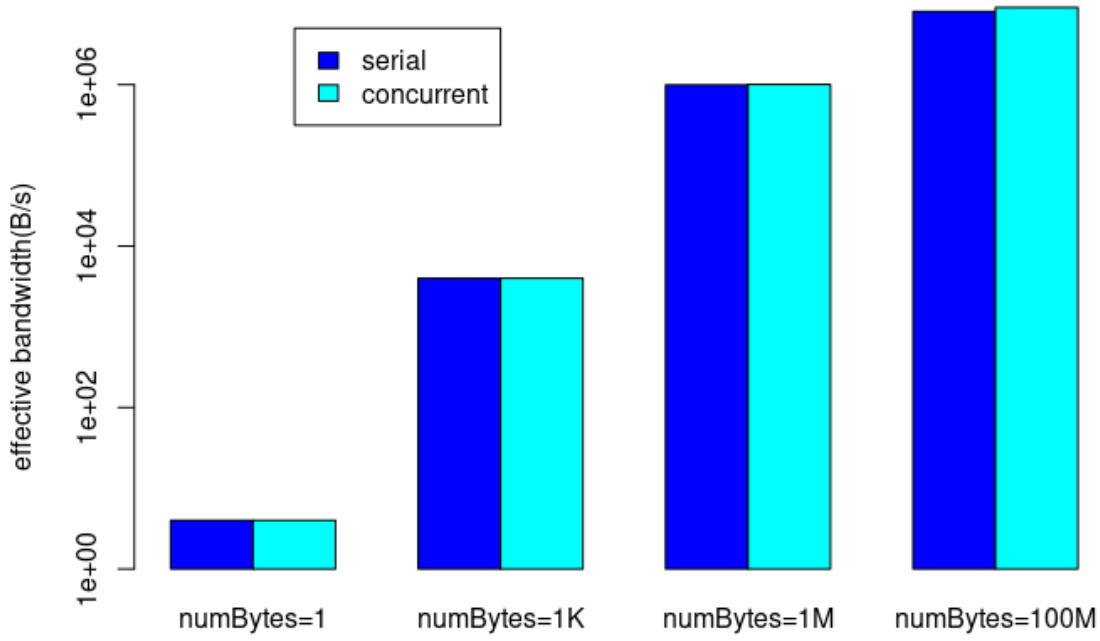
```
11.02s   100000000  http://54.244.98.195:8000/gendata?numBytes=100000000
11.02s elapsed
0.25s      1000  http://54.244.98.195:8000/gendata?numBytes=1000
0.25s         1  http://54.244.98.195:8000/gendata?numBytes=1
0.99s   1000000  http://54.244.98.195:8000/gendata?numBytes=1000000
9.59s   100000000  http://54.244.98.195:8000/gendata?numBytes=100000000
9.59s elapsed
0.25s         1  http://54.244.98.195:8000/gendata?numBytes=1
0.25s      1000  http://54.244.98.195:8000/gendata?numBytes=1000
1.00s   1000000  http://54.244.98.195:8000/gendata?numBytes=1000000
12.50s   100000000  http://54.244.98.195:8000/gendata?numBytes=100000000
12.51s elapsed

# Latency
➜ ~ ping -c 3 54.244.98.195
PING 54.244.98.195 (54.244.98.195) 56(84) bytes of data.
64 bytes from 54.244.98.195: icmp_seq=1 ttl=221 time=123 ms
64 bytes from 54.244.98.195: icmp_seq=2 ttl=221 time=123 ms
64 bytes from 54.244.98.195: icmp_seq=3 ttl=221 time=123 ms

--- 54.244.98.195 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 123.757/123.767/123.786/0.406 ms
```

The average bandwidth for each `numBytes` and request method(serial/concurrent) is shown in the following table.

Bandwidth	serial	concurrent
`numBytes`=1	4	4
`numBytes`=1K	4000	4000
`numBytes`=1M	993377	1006711
`numBytes`=100M	8053691	9060706

Graph

This bar plot graph in log scale represents the effective bandwidth under different `numBytes`, where the data are from the above table.



Report

- Q1. For the data from the serial experiment, how was the effective bandwidth affected by the size of `numBytes`?
  Answer: It is clearly shown by the blue bars that the effective bandwidth goes up as the size of `numBytes` goes up.
- Q2. Using the data from only the **Serial** experiment, estimate the bandwidth between your client and your server. Now incorporate the data from your **Latency** experiment to increase the accuracy of your bandwidth estimate. Describe how data from the **Latency** experiment improves accuracy.

  Answer: In Serial experiment, the biggest `numBytes` shows the real capacity of the bandwidth. So I will use the effective bandwidth calculated when `numBytes`=100M. The bandwidth is 8053691B/s, roughly 8MB/s. Actually this resonates with the outcome when I turn `numBytes` up to 200M.

  If the RTT is counted, then bandwidth will become `numByes/(average time - rtt)`. In my case it becomes 8134269B/s, which is roughly 8.1M/s.

  The Latency experiment improves the accuracy by taking out the travel time of requests and responses. So the remaining time purely contains how long the data flow through the network card.
- Q3. How did the data from the **Serial** experiment compare from the **Concurrent** experiment? Similar? Dissimilar? Explain these results as best you can.

  Answer: The data from the Serial experiment and the Concurrent experiment is similar. The bandwidth in the concurrent experiment is slightly higher than that in the serial experiment. I think it is due to the fluctuation.

- Q4. After carrying out these experiments, what is something that you learned about performance and networked applications?

  Answer: The speed of our network infrastructure is very fast, almost to its limit. The straight distance from Oregon to Seoul is about 8,892km, while light travels about 200,000,000m/s in optical fiber. So the theoretical minimal latency is 2*8892/200000=0.08892s. The real-life latency 0.123s is very close to the limit.

- Q5. After carrying out these experiments, what is one (or more) unanswered question(s) you still have about network performance?

  How does the ping application and the NTP protocol calculate the exact time latency, since the route from the server to the client is not deterministic, and the time stamp on different machine may not be exactly the same.