Report 1: Homework Report 1

Yanghang Zeng

September 14, 2022

1 Introduction

In this assginment, I learn to build a simple web server using Erlang, and understand how a web server works. The program used functions in the gen_tcp library to handle the server connections. At last, the performance of the server is evaluated.

2 Main problems and solutions

To build a rudimentary web server, a HTTP parser is needed to parse received requests. Here, a module named http is used to extract the request line, headers and the message body.
To open a connection which is able to wait for incoming request, a module named rudy is declared. The module has follwing functions:

1. init(Port): this function create a server that listens to request on given port number.
2. handler(Listen): this function listen for incoming connections and pass the connections to the request function.
3. request(Client): this function read the request and then parse it using http module declared before, then pass the request detials, call the reply function.
4. reply(Request): this function takes a parsed request as parameter, and then generate a HTTP reply with a 200 status code to send back to the client.
5. start(Port): in the Erlang shell, use rudy:start(8080) to start the server on port 8080.
6. stop(): rudy:stop() to terminate the server.

In order to make the server running until it's terminated, i.e., listen to new connection after handling one, the handler() function calls itself recursively after executing the request parsing function.

3 Evaluation

After running command rudy:start(8080). , accessing the address "http://localhost:8080/foo" will get a response with 200 status code.

In order to evaluate the performance of the server, a small benckmark test was used to measure the time it costs to receive answers. The command to run the test is: test:bench("localhost", 8080). First, start the server and run the benchmark test for 5 times. Second, add a 40ms delay in the handling of request, and run the test agian. The result is the average difference between start timestamp and finish timestamp.

The result is as follow:

	Average Time
Without delay	138198.8
With 40ms delay	4687811.6

Table 1: Result of benckmark test

4 Conclusions

From the result of bench test, it is easy to tell that the artificial delay is significant in the request handling. Some possible improvements can be done for the server. One possible solution is creating multiple threads to handle multiple requests.

Through this assignment, I understand how Erlang works in terms of functional programming and concurrency. Besides, I learned how to build a simple web server and implement its rudimentary functions.