ECE 454 - Assignment 1 Report

Michaela Farova (mfarova) & Nishad Krishnan (n9krishn)

Server Type

Our design uses the synchronous RPCs due to the fact that it requires a much simpler design and is generally less processor intensive compared to the asynchronous alternative. Our design uses a single-threaded server type, the TSimpleServer. This server type is a blocking I/O server, but it was chosen due to the fact that it is easier to process incoming and outgoing information rather than having a multi-threaded server that requires locks and concurrency control. The protocol used for data transmission is the TBinaryProtocol as it is one of the faster protocols to process and thus reducing processing time on our servers. We chose this protocol over a more space-efficient one due to the nature of the server layout as it is a short distance between all nodes (University network is small) and thus latency and space efficiency were lower in priority versus processing time on our server.

Load Balancing

Load balancing is achieved by using a weighted item picker algorithm. The weights are the number of cores available and the items are the BE nodes currently connected to the FE node. The algorithm is as follows:

1. Get total weight (sum of all BE ncores)
2. Choose random number between [0, total weight]
3. Go through BE nodes, see which one is picked by the random number

The node chosen is the BE that will receive the current request. This algorithm creates a statistically well balanced system that takes into consideration the number of cores at each node's disposal and thus ones with more cores have a higher probability of being chosen.

Detecting and Dealing with Crash Failures

Crash failures are detected by try-catch blocks that catch any connection error exceptions. When this type of exception is caught a proper error handler function is called to remedy the situation:

- BE nodes:
    - **(Startup)** If cannot connect to FE seed try another randomly chosen seed
- FE nodes:
    - **(Startup)** If cannot connect to FE seed try another randomly chosen seed
        - If go through all seeds and is a seed itself, determine that it is the first seed up
        - If go through all seeds and is not a seed, keep trying to connect to seeds
    - **(Runtime)** If cannot connect to BE node remove it from the BE nodes list and try next one
    - **(Runtime)** If connection to BE node lost in the middle of an operation remove it from BE node list and move work onto another available BE node
    - **(Runtime)** If cannot connect to available FE nodes to forward Gossip data, remove from available list and forward data to next FE node

Gossip Protocol

A gossip protocol is implemented such that the two lists of available nodes (FE and BE) are sent on a 100ms time interval to all other available FE nodes. All receiving nodes take the list and populate their own lists with any new values. Thus every node will know of any new BE or FE nodes within 100ms of it joining which is < 1s as per the specifications.