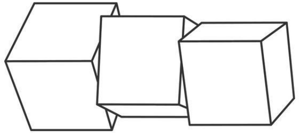
Partitioned Point-to-Point Communication

Partitioned Point-to-Point Communication

- In MPI, partitioned point-to-point communication refers to a method of communication where data is divided into non-contiguous blocks or partitions and sent/received between processes.

- This can be particularly useful when dealing with large datasets that are logically divided into segments.



Problem Example

Write a MPI program that consists of two processes. Process 0 should initialize a **persistent** send operation and send data to process 1, while process 1 should initialize a **persistent** receive operation and receive data from process 0. The data to be sent/received should be stored in array of doubles. This array should be logically divided into `PARTITIONS` number of partitions, each of size `COUNT`. The program should print informative messages at each step to indicate the progress of the communication process. Once the communication is completed, each process should print a message indicating the completion of communication and exit.

Persistent Communication

Persistent means that the communication object remains associated with a particular communication buffer, allowing multiple sends/receives without needing to reinitialize the operation each time.

Persistent Communication

- In non-persistent communication, each message is sent and received individually using functions like MPI_Send and MPI_Recv.

- Once the message is sent or received, the communication buffer associated with that message is free to be used for other purposes.

Non-Persistent Communication

- In non-persistent communication, each message is sent and received individually using functions like MPI_Send and MPI_Recv.

- Once the message is sent or received, the communication buffer associated with that message is free to be used for other purposes.

Methods for Sending (Analogous in Receiving)

```
int MPI_Send_init(const void *buf, int count, MPI_Datatype datatype, int dest,
                  int tag, MPI_Comm comm, MPI_Request *request)
```

```
int MPI_Psend_init(const void *buf, int partitions, MPI_Count count,
                   MPI_Datatype datatype, int dest, int tag, MPI_Comm comm,
                   MPI_Info info, MPI_Request *request)
```