Lab 3: MPI File Transfer

1. Rationale for MPI Implementation

We chose **mpi4py** for its efficient integration of the Message Passing Interface (MPI) with Python. It offers straightforward APIs for parallel communication, supports essential MPI features (e.g., point-to-point messaging), and is compatible with widely used MPI distributions such as MPICH and OpenMPI. This combination of flexibility and performance makes mpi4py an ideal choice for our file transfer system.

2. System Design

The system is composed of two main processes: the client (Rank 0) and the server (Rank 1). The client reads the file in chunks and transmits them to the server, which reconstructs the file and writes it to disk. Communication is facilitated via **MPI_COMM_WORLD**, with unique message tags used to identify data types.

Figure: MPI File Transfer Architecture

Figure 1: Architecture of the MPI File Transfer System

3. Workflow and Process Description

The workflow is divided into two distinct processes:

3.1 Client Process (Rank 0)

- Reads the file from the local system in chunks of 1 KB.
- Sends the file name to the server, followed by the file data in chunks.
- Signals the end of the transfer using a termination message (‘DONE‘).

3.2 Server Process (Rank 1)

- Receives the file name from the client and prepares to write to disk.
- Collects file chunks and reconstructs the file.
- Completes the transfer when the termination message (‘DONE‘) is received.

Figure: Data Flow in the MPI Service

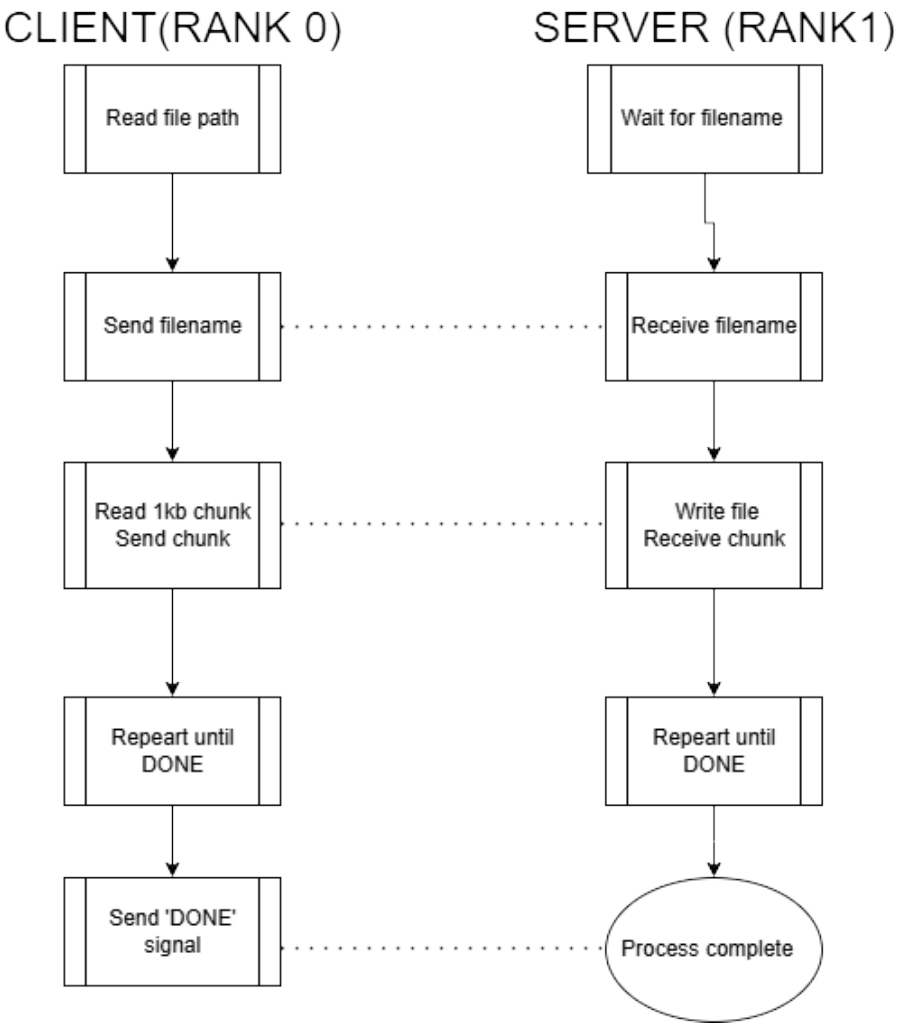

Figure 2: MPI Data Flow and Communication

4. Implementation Summary

4.1 Client Implementation

The client opens the specified file, reads it in 1 KB chunks, and sends the data to the server via ‘comm.send()‘. After all chunks are sent, it transmits a ‘DONE‘ signal to indicate the transfer is complete.

4.2 Server Implementation

The server listens for incoming messages from the client. It receives the file name first, followed by the data chunks. The chunks are written sequentially to reconstruct the file. The process ends upon receiving the 'DONE' signal.

5. Task Allocation and Contributions

- **Research on MPI and mpi4py**: Cao Nhat Nam (22BI13320) and Pham Ngoc Minh Chau (22BI13063)
- **Client-Side and Server-Side Code Development**: Do Thi Huong Tra (BA12-174)
- **Testing and Debugging**: Vu Hoang Mai Nhi (22BI13352)
- **Report Writing and Diagrams**: Bui Nguyen Ngoc Huyen (22BI13199)