Natalie Petrosian
CSE 130 Principles of Computer Systems Design
Prof. Peter Alvaro

Assignment 2: Writeup

Testing

In this assignment, I implemented whole-system testing. The following is a subset of tests that the program should handle appropriately.

GET a small binary file (with __-- in the name)
GET a large binary file
PUT a small binary file
PUT a large binary file
Test HEAD request for a proper response header with 200 OK
Test for 400 with a bad HTTP version (HTTP 1.0)
Test if server creates a default number of threads
Test if server can read optional arguments in any order
Test if server can handle multiple GET requests simultaneously
Test if server can handle many requests greater than the number of threads
Test if server can log a single valid request
Test if server truncates log files every time it starts
Test if server can log many valid requests sent simultaneously
Test if server can log many valid and invalid requests (400) with multiple requests sent
Test if server can log many valid and invalid requests (400 and 404) with multiple requests sent
GET request to ‘healthcheck’ with no logging enabled (404 error)
GET request to ‘healthcheck’ with logging enabled
Test 403 error for a request to ‘healthcheck’
Test logging of many valid and invalid requests including a request to ‘healthcheck’

- Using either your HTTP client or `curl` and your *original* HTTP server from Assignment 1, do the following:
    - Place eight 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 400 MiB long.
    - Start `httpserver`.
    - Start eight *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. The best way to do this is to write a simple shell script (command file) that starts eight copies of the client program in the background, by using `&` at the end.
- Repeat the same experiment after you implement multi-threading. Is there any difference in performance? What is the observed speedup?
- 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?
- For this assignment you are logging the entire contents of files. In real life, we would not do that. Why?

Answers to Write Up Questions

Perhaps due to the large size of files, the difference in speed was not as impressive as I thought it would be. I'm assuming with smaller sized files, we may achieve closer to 30-40% performance increase.

The likely bottleneck in the system is probably due to the input/output operations, especially when they are in the critical sections of the code, protected by mutex locks.

For this assignment, we are logging the entire contents of files. In real life, we would not do this because of the bottleneck it creates. In real life applications, efficiency and speed are of utmost importance, and giving up verbosity in log files in order to achieve speed is desired.