HPC (U-3).

Page No.
Date

Parallel Communication

* One to all broadcast & All to one Reduction -

→ • Processes → exchange data → other process → parallel algorithms.
• change of data → efficiency affected → interaction delays → execution.
• communication → various parallel architecture → improve performance & reduce development effort.

* Different architecture are -
1) One to all Broadcast (scatter).
2) All to One Reduction (Gather).
3) All to All Broadcast & Reduction.
4) All - Reduce & Prefix - Sum.
5) Scatter & gather.
6) All to All personalized.
7) Circular Staff.

Linear arrays, meshes & hypercubes.

t_s = It is startup time for data transfer

t_w = per word transfer time

It is same b/w all pair of nodes.

M_1 = communication cost in parallel machine.

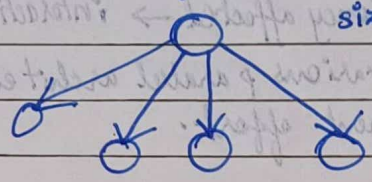
0 → 1 → 2 → 3 sends msg
↓ ↓ ↓ ↓
4 5 6 7

* One to One Broadcast with Recursive Doubling.
• source node broadcasts msg to all others efficiency
• eight node ring with Node 0 as source.
• each step doubles distance b/w sender & receiver nodes, forms bidirectional comm^n plan.
• recursive dubling technique ensures efficient one to all broadcast access eight^ node ring
• Nodes with msg forward to opp neighbor (not already sent to)
• Repeats for log 2 (no. of nodes) steps log(8)

Page No.
Date

① One to all Broadcast -

1. Parallel algorithm often require a single process.
2. It is used to send idential data to all processes or subset of them.
3. one to all Broadcast.
4. one processor has a piece of data size 'm' it needs to send everyone.

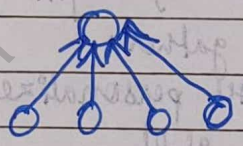


② All to one Reduction -

1. Reduction can be performed by simply reversing direction.
2. In all to one reduction each processor has m units of data.
3. these data items must be combined piece-wise using same associative operator.

eg → addition or min.

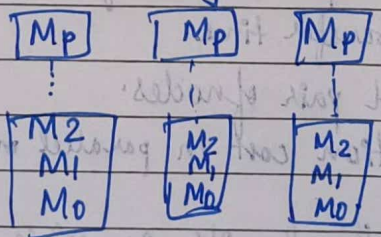
0 1 2 3
4 5 6 7

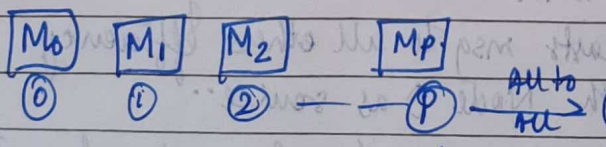


③ All to all broadcast -

1. All to all broadcast is generalisation of broadcast → processor is source.
2. simultaneously every process out of p processo iniates broadcast.
3. A source process ends same m-word message to every other process but different processes may broadcast diff messages.

Example -





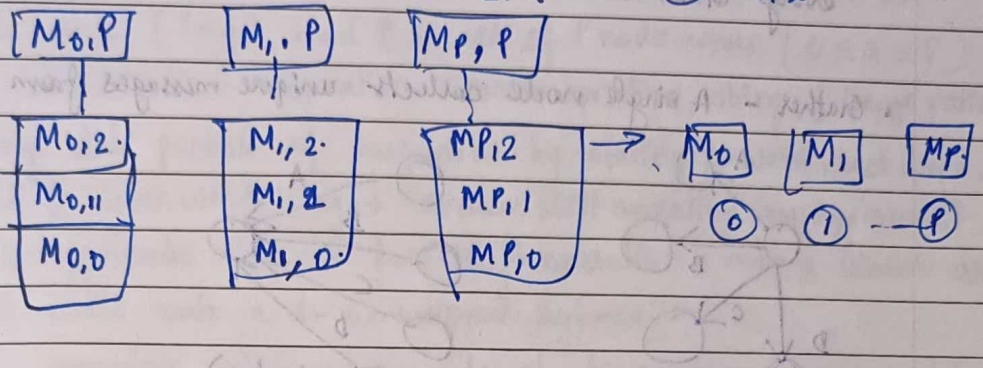
All to all Broadcast

Page No.
Date

④ All to All Reduction –

All to All Reduction is reverse of all to all broadcast in every node. destination of all to one reduction.

$M x = M_{0,x} \oplus m_{1,x} \oplus M_{2,x} \oplus \dots \oplus M_{P,x}$.



* All reduce & prefix sum operations –

- All reduce communication operation –
  1. Identical to all to one reduction followed one to all broadcast.
  2. formulation is not efficient.
  3. faster way to perform all reduce by using butter All to all broadcast.
  4. only diff is message size doesn't increase here.

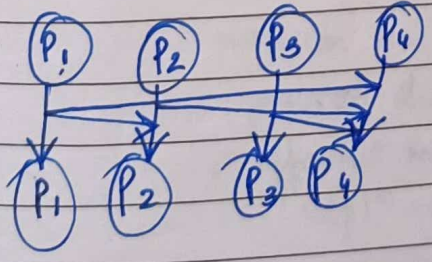
  $CA - T = (t_s + t_w M) \log P$.

