Experiment 5 – MapReduce

Aim: Develop a Distributed Application using MapReduce.

Theory

MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

Further theory: https://ashnehete.in/mapreducejs/

Steps

1. Go to https://ashnehete.in/mapreducejs/
2. Click on **Start**
3. Read and understand the theory on the page about mapper and reducer
4. Click on **Word Counter**
5. The left side of the page has 3 input components:
    a. Input – This is a newline separated text which is fed as input to the mapper function.
    b. Mapper – This is where you'll write the code for mapper function.

```
function mapper(line, context)
{
      // Insert your code
      // context.write(key, value)
      return context
}

Parameters:
      line - each new line of the input
      context - global mapper object to store output
```

    c. Reducer – This is where you'll write the code for reducer function.

```
function reducer(key, list, context)
{
     // Insert your code
     // context.write(key, value)
     return context
}

Parameters:
      key - each key from the mapper
      list - list of values associated with the key
      context - global mapper object to store output
```

6. Click on **Run**
7. The right side of the page has 2 output components:

    a. Mapper Output – This will display the output after the mapper stage is complete. You can clearly see the key-value pairs that are formed in Word Count by comparing the mapper code.
    b. Reducer Output – This will display the final output after the reducer stage is complete. The Mapper Output is fed key-by-key to the reducer code.
8. Click on the dropdown on the top of the page and select **Count**
9. Aim of the Count example is to find the sum of all values associate with the corresponding letter. I'll leave it to you to figure out the code already provided.

Additional Task

1. Average
   Change the count code to display the average of all values instead of the sum associated with a letter

   Answer:
   Only change will be in reducer code which is,

```
function reducer(key, list, context)
{
    // Get the sum of all values in a list
    let sum = 0
    for (let i = 0; i < list.length; i++) {
        sum += list[i]
    }
    context.write(key, sum / list.length)
    return context
}
```

2. Word Length
   For the given input of words, find the number of words of each word length.
    a. Click on the dropdown on the top of the page and select **Blank** (This will load a blank template to start coding)
    b. Test against a custom input

   Answer
   Mapper

```
function mapper(line, context)
{
    // The input parameter line will contain
    // each individual word
      context.write(line.length, 1) // key = word length, value
= 1
      return context
}
```

   Reducer

```
function reducer(key, list, context)
```

```
{
    // In the reducer function we will just count
    // how many times the word has occurred
    context.write(key, list.length)
    return context
}
```