Despina Patronas
CSE 130

Write Up Document Assignment 2: httpserver

Testing

httpserver, should be similar to an http protocol server.

i. When Client sends curl requests (w/ GET, HEAD, or PUT) to *httpserver*, *httpserver* should perform the operation and respond back to client properly.

Testing Code for the spec:

Tests were done using Scripts provided by ***Christopher Lim*** and ***Donovan Henry***.

Other tests include various curl commands for *httpserver* and compares expected outputs to *httpserver outputs*.

Example Client Sends:

1. *(PUT)*
   curl -T file1 http://localhost:8080/file2 &
   curl -T file2 http://localhost:8080/file3 &
   curl -T file3 http://localhost:8080/file4

   Each Request should run in independent threads on server

2. *(GET)*
   curl -s http://localhost:8080/file1 -o file4 &
   curl -s http://localhost:8080/file2 -o file5 &
   curl -s http://localhost:8080/file3 -o file6

   Each Request should run in independent threads on server

3. *(HEAD)*
   curl -I http://localhost:8080/file7 &
   curl -I http://localhost:8080/file8 &
   curl -I http://localhost:8080/file89 &

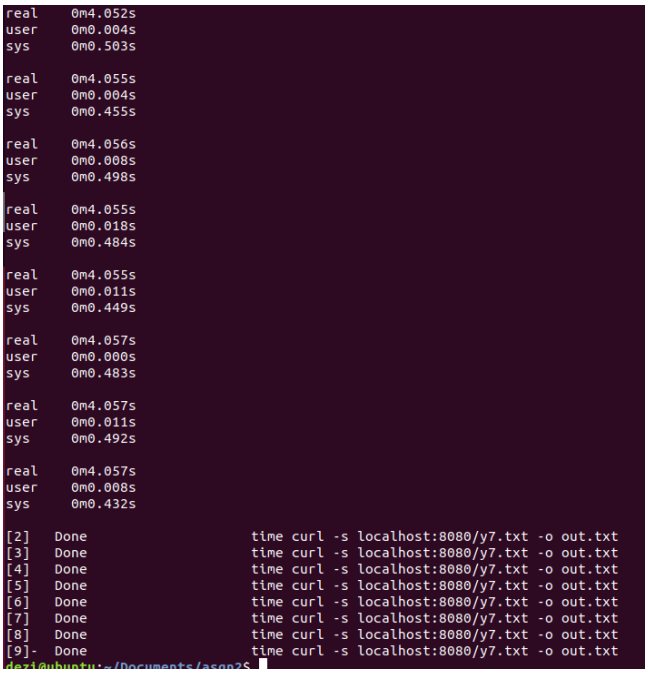
   Each Request should run in independent threads on server

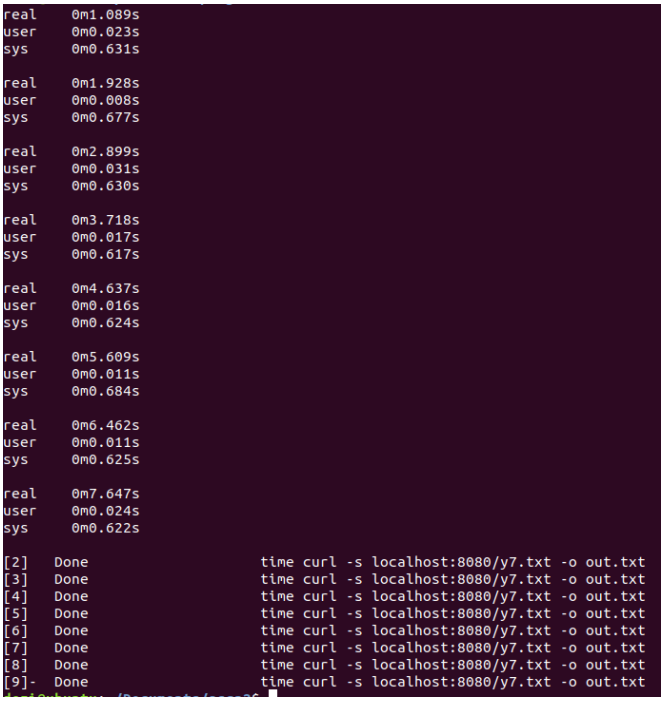
4. Mixture of *(HEAD) (GET)* and *(PUT)*requests
   curl -T file1 http://localhost:8080/file2 &
   curl -s http://localhost:8080/file1 -o file4 &
   curl -I http://localhost:8080/file7 &

5. Testing logging and healthchecks were done as well.

1) **Repeat the same experiment after you implement multi-threading. Is there any difference in performance? What is the observed speedup?**

Test file: **y7.txt**
size: **555.7 MB (555,666,777 bytes)**





The time it takes to do all threads at once on the multithreaded httpserver is constant. Each thread takes around 4 seconds concurrently.

The time it takes to do each thread on the single threaded system grows sequentially. After 4 processes it takes longer to do the work on equivalent test in multithreaded environment.

The speedup for the entire job is about 2x in the multithreaded environment.

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

Bottlenecks:
Varying time spent working on varying sized requests. Some threads take longer to complete jobs than others so there is an issue where you are only as fast as the slowest worker.

Dispatch:
Concurrency can be improved by having the listener in its own thread.

Workers:
Workers can be improved by distributed jobs more evenly across workers. Ex: having threads handle work that is divided by request / content length size type. Head requests are evidently much faster than arbitrarily sized GET or PUT requests. Having threads take turns handling varying sized content length files and Head requests could increase performance by distributed the work load evenly thus reducing the slowest threads time working.

Logging:
Logging may be considering a slow process since it is writing data and processing bytes in validation of logs. Logging may be contained in its own thread to increase speed up.