Homework 1 Report

Rafat Khan

September 15, 2021

1 Introduction

The purpose of this homework is to implement a small web server in Erlang and learn about protocols in the process. Client server architecture is basic to distributed systems. This seminar is about learning the basics of this communications. Learning about the implementation of socket API, structure of a server process, client server communication using TCP through HTTP request, distribution and communication are the main aspect of this seminar.

2 Main problems and solutions

The assignment is to create a simple server and do some experiments. Most of the basic functionality was described and the code was given. There were some missing piece of code which needed to be fixed in order to implement the server.

For example, gen_tcp:listen(Port, Opt) opens a socket, which should be passed to handler/1

```
{ok, Listen} -> handlers(Listen)
```

When a client connects, the handler should pass the communication channel to request/1

```
{ok, Client} ->request(Client)
```

After that, the byte stream of client request should be parsed (interpreted according to the HTTP specification) using parse_request/1 function from the http module

```
Request = http:parse\request(Str)
```

Another problem with this server is the low throughput. It initiates only one handler which runs as a single process. This means all the requests are processed sequentially which results in bigger latency. Using multiple handlers which will run in parallel, this problem can be minimized. On this case, requests will be divided among the handlers which will result in faster execution.

For creating multiple handlers, handlers/2 function is used. It run a loop to create N numbers of handlers. spawn/1 is used to create a connection between the caller and the process. In this way, N numbers of handlers keep listening to a single socket concurrently and they can attend to multiple the client requests simultaneously.

```
handlers(Listen,N)->
case N of
0 ->
    ok;
N ->
    spawn(fun() -> handler(Listen,N) end),
    handlers(Listen, N-1)
    end.
```

3 Evaluation

After running the server, http request were made to the server using different shell. Various tests have been performed within the system. A small delay (40ms) was added in the handling of the request to simulate file handling, server side scripting etc.

In the first experiment, 5 client processes sent 50 requests each to the server. In each test case the response time was measured against various number of active handler processes. In the first test case, only one handler process was initiated which processed the requests in 12031 ms and in the last test case, eight handler processes processed the requests in 2406 ms, which indicates a 80% speedup in the response time. These test cases shows that the increment of the handler process significantly reduce servers response time.

Test No.	Handler Processes	Time (ms)	Speedup
1	1	12031	0
2	2	6000	50.2%
3	4	3250	72.9%
4	8	2406	80%

Table 1: Response time for different numbers of active handlers.

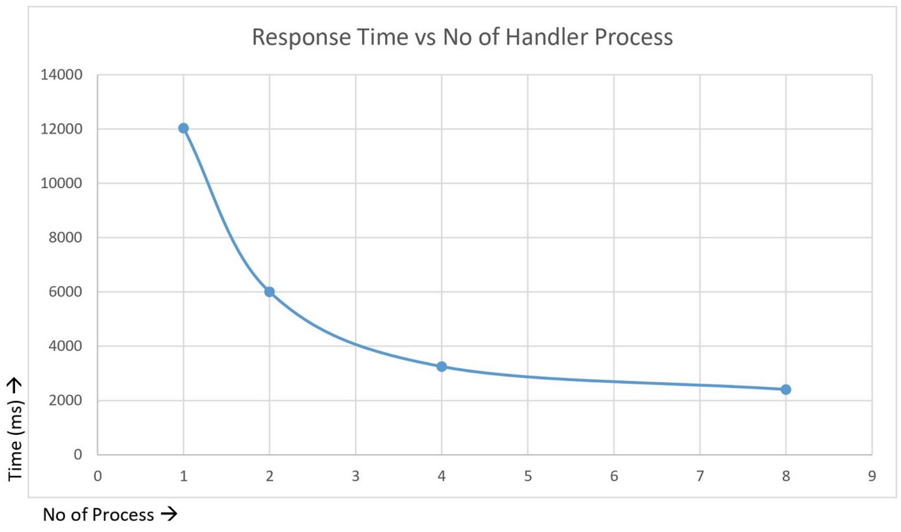

Figure 1: Response time for different numbers of active handlers.

In the second experiment, 100 requests were to the server from 1 client process (sequential) and from 100 client process (parallel). In each test case the response time was measured against various number of active handler processes. requests are processed sequentially and the extra handler processes remain idle. On the other hand parallel execution shows that the servers response time decrease with the increase of the number of handler processes because multiple requests can be processed in a parallel manner.

Table 2 and Figure 2 shows the output and it's graphical representation for each test case.

Test No.	Handler Processes	Sequential Execution (ms)	Parallel Execution (ms)
1	1	5208	1734
2	2	5375	1335
3	4	5328	1063
4	8	5375	719

Table 2: Response time for sequential and parallel request with different numbers of active handlers.

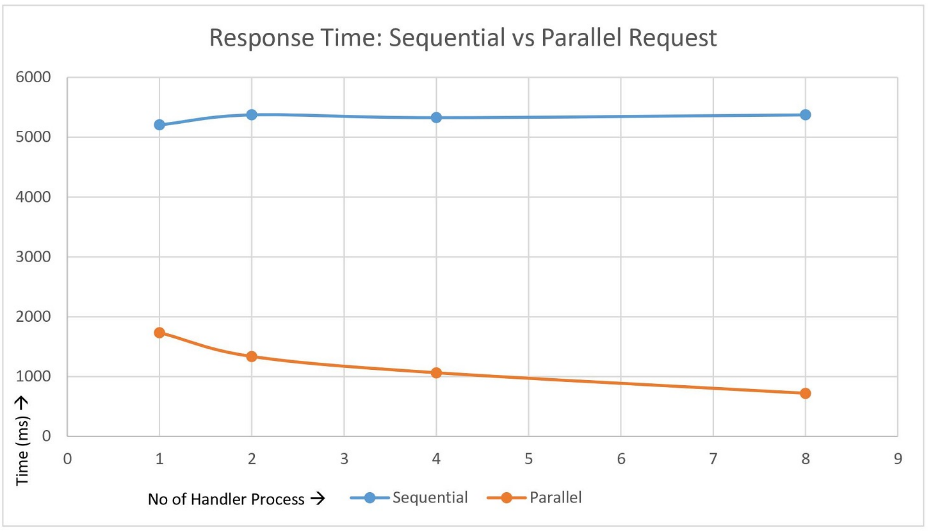

Figure 2: Response time for sequential and parallel request with different numbers of active handlers.

4 Conclusions

The problem that was presented made me understand the importance of concurrency for a server. It also made me understand the concept of a client server communication model. It also made me curious about the error handling issues and the robustness. As a bonus, I have learnt to write and execute Erlang code.