Report 1: Rudy

Nagasudeep Vemula

September 11, 2019

1 Introduction

The project mainly deals with the creation of a simplistic web server named Rudy using the Erlang programming language.The main purpose of this exercise is to gain knowledge of the following concepts:

1.Understanding of the client-server architecture which forms the foundation of distributed systems.

2.Process for implementing a Socket API.

3.Passing message through TCP protocol.

4.Parsing of a http message and in turn learning about the components of the http header.

This project largely touches upon the topics of communication and implementation of a server to pass messages and receive requests for both single and multi-threading. This also gives an introduction to the concept of concurrency that is useful in configuring for systems which need a lot of scalability and efficiency at the same time.

2 Main problems and solutions

The problem to be solved mainly is to send repeated requests to the server and receive the response quickly. These requests are sent using the gen-tcp library which provides functions to communicate with sockets using the tcp ip protocol. Setup of a communication channel between the client and host needs to take place to enable the transfer. The message should be parsed in order to identify its header and body portions and then should be sent back to the client. Handlers have been used to take care of the scenario where multiple requests sent by the client can be efficiently handled on setting up the connection.

Furthermore the enabling of multi-threading is done in the application with the help of handlers which reduce the overall time for sending and receiving messages. Every handler has a separate process spawned for handling the request.

3 Evaluation

The results of the project are displayed with the help of graphs and tables below. Measured benchmarks are mainly for throughput and response time and this has been done for both single and multi-threading.

Single Threading: The below graph is to show the performance while using a single thread for the execution of the process. It can be seen that time taken is directly proportional to the number of requests.

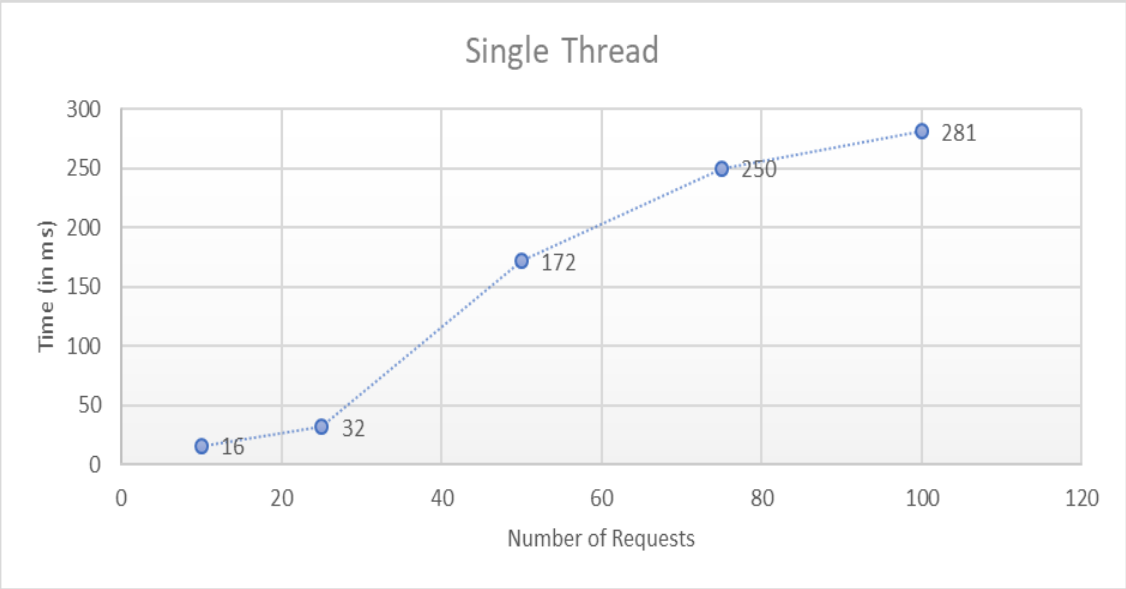

Figure 1

Number of Requests	Time taken to execute(in ms)
10	16
25	32
50	172
75	250
100	281

Table 1

Multi Threading: The below graph is now to show the performance when multiple threads are used.It is seen that with the increase in the number of threads the time decreases and this is very useful for systems that need to handle multiple requests at a time.

