Report 1: Rudy - a small web server

Jiayi Feng

September 15, 2021

1 Introduction

Our task is to implement a small web server in Erlang. The aim of this exercise is that you should be able to:

- describe the procedures for using a socket API
- describe the structure of a server process
- describe the HTTP protocol

In this homework, a simple server is built and their mutual communication between the server and clients is investigated and record.

2 Main problems and solutions

Following the instruction of both HW0 and HW1, three erlang files, http, rudy, and test can be formulated. Codes are given in zip file, and then the battlefield turns to the power shell.
Building upon HW0's part 4 distributed system, two systems can be built with the command: **erl -sname foo -setcookie secret** or **erl -sname bar -setcookie secret**, and names are arbitrary decided.
Then, both rudy and test file can be executed with the command **c().**.
The server starts to work at port 8080 with command **rudy:start(8080).**, which is followed by the test in command **test:bench("localhost", 8080).**, for ethernet via optical fiber provides the Internet.
Several data emerges, three groups data, in Figure 1 is recorded with different sleep times.
Then, the bonus point comes. Throughput is investigated in this part, and the variable is controlled, where 4 clients, each N requests, are fixed and the number of servers/handles/ports are ranging from 1 to 5, in Figure 2
rudy:stop(). command is useful in both of these two tasks, for parameters' modification.

