**Project1: Web Server Programming**

M2608.001200 Introduction to Data Communication Networks (2021 Fall)

Name: Yu Sangyoon Student ID: 2016-13343

**1. Describe what you implemented for 1) single-thread server, 2) multi-threaded server, and 3) thread-pooled server.**

All servers are share the structure of initializing the needed parts then wait for connection, if connection, get the integer for socket descriptor then process connection and send response using the implemented respond function.

Text

Description automatically generated Text

Description automatically generated Text

Description automatically generated

Figure Figure2 Figure 3

Text

Description automatically generated

Figure 4

In Fig1, receive and copy bytes from socket to buffer. recv() returns below 0 when error and returning 0 means disconnection. So handle error by exiting function. Then, from received buffer, get informations needed to process the request. Check for http version and send error code 505 if http version is above 1.1. After checking http version, we will match the response http version with the request http version.

In Fig2, if method is GET, check if the file requested exists. If file does not exist, send 404 error.

If file exits, send header first. Header includes Content-type using custom function sendContentTypeHeader() in Fig4.

Get file byte length(size), then send file bytes reading bytes by buffer, then send end “\r\n\r\n”.

Only GET is implemented, so other methods will send error code 405 METHOD Not Allowed.

In Fig3, shutdown and close sock.

Text

Description automatically generated Text

Description automatically generated Text

Description automatically generated

Figure 4 Figure 5 Figure 6

1) Single-Threaded server simply processes the connection synchronously. So as in Fig4, synchronous flow in single thread.

2) Multi-Threaded server creates new thread for every new connection as Fig5. To use pthread\_create(), modified respond function type and argument type as void\*. At end of respond, destroy thread.

3) Thread-Pooled server as in Fig6 initializes thread\_pool before accepting connections. Then for new connections, throw into the thread\_pool.

**2. Describe the characteristics of single-threaded, multi-threaded, thread-pooled methods  
(pros and cons)**  
  
Single-Threaded:

Pros: Low computation. Easy to Implement.

Cons: Bad for processing multiple requests at the same time, might loose connection due to queue or timeout

Bad when one client is using the thread by downloading a large file.

Multi-Threaded:

Pros: Good for processing multiple requests at the same time. Does not loose connection.

Cons: Number of threads to make is not known, so too much threads can be made. Creating and ending threads have computational overhead. Also if many threads are made, out of memory can occur. Switching tasks in multi-threads also causes overhead.

Bad when many http requests are sent at same time.

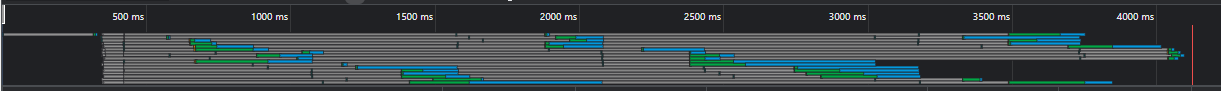
Thread-Pooled:

Pros: Can decide how many threads to be make so computationally safe. Creating threads only at the first place so low overhead.

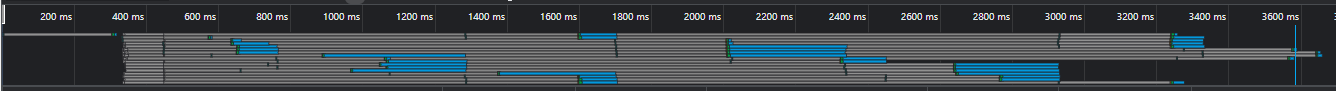
Cons: Multiple threads existing even when no requests are made.

**3. For the provided web page sources, capture network loading graphs (single threaded, multi-threaded, and thread-pooled) from Chrome developer tool and attach them here. Analyze and discuss each method based on the graph you captured.**

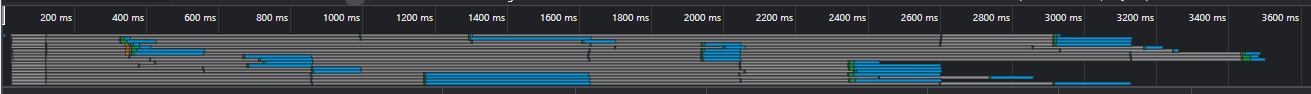
To maximize difference, I compared with loading the test.html page and it’s contents.



