ECE 454: Assignment 1 Design Document

Simarpreet Singh ID#20434209
Caitlin Pinn ID# 20368217

The chosen Thrift synchronization interface was IFace since it provided reliability over speed. The application needs to guarantee that a service request will have a response rather than risk the chance a response may never get sent back.

In order to try and maximize throughput, the TThreadPoolServer was used for both the management and password ports of FE Seeds and FE Servers. The TThreadPoolServer is not the strongest in terms of latency since many concurrent connections are held but it was decided to try and maximize the throughput since there would be many clients making many requests on the BE Servers so a lot of requests would need to be processed. A TThreadPoolServer was also used to deal with both the password port of BE Servers since each BE needed to perform the password hashing and checking quite often for the clients. The BE Server management port uses a TNonblockingServer since the management port is likely to be less used and doesn’t require as many extra connections as the password port and can be used with the one thread TNonblockingServer utilizes.

Transport protocols in the application utilized the TBinaryProtocol for serializability and de-serializability purposes. The TBinaryProtocol was selected since it is able to transport the data most efficiently and for the implementation there was no need for human readability of the data.

The gossip protocol is implemented using the FE Seeds. BE Servers connect to the FE Seed on arrival and periodically send a message that they are alive in case the FE Seed has been rebooted. On arrival, a FE Server will connect to one of the FE Seeds. The FE Seed will send its list of BE Servers to the FE Server and each 100ms the list will be resent. Clients will connect to FE Servers and their request will be forwarded to one of the many BE servers. Essentially, BE Servers contact FE Seeds, FE Servers contact FE Seeds and the information is propagated every 100ms therefore ensuring all nodes in the system have the correct information within 1s.

When a Client wants to send a request, it attaches to an FE server which forwards the request to one of many BE servers. The BE servers are picked at using a weighted random based on the number of cores each server contains. If a BE Server leaves the system, the list will be updated and the request will be rerouted to another available BE server. FE Servers can also periodically go down without causing issue to the client.

In terms of functionality, the password check and hash are performing according to the jBcrypt library standard. If any FE or BE node is queried by a client, it can return a PerfCounter which records data on server uptime, number of requests and number of completed requests and the group members names via the methods getPerfCounters() and getGroupMemembers(). Unfortunately parts of the final steps were not implemented including connection pooling and the ServiceUnavaliableException is thrown but occurs too early.

README

Instructions for Building

ant compile - to compile all the code and generate the appropriate gen-java files

Instructions for Running

Running any of the below commands will generage the ece454750s15a1.jar file

ant FESeed - to start seed
ant BEServer - to start a BE server
ant FEServer - to start an FE server
ant client - to run our testing client