Write-up Multi-threading HTTP server with logging

Jake Armendariz

Original HTTP Server:
3267433 bytes
real 0m0.506s
user 0m0.066s
sys 0m0.115s

352882764 bytes
real 0m25.851s
user 0m0.661s
sys 0m6.062s

After Threading:
3267433 bytes
real 0m0.303s
user 0m0.031s
sys 0m0.050s

352882764 bytes
real 0m12.754s
user 0m0.623s
sys 0m4.887s

• 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?

There are two possible bottle necks. The first, is how long it takes for threads to handle each request and print to log file. I lock the program when reserving space for log files, so I think this could slow down the process The other is creating an instance of the request object. When I add a request, it lock, add request to the end of queue. Only one request can be added at a time,

this can bottleneck the process as well.

The program runs sequentially to add the requests, and then to reserve space in logfile. Other than that, reading the request, processing, writing and sending is done concurrently.

To improve overall speed. I think I could combine writing to client/file in GET/PUT to only read once, but write twice. Instead my current process reads the file again when writing to log. BUT this would dramatically slow down http responses.

• **For this assignment you are logging the entire contents of files. In real life, we would not do that. Why?**

Two reasons:

1. It takes a very long time to log each file. Longer to log a file because it has to be processed and its log size > actual size. Thus logging an entire request would quickly create a massive file.
2. Its unnecessary. Why would we ever need the entire contents of the file. It is just unreasonable to keep the entire file conentes of every request on the log file.