Rudy: a small web server

Mikael Karlsson

September 9, 2019

1 Introduction

This report describes to work of implementing a simple web server in Erlang. The aim of the work was to gain an understanding of a server process, how to use a socket API and gain basic knowledge about the http protocol.

A web server is a distributed system where a server and client can communicate with each other through message passing.

2 Main problems and solutions

The first part of the assignment was to implement a simple HTTP parser. The HTTP parser code was provided so the main problem was to get an understanding of how the code handles a HTTP request.

The code is traversing a string recursively searching for three components: request lines, headers and body. Request and Headers and each header are separated by (\r\n). The headers and body are separated by (\r\n\r\n). A request with two headers and a body could look like this:

```
GET /index.html HTTP/1.1\r\nHEADER1\r\nHEADER2\r\n\r\nBODY
```

The second part of the assignment consisted of implementing a simple web server that can reply to a HTTP request. Most of the code was provided, so the task was to fill in small sections of code. The server could only handle one request at the time, so I implemented a function that increases the throughput.

The throughput is increased by creating several handlers. Each handler is created by spawning a new process:

```
worker(Listen , Number_of_workers) when Number_of_workers == 0 -> ok;
worker(Listen , Number_of_workers) when
Number_of_workers > 0 ->
```

```
spawn(fun() -> handler(Listen, Number_of_workers) end),
io:fwrite("worker ~p created\n" , [Number_of_workers]),
worker(Listen, Number_of_workers - 1).
```

3 Evaluation

Each request to the server has a 40 micro-seconds artificial delay. All tests were conducted on my personal computer. This means that the server and clients were separate terminal windows on the same computer and not individual computers

Test results			
Requests	Handlers	Response time	Clients
100	1	4197	1
100	4	4198	1
1000	1	42066	1
1000	4	41946	1
100	1	14411	4
100	4	4192	4
1000	1	161887	4
1000	4	42028	4

4 Conclusions

The result shows a server with 4 handlers will handle 4 clients approximately 4 times as fast as if the server only have 1 handler. The result also shows that number of handlers doesn't decrease the response time when there only is one client. This is because of the provided test-program. The test-program that is started by a client is waiting for a response before sending next request.

In summary, this assignment provided a good introduction to the HTTP-protocol. It was a good first assignment to be introduced to the programming language Erlang and how powerfull it can be in a client server implementation.