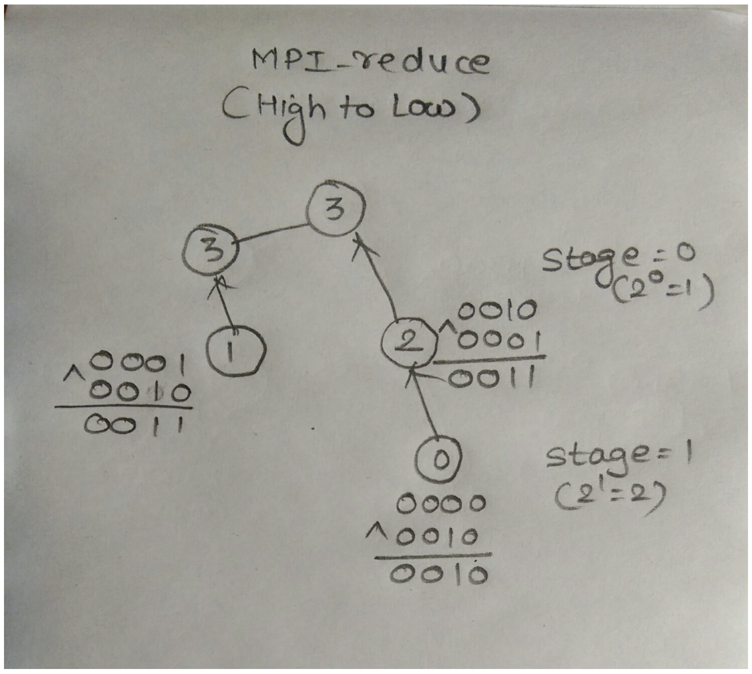
HPSC / CSCI 5576 /Rashmi Oak

Assignment 2

Task 1: Draw node diagrams



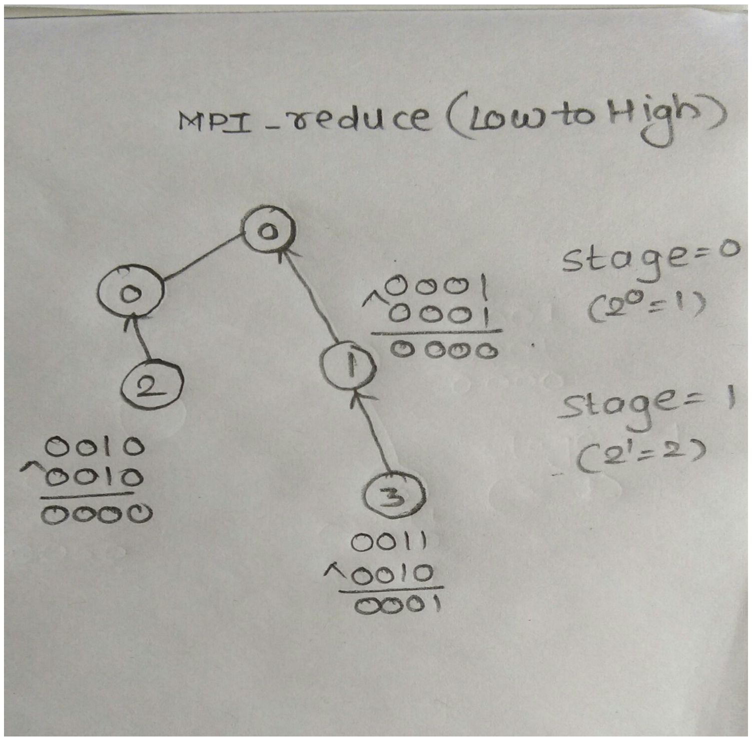


Figure 1. MPI reduce (High to Low and Low to High)

