Chapter 4
Basic Communication Operations
(Selected slides)

A. Grama, A. Gupta, G. Karypis, and V. Kumar

Topic Overview

- One-to-All Broadcast and All-to-One Reduction
- All-to-All Broadcast and Reduction
- All-Reduce and Prefix-Sum Operations
- Scatter and Gather
- All-to-All Personalized Communication
Basic Communication Operations: Introduction

- Many interactions in practical parallel programs occur in well-defined patterns involving groups of processors.

- These collective communication operations are important for performance, development effort and cost, and software quality.

- Efficient implementations should consider the underlying architecture.

- A descriptive set of architectures are selected to illustrate the process of algorithm design.
Basic Communication Operations: Introduction

- Group communication operations are built using point-to-point messaging primitives.

- Time usage of communicating a message of size $m$ over an uncongested network: $t_s + t_w m$.

- We use this as the basis for our analyses. Where necessary, we take congestion into account explicitly by scaling the $t_w$ term.

- We assume that the network is bidirectional and that communication is single-ported.
One-to-All Broadcast and All-to-One Reduction

- 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 reduction.

- In all-to-one reduction, each processor has $m$ pieces of data. These data items must be combined piece-wise (using some associative operator, such as addition or min), and the result made available at a target processor.
One-to-all broadcast and all-to-one reduction among \( p \) processors.
One-to-All Broadcast and All-to-One Reduction on Rings

- The simplest way is to send \( p - 1 \) messages from the source to the other \( p - 1 \) processors – this is not very efficient.

- Use recursive doubling: source sends a message to a selected processor. We now have two independent problems defined over halves of machines.

- Reduction can be performed in an identical fashion by inverting the process.
One-to-all broadcast on an eight-node ring. Node 0 is the source of the 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 indicates the time step during which the message is transferred.
All-to-One Reduction

Reduction on an eight-node ring with node 0 as the destination of the reduction.
Broadcast and Reduction: Example

Consider the problem of multiplying a matrix with a vector.

- The $n \times n$ matrix is assigned to an $n \times n$ (virtual) processor grid. The vector is assumed to be on the first row of processors.

- The first step of the product requires a one-to-all broadcast of the vector element along the corresponding column of processors. This can be done concurrently for all $n$ columns.

- The processors compute local product of the vector element and the local matrix entry.

- In the final step, the results of these products are accumulated to the first row using $n$ concurrent all-to-one reduction operations along the columns (using the sum operation).
One-to-all broadcast and all-to-one reduction in the multiplication of a $4 \times 4$ matrix with a $4 \times 1$ vector.
Broadcast and Reduction on a Mesh

- We can view each row and column of a square mesh of $p$ nodes as a linear array of $\sqrt{p}$ nodes.

- Broadcast and reduction operations can be performed in two steps – the first step does the operation along a row and the second step along each column concurrently.

- This process generalizes to higher dimensions as well.
Broadcast and Reduction on a Mesh: Example

One-to-all broadcast on a 16-node mesh.
Broadcast and Reduction on a Hypercube

- A hypercube with $2^d$ nodes can be regarded as a $d$-dimensional mesh with two nodes in each dimension.

- The mesh algorithm can be generalized to a hypercube and the operation is carried out in $d (= \log p)$ steps.
Broadcast and Reduction on a Hypercube: Example

One-to-all broadcast on a three-dimensional hypercube. The binary representations of node labels are shown in parentheses.
Broadcast and Reduction on a Balanced Binary Tree

- Consider a binary tree in which processors are (logically) at the leaves and internal nodes are routing nodes.

- Assume that source processor is the root of this tree. In the first step, the source sends the data to the right child (assuming the source is also the left child). The problem has now been decomposed into two problems with half the number of processors.
One-to-all broadcast on an eight-node tree.
Cost Analysis

• The broadcast or reduction procedure involves $\log p$ point-to-point simple message transfers, each at a time cost of $t_s + t_w m$.

• The total time is therefore given by:

$$T = (t_s + t_w m) \log p.$$  (1)
All-to-All Broadcast and Reduction

- Generalization of broadcast in which each processor is the source as well as destination.

- A process sends the same $m$-word message to every other process, and different processes broadcast different messages.
All-to-all Broadcast and Reduction

\[ M_0 \quad M_1 \quad \ldots \quad M_{p-1} \]

All-to-all broadcast

\[ \begin{array}{ccc}
M_{p-1} & M_{p-1} & M_{p-1} \\
\vdots & \vdots & \vdots \\
M_1 & M_1 & M_1 \\
M_0 & M_0 & M_0 \\
\end{array} \]

All-to-all reduction

\[ \begin{array}{ccc}
0 & 1 & \ldots & p-1 \\
\end{array} \]

All-to-all broadcast and all-to-all reduction.
All-to-All Broadcast and Reduction on a Ring

- Simplest approach: perform $p$ one-to-all broadcasts. This is not the most efficient way, though.

