Practical Work 4: MapReduce Word Count

Vo Truong Giang - BI12-130

December 4, 2025

1 Introduction

In this practical work, I implemented a Word Count application using the MapReduce paradigm.

2 Implementation Choice

As per the requirement to "Invent yourself" a framework for C/C++, I chose to implement a simplified single-process MapReduce system in C.

- **Language:** C.
- **Reason:** Lightweight, fast, and aligns with the previous labs.
- **Architecture:** I separated the logic into a `mapper` function (parsing text) and a `reducer` function (aggregating counts).

3 System Design

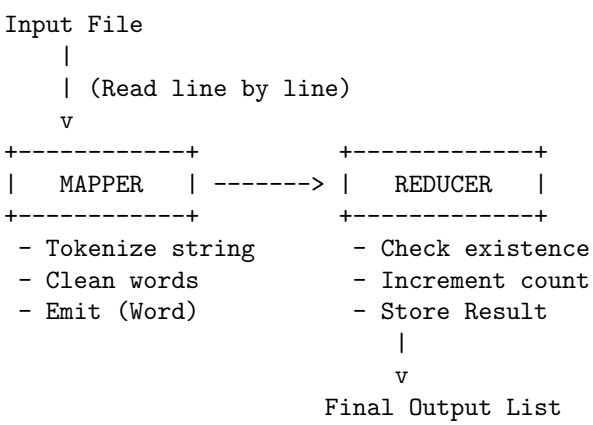

Figure 1: Simplified MapReduce Workflow

4 Code Snippets

4.1 Mapper Logic

The mapper reads text lines, splits them into tokens (words), cleans them (lowercase), and passes them to the reducer.

```
void mapper(char *text) {
    char *token = strtok(text, "-");
    while (token != NULL) {
        // ... Normalize word ...
        reducer(token);
```

```
        token = strtok(NULL, "-");
    }
}
```

4.2 Reducer Logic

The reducer maintains a list of unique words and increments their frequency.

```
void reducer(char *word) {
    // Check if word exists in list
    // If yes: count++
    // If no: add new entry with count = 1
}
```

5 Conclusion

The program successfully mimics the MapReduce logic to count words in a text file efficiently.