Design Document

For the design of the system, synchronized RPCs were chosen because even though only one thread can be used per call, we can use more than one thread. The client doesn't have much to do in the background in any case, so performance differences are quite small. However, using synchronized RPCs make it easy to detect errors because exceptions will be passed back right away. For an asynchronous RPC, if a request gets lost on the way, the thread may be kept open indefinitely.
A TThreadedSelectedServer was chosen because it has good performance without needing to handle distributing a thread per client like the TThreadedPoolServer. TThreadedSelected has the best performance next to the TThreadedPoolServer.
The protocol that's used is TBinaryProtocol. It's performs better than TJSONProtocol and is easier to use than TCompactProtocol in terms of processing the variable bit lengths. For smaller messages, the TBinaryProtocol will have good performance.
Load balancing is achieved by using an arraylist for the BEs inside each FE. Each BE has one entry in the arraylist per core. When the FE gets a request, a random number is generated between zero and the size of the list. The BE that corresponds to the index of the list is chosen to serve the request.
A TTransportException is caught when a BE goes down. This BE is then removed from the list of live BEs in the FE. Then we try try one of the other BE in the list. When the list of BEs is empty, we know that all BEs are down and a ServiceUnavailableException is thrown.
When a BE node joins the cluster, it lets the FE seed know and the FE seed will at it to its list of live BEs. Then, the gossip protocol which runs once every 100ms passes the list to other FE nodes and the BE node will be able receive requests.