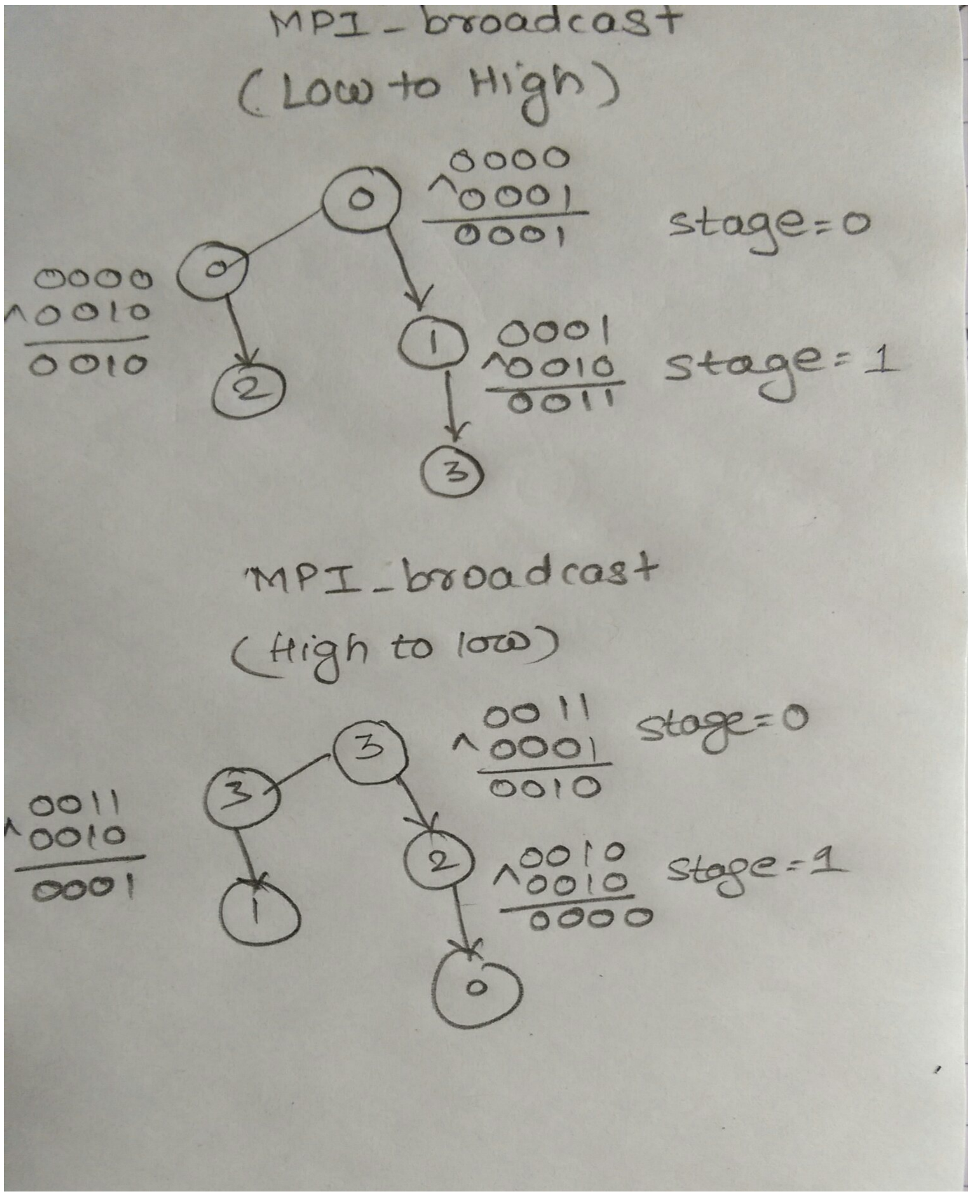
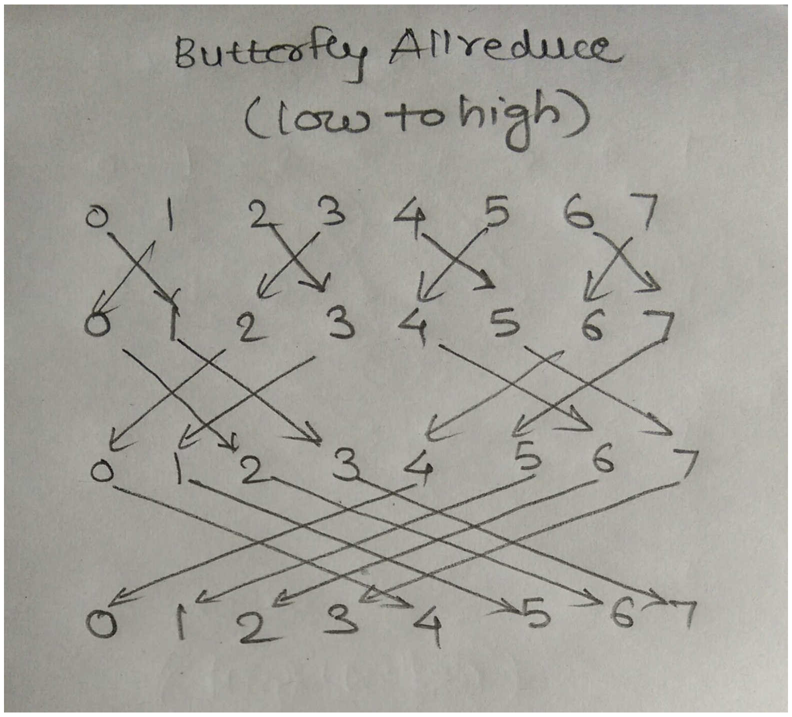


