Practical Work 4: Word Count using MapReduce

Group ID: 16

December 16, 2025

1 Introduction

This report presents the implementation of the classic **Word Count** problem using the MapReduce programming model. The goal is to count the frequency of each word in a given text dataset by distributing the task across Mapper and Reducer phases.

2 Implementation Choice

We chose **Python** to simulate the MapReduce framework for this practical work.

- **Reasoning:** While C/C++ offers performance benefits, Python allows for rapid prototyping of the MapReduce logic (Map, Shuffle, Reduce) without the overhead of memory management. This allows us to focus on the algorithmic design and data flow demonstration.
- **Environment:** The simulation runs locally, where the main program acts as the Master node orchestrating data splits, mapping, and reducing.

3 System Design

3.1 Data Flow Architecture

The diagram below illustrates how input text is processed through the MapReduce pipeline.

3.2 Mapper and Reducer Logic

The **Mapper** takes a line of text, normalizes it (lowercase, remove punctuation), and emits a key-value pair `(word, 1)` for every word found.

```
def mapper(self, text_chunk):
    output = []
    for line in text_chunk:
        words = clean_text(line).split()
        for word in words:
            output.append((word, 1))
    return output
```

Listing 1: Mapper Implementation

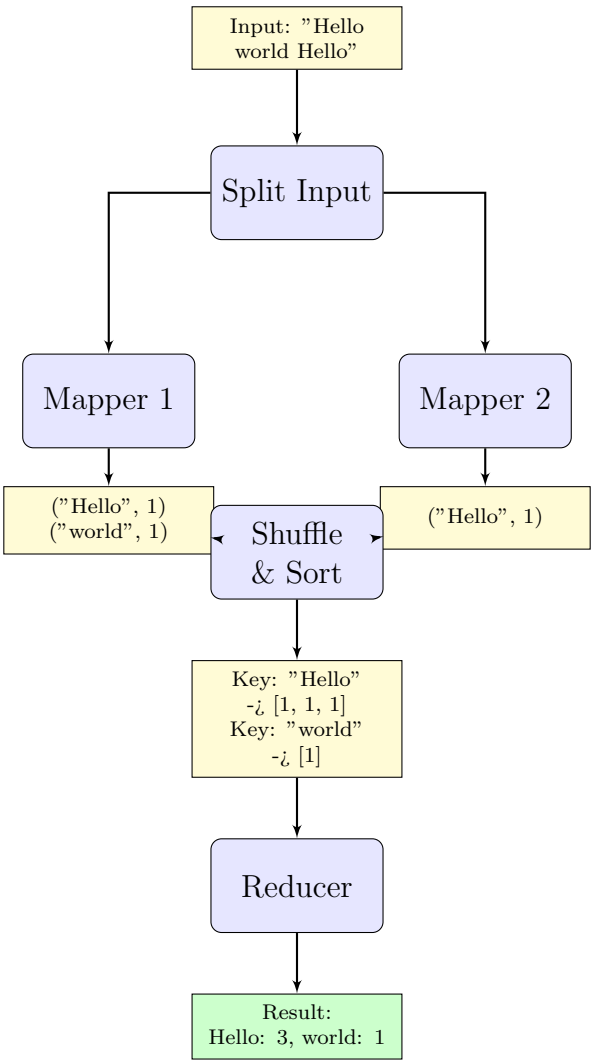

Figure 1: MapReduce Data Flow for Word Count

The **Reducer** receives a key (word) and a list of counts (e.g., `[1, 1, 1]`). It sums these values to get the total frequency.

```
def reducer(self, key, values):
    return (key, sum(values))
```

Listing 2: Reducer Implementation

4 Roles and Responsibilities

No.	Member	Task
1	Member 1	Mapper logic and Input splitting
2	Member 2	Reducer logic and Shuffle implementation
3	Member 3	Report writing and Architecture Diagram

Table 1: Group Roles