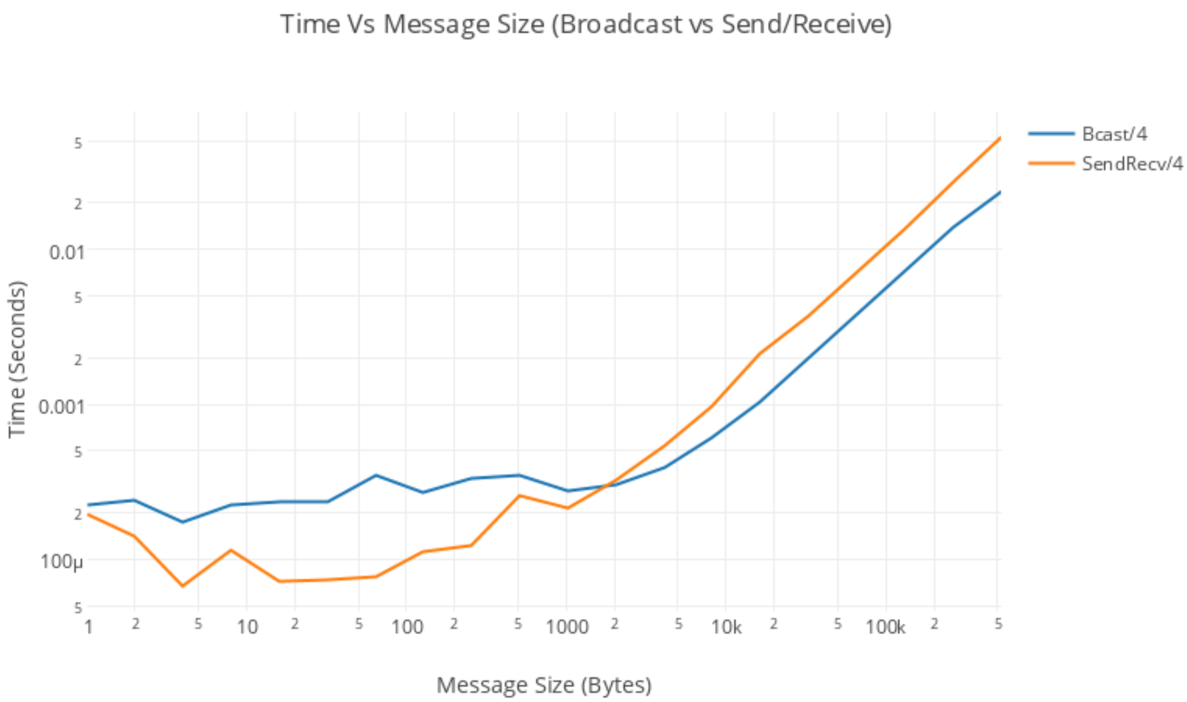
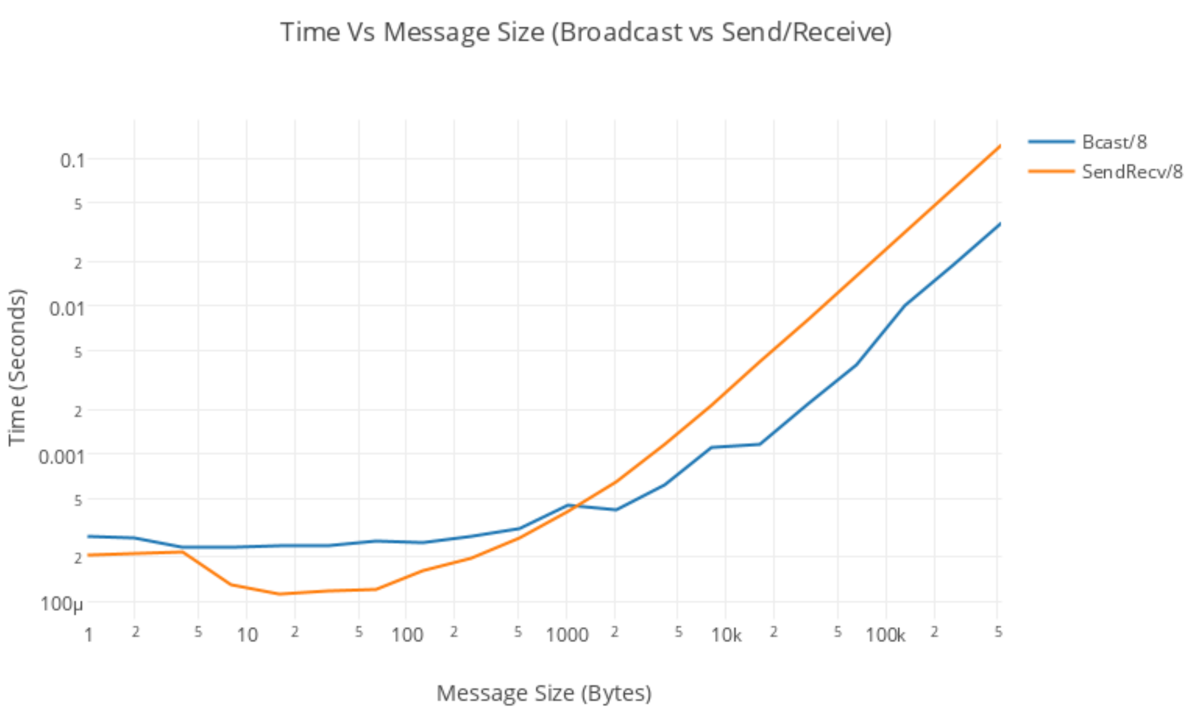
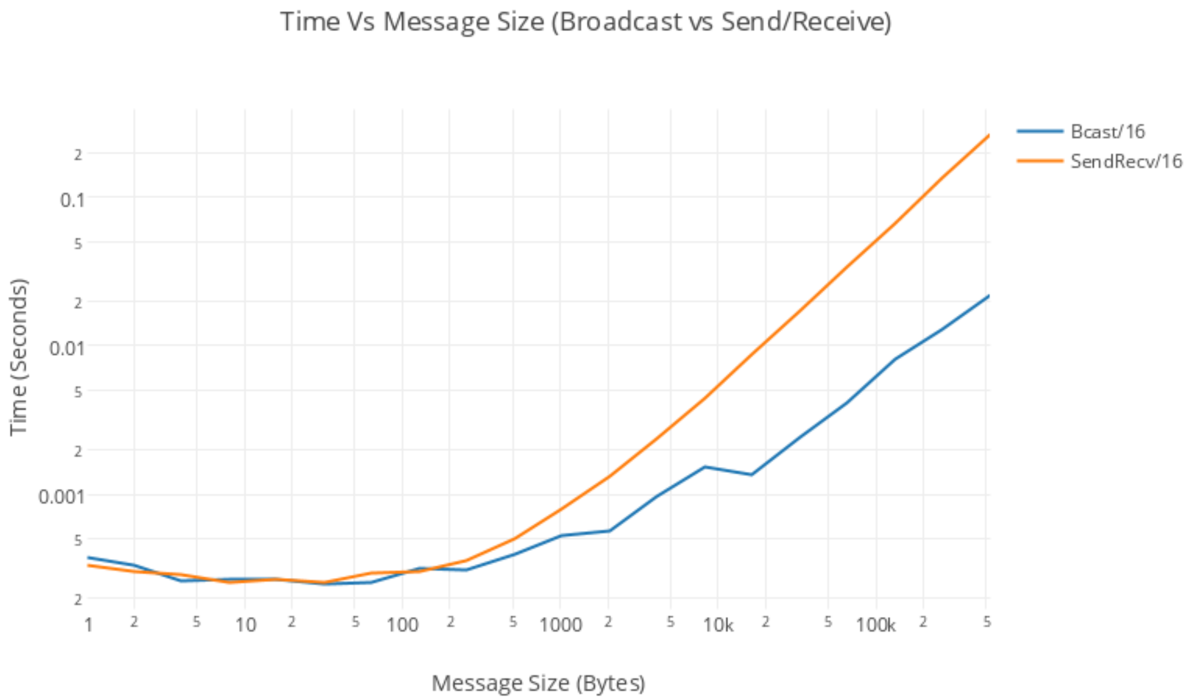
Question 1-a

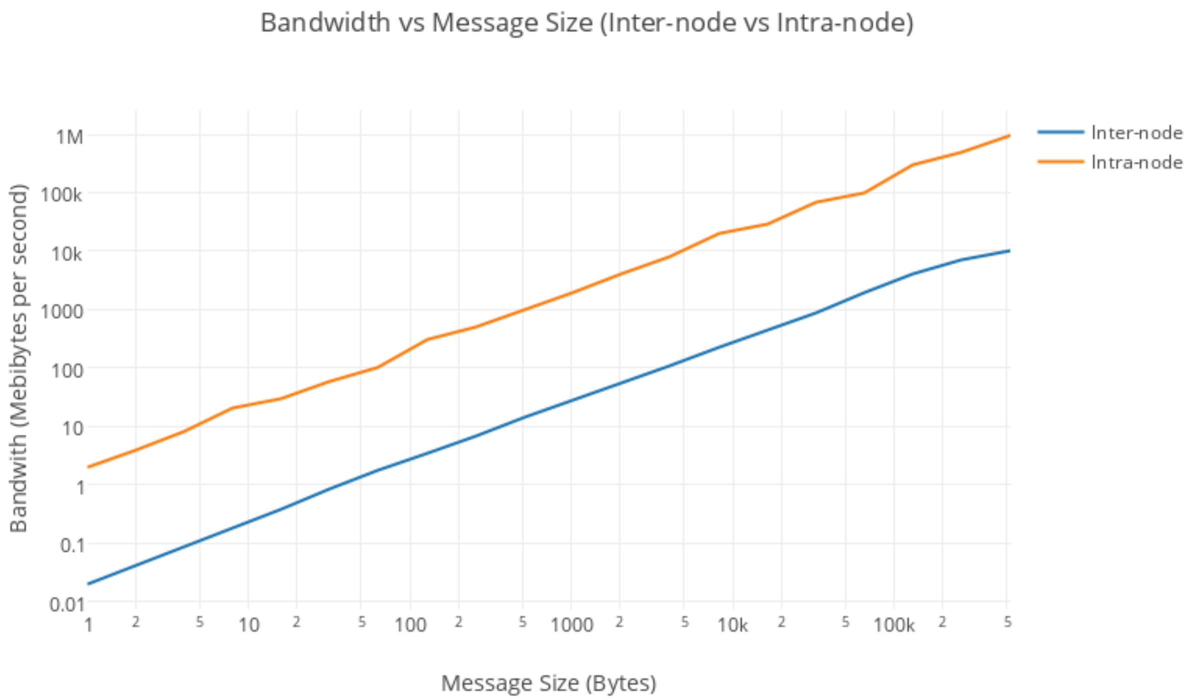

Time Vs Message Size (Broadcast vs Send/Receive)