Figure 2. MPI_broadcast (Low to High and High to Low)



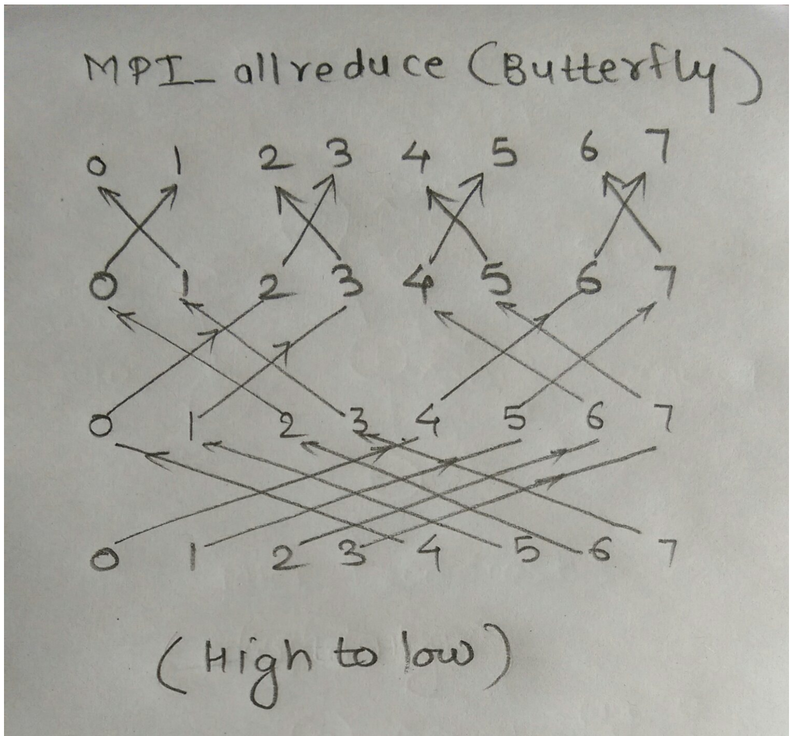


Figure 3. Butterfly Allreduce (Low to High and High to Low)

Note: Figure 1,2 and 3 are drawn for a 4 node network.

Task 2: Show outputs for all implementations

I am attaching nine text files along with this report. They include output of all the implementations for 4nodes.

MPI broadcast low to high and high to low output (2)
MPI reduce low to high and high to low output (2)
MPI butterfly Allreduce low to high and high to low (2)
MPI custom Allreduce using custom reduce and broadcast routines low to high and high to low (2)
MPI all to one and one to all Allreduce (1)

Task 3

For performance analysis, I commented out all "printf" statements in the code and ran each code 10 times and took average of 10 timings.
I used following to calculate execution time:

