High-level Description of Eau2 System

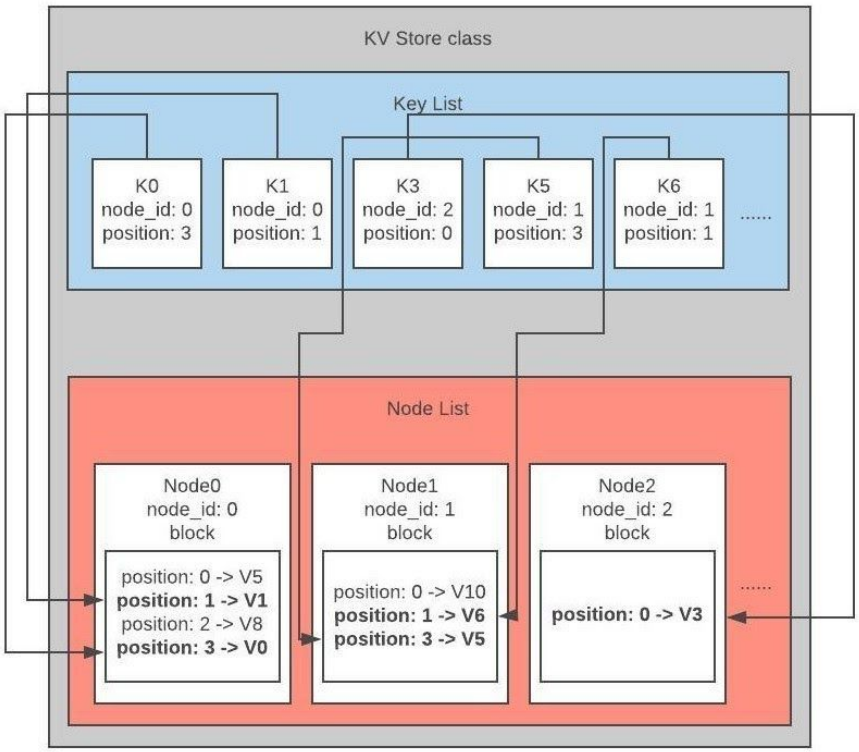
Xiaocheng Zhang, Tian Xia

- Introduction:

For this project, we are going to design an eau2 system, which is a distributed system designed for storing, retrieving, and managing related data in an array and a dataframe. In this system, each client will interact with one application where multiple nodes will be running. Each node contains one kv-store where all serialized messages will be stored and retrieved. By using the eau2 system, a customer will be able to efficiently store and retrieve data in a large scale as well as greatly reducing memory usage.

- Architecture:

The architecture of this project will be separated into three sections. The first one is the utility part. It contains all the basic data structures such as object, string, array, map, etc. The second part includes what client will interact with such as data frame adaptor, serializer and deserializer, and application file. The third part takes files which are related to storage such as key, value, kv-store, message etc.



- Implementation:

Each kv-store is similar to a map, in which a key has its own name and home address, and a value is a selized message in the char array. There is also a network layer that connects each kv-store together. Each application can call get functions to get values from kv-store in

other applications delivered by network layer. Once the application got that value, it will store the value locally. In order to get the value given the key, the application will not broadcast the request to every application. Instead, because each key has been encoded with the home field, the get function will know exactly which kv-node it should send the request to.

Each application has multiple nodes to execute different tasks. The field of application includes a kv-store. Each application will be started by calling the run method, and then be directed to the specific function based on the node itself.

- Use Cases:

We haven't finished the implementation of detail methods now. We have a basic structure of KVStore that saved data. The ideal usage is: big data will not be read into only one computer, and users are able to visit value whether it is on the current computer memory. In this case, the user will call a method "get(Key *k)". It returns a Value* which represented one of the basic types: String, int, float, bool. We tried to saved key and values by using a specific hash value. HashMap is the general buffer that saved data. In order to decrease the use of memory, we did not save the key. When users want to get value by using an existed key, the application will check the key's node id, and go through the node that saved it, caught the value by using the position (hash value of key's name).

- Open Questions:

We are wondering what kind of value our user required. As far as we know, this is defined by ourselves. So we just leave it empty now.

The most important question is how the application works. Since we have a little bit of application code which is a lack of description. We want to know more in detail about what our customers want.

- Status:

We just reimplemented the dataframe based on C++. This time we deduced some duplicate code and memory wasted. KV-Store doesn't have tests know. We are focusing on making it general and flexible now.