Word Count MapReduce Implementation

Group Project

3/12/2024

Abstract

This report describes the implementation of a Word Count problem using a Python-based MapReduce-like approach. The program utilizes multi-threading and synchronization to efficiently count the frequency of words across multiple input text files. The tasks were divided among five group members for collaborative work.

1 Introduction

The Word Count problem is a classic example of distributed computing, where the goal is to count the frequency of each word in a large collection of text. Our implementation employs Python's threading module and shared data structures to simulate a MapReduce framework. The tasks were divided among group members as follows:

- **Vu Hai Thien Long:** Implemented the core logic for cleaning and processing words.
- **Luu Linh Ly:** Designed the multi-threading structure for processing files concurrently.
- **Nguyen Ngoc Nhi:** Implemented thread-safe mechanisms using locks.
- **Le Viet Hoang Lam:** Created the functionality for saving results to the output folder.
- **Nguyen Duc Duy:** Managed integration and testing of the overall program.

2 Choice of MapReduce Implementation

The decision to implement this problem in Python with threading was motivated by the following considerations:

- **Ease of Implementation:** Python's threading module provides a simple interface for concurrent execution.
- **Efficiency:** Multi-threading allows parallel processing of files, reducing runtime.
- **Thread Safety:** Shared data structures are protected using locks, ensuring correctness in concurrent operations.
- **Scalability:** The approach can handle multiple files simultaneously, making it suitable for moderate-sized datasets.

3 Mapper and Reducer

In our implementation, the Mapper and Reducer roles are defined as follows:

3.1 Mapper

The Mapper function, `count_words_in_file`, is responsible for:

- Reading a file line by line.
- Splitting lines into words.
- Cleaning words by converting them to lowercase and removing punctuation.
- Counting word occurrences locally using a dictionary.

3.2 Reducer

The Reducer functionality merges the local dictionaries from all threads into a global word count map. This process is synchronized using a lock to ensure thread safety. The final output is a comprehensive word frequency map across all input files.

4 Flowchart of the Process

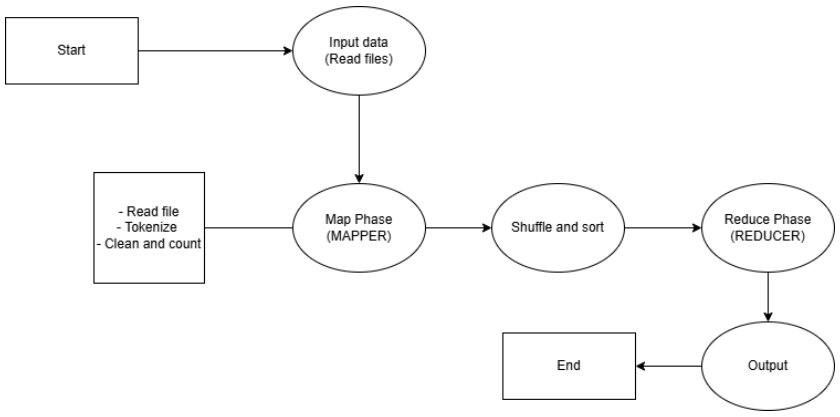

Figure 1: Flowchart of the Mapper and Reducer process.

5 Conclusion

This implementation demonstrates the effectiveness of a Python-based MapReduce-like framework for solving the Word Count problem. The use of threading and locks ensures efficient and thread-safe processing of multiple files. This collaborative project allowed us to learn and apply parallel programming concepts in Python.