# **OS Final Project**

# Map-Reduce Implementation Using Operating System Concepts

### **Submitted by:**

**Taha Tariq**22I-2014

**Umer Siddiqui**22I-2038

### **Submitted to:**

**Miss Mubrra Asma**

### **Date:**

8 December, 2014.

# 

## **Introduction**

This document outlines the design, functionality, and implementation details of a word count program utilizing the MapReduce paradigm. The program demonstrates fundamental operating system concepts such as threads, named pipes (FIFOs), and interprocess communication (IPC). It is divided into two components: mapper.cpp and reducer.cpp.

## **Program Structure**

The program simulates a distributed processing system with the following stages:

1. Mapper: Processes an input sentence, generates key-value pairs for each word, and prepares data for shuffling.
2. Shuffle: Groups identical keys together and prepares data for reduction.
3. Reducer: Aggregates the counts of identical keys to produce the final word count.

### **Mapper Function:**

* The Mapper is responsible for tokenizing a user-provided sentence into individual words.
* Each word is processed in a separate thread, and the output is stored in a shared array protected by a mutex.
* Each thread emits a key-value pair in the form (word, 1).

**Input:** A single sentence provided by the user.

**Processing:**

* Sentences are split into words using a loop.
* A thread is created for each word to store the key-value pair (word, 1) in a global array.

**Output:** An array of words to be passed to the Shuffle phase.

**Example:** For the input “hello hello hello world”, the mapper output is   
hello,1  
hello,1  
hello,1  
world,1

**Shuffle Function:**

* The Shuffle phase groups words emitted by the Mapper based on their keys.
* It uses a named pipe (pipe1) to transfer grouped data (words and their counts) to the Reducer process.
* Words are grouped by uniqueness, and their counts are stored as strings (e.g., "1,1,1").

**Input:** The array of words generated by the Mapper.

**Processing:**

* Groups identical words and concatenates their counts (e.g., "1,1,1" for three occurrences).
* Writes the number of unique words, each word, and its count to the named pipe.

**Output:** Data written to the named pipe for the Reducer phase.

### **Reducer.cpp**

This component handles:

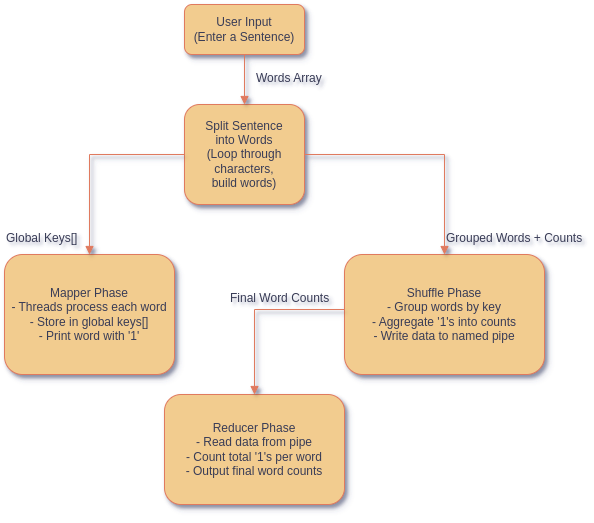
* Receiving grouped data from the mapper via the named pipe.
* Aggregating the counts for each word.
* Printing the final reduced output (word,count).

**Input:** Data from the Shuffle phase received via the named pipe.

**Processing:**

* Read each word and its count string.
* Aggregates the counts for each word by counting the 1s in the count string.

**Output:** Word frequencies displayed on the console.



## 

## 

## 

## 

## **Test Cases:**

### **Basic Test Case** **Input:** hello world hello **Expected Output:** hello,2 world,1 **Actual Output:** hello,2 World,1

### **Case Sensitivity Test: Input:** Hello hello HELLO **Expected Output:** Hello,1 hello,1 HELLO,1 **Actual Output:** Hello,1 hello,1 HELLO,1

### **Input with Punctuation: Input:** hello, world! Hello. **Expected Output:** hello,,1 World!,1 Hello.,1 **Actual Output:** hello,,1 World!,1 Hello.,1

### **Edge Case: No Input and Input with Only Space Input:** “ “ or no input. **Expected Output:** The program should say no input received. **Actual Output:** No output - Empty input received