**NAMES :**

**SARMAD ALI**

**&**

**ABYAZ ISRAR**

**ROLL NUMBERS:**

**22I-1997**

**&**

**22I-2056**

**OS PROJECT REPORT:**

Project Overview

In this project a basic MapReduce framework has been developed using C++. It also shows concepts of multiple processing units of an operating system including synchronization and concurrency control. The framework processes input data in three stages:

**Map Phase:** Carries out the computations in parallel for the input data and produces some intermediary key value pairs.

**Shuffle Phase:** Aggregates values associated with keys in the data to compose subgroups consisting of NV pairs.

**Reduce Phase:** The required aggregates brought together values in order to generate the output.

1. Features

**Concurrency:** Implemented with threads and synchronization tools ((std::thread), mutex).

Synchronization: When working with the shared data, Mutex locks help avoid race conditions as a result.

**Parallel Processing**: During the Map phase, we have multiple threads to handle data chunks at the same time.

**Customizable Input**: It also means that the input from the user is taken and processed functionally.

1. **System Workflow**

The following steps outline the workflow:

**Input Phase:**

Take a string of words from the user.

Divide the input data smaller data units so that they can be easily processed.

**Map Phase:**

All of the chunks are processed through a separate thread.

The mapFunction is to get the number of how many times each word appears in the chunk.

**Shuffle Phase:**

Key-value pairs with an intermediate level of importance are organized according to the names of keys.

The details from all the threads are then compiled into one structure.

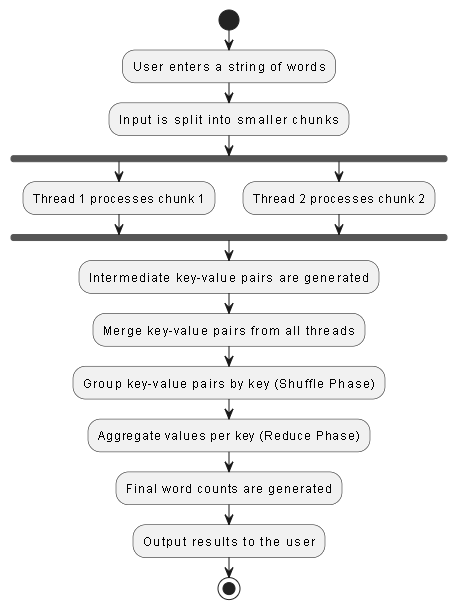
**Reduce Phase:**

Denominates the key values thus summing them in cases where the parameter is counted (e.g. number of patients).

Enters the final word frequency out.

1. **Activity Diagram :**

* Start: User provides input.
* Map Phase: It is always good practice to process input in parallel, which means that the input is divided into chunks.
* Shuffle Phase: An additional step is to merge and group intermediate results into higher level outcomes and results.
* Reduce Phase: Summarize and produce output of the aggregation of SUPERVALU’s performance measures.
* End: Display results to the user.



1. **Code Walkthrough**

**Input Handling:** Takes a sentence of words from a user through the keyboard.

**Map Phase:**

The function mapFunction can be seen as execution of operations on data chunks in parallel.

Mutex locks help in making threads safe for use while dealing with shared data.

**Shuffle Phase:**

Answers gathered from different discussions are combined and localized by subject.

**Reduce Phase:**

As a final result, the reduceFunction combines reported the aggregated key-value pairs.

1. **Test Cases:**
2. **Potential Enhancements**

Backing for computing across different machines.

DBMS Thread pool which can dynamically launch new threads when there is high traffic.

Scalability for complex large-sized sets.

1. Conclusion

The outcome of this project includes illustrating the basic principles of OS concepts within a real-life challenge. Employing threads and synchronization it achieves rather fast and concurrent processing and data accumulation.