Part2: Implementation with gRPC and Built in Thread Pool Design Document

Creating distributed applications and services is easy with gRPC, as it makes it easier for a client application to call the methods on a server which is existing in another machine directly.In gRPC, we will define the services, specify the methods that can be called remotely by the client.

gRPC uses protocol buffers for serializing the structured data. In the proto file(toy_store.proto), the protocol buffer data is structured as messages - ToyStoreSearchQuery, ToyStoreSearchResponse, ToyStoreBuy and ToyStoreBuyResponse. Each of these messages has name-value pairs in it. Once the .proto file is created, we have to generate the data access classes using the protocol buffer compiler-protoc. We have generated the protocol buffer compiler in python using the below command :
python3 -m grpc_tools.protoc --proto_path=. ./toy_store.proto --python_out=. --grpc_python_out=.

The generated data access classes provide accessors for the fields mentioned in the messages- ToyStoreSearchQuery, ToyStoreSearchResponse, ToyStoreBuy and ToyStoreResponse. We have also specified the simple gRPC services in the proto files - Query, BuyItem here. When protoc is used above, a gRPC server and client code is generated and also the protocol buffer code for populating, serializing and retrieving the message types will be generated. The generated server and client files are called as toy_store_pb2.py, toy_store_pb2_gRPC.py

The server and client files have messages that are defined in toy_store.proto. It has classes-ToyStoreStub, ToyStoreServicer for the service defined in toy_store.proto. ToyStoreStub is used by toystore_client to invoke the RPCs and the servicer defines an interface for implementation of the ToyStoreServicer service.

The next step in designing this ToyStore is creating the ToyStore server. Here we have implemented rpc services Query(ItemName) and BuyItem(ItemName). Based on the toy name provided by the clients, the server returns the response to the client using the structured response messages- ToyStoreSearchResponse, ToyStoreBuyResponse. We can start the server using the serve start() method which is a non-blocking method. A new thread will be instantiated to handle requests. We have implemented toystore_client.py for calling these rpc methods. We are retrieving the response from the server and displaying its values.

Part2: Implementation with gRPC and Built in Thread Pool Design Document