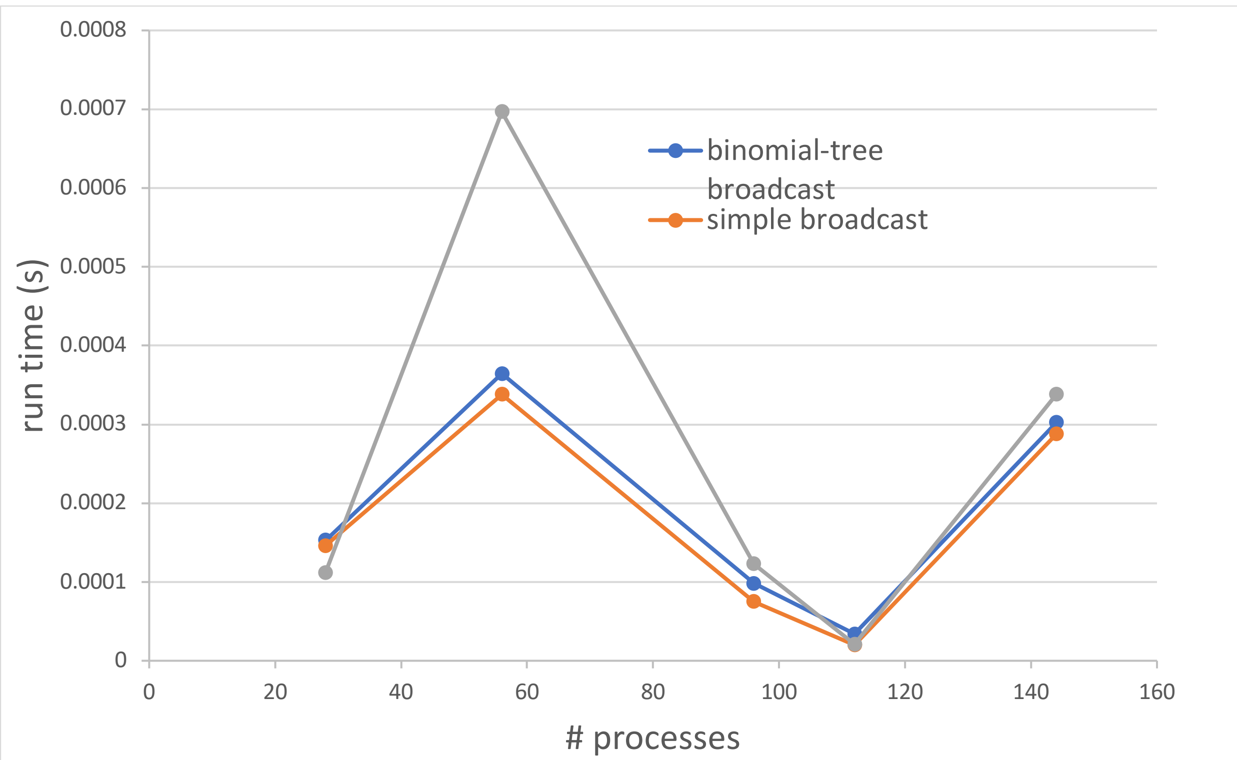
Homework 5: Collectives
Han Tran

Task 1
The code is attached as hw5_task1.zip

I ran the code on kingspeak wih 1, 2, 4 nodes. However, I could not manage to run on kingspeak with 8 nodes because it took forever to wait on the queue. Therefore I ran the code on ash-guest.chpc.utah.edu with 8 and 12 nodes (12 procs/node). The run time is shown in the table below. The log_bcast can run with arbitrary *p* and arbitrary *counts*.

np	28	56	96	112	144
log_bcast	0.000153	0.0003644	0.00009799	0.00003409	0.000302
simple_bcast	0.000146	0.00033784	0.0000751	0.00002003	0.000288
MPI_Bcast	0.000112	0.00069661	0.00012302	0.00002098	0.000338



As seen, the run time using log_bcast (i.e. broadcast using binomial tree algorithm) and using MPI_Bcast both are not different with simple_bcast. This is not understandable because, even with MPI_Bcast which is supposed to be the best algorithm, the run time is still bigger than simple_bcast.

Task 2

To implement Gather or Scatter efficiently, I would use the algorithm similarly to the broadcasting, i.e. using the binomial tree. For illustration, let's take the case of Scatter for 8 processes (*p* =8), the root will need to send each element of an array A= [a0, a1,…,a7] to all processes so that each process will receive each element corresponding to the process #. Following is the procedure:

Step *i* = 0:

p0 → p1: [a1, a3, a5, a7]

Step *i* = 1:

p0 → p2: [a2, a6]

p1 → p3: [a3, a7]

Step *i* = 2:

p0 → p4: [a4]

p1 → p5: [a5]

p2 → p6: [a6]

p3 → p7: [a7]

<u>Parallel runtime:</u>

$$(p-1)t_s + \left(\frac{p}{2}log_2p\right)\left(\frac{n}{p}\right)t_w$$

where *p* is the number of processes; t_s and t_w are start-up time and per-word transfer time respective, n is the total size of data (i.e. chunksize = *n*/*p*).

<u>Pseudo code:</u>

m = $\log_2$(p)

for i = 0, 1, …, m-1

if (rank < 2^i)

"*rank*" sends to "*rank* + 2^i" the data of $\left({}^{p}/_{2^{(i+1)}}\right)$

elseif (2^i <= rank < 2^{i+1})

"*rank*" receives from "*rank* - 2^i) the data of $\left({}^{p}/_{2^{(i+1)}}\right)$

endif

end for

For the case of Gather, we would do the same algorithm as Scatter but now we go in the opposite direction, i.e. starting from the bottom of the tree (i.e. each process) and go up to the top (i.e. the root process).

The above algorithm is for the case of *p* is the power of 2. For arbitrary *p*, we need to replace *m*=$\log_2 p$ by *m* = ceil($\log_2 p$) (i.e. *m* is the smallest possible integer value which is greater than or equal $\log_2$p), and in the binomial tree will miss the positions starting from p to 2^m.

Task 3

a) The tests for v variant communications (MPI_Gatherv, MPI_Scatterv, MPI_Allgatherv, and MPI_Alltoallv) are included in the attached hw5_task3.zip

b) This algorithm for doing the transpose is to use MPI_Alltoall with sendbuff of each process is the row of the matrix. After MPI_Alltoall, the recvbuff of each process will be the row of the transpose matrix.
   I do not have enough time to finish this task. I may go back to complete this part at a later time.

Task 4

The debugged codes are in the attached hw5_task4.zip.
For each file, please see the description of the bug and how to fix is presented at the beginning of the program.