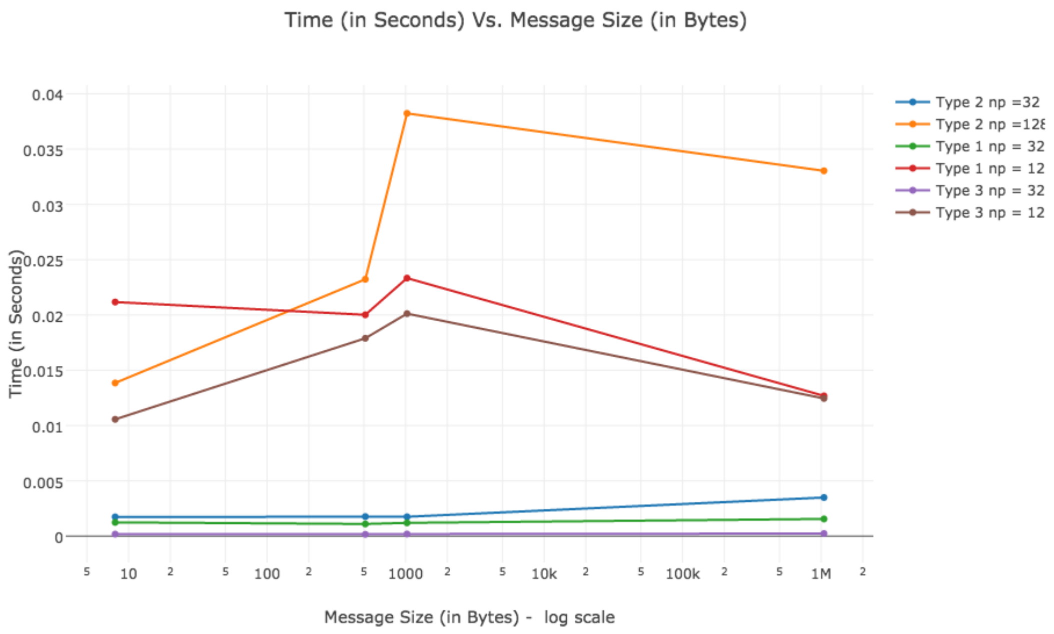
```
MPI_Init(&argc, &argv);

   /* Get my process rank */
   MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

   /* Find out how many processes are being used */
   MPI_Comm_size(MPI_COMM_WORLD, &p);

   MPI_Barrier(MPI_COMM_WORLD);
   start = MPI_Wtime();

(piece of code)

MPI_Barrier(MPI_COMM_WORLD);
   finish = MPI_Wtime();

   local_elapsed = finish - start;

   MPI_Reduce(&local_elapsed, &elapsed_time, 1, MPI_DOUBLE, MPI_MAX, 0,
MPI_COMM_WORLD);
   if(my_rank == 0){
   printf("execution time = %f\n",elapsed_time );
   }
   MPI_Finalize();
```

Task 4

Performance analysis

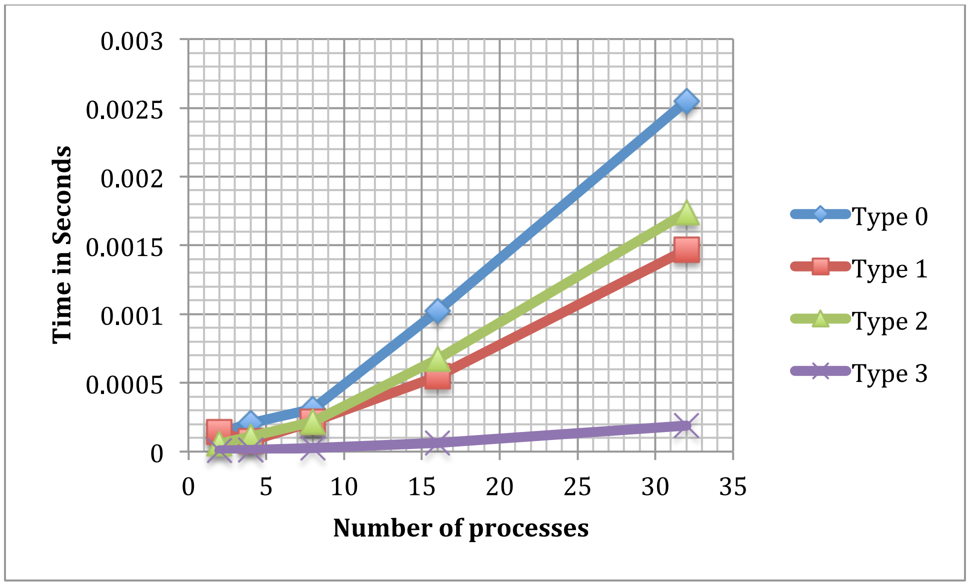
Time Vs. Message size for np =32 and np = 128

Type 1- butterfly allreduce
Type 2- custom allreduce using custom broadcast and custom reduce
Type 3 - MPI_Allreduce

