Project Title : Asynchronous TCP Server in Modern C++

Name : Shakil Ahmed
Reg : 2017331024
Session :2017-18

Project Github Repository : https://github.com/ki9gpin/TCPServer
Deployed on Heroku at : https://cppy.herokuapp.com

Introduction:

This project implements an Asynchronous TCP Server using Boost Beast library .
This server serves a full fledged WebApp with . The WebApp functions as an online compiler for C++ sources code. From frontend C++ source code is sent to server .

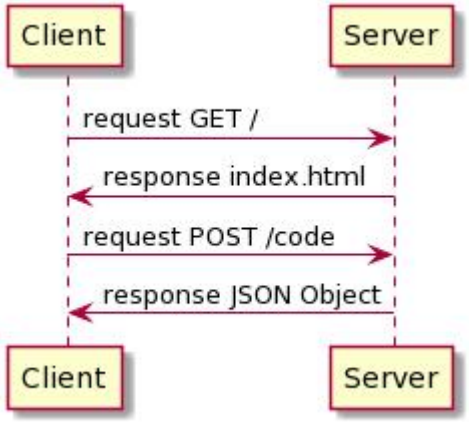

Figure - Client Server communication

Figure depicts communication state once client has successfully connected to server over SSL.
The server can handle such multiple connections simultaneously.Asynchronous I/O of Boost Asio library is helps to get that behaviour.

Motivation :

Web Servers are typically developed in high level languages like Java ,C#,JavaScript,C#, Ruby and others using provided standard web framework.C++ has evolved a lot and Modern C++ brings many new features like lambda expression , smart pointers which makes programming in C++ easier. Many handy libraries are available like JSON library(JSON parsing) , libfmt(logs and pretty prints) format library which help finish a specific programming easier.So now many help is available around . Boost is a collection of libraries that fall into different categories like mathematics ,graphics,statistics.Developing a Software Solution in C++ compared to other languages but it pays off by high performance and maintenance in the long run and survey shows despite being harsh a lot of developers' favourite language is C++.

Libraries Used:

Boost Beast is a communication library which is written in modern C++ ,a wrapper for Boost Asio and makes HTTP protocol implementation easier.
Boost Asio is a bare metal TCP/IP protocol implementation creating abstraction around winsock on windows and socket on unix platforms.
Other C++ networking frameworks are available such as WT by QT foundation and CPPRESTSDK by microsoft but using Beast and Asio provides control and flexibility.
For JSON parsing the famous nlohmann/json library is used which is a single header file.

Server :

On start up Server reads PORT environment variable.If it is not set then server uses port 8080. It starts a TCP acceptor method on a socket bound it to that port waiting asynchronously for listening to incoming requests . When a new request arrives it creates a new socket for the client. The client from then on to uses that socket to start a new session which is automatically bound to a port. The session then manages the client that is communicating that port.

Server takes source code input from HTTP request and saves on server file system storage. Then using system call popen we create a pipe whose other end executes our provided command to compile the source code file and execute it. The output result is read from pipe. Then result is sent back to frontend as a JSON object .

Client :

In the frontend the user gets a textarea to code and a button. It sends C++ source from client to server through an AJAX call. And then waits asynchronously for server response through another AJAX call.
When server sends the JSON Object as body of response it parse it out and shows the desired result to output area.

Outcome:

C++:

Acquiring skill to be able to read C++ Source Code of real life implementation of software such as Open Source Projects . These Open Source Projects provides us with practical implementation of CS topics like efficiency of CPU Time and disk space , OS,Thread and Concurrency,Memory management , Compiler,Networking, Software Design Patterns (like Boost uses Executor Pattern to dispatch event handler and then use these handlers handle event) and Design Choices. So removing language barrier for entry to these resources.

Memory management techniques using smart pointers like shared_ptr,unique_ptr
Object Oriented design choices , Inheritance ,
Object lifetime , Curiously recurring template pattern,Compile time Computing,
Runtime and Compile time
Lambda expression ,Functional aspect of C++
Efficient use of Standard Template Library
Using external libraries ,Dynamic linking and static Linking to them and how this works

Networking:

Gathering practical insights into the TCP/IP implementation process and necessity of different protocols and often their different versions.

Build System and Development tools:

Cmake and Make combination is a useful build system for C++ development.
Debugging in Visual Studio is a great way to gain insight of a program runtime as it can visualize Individual instruction and assembly execution .

Deployment :

Docker is used to containarize a software with its dependency . Deploying a C++ software system to cloud helps gather practical skill for real world deploy and shipping scenarios.

Security :

Using SSL and TLS in server provides practical knowledge of SSL handshake, certificate generation and concept of issuing certificate from Certificate authority. (Heroku manages security themselves and trying to perform SSL handshake manually results in error. So the version deployed on heroku does not use SSL encrypted communication but local one while developed on Visual Studio uses SSL encrypted communication)

Posix System API:

Gain Posix System API familiarity by implementing Online Compilation process on server. Using Pipe to communicate server process with GCC compiler and final executable on server side for acquiring compilation and execution output.