TESTING

Almost the same as testing for last assignment, adding up concurrency request

- Setting Up Stage - arguments
  - `./httpserver -N x(>0) -c y(>0) localhost:8000` OK
  - `./httpserver -N x(<=0) -c y(<=0) localhost:8000` FAILED
  - The rest is the **same** from last assignment
- Transaction Stage - request
  - `./httpclient s:file:aaaa-40 characters-aaaa & ./httpclient r:aaaa-40 characters-aaaa:test` SUCCEED – 200 OK
  - `./httpclient s:file:aaaa-40 characters-aaaa & ./httpclient s:file:aaaa-40 characters-aaaa & ./httpclient r:aaaa-40 characters-aaaa:test` SUCCEED – 200 OK
  - `./httpclient s:file:aaaa-40 characters-aaaa r:aaaa-40 characters-aaaa:test & ./httpclient s:file:aaaa-40 characters-aaaa & ./httpclient r:aaaa-40 characters-aaaa:test` SUCCEED – 200 OK
  - The rest is the **same** from last assignment
- File Transportation - file
  - 0 B file SUCCEED – 200 OK
  - 30 B file SUCCEED – 200 OK
  - 4 KB file SUCCEED – 200 OK
  - 4 MB file SUCCEED – 200 OK
- Other situations
  - Sending wrong request to both client and server SUCCEED – 200 OK
  - Work with `SimpleHTTPserver` SUCCEED – 200 OK
  - Work with `curl` SUCCEED – 200 OK
  - Work with web browser SUCCEED – 200 OK
  - Work with others' server and client SUCCEED – 200 OK
  - The rest is the **same** from last assignment

QUESTION: Repeat the same experiment after you implement multi-threading. Is there any difference in performance?

Without multi-threading

```
hang@hang-VirtualBox:~/Desktop/CMPE105/asgn1$ time(./httpclient localhost:8080 r
:aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaab:f1 & ./httpclient localhost:8080 r:aaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaab:f2 & ./httpclient localhost:8080 r:aaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaab:f3 & ./httpclient localhost:8080 r:aaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaab:f4)
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK

real    0m1.718s
user    0m0.005s
sys     0m0.204s
hang@hang-VirtualBox:~/Desktop/CMPE105/asgn1$
```

With multi-threading

```
hang@hang-VirtualBox:~/Desktop/CMPE105/asgn1$ time(./httpclient localhost:8080 r
:aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaab:f1 & ./httpclient localhost:8080 r:aaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaab:f2 & ./httpclient localhost:8080 r:aaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaab:f3 & ./httpclient localhost:8080 r:aaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaab:f4)
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK
HTTP/1.1 200 OK

real    0m0.718s
user    0m0.001s
sys     0m0.059s
hang@hang-VirtualBox:~/Desktop/CMPE105/asgn1$
```

Obviously, with the multi-threading methodology, the whole running time was reduced a lot. The multi-threading makes the multi request into different thread which can execute at the same time. In this way, much of time is saved during high volume of request received. This improve the ability for server to respond to client in a short time.

QUESTION: What is likely to be the bottleneck in your system? How much concurrency is available in various parts, such as dispatch, worker, cache? Can you increase concurrency in any of these areas and, if so, how?

The bottleneck would be the plenty of mutex lock. In order to make sure thread safety, I have to lock almost entire real processing part to avoid fd collision and read-write collision. Even though they may try to read and write different file with different fds at the same time, my server cannot distinguish the difference and has to lock all of them.

One way to enhance the performance is to find a way to distinguish different executions such as reading and writing different files and take off unnecessary concurrency controls.

In the other way, I should reduce shared variables in advance. With less shared data, less currency controls are needed, improving the execution time for each request.