MPI

Spring Training 2025

Yoshihiro Izawa

2025/05/13

Contents

1. MPI Overview

1.1 What is MPI (Message Passing Interface)?

- **A standard API** for message passing between distributed memories in parallel computing.
- MPI assumes a **distributed-memory computing system**
- MPI can run on **shared-memory computing system**
- MPI programming model (basically) uses **SIMD**

1.2 Parallel Programming Classification

- **Multi-Process**: MPI(Message Passing Interface), HPF(High Performance Fortran)
- **Multi-Thread**: OpenMP, Pthread(POSIX Thread)

1.3 MPI Features

- **Communication Model**: Uses message passing for communication between processes.
- **Distributed Memory Support**: Each process has its own memory space, no shared memory.
- **Multi-node Capacity**: Can run across multiple nodes; abstracts network communication.
- **Standardized API**: Standardized interface in C, C++, and Fortran; hightly portable.
- **Multiple Implementation**: Available implementations include OpenMPI, MPICH, and Intel MPI, etc.
- **Difficalt to Debug**: Debugging is challenging due to concurrency and communiaction complexity.

1.4 Typical example of Usage

- Simulation on a supercomputer(Physics, Meteorology, Chemistry, etc.)
- Data processing in large-scale data analysis (e.g., genomics, astronomy).
- Machine learning training on large datasets (e.g., distributed deep learning).

1.5 Comparison between implementations

	OpenMPI	MPICH	Intel MPI
Developer	Universities, Companies	Argonne National Laboratory	Intel Corporation
Distribution	Open source	Open source	Free version included
Optimization Target	General purpose	Lightweight, stable	Optimized for Intel architecture
Performance	Medium to high	Lightweight, stable, scalable	Best performance on Intel CPUs
Main Use	Academic clusters, general HPC	Research, education	Commercial HPC, Intel clusters

1.6 Key Communication Primitives

- **System function**: `MPI_Init`, `MPI_Finalize`, `MPI_Comm_size`, `MPI_Comm_rank`
- **Point-to-point communication**: `MPI_Send`, `MPI_Recv`
- **Collective communication**: `MPI_Bcast`, `MPI_Reduce`, `MPI_Alltoall`
- **Synchronization**: `MPI_Barrier`, `MPI_Wait`, `MPI_Test`
- **Derived data types**: `MPI_Type_create_struct`, `MPI_Type_vector`
- **Non-blocking communication**: `MPI_Isend`, `MPI_Irecv`
- **Remote memory access**: `MPI_Put`, `MPI_Get`
- **Process management**: `MPI_Comm_spawn`, `MPI_Comm_free`

1.7 Minimum MPI Program

1.7.1 Hello World (C)

```c
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    MPI_Init(&argc, &argv);

    int num_procs;
    int my_rank;

    MPI_Comm_size(MPI_COMM_WORLD, &num_procs);
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

    printf("Num of Proc : %d\n", num_procs);
    printf("My Rank     : %d\n", my_rank);

    MPI_Finalize();
    return EXIT_SUCCESS;
}
```

```
mpicc mpi_hello.c -o mpi_hello
mpirun -np 4 ./mpi_hello
Num of Proc : 4
My Rank     : 3
Num of Proc : 4
My Rank     : 2
Num of Proc : 4
My Rank     : 0
Num of Proc : 4
My Rank     : 1
```

1.7 Minimum MPI Program

1.7.2 Hello World (C++)

```cpp
#include <mpi.h>
#include <iostream>
#include <cstdlib>

int main(int argc, char *argv[])
{
    MPI_Init(&argc, &argv);

    int num_procs;
    int my_rank;

    MPI_Comm_size(MPI_COMM_WORLD, &num_procs);
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

    std::cout << "Num of Proc : " << num_procs <<
std::endl;
    std::cout << "My Rank     : " << my_rank << std::endl;

    MPI_Finalize();
    return EXIT_SUCCESS;
}
```

```
mpic++ mpi_hello.cpp -o mpi_hello
mpirun -np 4 ./mpi_hello
Num of Proc : 4
My Rank     : 3
Num of Proc : 4
My Rank     : 1
Num of Proc : 4
My Rank     : 0
Num of Proc : 4
My Rank     : 2
```

1.7.3 Hello World (Fortran)

```
program hello_mpi
  use mpi
  implicit none

  integer :: ierr, rank, size

  call MPI_Init(ierr)
  call MPI_Comm_rank(MPI_COMM_WORLD, rank, ierr)
  call MPI_Comm_size(MPI_COMM_WORLD, size, ierr)

  print *, "Num of Proc:", size
  print *, "My Rank:    ", rank

  call MPI_Finalize(ierr)
end program hello_mpi
```

```
mpif90 mpi_hello.f90 -o mpi_hello
mpirun -np 4 ./mpi_hello
Num of Proc:            4
My Rank:                2
Num of Proc:            4
My Rank:                0
Num of Proc:            4
My Rank:                3
Num of Proc:            4
My Rank:                1
```

1.8 Communicator

- **MPI_COMM_WORLD**: Default communicator for all processes in MPI. Determines the processer group.
- At

2. Basic Learning of MPI

3. References

3.1 MPI Reference

- MPI「超」入門（C 言語編）- 東京大学情報基盤センター
- 並列プログラミング入門