* Prefix - Scan communication Operation –

- Scan operation.

example - Input - 1 2 3 4 operation - ADD. Output - 1 3 6 10.

- each output → sum of all numbers → input upto given point.
- histogram uses prefix scan. Also used for data compression.
- scan operation perform parallel prefix.



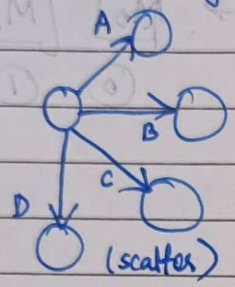
Scan communication.

Page No.
Date

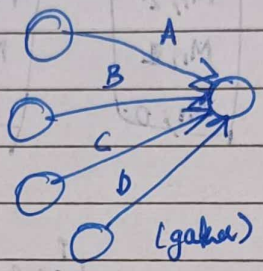
* Scatter & Gather -

- Scatter - single node sends a unique message of size m to every other node. called as one to all personalized communication.

- Gather - A single node collects unique messages from each node.



(scatter)



(gather)

C.A P Processes :- No. of steps log P.
Total time $T = t_s \log P + t_w (P-1)$.

* All to All Personalised Communication -

- Communication → total exchange.
- each node → sends same msg → nodes in Broadcast.
- each node sends → direct msg of size m → every node.
- parallel algorithm → parallel sorting, fast fourier transform.

CA = processes = P-1 Transmitted mess size = $m(P-1)$.

Total time = $T = t_s(P-1) + \sum_{i=1}^{P-1} t_w m$ $m(P-2) \dots m$.

Page No.
Date

* Circular shift -

- particular permutation [redistribution in set]
- circular q-shift - special permutation in which node i sends a data packet to node $(i+q) \bmod P$ in set of P nodes where $(0 \leq q \leq P)$.
- circular communication → operation → matrix operation, string & image pattern.
- permuting data packets of each node by rotating forward, backward, upward & downwards. (eg → circular shift operation on hypercube).
- develop a hypercube algo → for shift operation, map a linear array with 2d nodes onto a d-dimensional hypercube.
- perform mapping → assign node of linear array to node j of linear array of node i of any q in p-node hypercube is most $2 \log(p-1)$
- $T = (t_s + t_w m)(2 \log p - 1)$.

* Improve speed of commun operation -

- need speed of communication → 3 operations

1) one to all broadcast -
   - splitting & routing messages into parts scatter + All to all broadcast.
   - improve speed of one to all, split message int P part & perform scatter operation.
   - scatter - divide msg into n part & every node carry each part of msg.
   - CA - $T = 2 \times (t_s \log p + t_w m)$.

- shift repositions elements of array cyclically by specified distance
  - moves element from one end of array to another.
  - data reording & rotation in parallel algo
  - shift can be left or right
  - preserves data locality & reduce memory access impact pattern
  - efficient manipulation of large dataset.
  - Appln → cryptography, signal processing & image manipulation

2) All to one Reduction –

- performed by gather operation. All to all reduction + Gather.
- msg are reduced to one message & gathers reduced msg.

3) All reduce –

- performed by performing all to one reduction followed by one to all broadcast.
- asymptotically optimal algorithms for two operations.
- construct a similar algorithm all reduce operation.
- gather & scatter operations – cancel each other require all to all reduction & broadcast.

- All to one reduction + One to all broadcast.

↓ ↓

All to all reduce + Gather

Scatter

All to all ~~broadcast~~

(scatter & gather) (cancel operation)

↓

All to all reduction + All to all Broadcast.

Fig. All reduce operation.

Page No.
Date

Block communication using MPI in HPC -

- Initialization - MPI → Message Passing Interface
  - ↳ Is initialized to setup communication.
- Process Setup → multiple processors are created each on seprate node or processor.
- Blocking send → sender process executes MPI-send, halting until msg is sent.
- Blocking Receive → receiver process executes MPI-recv, waiting until msg is received.
- completion - Both resume after msg transn, continuing further execution.

Adv → • Performance Overhead
• Ensure data exchange before continuing

Disad → • performance overhead
• scalability with large processes.

For Usecase → • simple comm^n patterns • small msg & low comm^n
• critical synchronization needs.

Non-blocking using MPI -

- Initialization → to set up comm^n (MPI)
- Process Setup → Multiple processors are created each on seprate node.
- Non-blocking send → sender process executes MPI_Isend, allows to continue execution immediately
- Non-blocking receive → Receive process executes MPI_Irecv, allows continue execution immediately
- completion check → Both sender & receiver process check completing using MPI_Test or similar function.

Adv → • Improved performance
• Better Scalability
• Flexibility

Disadv → • more complex programming
• Potential for deadlock

Usecase → • complex communication patterns • overlapping comm^n.
• Large msg transfers with

Page No.
Date

Prefix Sum Operation

- also known as scan
- computes cumulative sum of elements in sequence
- crucial for various algo. like parallel reduction.
- sum of all elements upto certain index in array
- Result at each index is sum of all elements preceding it.
- application like → computational geometry, image processing, cryptography
- requiring prefix info for each element.
- efficiency & applicability in various domains