Introduction to Parallel & Distributed Programming

Lec 18— MPI

Subodh Sharma | March 20, 2026

Message Passing

- Inter-process communication
  - System calls, Network Infrastructure (Ethernet, Infiniband)
- Process-local memory: Processes don’t share address space
- Computation model is Single Process Multiple Data (SPMD)
- Access to other process’s data through explicit `send` and `recv` instructions
- MPI — Language of SuperComputing
  - A user library (with more than 150 API calls) for implementing parallel programs

Message Passing

- Two significant implications:
  - Data is local to each process
    - User's responsibility to partition and place the data in appropriate machines
      - **Side-effect — complex programming**!
  - All communication requires at least two processes — sender and receiver
- **Message Semantics**:
  - Machine-local buffering?
  - Communication links lossy? (How to deal with the loss)
  - FIFO? **Point-2-Point or Collectives?** Addressing?

A Simple MPI Program

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

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

    int world_rank, world_size;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);

    printf("Hello world from process %d out of %d\n", world_rank, world_size);

    MPI_Finalize();
    return 0;
}
```


```
mpicc hello_mpi.c -o hello_mpi
mpirun -np 4 ./hello_mpi
```

Process Organisation

- **Communicator** — group of processes sharing a context
  - Used to support a communication topology
  - Intra and Inter communicator communication is possible
  - MPI_COMM_WORLD is a predefined constant
- **Context:** Used to define a communication universe (part of communicator)
  - Message across contexts can’t interfere
- **Groups:** Ordered collection of processes (can build hierarchy)

P-to-P Communication: Send & Recv

- **MPI_Send**:
  - Address of data
  - Count of elements
  - Datatype of elements
  - Destination rank
  - Message tag
  - Communicator

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

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

    int rank, size;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    if (rank == 0) {
        int x = 42;
        MPI_Send(&x, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
        printf("Process 0 sent value %d to process 1\n", x);
    }

    if (rank == 1) {
        int y;
        MPI_Recv(&y, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
        printf("Process 1 received value %d from process 0\n", y);
    }

    MPI_Finalize();
    return 0;
}
```

P-to-P Communication: Send & Recv

- **MPI_Recv**:
  - Source rank
    - **MPI_ANY_SOURCE**
  - Message tag
  - Communicator
  - Status object — information about the message received

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

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

    int rank, size;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    if (rank == 0) {
        int x = 42;
        MPI_Send(&x, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
        printf("Process 0 sent value %d to process 1\n", x);
    }

    if (rank == 1) {
        int y;
        MPI_Recv(&y, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
        printf("Process 1 received value %d from process 0\n", y);
    }

    MPI_Finalize();
    return 0;
}
```

Semantics of Send

- **MPI_Send**:
  - It is a blocking send call
  - Under Buffering:
    - Send can return once the buffering of the message is complete
  - Under no buffering:
    - Send acts as a rendezvous send
  - Completion semantics;
    - Complete when the payload is copied out of the sender’s address space

Semantics of Recv

- **MPI_Recv**:
  - It is a blocking call (with or without buffering)
  - Completion semantics;
    - Complete when the payload is copied out of the receiver’s address space
  - MPI_ANY_SOURCE: Can non-deterministically match with any one of the racing sends
    - Source of non-determinism
    - Can also be a source of non-deterministic deadlocks