MPI Tutorial

Aditya

What is MPI?

- The Message Passing Interface (MPI) is a standard that defines an interface for message-passing libraries.
  - It is NOT a language, a compiler or a library.
- MPI is language-independent.
- OpenMPI is an open-source implementation of MPI in C, C++ and Fortran.
- mpi4py is a Python library that implements MPI.

Why use MPI?

- Allows communication between processes, which is crucial for parallel computing.
  - Why is this crucial?
- Is powerful, portable, scalable, and widely used.

Point-to-point Communication

```
int MPI_Send(const void* buf,          // what to send
             int count,                // number of elements to send
             MPI_Datatype datatype,    // type of elements
             int dest,                 // rank of destination
             int tag,                  // message tag
             MPI_Comm comm             // communicator
);

int MPI_Recv(void* buf,                // where to receive
             int count,                // number of elements to receive
             MPI_Datatype datatype,    // type of elements
             int source,               // rank of source
             int tag,                  // message tag
             MPI_Comm comm,            // communicator
             MPI_Status* status        // status object
);
```

Group Communication – Send Identical Copies

```
int MPI_Bcast(void* buffer,            // what to broadcast
              int count,               // number of elements to broadcast
              MPI_Datatype datatype,   // type of elements
              int root,                // the rank of the process broadcasting the data
              MPI_Comm comm            // communicator
);
```

```
int MPI_Reduce(const void* sendbuf,    // what to send
               void* recvbuf,          // where to receive
               int count,              // number of elements to send/receive
               MPI_Datatype datatype,  // type of elements
               MPI_Op op,              // operation to perform
               int root,               // the rank of the process receiving the result
               MPI_Comm comm           // communicator
);
```

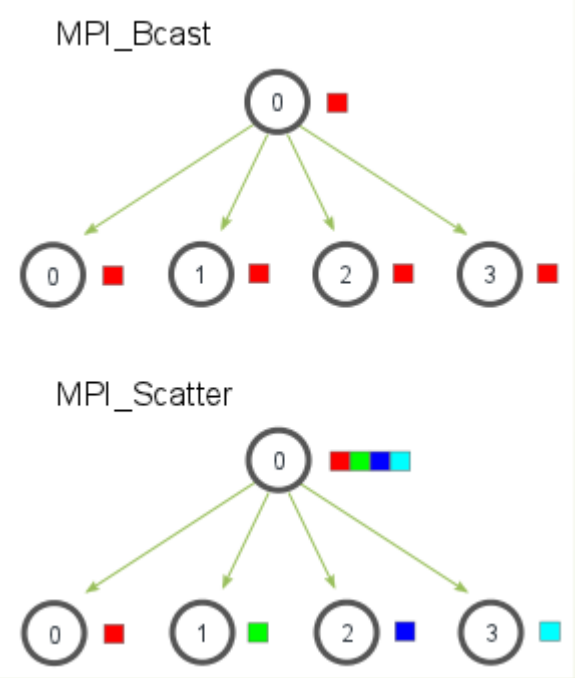
Group Communication – Partition & Transfer an Array

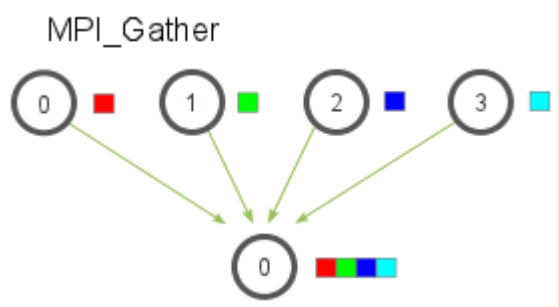
```
int MPI_Scatter(const void* sendbuf,  // what to send
                int sendcount,        // number of elements to send to each process
                MPI_Datatype sendtype, // type of elements
                void* recvbuf,        // where to receive
                int recvcount,        // number of elements to receive
                MPI_Datatype recvtype, // type of elements
                int root,             // the rank of the process scattering the data
                MPI_Comm comm         // communicator
);

int MPI_Gather(const void* sendbuf,   // what to send
               int sendcount,         // number of elements to send from each process
               MPI_Datatype sendtype, // type of elements
               void* recvbuf,         // where to receive
               int recvcount,         // number of elements to receive from each process
               MPI_Datatype recvtype, // type of elements
               int root,              // the rank of the process gathering the data
               MPI_Comm comm          // communicator
);
```

Communication

- MPI_Send, MPI_Recv, MPI_Bcast, MPI_Reduce, MPI_Scatter and MPI_Gather are blocking calls.
- Senders and receivers must explicitly call these functions.





Hello, World!

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

int main(int argc, char** argv) {
    MPI_Init(&argc, &argv);  // initialises the MPI execution environment

    int rank, size;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);  // gets the rank of this process
    MPI_Comm_size(MPI_COMM_WORLD, &size);  // gets the number of processes

    printf("Hello, World! I am process %d out of %d processes.\n", rank, size);

    MPI_Finalize();  // terminates MPI execution environment
}
```

Compile & Run

Compile:

- mpicc hello-world.c –o hello-world

Run:

- mpiexec -n 4 ./hello-world

```
[aditya@indus mpi-tutorial]$ mpiexec -n 4 ./hello-world
Hello, World! I am process 3 out of 4 processes.
Hello, World! I am process 1 out of 4 processes.
Hello, World! I am process 0 out of 4 processes.
Hello, World! I am process 2 out of 4 processes.
```

Suggested Resources

- https://rookiehpc.github.io/mpi/docs/index.html
- https://docs.open-mpi.org

Q&A