# Report for CSC3150 assignment 2

Feng Yutong 120090266

**Design**

- hw2
  - Intro

    This program implements the game "Forg crosses river". A frog crosses the river by jumping on the logs as they pass by.

  - Implementation

    1. A 2D array map represents the log and river, witch character '=' and ' ' respectively. Since each log has similar movement, multiple threads can be used to simulate each log . `thread[i]` represents the log in ith row.
    2. Each thread is bind to `logs_move`. Threads with odd id moves right and with even id moves left. Each thread updates new `start_pos` after a tiny period. If `frog.x` equals thread id, move the frog with the log. Position is mod by `COLUMN-1` to ensure within the river width range.
    3. Use `kbhit()` to read keyboard input. A STL map stores key value and its corresponding moving direction. If `q` is pressed, set flag `end` as 3 (quit). Otherwise check the stauts of frog. If it is in the river or reach the edge, set flag `end` as 2 (lose). If `frog.x=ROW+1`, then set `end` as 1 (win).
    4. Flush and print the new map. Though every log moves at the same time, the program uses `pthread_mutex_lock` to lock each movement. Otherwise, frog's position will be updately wrongly.
    5. Repeat step 2-4 until the `end` is set a value.

- Bonus
  - Intro

    This program implements async.c and async.h to handle web request by multi-threads.

  - Design

    - `async_init` creates a thread poll, initializes mutex and conditional variable. Each thread is bind to `init`.
    - Everytime `async_run` is called, it puts a request with function pointer and args in a request queue. Then it use `pthread_cond_signal` to indicate a new job coming

in.
        - `init` first locks the thread. It uses `pthread_cond_wait` in a while loop waiting for request to enter. Once waking up, it pops a request from the request queue (globle varible changing is locked). Then it unlocks and handles the request (handling request is in parallel). Then it goes back to sleep and waits for another signal.

## Environment and execution

- Environment
    - OS version: Linux Ubuntu 22.04.1
    - Kernel version: 5.15.0-50-generic
- Execution
    - hw2

      cd to the directory of Makefile

      ```
      make
      ./hw2
      ```

    - bonus

      1. cd to the directory of Makefile

      ```
      make
      ./httpserver --files files/ --port 8000 --num-threads T
      ```

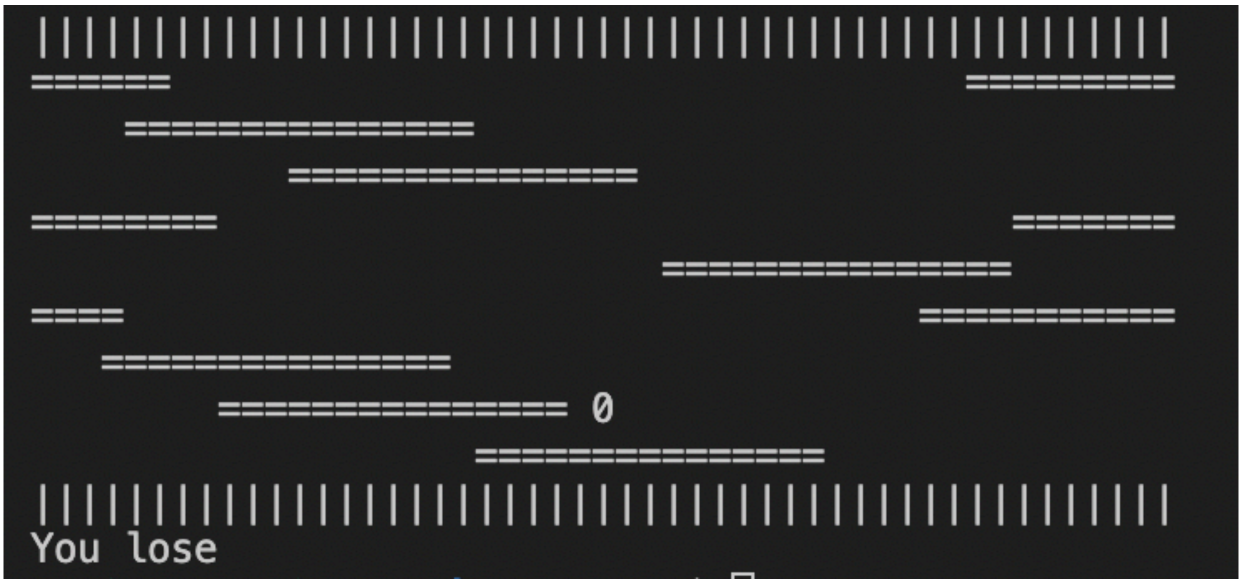
      please replace T with a thread number, then open another terminal,

