REPORT

While implementing abstract class MapReduce I want to point out some parts for functions:

Producer: partitions the data into equal size, checks the remainder and splits the remainder to first remainder workers. Opens a push zmq socket and after pushing one piece of chunk it waits for consumer to get the data for 0.1 seconds. Having this kind of mechanism looks iffy, it would be better to use a mechanism that looks like a semaphore. (I belive consumer should be the one that wakes producer).

Consumer: It repedeatly tries to receive the input from producer by connecting the channel that producer opens. When it gets, it terminates the channel. Maps the piece and opens a push socket to push the mapped piece to resultcollector.

Result Collector: Opens and connects to zmq channel that consumer opens, pulls partial results then by calling reduce function gets final results.

FindCylicReferences -> Map implementation represents a graph for given input. Reduce function makes small change in the graph by iterating keys to generate values as an array of corresponding keys. ((1,3), (1,2), (2,1)) represented as ((1, [2,3]) , (2,[1])). Then starts a for loop, it checks key , values of key, then if values of key is represented as key and the initial key is in the array of values of key cycle is found. For example two tuples (1, [2,3]) and (2,[1]) key 1 has edge to 2 and 3, it checks 2 as key and values of 2 includes 1 then there is a cycle found.

FindCitations -> For me this was easier, Map function creates an array and for all cited paper it pushes (paperid, 1) pairs. Reduce function opens a dictionary, iterates over the tuples, for every occurrence key value of paper id is incremented by one. Returns the dictionary.