Bcast/4
SendRecv/4
Time (Seconds)
Message Size (Bytes)
0.01
0.001
100μ
1
10
100
1000
10k
100k


Time Vs Message Size (Broadcast vs Send/Receive)
Bcast/8
SendRecv/8
Time (Seconds)
Message Size (Bytes)
0.1
0.01
0.001
100μ
1
10
100
1000
10k
100k



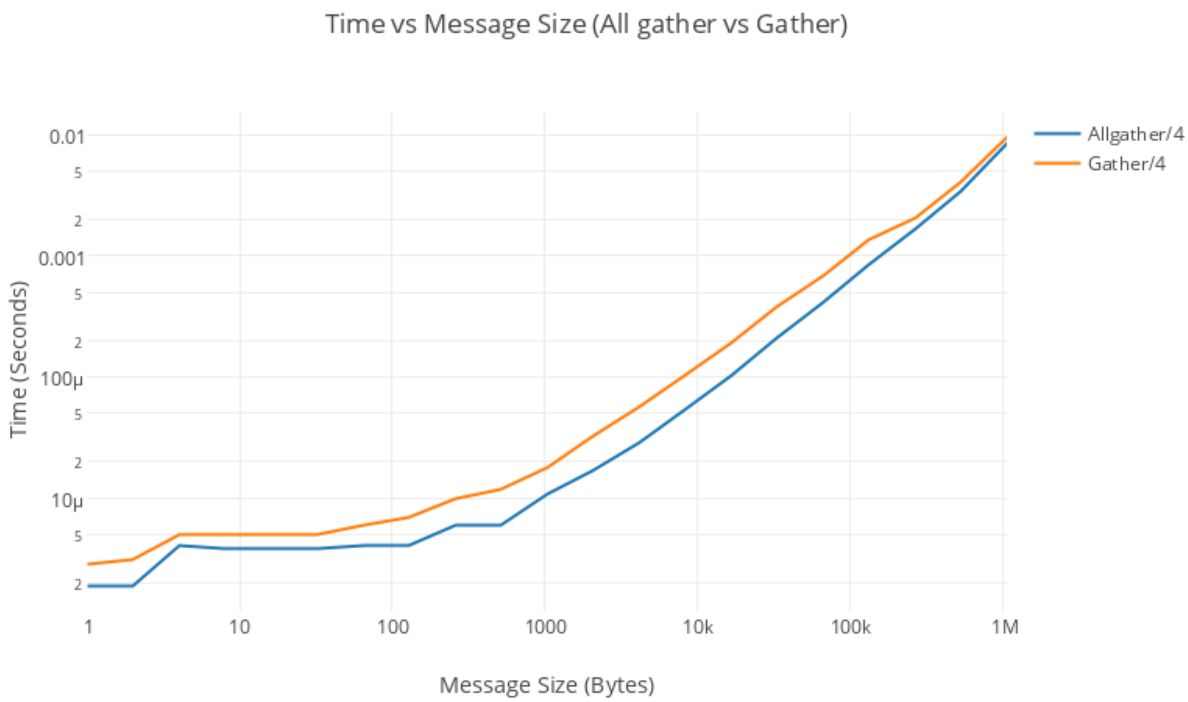
These graphs measure the time it took for a process to broadcast a message of different sizes to all other processes. With