      ```
      ab -n X -c T http://localhost:8000/
      ```

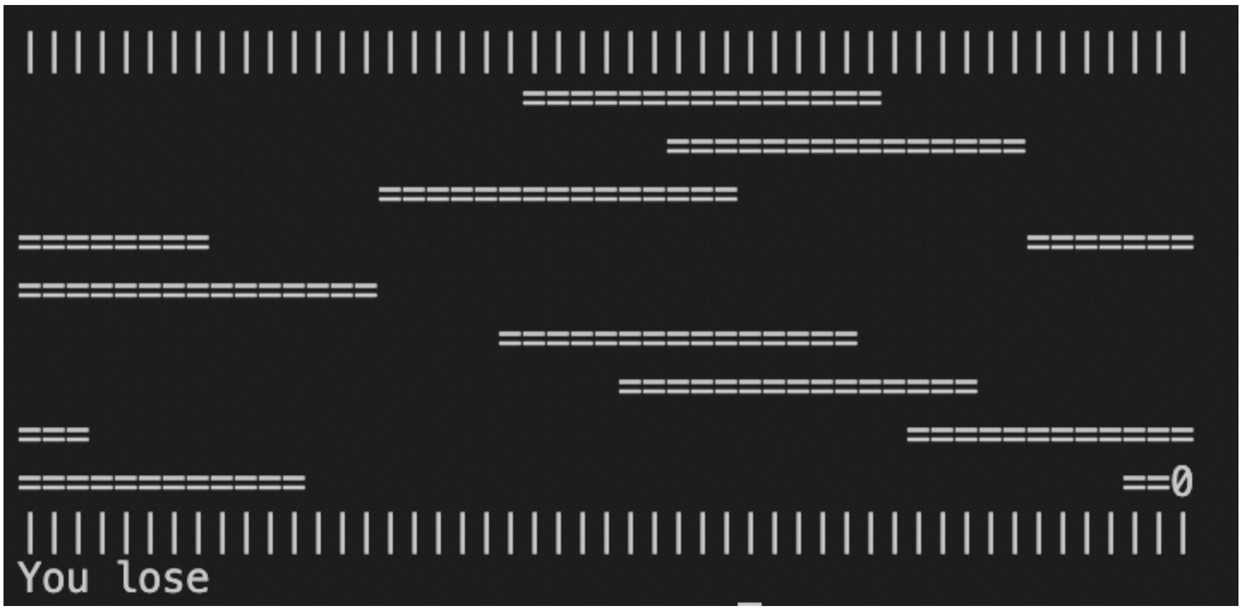
      please replace X with the total request number, say 5000, and T with the thread number, say 10) to benchmark the thread pool implementations.

## Output

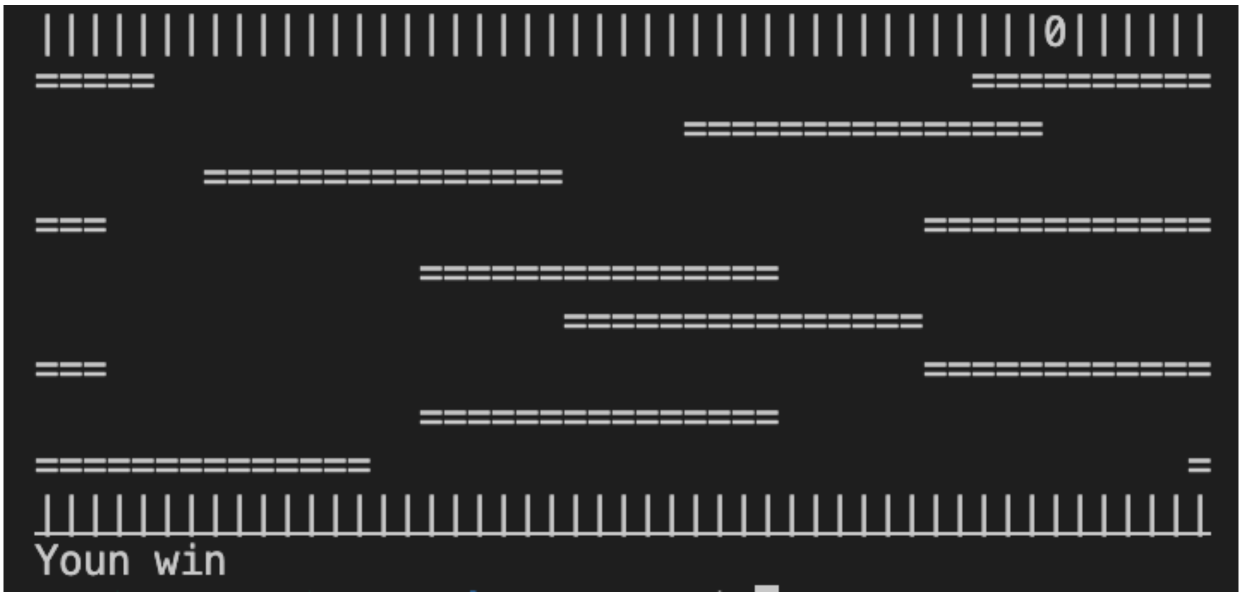
- hw2: It shows two cases of losing (in the river / hit the edge), one case of winning and one case of quit. The final map is not flushed so you can better know the situation

