Nathan Ang
nathanan

18-441/741 Project 3 Design Document

a. Your server design, model and components (add pictures if it helps)

My server class is called VodServer. In the main function it waits for connections from clients - if it has one, it will launch a thread to serve the client.

When serving the client, it continuously reads in from an InputStreamReader for requests. Upon receiving a GET request, the server will fulfill it based on what it requests. Based on the request, the server will call replyFileNotFound, replyWholeData, and replyPartial data appropriately.

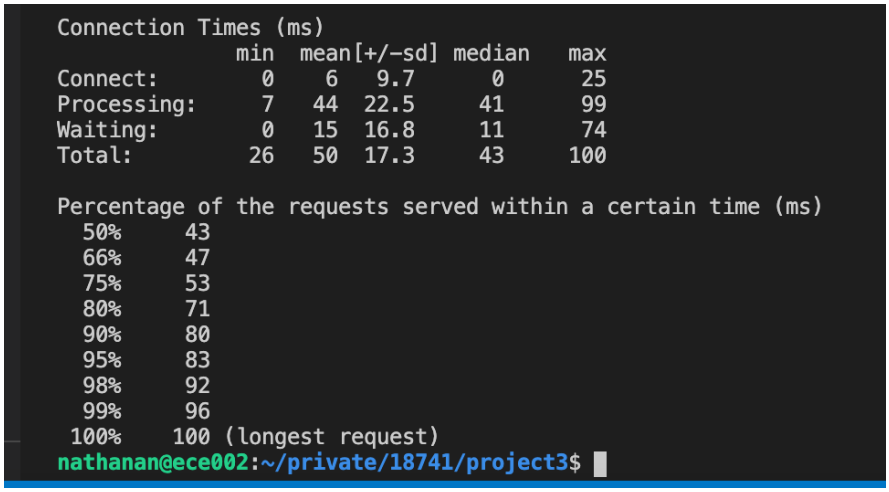
When sending replies, I followed HTTP for correct specific headers. These include the status code, Date, Connection: Keep Alive (so connection stays alive after response), Content-Length, Content-Type, Accept-Ranges, Content-Range. This is then followed by the data payload in the form of bytes. A special case is for serving video files, where if needed (video is larger than CHUNK_SIZE, or 1MB), we send Partial Content so the video can buffer efficiently.

After the client is finished, the server will close the connection and move on serving other clients.

b. How did you handle multiple clients? How many clients can your server handle effectively?

I used threading to handle multiple clients. I created my own class MyRunnable that implements Runnable in order to spin up a new thread to call serveClient. Whenever a new client connects, this thread is started, taking in the clientSocket as a parameter.

I was only able to test concurrency on the ece machine, which allows for 1000 users. This is my benchmark for serving 1000 concurrent clients requesting a 10KB image:



The handout states: “Try to see how many concurrent requests (for a 10KB content) you can have, while still allowing 95% of them to be served within **500ms**” and mine is able to serve 95% of them within **83ms**. Thus, I know for sure my server can handle over 1000 clients effectively and I think it will be able to serve 5000 clients effectively, as the handout asks for, due to the very low relative time of my benchmark.

I achieved this efficiency with threading and being strict in the sense that the server only does work when absolutely necessary. When the client is not asking for anything, the server will not do any work for the client.

c. Libraries used (optional; name, version, homepage-URL; if available)

SimpleDateFormat for date formatting.
DataOutputStream, InputStreamReader for file and socket I/O.
RandomAccessFile for serving requests that have Range within files.

d. Extra capabilities you implemented (optional, anything which was not specified in the Server Requirement section)

Last-Modified header to allow the browser to cache files.

e. Extra instructions on how to execute the code, if needed (WARNING: if we cannot compile and run your server on our cluster machine with the above execution method, your logistic points will be deducted. However, please do tell us any runtime parameter your server may have which may help in tuning)

Compile with “ant”.
Run with “java VodServer XXXXX” where XXXXX is the port number.