Single-Threaded Server



Multi-Threaded Server



Thread-Pooled Server

The overall time was similar. But the 2 main differences in the graph are proportion of blue/green part, which is waiting time after first connection. Parallel distribution of colors.

1. Green part is Time To First Byte(TTFB), which is waiting time for the first byte. Single-threaded server is much longer than other two ways. It is because even though the connection is reached, single-threaded server sequentially handle requests, so waits longer time.
2. Blue part seems to be parallel distributed in multi-threaded, thread-pool servers, but stairway shape is shown in single-threaded server. Blue part is content download, so we can check that multiple responses are sent at same time. Up to 6 responses are parallel, it is because chrome allows only max 6 TCP connections from the same origin at same time.

Why is there little overall time difference?

There are bottleneck of multiple threads such as context switching, and sending response is relatively not big enough than the bottleneck to show the difference.

Then can we show the difference?

We need another client!

It is because file size is small to cover the bottleneck and show the difference.

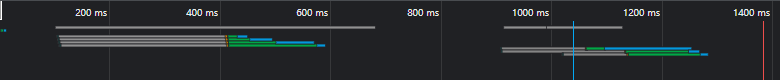
This will be tested at 4).

**4. Download web page sources from a web site (whichever you want to download) and put them into your web server. For those web page sources, capture network loading graphs (single threaded, multi-threaded, and thread-pooled) from Chrome developer tool and attach them here. Analyze and discuss each method based on the graph you captured.**

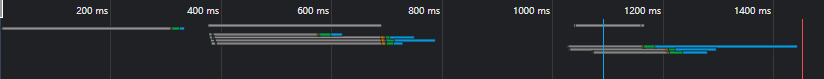
Graphical user interface, website

Description automatically generated

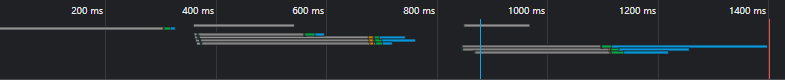
I tested with a website I made looking like above.



Single-Threaded Server



Multi-Threaded Server

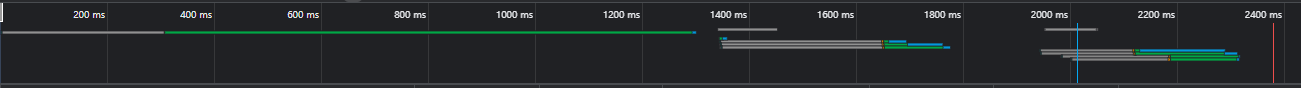
  
Thread-Pooled Server

As it is a simple website with few files, overall time was faster in single-threaded server very slightly.

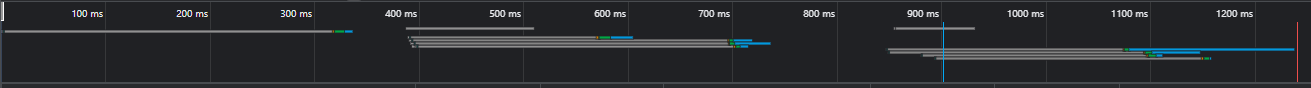
We can still see the 2 differences I mentioned at 3).  
  
But as we mentioned at 3), lets see if another client exists. This website contains a video, so lets make a scenario when some clients are requesting for video file and I want to open the webpage.

Python script code under will play the role for another client.

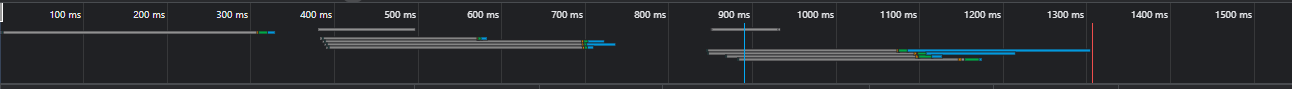
Text

Description automatically generated  
  
  


Single-Threaded Server



Multi-Threaded Server

  
Thread-Pooled Server

We can see that Multi-Threaded, Thread-Pooled servers perform similarly with no other client scenario, but single-threaded server had to wait because other client was requesting for the video.