COMMUNICATION OPERATIONS AND MESSAGE PASSING

- Introduction to programming with message passing
- Broadcast operations
- All-to-all broadcast and reduction operations
- Scatter and Gather operations
- All-to-all personalized communication
Introduction to message-passing

➢ Need to explicitly code the exchange of messages [data, control,..]

Example: Revisit the sum example seen earlier

**ALGORITHM : 1. Parallel Sum of n numbers**

1. for \( j = 0; \ j < p; \ j++ \) \{ \ // Parallel Loop
2. \( tmp(j) = 0; \)
3. for \( i = j \ast m; \ i < (j + 1) \ast m; \ i++ \)
4. \( tmp(j) = tmp(j) + x(i) \ \}; \)
5. \( s = 0; \)
6. for \( j = 0; \ j < p; \ j++ \) \ // Sequential loop
7. \( s = s + tmp(j); \)
Let “root” = ‘master’ node where the sum ends up. Recall: 
\[ m = n/p \]

**ALGORITHM : 2.** Parallel Sum with communication

1. If \((myid == root)\) {
2. read array \(x\);
3. For \((j = 0, j < p && j! = root; j++)\) 
4. send \(x(j \times m : (j + 1) \times m - 1)\) to proc. \(j\) }
5. else 
6. receive \(xloc(1 : m)\) from \(root\);
7. \(tmp = 0;\)
8. for \((i = 0; i < m; i++)\) 
9. \(tmp += xloc(i);\)
10. REDUCE\((sum, tmp, ADD)\) \(\) Reduction oper.
REDUCE\((sum, \textit{tmp},'\textit{ADD}')\) adds 'tmp' from each PE into 'sum'

- Can do reductions with add, multiply max, min, etc..
- More on reductions later.
- Next: we will see some of the common communication functions used –
Communication ‘kernels’

Typical questions addressed:

1. Identify the important communication operations
2. Find effective algorithms for performing these on distributed memory computers
3. Analyze their cost

A by-product: some framework for generic algorithms
Example: Broadcast operation

Sending a message from a 'root' node to all nodes is a Broadcast

Questions: Best way to broadcast a message from a root node to all others in a ring? In hypercube?
Reduction does a global operation (e.g. a sum) on items located on all processors onto a 'root' processor.

Can be viewed as a sort of inverse of the broadcast.
In parallel sum example, could replace the sends of $x(j \times m : (j + 1) \times m - 1)$ from root to all others by a broadcast of all $x$ from root of the vector $x$. Lines 1 – 6 replaced by:

1. broadcast($x$, root)

Note however that each PE will get the whole vector.

Corresponding MPI code provided in class web-site
All-to-all broadcast and reduction

- All-to-all broadcast can be viewed as $p$ broadcasts, one from each node.
- Similarly: All-to-all reduction is a reduction to each node (different for each node).

**Note:** All-reduce ($\neq$ all-to-all reduce) is a reduction operation in which the result of reduction is available in each processor

- All-reduce achievable by a reduce followed by a broadcast [not best way]
Important application of all-reduce: testing if an algorithm has “converged”.

**Example:** Test would be something like:

\[
\text{if } \max_{i=0,\ldots,p-1} |x^i_k - x^i_{k+1}| < \text{ then stop}
\]

- Variable \(i\) = processor, variable \(k\) = iteration number
- Need to know \(\max_i |x^i_k - x^i_{k+1}|\) in each processor.
- See text for algorithms on linear array, ring, and hypercubes
**Gather and scatter operations**

- Scatter is similar to a broadcast – but a different item is sent to each processor -

- Gather does the inverse operation.
How would you implement a Scatter operation on a hypercube?

Cost?
Example: For the parallel sum example – we can “scatter” the subvectors to be summed up in each processors.

In parallel sum algorithm, the lines

```c
for (j = 0; j < p && j != root; j++) {
    send x(j*m : (j+1)*m-1) to process j
} else {
    receive xloc(1:m) from root;
}
```

are replaced by

3. scatter(x)
**All-to-All personalized communication**

- Can be viewed as a scatter from each node: each node sends a distinct message to every other node.

```
<table>
<thead>
<tr>
<th>P0</th>
<th>A0</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>B0</td>
<td>B1</td>
<td>B2</td>
<td>B3</td>
</tr>
<tr>
<td>P2</td>
<td>C0</td>
<td>C1</td>
<td>C2</td>
<td>C3</td>
</tr>
<tr>
<td>P3</td>
<td>D0</td>
<td>D1</td>
<td>D2</td>
<td>D3</td>
</tr>
</tbody>
</table>
```

Equivalent to $p$ gathers too (0ne to each node)

Notice: operation amounts to transposing a $p \times p$ array!

How would you code an all-to-all communication on a hypercube?