- 

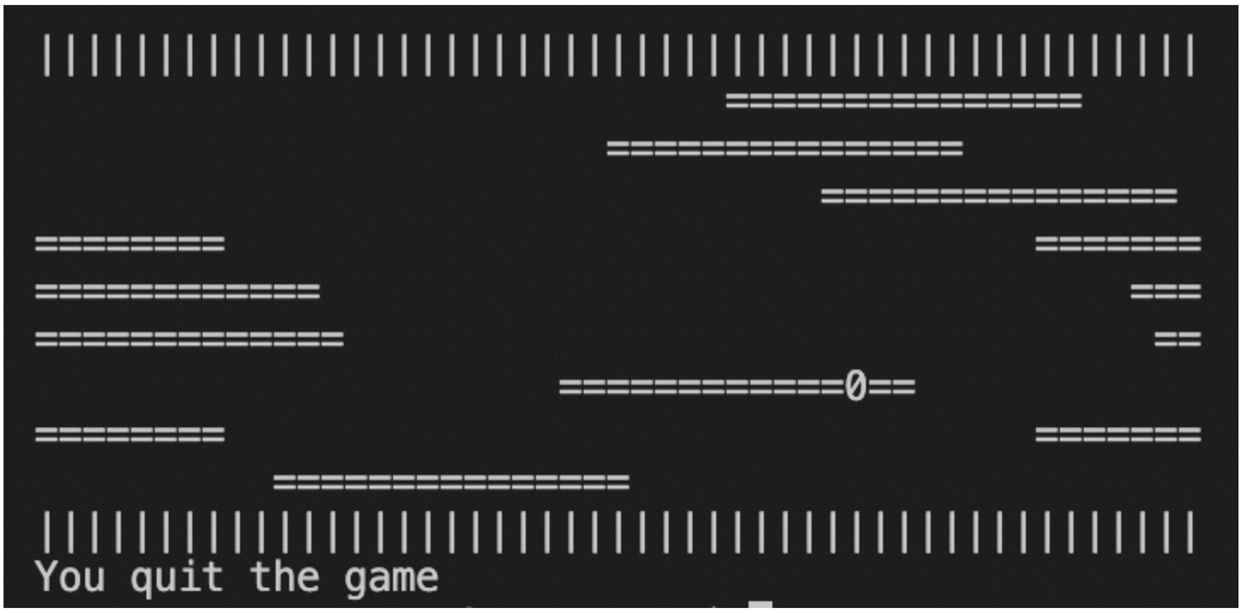
```
|||||||||||||||||||||||||||||||||||||||||||||||||
======                                  =========
    ===============
           ===============
========                                  =======
                           ===============
====                                  ===========
   ===============
        =============== 0
                   ===============
|||||||||||||||||||||||||||||||||||||||||||||||||
You lose
```

- 

```
|||||||||||||||||||||||||||||||||||||||||||||||||
                     ===============
                           ===============
               ===============
========                                  =======
===============
                    ===============
                         ===============
===                                  ============
============                                  ==0
|||||||||||||||||||||||||||||||||||||||||||||||||
You lose
```

