HPC Assignment - 3

Q.1 What are types of basic communication? Explain anyone in detail

Ans: Many interaction in parallel programs occur in well defined patterns involving groups of processors

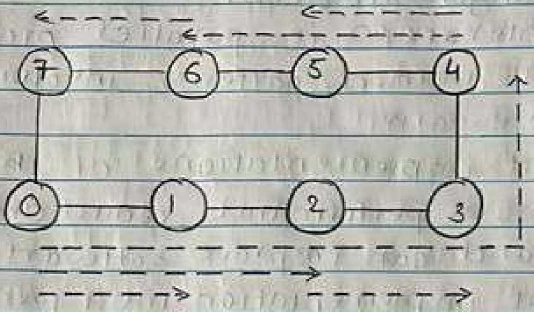
- Efficient implementations of these operations can improve performance, reduce development effort and can improve software quality.
- Efficient implementation must leverage underlying architecture.
- For this reason, we offer to specific architectures here.
- We select a descriptive set of architectures to illustrate the process of algorithm design.
- Group communication operations are built using point to point messaging primitives.

→ One to all broadcast

- One processor has a piece of data (of size m) it needs to send to everyone.
- The dual of one to all broadcast is all to one.
- One to all broadcast on an eight node ring, Node 0 is the source of ~~pt~~ broadcast.
- Each message transfer step is shown by a numbered, dotted arrow from the source

of the message to its destination.

- The number on an arrow inidicates the time step during which message is transferred.



Q.2 Explain blocking and non blocking MPI

Ans: The syntax of the broadcast operation takes the following form:

- In MPI_Bcast (void *shared_data, int number,
- MPI_Datatype datatype, int source_process,
- MPI_Comm communicator)

- The broadcast operation is achieved through the MPI_Bcast command in MPI.
- The operands define the form and source of the data to be sent to all processes
- The broadcast is performed within the scope of a communicator specified by last argument or the communicator of type MPI_comm and broadcast data are sent to all processes within it.
- The data come from a single process

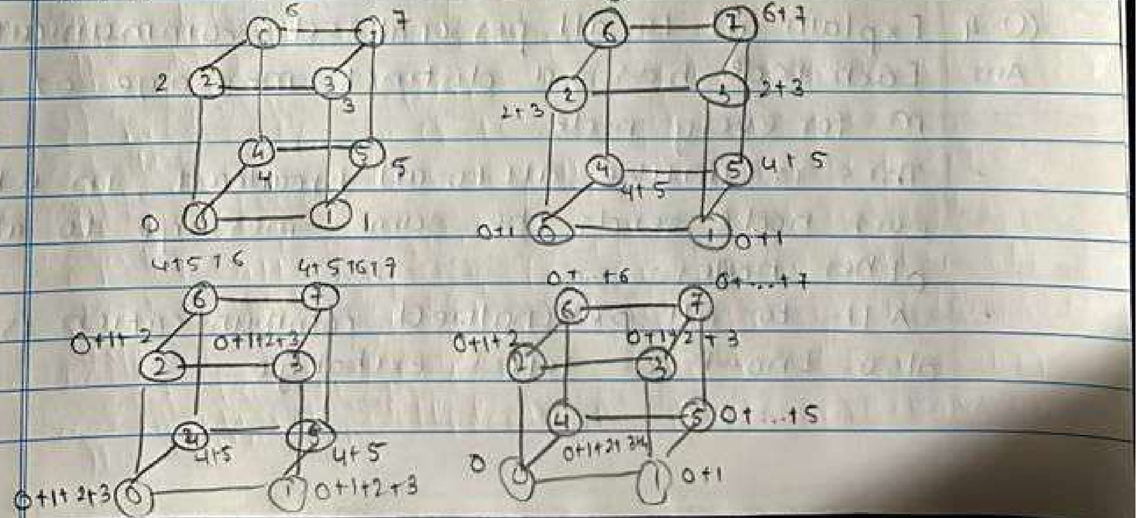
identified by its rank within the communictor by source process which is the penultimate argument of MPI_Bcast

- Like many other message passing commands, the data to be sent is determined by the first three arguments: the name of variable pointing to data buffer, here "shared data", the type of data element which is composed, here "datatype" and the "number" of elements of data type making up the data to be broadcast.