- Each node first sends to one of its neighbors the data it needs to broadcast.

- In subsequent steps, it forwards the data received from one of its neighbors to its other neighbor.

- The algorithm terminates in $p - 1$ steps.
All-to-All Broadcast and Reduction on a Ring

All-to-all broadcast on an eight-node ring.
All-to-All Broadcast and Reduction on a Ring

1. procedure ALL_TO_ALL_BC_RING(my_id, my_msg, p, result)
2. begin
3.   left := (my_id − 1) mod p;
4.   right := (my_id + 1) mod p;
5.   result := my_msg;
6.   msg := result;
7.   for i := 1 to p − 1 do
8.     send msg to right;
9.     receive msg from left;
10.    result := result ∪ msg;
11.  endfor;
12. end ALL_TO_ALL_BC_RING

All-to-all broadcast on a p-node ring.

All-to-all reduction is simply a dual of this operation and can be performed in an identical fashion.
All-to-all Broadcast on a Mesh

- Performed in two phases – in the first phase, each row of the mesh performs an all-to-all broadcast using the procedure for the linear array.

- In this phase, all nodes collect \( \sqrt{p} \) messages corresponding to the \( \sqrt{p} \) nodes of their respective rows. Each node consolidates this information into a single message of size \( m\sqrt{p} \).

- The second communication phase is a columnwise all-to-all broadcast of the consolidated messages.
All-to-all broadcast on a $3 \times 3$ mesh. The groups of nodes communicating with each other in each phase are enclosed by dotted boundaries. By the end of the second phase, all nodes get $(0,1,2,3,4,5,6,7)$ (that is, a message from each node).
All-to-all broadcast on a Hypercube

- Generalization of the mesh algorithm to $\log p$ dimensions.
- Message size doubles at each of the $\log p$ steps.
All-to-all broadcast on a Hypercube

(a) Initial distribution of messages
(b) Distribution before the second step
(c) Distribution before the third step
(d) Final distribution of messages

All-to-all broadcast on an eight-node hypercube.
All-to-all Reduction

- Similar communication pattern to all-to-all broadcast, except in the reverse order.

- On receiving a message, a node must combine it with the local copy of the message that has the same destination as the received message before forwarding the combined message to the next neighbor.
Cost Analysis

- On a ring, the time is given by: \((t_s + t_w m)(p - 1)\).
- On a mesh, the time is given by: \(2t_s(\sqrt{p} - 1) + t_w m(p - 1)\).
- On a hypercube, we have:

\[
T = \sum_{i=1}^{\log p} (t_s + 2^{i-1}t_w m) \\
= t_s \log p + t_w m(p - 1). \tag{2}
\]
All-Reduce and Prefix-Sum Operations

• In all-reduce, each node starts with a buffer of size \( m \) and the final results of the operation are identical buffers of size \( m \) on each node that are formed by combining the original \( p \) buffers using an associative operator.

• Same effect as all-to-one reduction followed by a one-to-all broadcast. (But this formulation is not the most efficient.) Uses the pattern of all-to-all broadcast, instead. The only difference is that message size does not change here. Time for this operation is \((t_s + t_wm) \log p\).

• Different from all-to-all reduction, in which \( p \) simultaneous all-to-one reductions take place, each with a different destination for the result.
The Prefix-Sum Operation

• Given $p$ numbers $n_0, n_1, \ldots, n_{p-1}$ (one on each node), the problem is to compute the sums $s_k = \sum_{i=0}^{k} n_i$ for all $k$ between 0 and $p - 1$.

• Initially, $n_k$ resides on the node labeled $k$, and at the end of the procedure, the same node holds $s_k$. 
Computing prefix sums on an eight-node hypercube. At each node, square brackets show the local prefix sum accumulated in the result buffer and parentheses enclose the contents of the outgoing message buffer for the next step.
The Prefix-Sum Operation

• The operation can be implemented using the all-to-all broadcast kernel.

• We must account for the fact that in prefix sums the node with label \( k \) uses information from only the \( k \)-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 the message comes from a node with a smaller label than the recipient node.

• The contents of the outgoing message (denoted by parentheses in the figure) are updated with every incoming message.
Scatter and Gather

- In the *scatter* operation, a single node sends a unique message to every other node (also called a one-to-all personalized communication).

- In the *gather* operation, a single node collects a unique message from each node.

- While the scatter operation is fundamentally different from broadcast, the algorithmic structure is similar, except for differences in message sizes (messages get smaller in scatter and stay constant in broadcast).

- The gather operation is exactly the inverse of the scatter operation and can be executed as such.
Gather and Scatter Operations

Scatter and gather operations.
Example of the Scatter Operation

The scatter operation on an eight-node hypercube.
Cost of Scatter and Gather

- There are $\log p$ steps, in each step, the machine size halves and the data size halves.

- We have the time for this operation to be:

$$T = t_s \log p + t_w m(p - 1).$$  \hspace{1cm} (3)

- This time is for a linear array as well as a 2-D mesh.