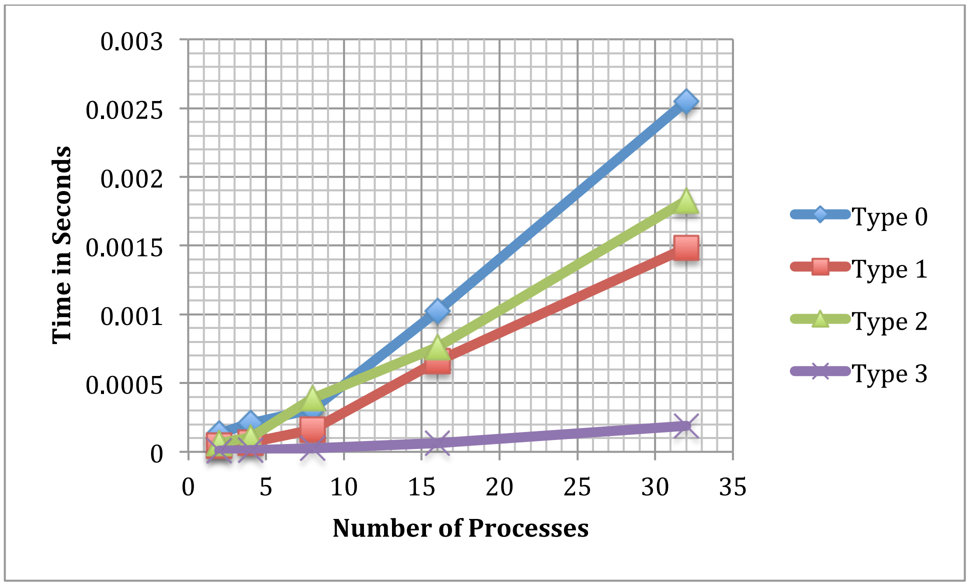


Time in Seconds Vs. Number of Processes

0 - One to all and all to one allreduce
1- butterfly allreduce
2- custom allreduce using custom broadcast and custom reduce
3 - MPI_Allreduce



(Low to High)



(High to Low)

Notes:
Network Analysis:

1. I plotted graphs for various data sizes 8,512,1024,1024*1024 bytes. The double pointer was used to allocate 2^20 bytes of memory.
2. All timings are collected for np = 32 and 128 on the system with following specifications:
3. System specification:

   Architecture: **x86_64**
   CPU op-mode(s): **32-bit, 64-bit**

```
Byte Order:          Little Endian
CPU(s):              32
On-line CPU(s) list: 0-31
Thread(s) per core:  2
Core(s) per socket:  8
Socket(s):           2
NUMA node(s):        2
Vendor ID:           GenuineIntel
CPU family:          6
Model:               63
Stepping:            2
CPU MHz:             1200.000
BogoMIPS:            4801.37
Virtualization:      VT-x
L1d cache:           32K
L1i cache:           32K
L2 cache:            256K
L3 cache:            20480K
NUMA node0 CPU(s):   0-7,16-23
NUMA node1 CPU(s):   8-15,24-31
```

4. It can be observed that as data size increases algorithm slows down. For np=32 a similar behavior is observed for all three MPI Allreduce techniques out of which MPI_Allreduce outperforms the other two by approximately factor of 10.

Latency analysis:

1. I plotted all four Allreduce techniques mentioned above for this analysis
2. I ran each test 10 times and took average of all timings for low to high and high to low.
3. It can be seen that all to one and one to all Allreduce underperforms while MPI_Allreduce routine performs better than all the other three.
4. This plot looks like a exponential function. Time is increasing exponentially as we increase the number of processes. But MPI routine, which is fully optimized routine, the graph is somewhat linear with time not changing much with the increase in number of processes.
5. For rest of the routines the time increases as we increase number of processes.

Physical topology and algorithm:

I think physical topology of the system will affect performance of the algorithm. But in this case, with respect to MPI_allreduce function it is had to beat the routine by any other algorithm. Even butterfly topology underperforms MPI_Allreduce routine.

Question 1

Built in MPI_Allreduce performs better than all custom Allreduce implementations.
Butterfly implementation performs best among all custom Allreduce implementations

Question 2

In my opinion, the order does not matter. There is a slight difference in the timings collected for high to low and low to high implementations but it is not significant to conclude that order does matter.

Question 3

It does matter. The implementation of “low to high reduce” and “high to low broadcast” will output bad results. This is because, it doesn’t make any sense to reduce the result to rank 0 and then broadcast result from rank 3 in case of 4-node network. It will not output the desired results.