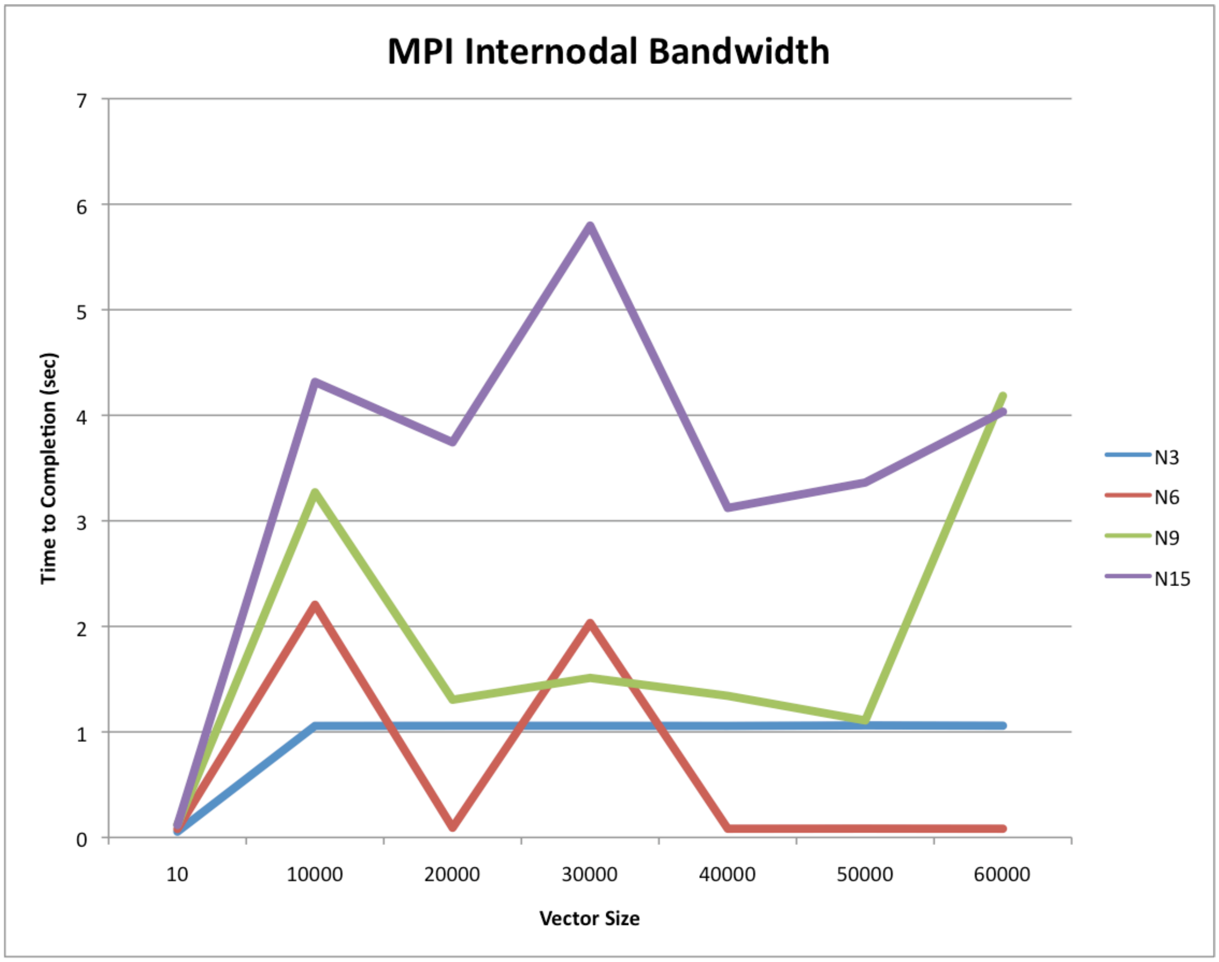
Project 3 : Reduce / Broadcast

Duane Johnson

For this project, the hypercube "reduce" and hypercube "broadcast" algorithms were implemented in MPI for an arbitrary number of nodes with the ability to reduce or broadcast to an arbitrary node number within the set of nodes. See page 2 for a sample demonstrating the summation of several vectors with arbitrary nodes and root.

Surprisingly, the internodal bandwidth was not very consistent, and so it was difficult to obtain a reasonable bandwidth number. For example, with 6 node (red line) the "40k messages" through "60k messages" were very fast. This analysis used the worst-case of a sample of 3 timing runs but consistently produced these low numbers.

Using this data, the maximum bandwidth achieved was about 60000 * log(15) / 4.03 seconds = 58,064 integer messages per second--i.e. an integer vector of 60000 on each of 15 nodes is passed back and forth log(15) times. We divide by the total time taken (4.03 seconds) to get messages/second.



```
$ mpirun -np 5 redbroad --root 1
Nodes: 5
Dimensionality: 3
Vector size: 10
Root node: 1
Initial vector values for each node: 0 1 2 3 4 5 6 7 8 9
[0]: 0 (1) recv from 1 (2)
[0]: 1 (2) send to 0 (1)
[0]: 2 (3) recv from 3 (4)
[0]: 3 (4) send to 2 (3)
[1]: 2 (3) send to 0 (1)
[1]: 0 (1) recv from 2 (3)
[2]: 0 (1) recv from 4 (0)
[2]: 4 (0) send to 0 (1)
Final vector values for all nodes: 0 5 10 15 20 25 30 35 40 45
Completed in 1.105300 seconds.

$ mpirun -np 5 redbroad --root 2
Nodes: 5
Dimensionality: 3
Vector size: 10
Root node: 2
Initial vector values for each node: 0 1 2 3 4 5 6 7 8 9
[0]: 3 (0) send to 2 (4)
[0]: 1 (3) send to 0 (2)
[0]: 0 (2) recv from 1 (3)
[0]: 2 (4) recv from 3 (0)
[1]: 2 (4) send to 0 (2)
[1]: 0 (2) recv from 2 (4)
[2]: 0 (2) recv from 4 (1)
[2]: 4 (1) send to 0 (2)
Final vector values for all nodes: 0 5 10 15 20 25 30 35 40 45
Completed in 1.108573 seconds.
```

**Sample output showing a Summation on 5 nodes,
reducing and broadcasting to node 1 followed by node 2.**