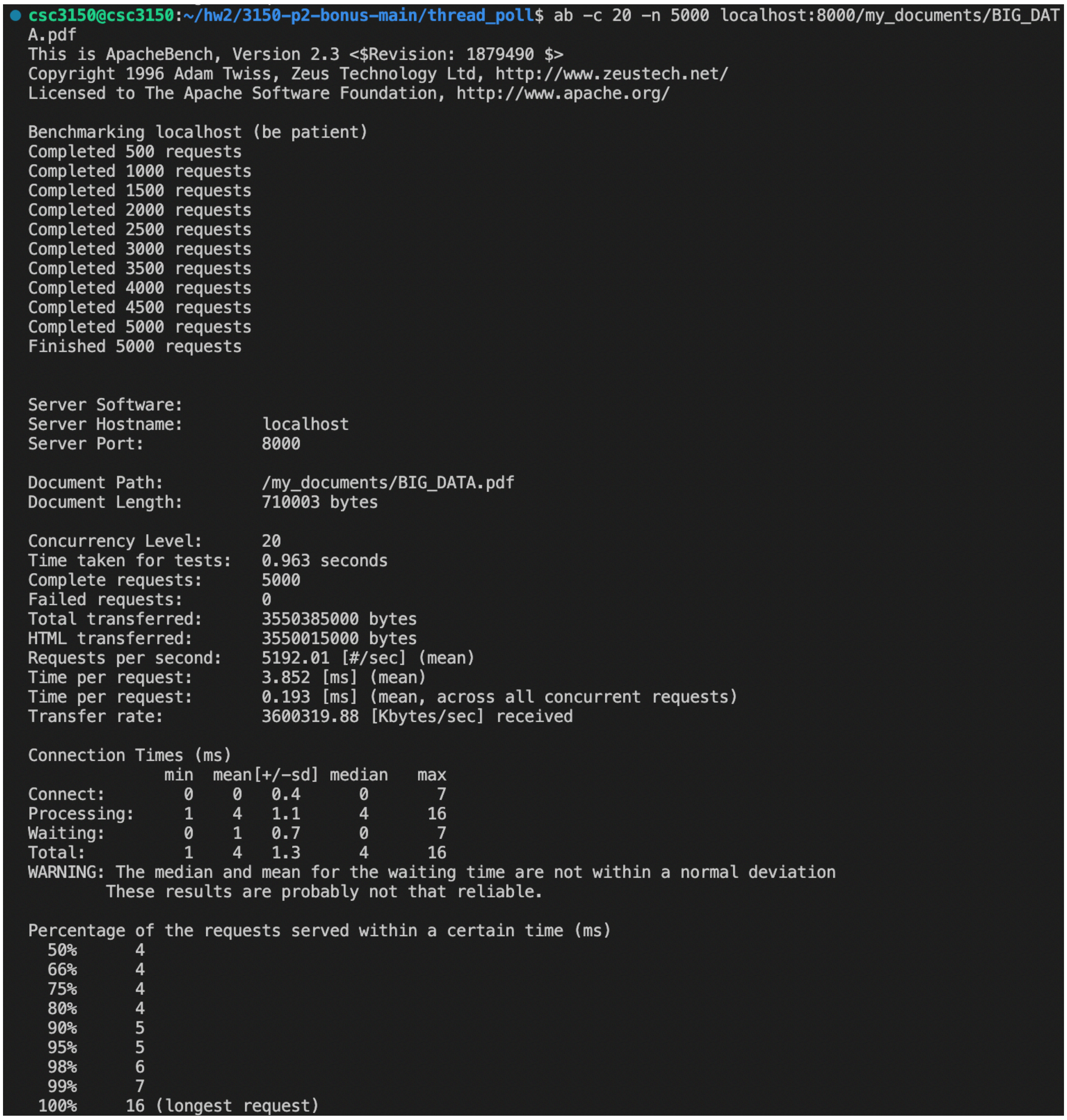
- 

```
|||||||||||||||||||||||||||||||||||||||||||0|||||
=====                                  ==========
                           ===============
       ===============
===                                  ============
                ===============
                      ===============
===                                  ============
                ===============
==============                                  =
|||||||||||||||||||||||||||||||||||||||||||||||||
Youn win
```

