Practical Work 3: MPI File Transfer

Nguyen Thi Minh Nguyet
ID: 22BA13241
Class: CS

December 3, 2025

1 Introduction

This practical work focuses on implementing a file transfer system using the **Message Passing Interface (MPI)** standard. Unlike previous client-server architectures (TCP, RPC), MPI is designed for high-performance parallel computing where processes communicate via messages within a "communicator."

2 Implementation Choice

For this implementation, we chose **mpi4py** (MPI for Python).

Reasoning:

- **Standardization:** It provides Python bindings for the standard MPI implementations (OpenMPI, MPICH, MS-MPI).
- **Ease of Use:** It allows sending generic Python objects (picklable) using lowercase methods ('send', 'recv') while still supporting high-performance buffer transfers ('Send', 'Recv').
- **SPMD Model:** It naturally supports the Single Program, Multiple Data paradigm required for this task.

3 Service Design

The system consists of two processes within the `MPI_COMM_WORLD` communicator.

- **Rank 0 (Sender):** Reads the file and initiates the transfer.
- **Rank 1 (Receiver):** Listens for messages and writes the file to disk.

4 System Organization

In MPI, the system organization is defined by the Communicator. Both processes are spawned simultaneously by the `mpiexec` command.

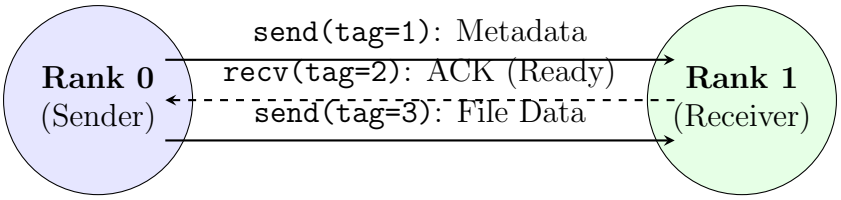

Figure 1: Point-to-Point Communication Design in MPI

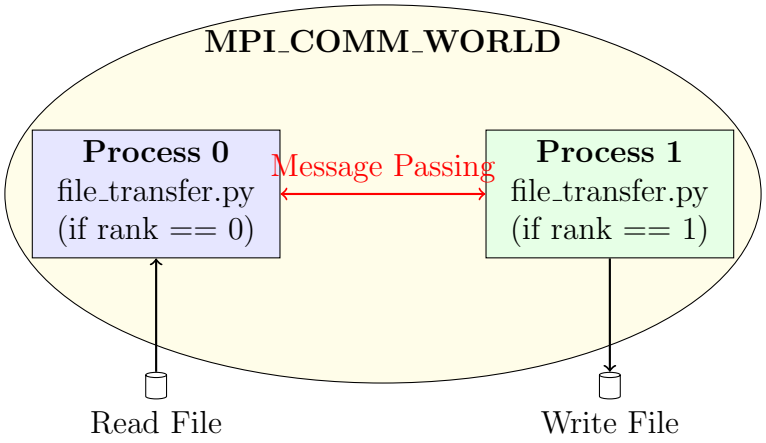

Figure 2: SPMD (Single Program Multiple Data) Organization

5 Implementation Details

The core logic relies on checking the process rank to determine the role.

```
from mpi4py import MPI
comm = MPI.COMM_WORLD
rank = comm.Get_rank()

if rank == 0:
    # Sender Role
    data = read_file("image.jpg")
    comm.send({'name': 'image.jpg', 'size': len(data)}, dest=1, tag=1)

    ack = comm.recv(source=1, tag=2)
    if ack == "READY":
        comm.send(data, dest=1, tag=3)

elif rank == 1:
    # Receiver Role
    meta = comm.recv(source=0, tag=1)
    comm.send("READY", dest=0, tag=2)

    file_content = comm.recv(source=0, tag=3)
    save_file(meta['name'], file_content)
```

Listing 1: MPI Transfer Logic

6 Roles and Responsibilities

- **Rank 0:** Acts as the source. It is responsible for reading the file I/O and serializing the data into messages.
- **Rank 1:** Acts as the destination. It handles the synchronization (ACK) and reconstructs the file on the storage system.