Introduction

The eau2 system is a distributed key/value store for big data analysis that can query information from data frames stored on different nodes in a network.

Architecture

Eau2 has three layers to it:

The bottom layer (network layer), consist of the distributed KV stores running on multiple nodes such that each of them have access to some part of the data and can communicate with each other to request and transfer data as and when needed.

*he middle layer provides abstraction in the form of distributed arrays and data frames that are used to store the data and represent them in a structured form if and when queried for.

The top layer is the application layer where users write code and can query for data.

Implementation:

We implemented the following classes as we believe they are needed for the development of the eau2 system:

application - this is used to create and store the kv store and stores a network for the application to interact with. Applications are meant to extend this class.

networkifc – This is an abstract class used for sending and receiving messages.

pseudonetwork - we use this for testing as it is a network that uses threads.

networkip - for actual transmission across a network - still in progress.

message - currently used for creating and sending messages across a network, include 5 different message types.

kvstore - we use this to hold the serialized dataframes.

dataframe - we use this to store and display the data after it parsed from a file.

parser - (the code is borrowed from team 4200NE) - we use this to parse SoR data from a file into our dataframe.

arraytemplate - we use this to store any type of data, a template was decided to be used instead of array classes to reduce clutter.

string - we use it to manipulate strings in our code.

object - we use it to manipulate different datatypes in our code.

wrapper - wrapper classes for ints, doubles, and bools. Used to create an array of objects in a row.

helper - Just used to make debugging / testing easier, supplying print methods.

Use cases

An application class can be written as a generic database with calls to create, and map over dataframes and store them in different nodes.

Status

- We cleaned our code and implemented the classes mentioned above.
- We ran valgrind on our old code to test for memory leaks, for milestone one and two code.
- We ran into trouble for using valgrind on milestone three code, as we had trouble deleting the KVStore as it has a thread running a while true loop, that we cannot join.
- We created a MakeFile that will run our tests for milestone1, milestone2, or defualt milestone 3.
- We created a static Method in DataFrame that creates a dataframe from a file location input.
- Serialization / Deserialization was added for dataframes, columns, and certain arrays.
- We created a pseudo network for testing the KVStore, and ran the DEMO application code on it.
- We confirmed that the word count application would work in the sudonetwork.
- Currently working on getting the actual network working
- Will then move on to distributing the dataframe / adding functionality to waitAndGet and local map.