3 Evaluation

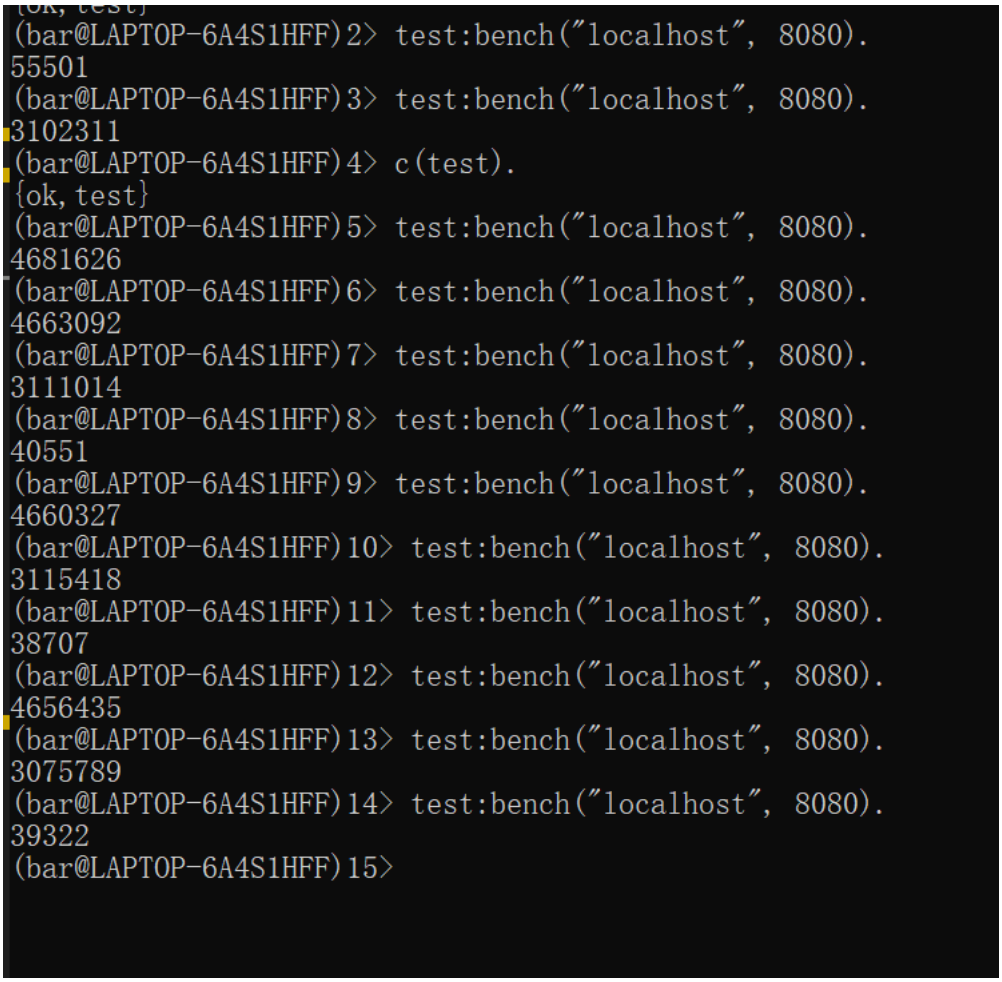

Figure 1: different sleep time under 100 requests, three group data

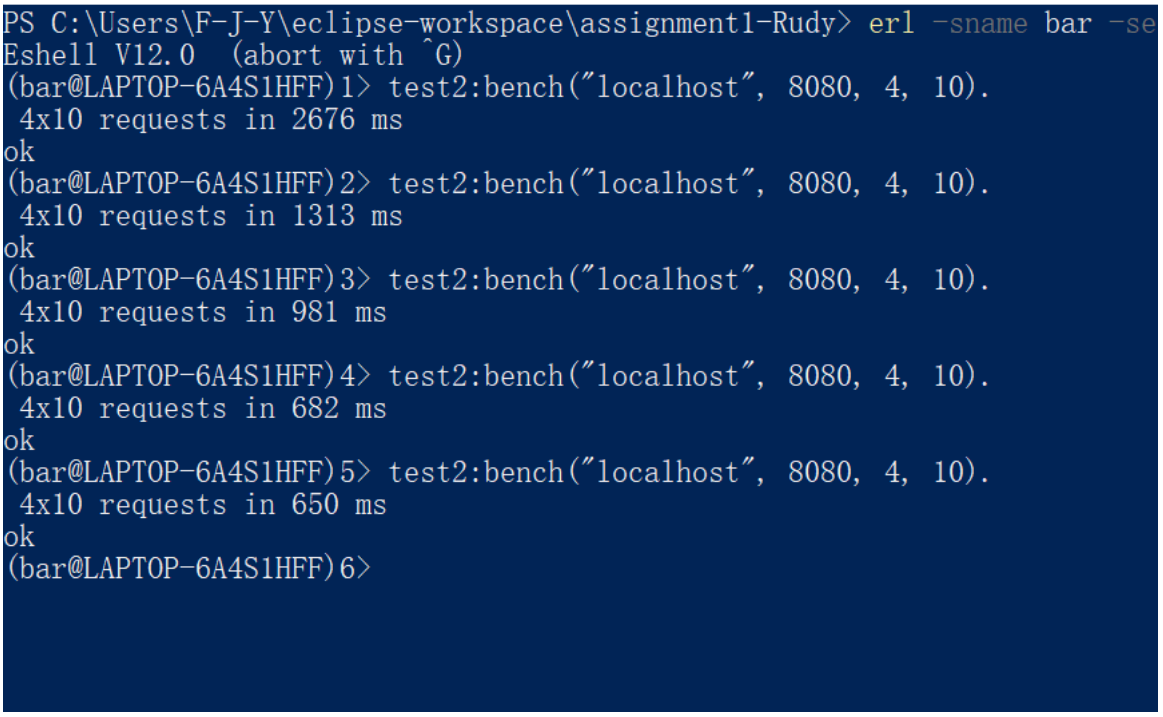

Figure 2: different server numbers under 40 requests, 4 clients, each 10 requests

These two figures are the experimental raw materials. They will be turned into visible tables instead, before analysis and evaluation.

40ms delay	20ms delay	0ms delay
4663092us	3111014us	40551us
4660327us	3115481us	38707us
4656435us	3075789us	39332us

Table 1: different sleep time under 100 requests, three group data

server number	time
1	2676ms
2	1313ms
3	981ms
4	682ms
5	650ms

Table 2: different server numbers under 40 requests, 4 clients, each 10 requests

Building upon tables originated from raw material, the average value will be given first in:

40ms delay	20ms delay	0ms delay
4659951us	3100761us	39530us

Table 3: averaged execution time over 100 requests

Then the request per second can be reached as:

40ms delay	20ms delay	0ms delay
21.46	32.25	2529.72

Table 4: request per second with delay

4 Conclusions

As for part 3, our programs have been successfully compiled, and data has been shown in the evaluation part.
When it comes to the bonus part, it is clear that when the server number exceeds the client(number 5), the system reaches its maximum efficiency, and other servers are in idle state. In addition, less time will be consumed with relatively higher server number, with fixed client number.
This is the first homework, so I have learned only the basic knowledge. Hoping to seeing you in HW2.