- 

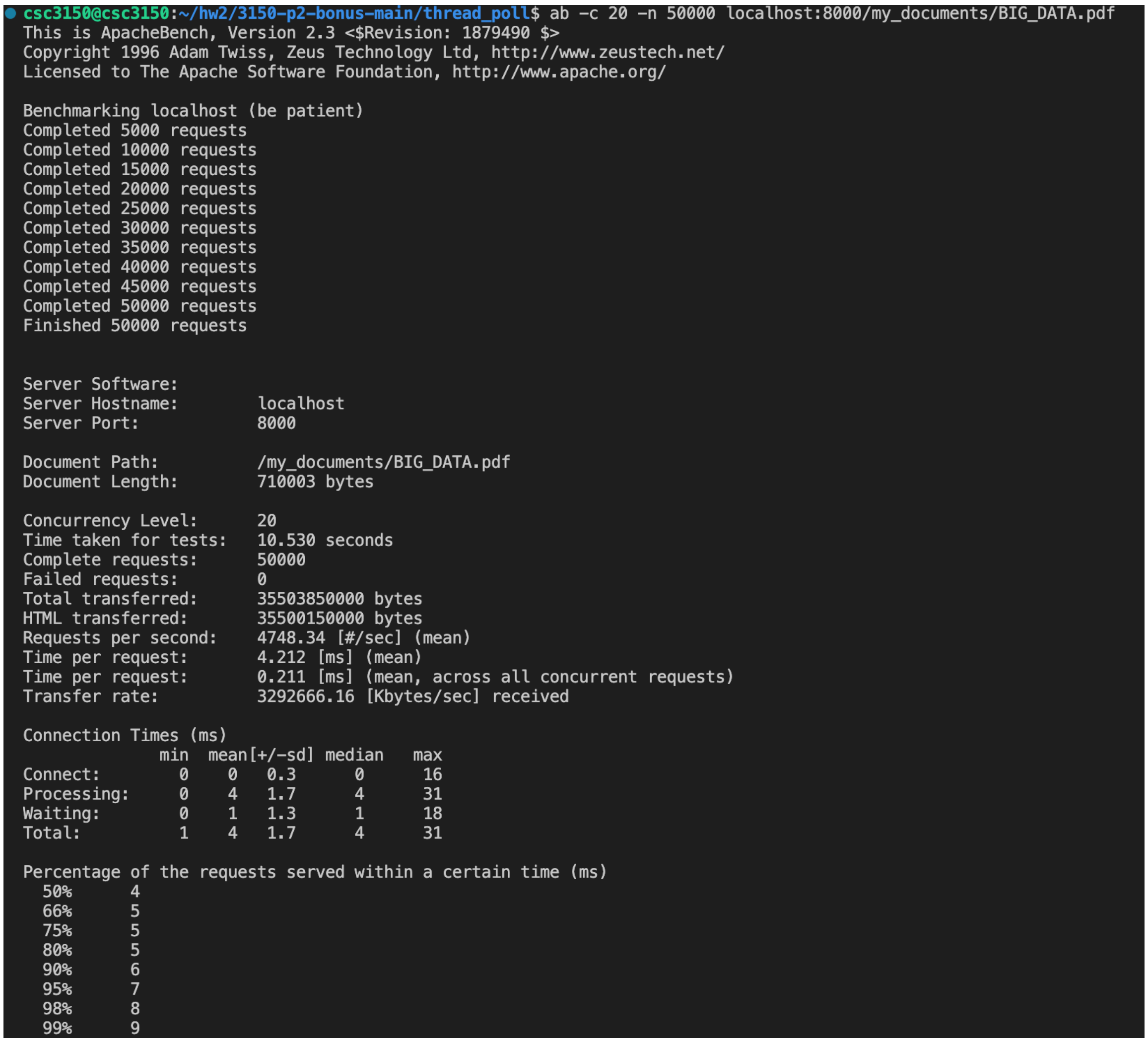
```
||||||||||||||||||||||||||||||||||||||||||||||||||
                             ===============
                        ===============
                                 ===============
========                                  =======
============                                  ===
=============                                  ==
                      ============0==
========                                  =======
          ===============
||||||||||||||||||||||||||||||||||||||||||||||||||
You quit the game
```