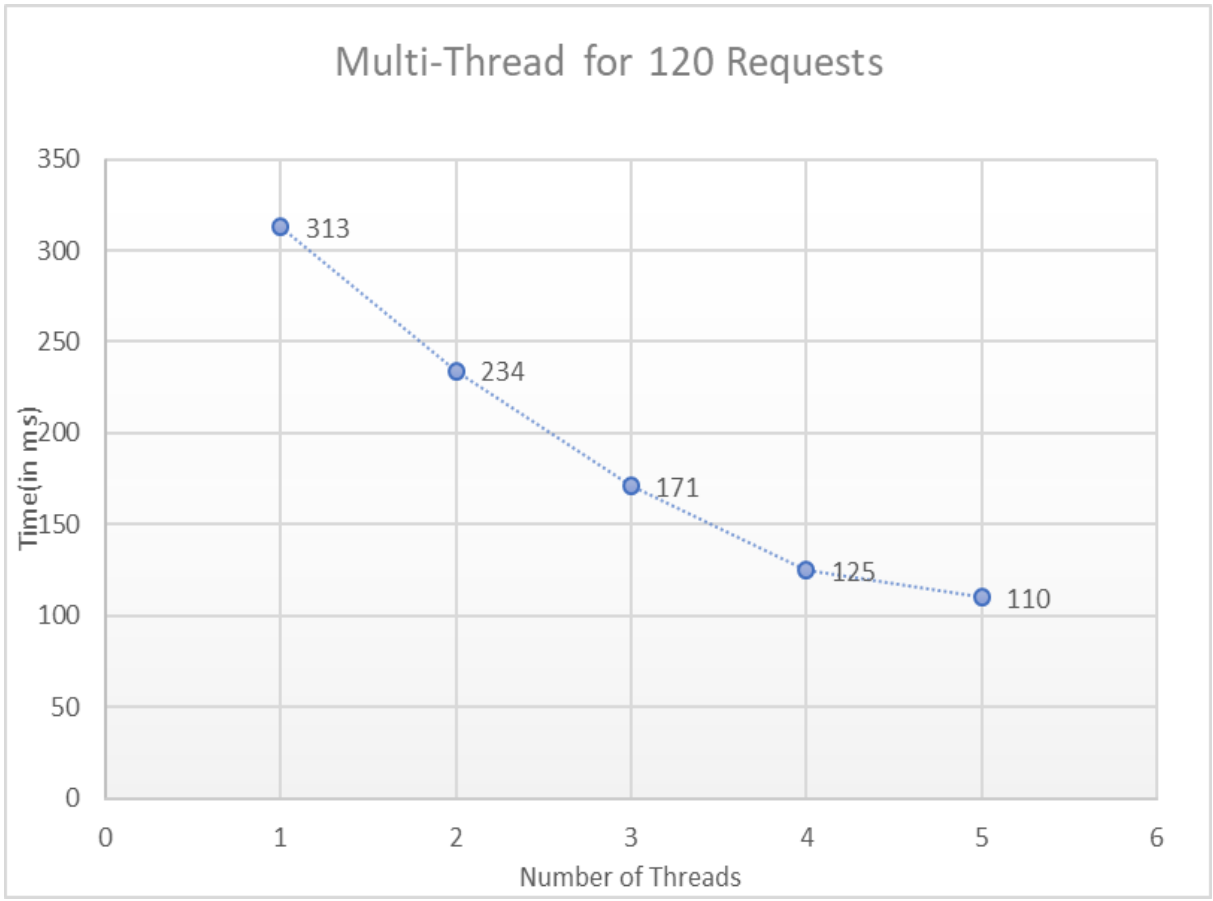

Figure 2

Number of Threads	Time taken to execute(in ms)
1	313
2	234
3	171
4	125
5	110

Table 2

4 Conclusions

The assignment has extended my knowledge of the erlang syntax and in turn given me greater understanding of how concurrency is enabled by it. Key takeaways would be the implementation of the server Rudy by learning to setup the connection with the help of init, handler, request and response. Also gained understanding of parsing of a HTTP header, Socket Api setup and architecture of client server.

Overall it has helped me strengthen my foundational knowledge in distributed systems by giving me a peek into the above mentioned concepts.