Practical Work 4: MapReduce Word Count

Nguyen Thanh Dat
Student ID: 23BI14091

1 Introduction

The objective of this practical work is to implement a **Word Count** application using the **MapReduce** programming model. MapReduce is a framework for processing large data sets with a parallel, distributed algorithm on a cluster.

2 Implementation Choice

I chose to implement a custom MapReduce framework using **Python and MPI (mpi4py)**.

- **Why Python:** It provides native dictionary structures (`defaultdict`) that are perfect for counting unique keys (words).
- **Why MPI:** Since C/C++ frameworks are currently limited, I utilized MPI to "invent" a parallel framework where processes communicate to scatter data and gather results.

3 Mapper and Reducer Design

The logic follows the standard MapReduce workflow:

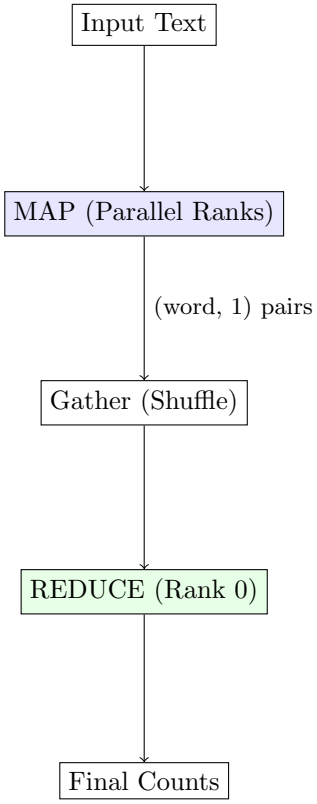

Figure 1: MapReduce Workflow Architecture

- **Mapper:** Each MPI Rank receives a subset of the text lines. It iterates through the words, converting them to lowercase and incrementing a local counter.
- **Reducer:** The "Master" process (Rank 0) gathers the dictionaries from all Ranks and aggregates the values for each unique word.

4 Code Snippet

```
# Mapper Logic
for word in line.lower().split():
    my_counts[word] += 1

# Reducer Logic
for partial_dict in all_counts:
    for word, count in partial_dict.items():
        final_counts[word] += count
```