CS-677 DESIGN DOCUMENT

INTRODUCTION

1. Project Overview

The overall project is divided into 3 sections –

a. Part 1 involves designing and implementing a stock bazaar server using socket connections and thread pool that can accommodate multiple client queries. Server maintains a list of stock names along with prices, which upon requests from clients should process and send the data back.
b. The second phase involves the development of a multi-client server that will handle gRPC-based lookup and trade requests from clients.
c. Part 3 involves performance evaluation of above-described models by varying the load on the server.

2. Goals and objectives

The project's objective is to create two distinct server-client models capable of handling multiple requests from multiple clients concurrently. Finally, assess both the model's performance by plotting graphs.

DESIGN AND IMPLEMENTATION

3. Architecture and Design

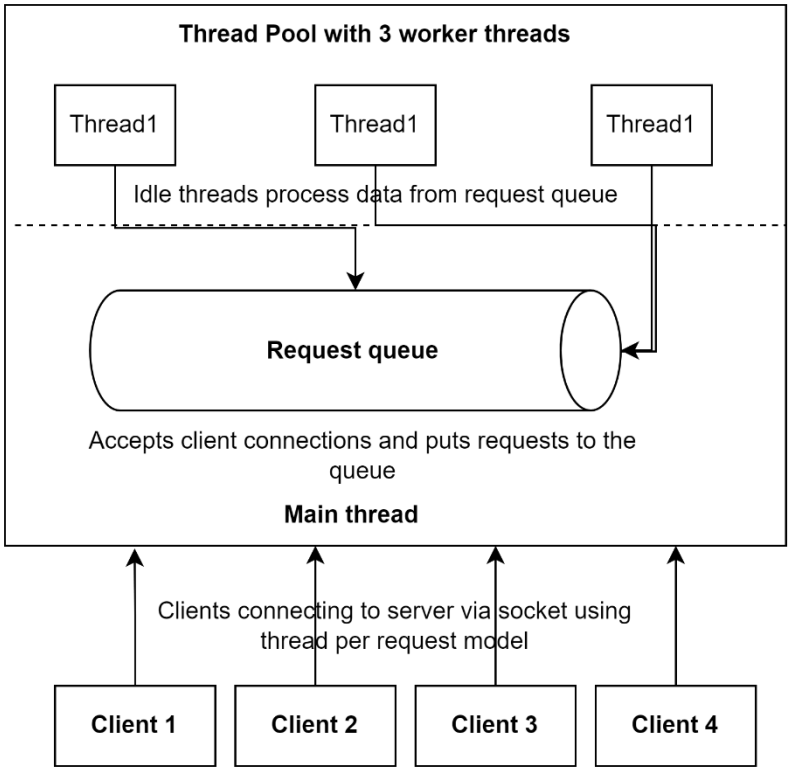

Fig. Part 1 design

The design includes creating a thread pool with fixed number of worker threads on startup which keep polling for requests from the request queue. Server is designed using thread per request model, where main thread keeps waiting for client connection. Whenever a client connects to the server via socket the request is added to the queue by main thread and connection is closed by the client thereby achieving thread per request model. If the client must send another request, it has to open a new socket connection to the server.

Since, on startup the thread pool is created, whenever there is an idle thread in the pool it picks up request from the request queue and processes it.

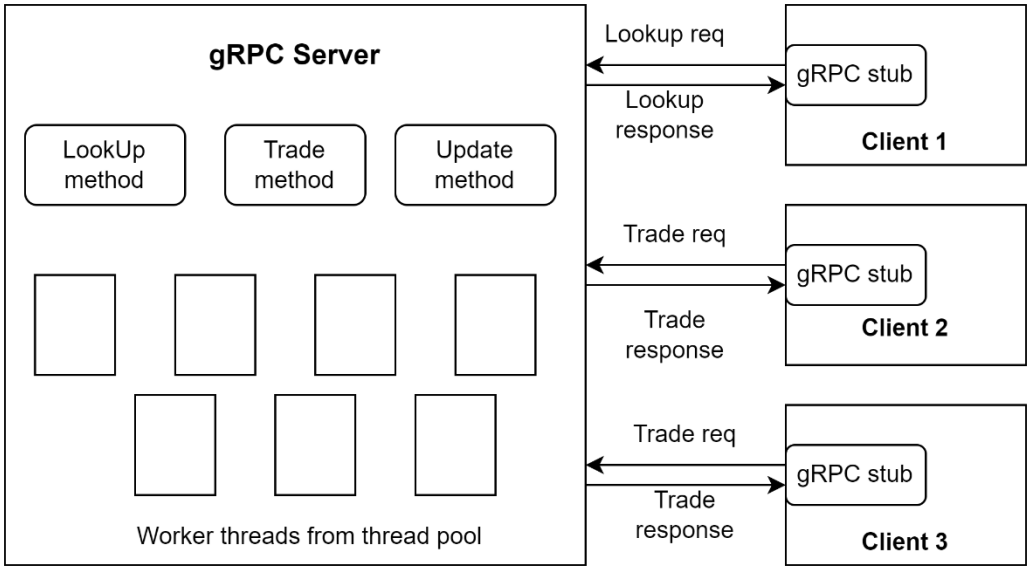

Fig. Part 2 design

Part 2 is designed using a built-in thread pool and gRPC for server client connections. A protobuff is implemented where all the methods, requests and responses are defined. Server design is such that client connections are accepted using gRPC and requests are handled by thread pool. Three methods are defined in server namely Lookup which returns stock price, Trade using which one can buy/sell any stocks in the stockbazaar and update method which updates the stock prices periodically.

4. Implementation Details

Part 1

I. ***Programming Language*****:** Python

II. ***Data storage:*** Dictionary is used to store all stock related data. Stock name is used as key, value is a list which has 3 items – price, volume and max trade possible.

connection_req_queue: is used to put all the incoming requests along with their connections.

III. Methods/Interfaces:

lookup(stock_name) method which returns stock price if the requested stock is in the maintained dictionary.

Process_connections(conn_q): method which gets the requests from the queue and internally calls lookup method to process the request and sends the response back to the client.

Part 2

I. Protobuff: stock bazaar.proto file is used to define all 3 services, their request and response messages

II. Methods/interfaces:

Lookup(self, request, context) method replies the stock prices for the requested stock name

Trade(self, request, context) method updates the available number of stocks in the stock bazaar dictionary based on buy/sell requests from the clients. This method uses lock to avoid discrepancies when both the methods are trying to access the dictionary.

Update(self, request, context) method updates the prices of the stocks periodically.