- bonus
    - A sample provided in piazza



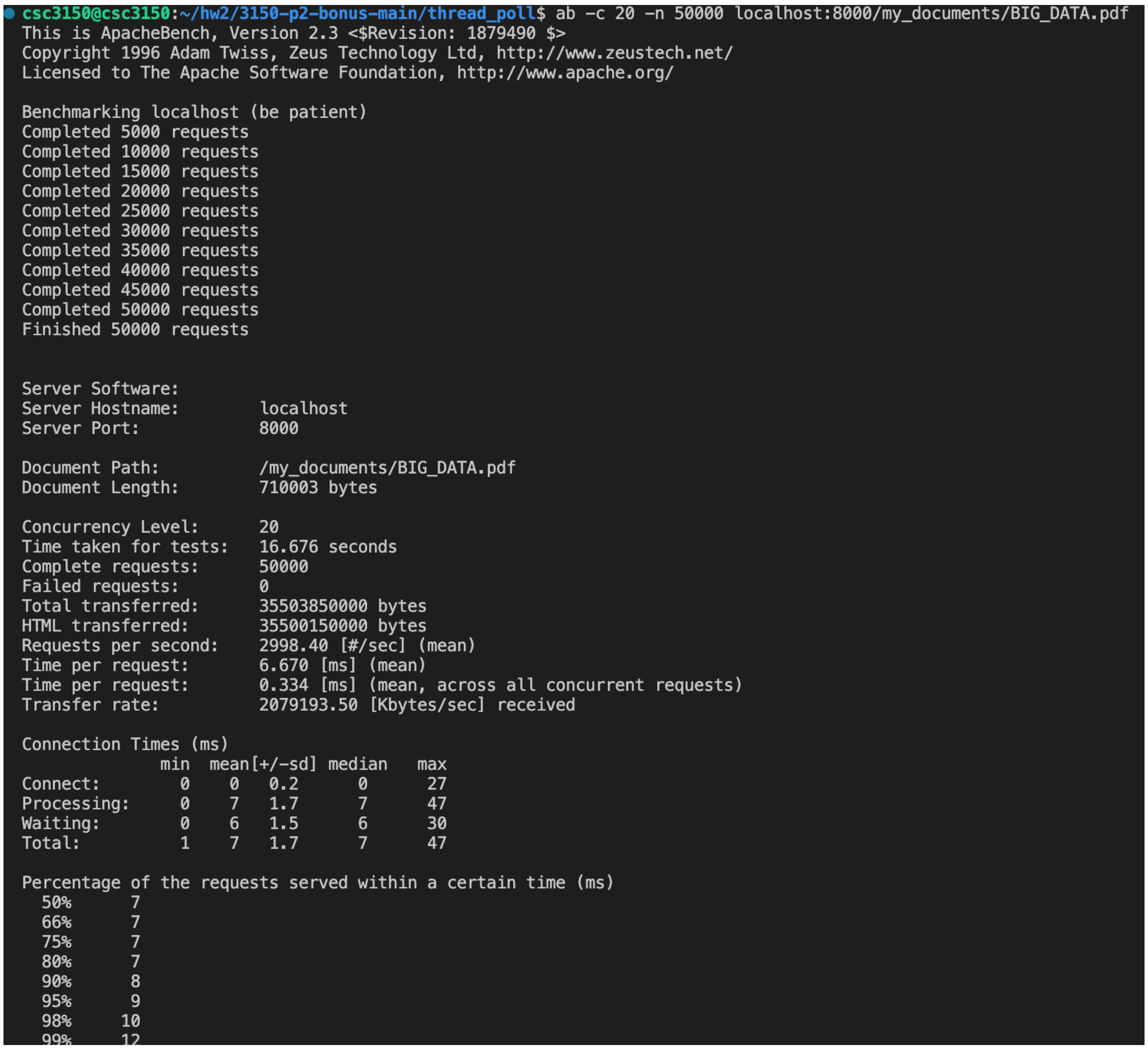
```
csc3150@csc3150:~/hw2/3150-p2-bonus-main/thread_poll$ ab -c 20 -n 5000 localhost:8000/my_documents/BIG_DAT
A.pdf
This is ApacheBench, Version 2.3 <$Revision: 1879490 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 500 requests
Completed 1000 requests
Completed 1500 requests
Completed 2000 requests
Completed 2500 requests
Completed 3000 requests
Completed 3500 requests
Completed 4000 requests
Completed 4500 requests
Completed 5000 requests
Finished 5000 requests


Server Software:
Server Hostname:        localhost
Server Port:            8000

Document Path:          /my_documents/BIG_DATA.pdf
Document Length:        710003 bytes

Concurrency Level:      20
Time taken for tests:   0.963 seconds
Complete requests:      5000
Failed requests:        0
Total transferred:      3550385000 bytes
HTML transferred:       3550015000 bytes
Requests per second:    5192.01 [#/sec] (mean)
Time per request:       3.852 [ms] (mean)
Time per request:       0.193 [ms] (mean, across all concurrent requests)
Transfer rate:          3600319.88 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.4      0       7
Processing:     1    4   1.1      4      16
Waiting:        0    1   0.7      0       7
Total:          1    4   1.3      4      16
WARNING: The median and mean for the waiting time are not within a normal deviation
        These results are probably not that reliable.

Percentage of the requests served within a certain time (ms)
  50%      4
  66%      4
  75%      4
  80%      4
  90%      5
  95%      5
  98%      6
  99%      7
 100%     16 (longest request)
```

