MapReduce System Report

Overview

This report provides an explanation of the MapReduce system implemented using gRPC for distributed processing. The system consists of three main components: **Master**, **Mapper**, and **Reducer**. Each component communicates via gRPC services to perform distributed data processing tasks such as WordCount and Inverted Index.

Services and RPCs

1. **MapperService**

The `MapperService` is responsible for handling the mapping phase of the MapReduce process. It processes input data splits and generates intermediate key-value pairs.

RPC: `map`

- **Input**: `MapRequest`
  - `query`: Specifies the type of query (e.g., WordCount or Inverted Index).
  - `input_location`: Path to the input data directory.
  - `input_split_files`: List of files assigned to the mapper.
  - `input_split_file_id`: List of file IDs corresponding to the input files.
  - `n_reducers`: Number of reducers to partition the intermediate data.
- **Output**: `MapResponse`
  - `intermediate_file_location`: Path where the intermediate data is stored.
  - `status`: Indicates whether the operation succeeded or failed.

Functionality

- The `map` RPC reads the assigned input files, processes them based on the query type, and writes intermediate data to the specified location. The intermediate data is partitioned based on the number of reducers.

---

2. **ReducerService**

The `ReducerService` handles the reduce phase of the MapReduce process. It processes intermediate data partitions and generates the final output.

RPC: `reduce`

- **Input**: `ReduceRequest`
  - `query`: Specifies the type of query (e.g., WordCount or Inverted Index).
  - `partition_files_path`: List of paths to intermediate data partitions.
  - `output_location`: Path where the final output will be stored.

- **Output**: `ReduceResponse`
  - `status`: Indicates whether the operation succeeded or failed.
  - `output_file_path`: Path to the final output file.

Functionality

- The `reduce` RPC reads the intermediate data partitions, aggregates the data based on the query type, and writes the final output to the specified location.

Components

1. **Master**

The `Master` is the central coordinator of the MapReduce system. It performs the following tasks:

- **Input Splitting**: Divides the input data into splits and assigns them to mappers.
- **Mapper Management**: Spawns mapper processes and assigns tasks to them.
- **Reducer Management**: Spawns reducer processes and assigns tasks to them.
- **Aggregation**: Combines the outputs from all reducers into a single final output file.

Key Functions

- `input_split`: Splits the input data and assigns files to mappers.
- `spawn_mappers`: Starts mapper processes.
- `spawn_reducers`: Starts reducer processes.
- `mapFunction`: Invokes the `map` RPC on mappers.
- `reduceFunction`: Invokes the `reduce` RPC on reducers.
- `aggregate_reducer_outputs`: Combines reducer outputs into a final file.

---

2. **Mapper**

The `Mapper` processes input data splits and generates intermediate key-value pairs. It implements the `MapperService` gRPC service.

Key Functions

- `map`: Processes input files based on the query type and writes intermediate data.
- `_wordCount`: Implements the WordCount logic.
- `_invertedIndex`: Implements the Inverted Index logic.

---

3. **Reducer**

The `Reducer` processes intermediate data partitions and generates the final output. It implements the `ReducerService` gRPC service.

Key Functions

- `reduce`: Processes intermediate data partitions based on the query type and writes the final output.
- `_wordCountAndInvertedIndex`: Implements the logic for both WordCount and Inverted Index.
- `wordCount_reduce_function`: Aggregates counts for WordCount.
- `invertedIndex_reduce_function`: Aggregates indices for Inverted Index.

Workflow

1. **Input Splitting**:
    - The `Master` splits the input data into chunks and assigns them to mappers.
2. **Mapping Phase**:
    - Each mapper processes its assigned input files and generates intermediate data partitions.
3. **Reducing Phase**:
    - Each reducer processes its assigned intermediate data partitions and generates the final output.
4. **Aggregation**:
    - The `Master` combines the outputs from all reducers into a single final output file.

Output Formats

- **Intermediate Data**: Stored in partitioned files (e.g., `P0`, `P1`, etc.) in the `folders` directory.
- **Final Output**: Stored in the `outputs` directory as `final_output.txt`.

Conclusion

This MapReduce system leverages gRPC for distributed processing, enabling efficient handling of large datasets. The `Master` coordinates the workflow, while `Mapper` and `Reducer` services handle the data processing tasks. The system supports two queries: WordCount and Inverted Index, and can be extended to support additional queries.