- These times are asymptotically optimal in message size.
All-to-All Personalized Communication

- Each node has a distinct message of size $m$ for every other 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.
All-to-All Personalized Communication

\[
\begin{array}{ccc}
M_{0,p-1} & M_{1,p-1} & M_{p-1,p-1} \\
\vdots & \vdots & \vdots \\
M_{0,1} & M_{1,1} & M_{p-1,1} \\
M_{0,0} & M_{1,0} & M_{p-1,0} \\
0 & 1 & \ldots & p-1
\end{array}
\]
All-to-All Personalized Communication: Example

Consider the problem of transposing a matrix.

- Each processor contains one full row of the matrix.
- The transpose operation in this case is identical to an all-to-all personalized communication operation.
All-to-all personalized communication in transposing a $4 \times 4$ matrix using four processes.
All-to-All Personalized Communication on a Ring

- Each node sends all pieces of data as one consolidated message of size $m(p - 1)$ to one of its neighbors.

- Each node extracts the information meant for it from the data received, and forwards the remaining $(p - 2)$ pieces of size $m$ each to the next node.

- The algorithm terminates in $p - 1$ steps.

- The size of the message reduces by $m$ at each step.
All-to-all personalized communication on a six-node ring. The label of each message is of the form \( \{x, y\} \), where \( x \) is the label of the node that originally owned the message, and \( y \) is the label of the node that is the final destination of the message. The label \( (\{x_1, y_1\}, \{x_2, y_2\}, \ldots, \{x_n, y_n\}) \) indicates a message that is formed by concatenating \( n \) individual messages.
All-to-All Personalized Communication on a Ring: Cost

- We have $p - 1$ steps in all.
- In step $i$, the message size is $m(p - i)$.
- The total time is given by:

$$T = \sum_{i=1}^{p-1} (t_s + t_w m(p - i))$$

$$= t_s (p - 1) + \sum_{i=1}^{p-1} it_w m$$

$$= (t_s + t_w mp/2)(p - 1).$$  \hspace{1cm} (4)$$

- The $t_w$ term in this equation can be reduced by a factor of 2 by communicating messages in both directions.
All-to-All Personalized Communication on a Mesh

- Each node first groups its $p$ messages according to the columns of their destination nodes.

- All-to-all personalized communication is performed independently in each row with clustered messages of size $m\sqrt{p}$.

- Messages in each node are sorted again, this time according to the rows of their destination nodes.

- All-to-all personalized communication is performed independently in each column with clustered messages of size $m\sqrt{p}$. 
All-to-All Personalized Communication on a Mesh

The distribution of messages at the beginning of each phase of all-to-all personalized communication on a $3 \times 3$ mesh. At the end of the second phase, node $i$ has messages ($\{0, i\}, \ldots, \{8, i\}$), where $0 \leq i \leq 8$. The groups of nodes communicating together in each phase are enclosed in dotted boundaries.
All-to-All Personalized Communication on a Mesh: Cost

• Time for the first phase is identical to that in a ring with \( \sqrt{p} \) processors, i.e., \((t_s + t_w mp/2)(\sqrt{p} − 1)\).

• Time in the second phase is identical to the first phase. Therefore, total time is twice of this time, i.e.,

\[
T = (2t_s + t_w mp)(\sqrt{p} − 1). \tag{5}
\]
All-to-All Personalized Communication on a Hypercube

- Generalize the mesh algorithm to $\log p$ steps.

- At any stage in all-to-all personalized communication, every node holds $p$ packets of size $m$ each.

- While communicating in a particular dimension, every node sends $p/2$ of these packets (consolidated as one message).

- A node must rearrange its messages locally before each of the $\log p$ communication steps.
An all-to-all personalized communication algorithm on a three-dimensional hypercube.
All-to-All Personalized Communication on a Hypercube: Cost

- We have \( \log p \) iterations and \( mp/2 \) words are communicated in each iteration. Therefore, the cost is:

\[
T = (t_s + t_w mp/2) \log p. 
\]  

(6)

- This is not optimal!
All-to-All Personalized Communication on a Hypercube: Optimal Algorithm

- Each node simply performs $p - 1$ communication steps, exchanging $m$ words of data with a different node in every step.

- A node must choose its communication partner in each step so that the hypercube links do not suffer congestion.

- In the $j$th communication step, node $i$ exchanges data with node $(i \text{ XOR } j)$.

- In this schedule, all paths in every communication step are congestion-free, and none of the bidirectional links carry more than one message in the same direction.
All-to-All Personalized Communication on a Hypercube: Optimal Algorithm
Seven steps in all-to-all personalized communication on an eight-node hypercube.
All-to-All Personalized Communication on a Hypercube: Cost Analysis of Optimal Algorithm

- There are $p - 1$ steps and each step involves non-congesting message transfer of $m$ words.

- We have:

$$T = (t_s + t_w m)(p - 1).$$  \(7\)

- This is asymptotically optimal in message size.