Write up of ASG2: Multithreading and server caching

Yujia Li
CruzID: yli302

Write up questions:

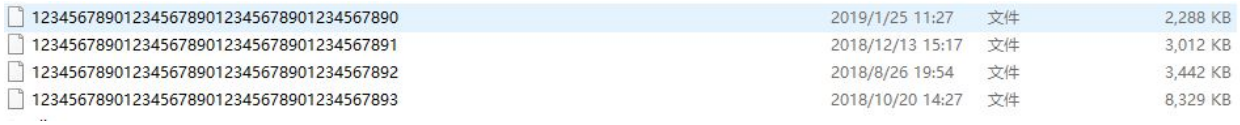
• Using either your HTTP client or curl and your original HTTP server from Assignment 1, do the following:

• Place four different large files on the server. This can be done simply by copying the files to the server's directory, and then running the server. The files should be around 4 MiB long.

• Start httpserver.

• Start four separate instances of the client at the same time, one GETting each of the files and measure (using time(1)) how long it takes to get the files. Perhaps the best way to do this is to write a simple shell script (command file) that starts four copies of the client program in the background, by using & at the end.

Filesize: 2288 KB, 3012 KB, 3442 KB, 8329 KB



Output:

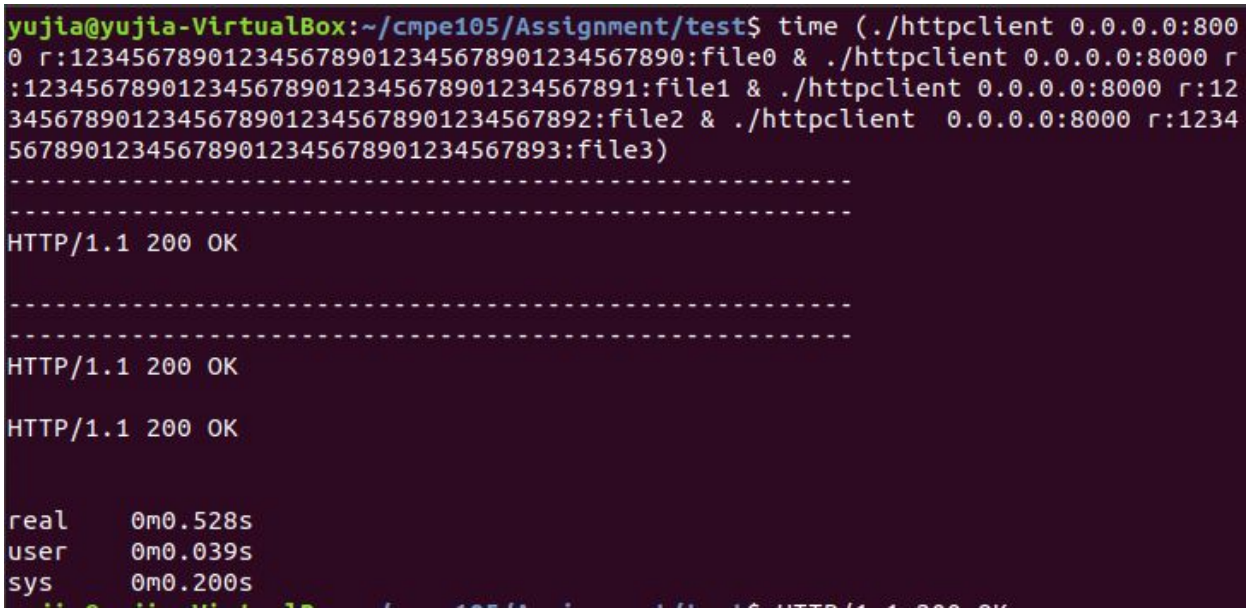
```
yujia@yujia-VirtualBox:~/cmpe105/Assignment/test$ time (./httpclient 0.0.0.0:800
0 r:1234567890123456789012345678901234567890:file0 & ./httpclient 0.0.0.0:8000 r
:1234567890123456789012345678901234567891:file1 & ./httpclient 0.0.0.0:8000 r:12
3456789012345678901234567890123456789 2:file2 & ./httpclient  0.0.0.0:8000 r:1234
5678901234567890123456789012345678 93:file3)
------------------------------------------------------------
------------------------------------------------------------
------------------------------------------------------------
------------------------------------------------------------
HTTP/1.1 200 OK

HTTP/1.1 200 OK

HTTP/1.1 200 OK

HTTP/1.1 200 OK


real    0m1.535s
user    0m0.031s
sys     0m0.221s
```

real 0m1.535s
user 0m0.031s

sys 0m0.221s

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

Output:



real 0m0.528s

user 0m0.039s

sys 0m0.200s

Multi-threading is much faster about 1s than original HTTP server.

• 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 must be the synchronization about shared variables, especially about read/write operation. In my implementation the semaphore about cache will affect dispatch thread, so the performance will decrease. I lock read and write starting at open() in the work threads, so most of the read/write operation wasted many time and this will lead to dispatch thread waiting for the read and write operation finish. Performance can be increased by separating the concurrency of cache and work threads. For example, when cache write/read to/from disk with different httpname file, the work threads do not need to wait. But it is difficult for me, because I don't know how to figure whether working work threads is working on a same httpname file.

TEST:

Multi-thread without cache can work with little bugs (see READEME)

Cache is not finished.