Title: MapReduce

Name: 曹寓恆

Student ID: 110521167

Implementation

1. Detail your implementation of the whole MapReduce program

(1) Input split function

Before the worker starts mapping, **job tracker will read the whole file and record the start position of each chunk**, and then put the tasks into a queue, so that they can be read in parallel between workers.

(2) Map function

After reading in a chunk data, the map function will cut the data with spaces and bind each word and an integer together to form a key value pair, which will be stored in the form of a vector.

(3) Partition function

There are many ways to map a string to an integer, but the MapReduce framework does not require a one-

to-one mapping relationship, so I obtain the reducer ID by **the summing of the ASCII codes of the characters in the string and taking the remainder of the number of reducer.**

(4) Sort function

Sorting key value pairs using std::sort. The

(5) Group function

After sorting, the same strings are stored in adjacent positions, so you can iterate over the whole vector and group the adjacent key value pairs into the same group.

(6) Reduce function

Add up all the values in the group to get the sum of word count.

(7) Output function

Write the word count pairs to the output file under

the specified directory.

2. **The challenges in your implementation, and how you solve them.**

(1) Thread pool

I am not so familiar with thread pools, and I have not been able to solve the **deadlock problem** when using the given thread_pool.h file. Therefore, I **first replaced this part with OpenMP to simulate the MapReduce architecture** to verify the correctness of the result, **and then tried to finish it with thread pool**.

(2) Data locality-aware scheduling algorithm

I **can't clearly understand the definition of locality** from the examples provided in the assignment, so I **use whether (chunkID mod #process) is equivalent to (nodeID mod #worker) to determine if the data is local**.

(3) Termination condition

Both the mapping task and the reduce task arguments that are positive integers, so I **define a negative number as the termination signal**, which is passed between the job tracker and the worker via MPI.

Experiences / Conclusion

Since finals week was quite busy, I regret that I was not able to fulfill all the requirements this time, but through this exercise I was able to grasp the MapReduce framework clearly. Thank you all for your teaching this semester.