Q.3 Describe prefix sum operations in detail.

Ans. Given p numbers $n_0, n_1, \dots n_{p-1}$ (one on each node) the problem is to compute the sums $S_k = \sum_{1}^{k}$ all k between 0 and p-1

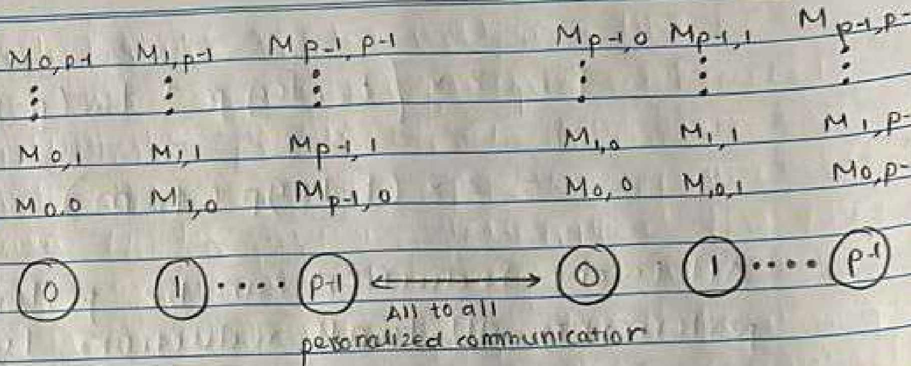
- Initially n_k resides on the node labeled k, and at the end of procedure, the same node holds S_k.



- computing prefix sums on an eight node hypercube, At each node, square bracket shows the local prefix sum accumulated in result buffer and parentheses enclose the contents of outgoing message buffer for next step.
- The operation can be implemented using all to all broadcast kernel.
- We must account for the fact that in prefix sums the node with label k uses information from only k node subset whose labels are less than or equal to k
- This is implemented using an additional result buffer.
- The content of an incoming message is added to the result buffer only if message comes from a node with a smaller label than receipient node.

Q.4 Explain all to all personalized communication.

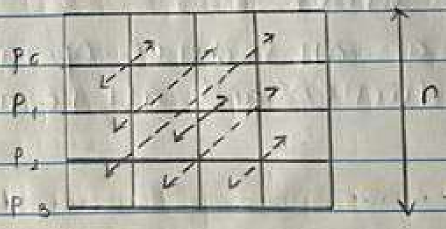
Ans: Each node has a distinct message of size m for every node.
- This is unlike all-to-all broadcast, in which each node sends the same message to all other nodes.
- All to all personalized communication is also known as total exchange.



$M_{0,p-1}$ $M_{1,p-1}$ $M_{p-1,p-1}$
$M_{0,1}$ $M_{1,1}$ $M_{p-1,1}$
$M_{0,0}$ $M_{1,0}$ $M_{p-1,0}$
0 1 .... p-1
$M_{p-1,0}$ $M_{p-1,1}$ $M_{p-1,p-1}$
$M_{1,0}$ $M_{1,1}$ $M_{1,p-1}$
$M_{0,0}$ $M_{0,1}$ $M_{0,p-1}$
0 1 .... p-1
All to all
personalized communication

Example

- Consider the problem of transposing a matrix.
- Each processor contains one full row of matrix
- The transpose operation in this case is identical to all to all personalized communication operation.
- All to all personalized communication in transposing a 4x4 matrix using four processes.



Q.5 How to improve the speed of some communication operations.

Ans: Splitting and routing messages into parts: If the message can be split into p parts, a one to all broadcast can be implemented as a scatter operation followed by an all to all broadcast operation. The time for this is:

$$T = 2 \times \left[t_s \log p + t_w (p-1) \frac{m}{p} \right]$$

$$\approx 2 \times (t_s \log p + t_w m)$$

- All to one reduction can be performed by performing all to all reduction followed by a gather operation.
- Since all reduce operation is semantically equivalent to an all to one reduction followed by a one broadcast, the asymptotically optimal algorithms for these two operations can be used to construct a algorithm for all reduce operation.
- The intervening gather and scatter operations cancel each other.
- Therefore an all reduce operation results an all to all reduction and all to all broadcast.