The main focus of program is to use restful get and put to transfer data between HPC systems, in our focus we chose to use Lustre system and parallel file system.

**Section one: Metadata class:**

The Metadata class is responsible to maintain the information about location of the data and the specific configuration which is needed such as port or interface.

Function get is responsible to check whether the information is valid or invalid by taking the requested object id.

Function delete is responsible to delete the information from repository after data being removed, function takes object id as a parameter.

Function post is responsible to create new information by using tuple of pairs based on imported json data from \_\_process\_obj\_input.

Function \_\_process\_obj\_input Process a single object (built in accordance with documented API requirements) Check for supported values and then INSERT into database NOTE: there is very little in the way to verification and no pre-processing being accomplished. Which needs to rely on correct usage of the API

:param single\_obj: single object (dict/json)

:return: Tuple: Boolean (successful/unsuccessful) and message (regarding possible failure)

**Section Two: Parallel Class:**

In parallel class program uses the same method as original GET and PUT with parallelism functionality, in this section functions such as get and put gets used in matter of splitting file into different segments based on size or provided number, in terms of clients node program needs a josn based file containing the information about nodes address.

Function get(): takes an object id as input and returns the hash result which is the location of file in requested system.

Function put(): takes object id, cloud vendor, and cloud location as a input parameter and returns the id of file. The id of file is based on md5 hash function to conserve the integrity and availability.

Function execute\_cloud\_get() takes the object id, cloud vendor, original cloud location and target location, including new object id and whether if the file should be removed or not.

Function execute\_cloud\_put takes original object id, original parallel system location, target object id and target cloud vendor, target cloud location, and Boolean remove after. These functions are working with google cloud and aws but same idea can be used to develop for wider systems.

Function execute\_parallel\_get(): is responsible to take object id and target parallel location and split the file into n number then make n number of connection to receive that file from multiple sources to increase the throughput and usage of resources.

Function execute\_parallel\_put(): takes original object id, original cloud vendor, original cloud location, target object id, target parallel system location, and Boolean remove after and returns the list of ids.

Function execute\_parallel\_get and Function execute\_parallel\_put both uses MPI to ensure not only use the maximum resources that we have also, use the asynchronous method to lock the objects globally.

**Section Three:**

Implementation can expand to delete and post functionality, supporting amazon AWS or any other system will be in project’s future work. Analyzing the load work of individual nodes and the statistical behavior of network between HPCs is another upgrade module for the system.