Assignment 3

Exercise 3 - Broadcast

Problem description

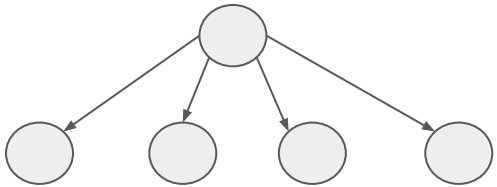
- Determine the bandwith of the system by performing broadcast of an array to all processes.
- The bandwith is computed as the array size / time.

Reference

- int MPI_Bcast(void *buffer, int count, MPI_Datatype datatype, int root, MPI_Comm comm)
  - Buffer - data to be sent
  - Count - number of elements
  - Datatype - type of each element (MPI_DOUBLE)
  - Root - root node which holds the initial data

Naive approach

- Send data from the root process directly to all other processes

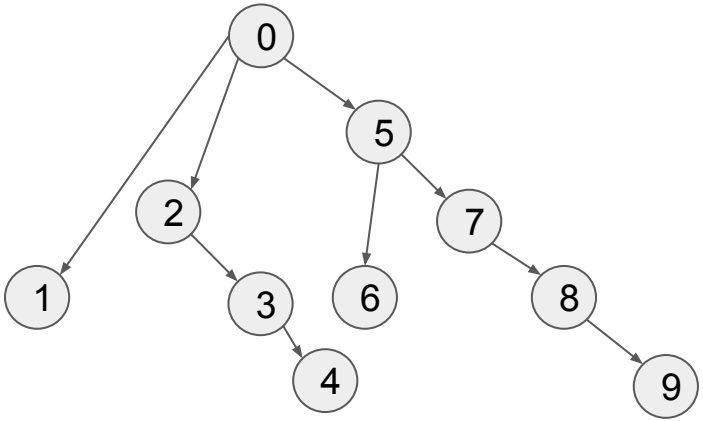


```
// processes
for (int i = 0; i < size; i++)
    if (i != root)
        MPI_Send(buffer, count, datatype, i, 0, comm);

// root
MPI_Recv(buffer, count, datatype, root, 0, comm, MPI_STATUS_IGNORE);
```

Tree approach

- Starting with the root node, send the array to process p/2, then repeat the procedure with all nodes that hold the data.



0	1	2	3	4	5	6	7	8	9

Tree approach

```
// determine parent and domain
while (left != rank) {
    int mid = (right + left) / 2;
    if (rank < mid) {
        right = mid;
    } else {
        parent = left;
        left = mid;
    }
}
```

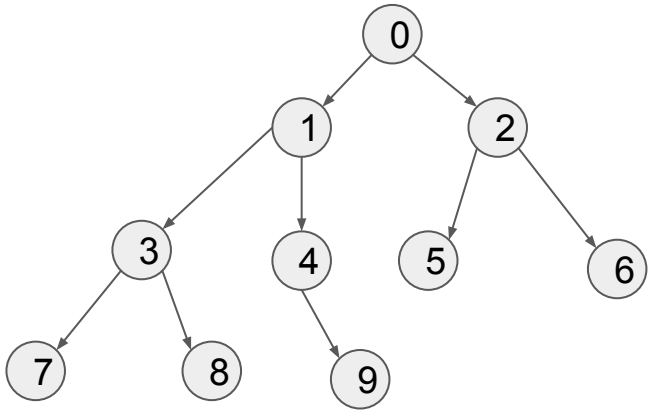
```
MPI_Recv(buffer, count, datatype, parent, 0, comm,
MPI_STATUS_IGNORE);


  while (1) {
      int dest = (right + left) / 2;
      if (dest == rank)
          break;

      MPI_Send(buffer, count, datatype,
  dest, 0, comm);
      right = dest;
  }
```

Tree approach - O(n) traffic - bonus

- Binary tree represented as an array
- Each node has two children (2i+1 and 2i+2 inside the array)
- Parent is computed as (i-1)/2
- Load balancing in terms of traffic per node



Tree approach - O(n) traffic - bonus

```
// receive
MPI_Recv(buffer, count, datatype, (rank-1)/2, 0, comm, MPI_STATUS_IGNORE);


// send
for (int i = 1; i <= 2; i++)
    if (rank*2+i < size)
        MPI_Send(buffer, count, datatype, rank*2+i, 0, comm);
```

Performance analysis

```
for (int i = 0; i < cases; i++) {
     MPI_Barrier(MPI_COMM_WORLD);
     double start = MPI_Wtime();
     bcasts[i].func(v, n, MPI_DOUBLE, 0, MPI_COMM_WORLD);
     MPI_Barrier(MPI_COMM_WORLD);
     double stop = MPI_Wtime();

     bcasts[i].duration = stop - start;
     if (!check_array(v, n))
         printf("For rank=%d; %s failed", rank, bcasts[i].name);
 }
```

Performance analysis

4 cases; 64 processes

Longest Time for Naive was 19.347412 seconds; array=100000000; bwidth=41349199.338776 B/s

Longest Time for Tree was 1.186976 seconds; array=100000000; bwidth=673981638.786326 B/s

Longest Time for Bonus was 1.894390 seconds; array=100000000; bwidth=422299502.822174 B/s

Longest Time for MPI_Bcast was 0.639517 seconds; array=100000000; bwidth=1250943937.153123 B/s

Performance analysis

4 cases; 128 processes

Longest Time for Naive was 39.220236 seconds; array=100000000; bwidth=20397633.598585 B/s

Longest Time for Tree was 1.441020 seconds; array=100000000; bwidth=555162310.996751 B/s

Longest Time for Bonus was 2.112662 seconds; array=100000000; bwidth=378669215.020269 B/s

Longest Time for MPI_Bcast was 0.723280 seconds; array=100000000; bwidth=1106072409.000889 B/s

Questions ?