Lab 3 - MPI

BI12-149 - Nguyen Duc Phuc
Tuong

April 2024

In this lab, we will use MPICH because of its open-source nature andhigh performance.

• System Architecture

The system consists of 2 processes: 1 server and 1 client. The server will transfer a file to the client through a buffer in a message with specific message tag:

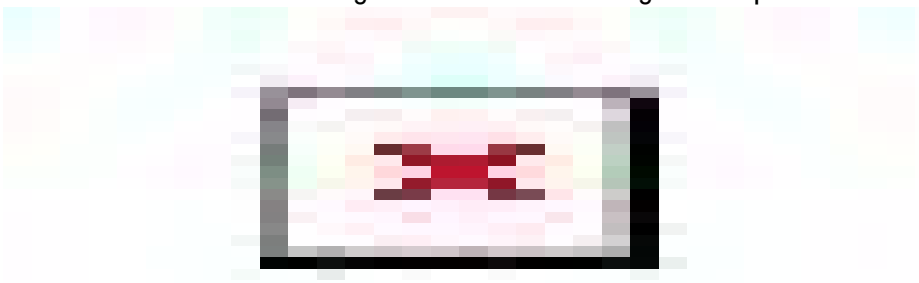

Figure 1: System Architecture

• Implementation

First, the MPI initialization begin, and the program check if there is exactly 2 processes take part in the file transfering process. Each process then queries its own rank in the MPI_COMM_WORLD:

```
MPI_Init(NULL, NULL);
```

```
int world_size; MPI_Comm_size(MPI_COMM_WORLD,
&world_size);if (world_size != 2)
{
    printf("Run: mpirun -n 2 exefile\n");MPI_Finalize();
    exit(EXIT_FAILURE);
}

int world_rank; MPI_Comm_rank(MPI_COMM_WORLD,
&world_rank);
```

For the server (with process rank 0), it first opens the file (in this example test.txt) and read the file content into a buffer. It then send that buffer using a message with a message tag (1234):

```
if (world_rank == 0)
{
    FILE *file = fopen("test.txt", "r");if (file == NULL)
    {
        perror("Error opening file for reading");MPI_Finalize();
        exit(EXIT_FAILURE);
    }

    fseek(file, 0, SEEK_END); long filesize =
    ftell(file);fseek(file, 0, SEEK_SET);

    char *buf = (char *)malloc(sizeof(char) * filesize);fread(buf, 1, filesize,
    file);
    fclose(file);

    MPI_Send(buf, filesize, MPI_BYTE, 1, 1234, MPI_COMM_WORLD);

    printf("Server sent file successfully!\n");
}
```

For the client (with process rank 1), it first probes the message with the specific message tag 1234 to get the file size for setting up the buffer length.After the buffer set up successfully, the client will now receive the message and store it in a local file (test2.txt):

```
else if (world_rank == 1)
{
    MPI_Status stat;
```

```
        MPI_Probe(0, 1234, MPI_COMM_WORLD, &stat);
        int filesize;
        MPI_Get_count(&stat, MPI_BYTE, &filesize);
        char *buf = (char *)malloc(sizeof(char) * filesize);


        MPI_Recv(buf, filesize, MPI_BYTE, 0, 1234, MPI_COMM_WORLD,&stat);
        FILE *file = fopen("test2.txt", "w"); if (file == NULL)
        {
            perror("Error opening file for writing");MPI_Finalize();
            exit(EXIT_FAILURE);
        }
        printf("Saved to test2.txt\n");

        fwrite(buf, 1, filesize, file);fclose(file);
        free(buf);
    }
    .MPI_Finalize();
    return EXIT_SUCCESS;
}
```