- Since pthread_create and enter/pop queue takes extra time, small IO cannot show the differences between multi-thread and one thread. Here we show a 50000 requests.
  - multi-thread



```
csc3150@csc3150:~/hw2/3150-p2-bonus-main/thread_poll$ ab -c 20 -n 50000 localhost:8000/my_documents/BIG_DATA.pdf
This is ApacheBench, Version 2.3 <$Revision: 1879490 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 5000 requests
Completed 10000 requests
Completed 15000 requests
Completed 20000 requests
Completed 25000 requests
Completed 30000 requests
Completed 35000 requests
Completed 40000 requests
Completed 45000 requests
Completed 50000 requests
Finished 50000 requests


Server Software:
Server Hostname:        localhost
Server Port:            8000

Document Path:          /my_documents/BIG_DATA.pdf
Document Length:        710003 bytes

Concurrency Level:      20
Time taken for tests:   10.530 seconds
Complete requests:      50000
Failed requests:        0
Total transferred:      35503850000 bytes
HTML transferred:       35500150000 bytes
Requests per second:    4748.34 [#/sec] (mean)
Time per request:       4.212 [ms] (mean)
Time per request:       0.211 [ms] (mean, across all concurrent requests)
Transfer rate:          3292666.16 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.3      0      16
Processing:     0    4   1.7      4      31
Waiting:        0    1   1.3      1      18
Total:          1    4   1.7      4      31

Percentage of the requests served within a certain time (ms)
  50%      4
  66%      5
  75%      5
  80%      5
  90%      6
  95%      7
  98%      8
  99%      9
```

- one thread



```
csc3150@csc3150:~/hw2/3150-p2-bonus-main/thread_poll$ ab -c 20 -n 50000 localhost:8000/my_documents/BIG_DATA.pdf
This is ApacheBench, Version 2.3 <$Revision: 1879490 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 5000 requests
Completed 10000 requests
Completed 15000 requests
Completed 20000 requests
Completed 25000 requests
Completed 30000 requests
Completed 35000 requests
Completed 40000 requests
Completed 45000 requests
Completed 50000 requests
Finished 50000 requests


Server Software:
Server Hostname:        localhost
Server Port:            8000

Document Path:          /my_documents/BIG_DATA.pdf
Document Length:        710003 bytes

Concurrency Level:      20
Time taken for tests:   16.676 seconds
Complete requests:      50000
Failed requests:        0
Total transferred:      35503850000 bytes
HTML transferred:       35500150000 bytes
Requests per second:    2998.40 [#/sec] (mean)
Time per request:       6.670 [ms] (mean)
Time per request:       0.334 [ms] (mean, across all concurrent requests)
Transfer rate:          2079193.50 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.2      0      27
Processing:     0    7   1.7      7      47
Waiting:        0    6   1.5      6      30
Total:          1    7   1.7      7      47

Percentage of the requests served within a certain time (ms)
  50%      7
  66%      7
  75%      7
  80%      7
  90%      8
  95%      9
  98%     10
  99%     12
```

## Learning

This is the first time I do multi-thread programming. I learned how to create thread poll, lock thread, and use conditional variable to implement thread communicaiton. Though hw2 can be implemented without multi-thread programming, I believe this programming skill can help me to deal with large data size with duplicate work in the future, which can greatly decrease the runtime.