Design document: Lab 1

Lab Description:

1) There is a pandemic caused.

2) Gauls are confined so taken up trading meme stocks.

3) Three parts in lab:

i. Design your own thread pool.

ii. Use gRPC and built-in thread pool to write server client application.

iii. Performance measure/Evaluation part

Part 1

(Python Socket architecture)`

Server

Overview:

The Stock Name Lookup Server is a server application that receives requests from clients, looks up the requested stock prices and volumes, and sends the requested information back to the clients. The server uses a thread pool to handle multiple clients simultaneously and is designed to be fast, responsive, and scalable.

Architecture:

The Stock Name Lookup Server is built on a client-server architecture. The server listens for incoming connections using Python's socket module and accepts incoming connections using the accept() function. When a connection is established, the server creates a new thread to handle the client's request using the handle_client() function. The handle_client() function receives the stock name from the client, looks up the stock price and volume, and sends the requested information back to the client using the send() function. The server uses a thread pool to handle multiple clients simultaneously, with each client being handled by a separate thread.

Functional Design:

The Stock Name Lookup Server is designed to be scalable and handle multiple clients simultaneously. The server listens for incoming connections using the listen() function and accepts incoming connections using the accept() function. When a connection is established, the server creates a new thread to handle the client's request using the handle_client() function. The handle_client() function receives the stock name from the client, looks up the stock price and volume, and sends the requested information back to the client using the send() function. The server uses a thread pool to handle multiple clients simultaneously, with each client being handled by a separate thread. The thread pool is designed to be dynamically scalable, with new threads being created as needed and old threads being reused when available.

Non-functional Design:

The Stock Name Lookup Server is designed to be fast and responsive, with minimal latency between receiving a request and sending a response. The code is well-structured and easy to read, with comments to explain each step of the program. The server is designed to be robust and handle errors gracefully, with error messages displayed to the user when necessary. The server is also designed to be secure, with data encrypted using UTF-8 encoding before being sent to the client. Overall, the Stock Name Lookup Server is a reliable and efficient tool for retrieving information about stocks.

Part 2:

(gRPC Implementation)

Overview:

This is an implementation of stock trading service which uses gRPC, allows users to perform stock lookups and trades and update stock prices. This is a server-client architecture where servers run as a gRPC server and client communicates via gRPC. The service will be implemented in Python 3, using the gRPC library for Python.

Architecture:

The Stock Trading service will have two primary components: the server and the client. The server will implement the StockTradingService class, which will contain the methods for performing stock lookups, trades, and updates. The client will be any system that can communicate with the server using gRPC such as a web application.

Stock Catalog:

The server will maintain a stock catalog that contains information about the stocks available for trading. The catalog will be initialized with four stocks: GameStart, FishCo, BoarCo, and MenhirCo. Each stock will have an associated price and volume, which will be updated as trades are performed.

Service Methods:

The StockTradingService class will implement three methods: Lookup, Trade, and Update.

Lookup:

The Lookup method will take a stock name as input and return the current price and volume of the stock. If the stock does not exist in the catalog, the method will return a price of -1 and a volume of 0.

Trade:

The Trade method will take a stock name, amount, and buy/sell flag as input and update the volume of the stock based on the trade request. If the volume of the stock exceeds its max volume, trading for that stock will be suspended. If the stock does not exist in the catalog, the method will return a status code of -1. If the trade is successful, the method will return a status code of 1. If trading is suspended, the method will return a status code of 0.

Update:

The Update method will take a stock name and price as input and update the price of the stock in the catalog. If the stock does not exist in the catalog, the method will return a status code of -1. If the price is invalid that means it is less than 0, the method will return a status code of -2. If the update is successful, the method will return a status code of 1.

References:

1) Implementing gRPC in Python: Step by Step Guide - https://www.velotio.com/engineering-blog/grpc-implementation-using-python

2) Socket Programming HOWTO - https://docs.python.org/3/howto/sockets.html

3) Tutorial to gRPC in Python - https://grpc.io/docs/languages/python/basics/

4) Socket Programming in Python(Guide) - https://realpython.com/python-sockets/

5) An intro to threading in Python - https://realpython.com/intro-to-python-threading/#starting-a-thread

6) Parallel Programming in python - https://docs.python.org/3/library/concurrent.futures.html

7) ChatGPT - https://chat.openai.com