one process per physical node. The graphs show that when the message is small a combination of sends and receives is faster than a call to broadcast. However, as the message gets larger it seems that broadcast becomes more efficient. This probably means that broadcast is doing some sort of optimization for large messages.

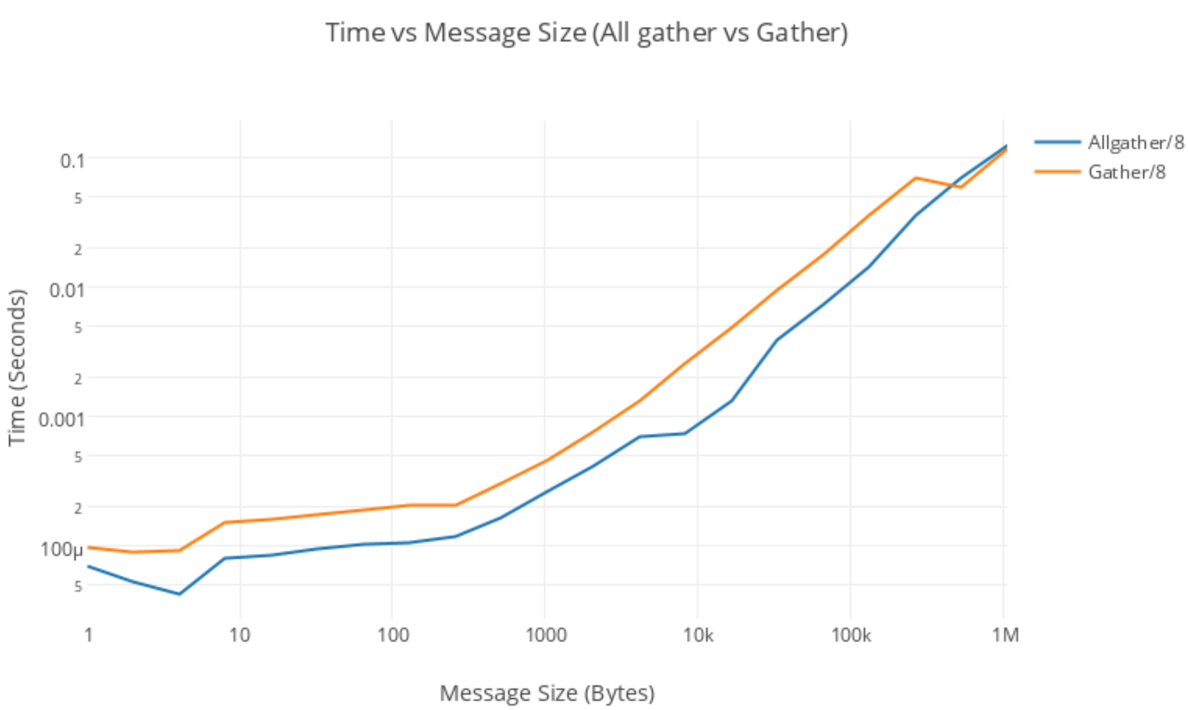
Question 1-b

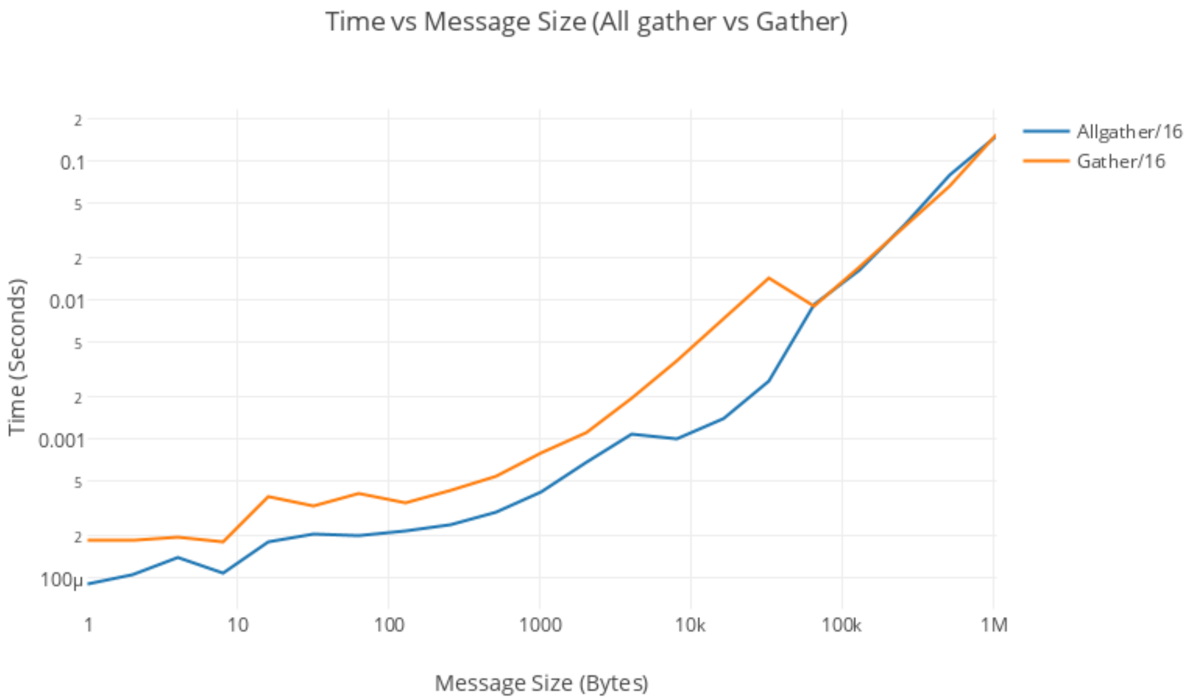


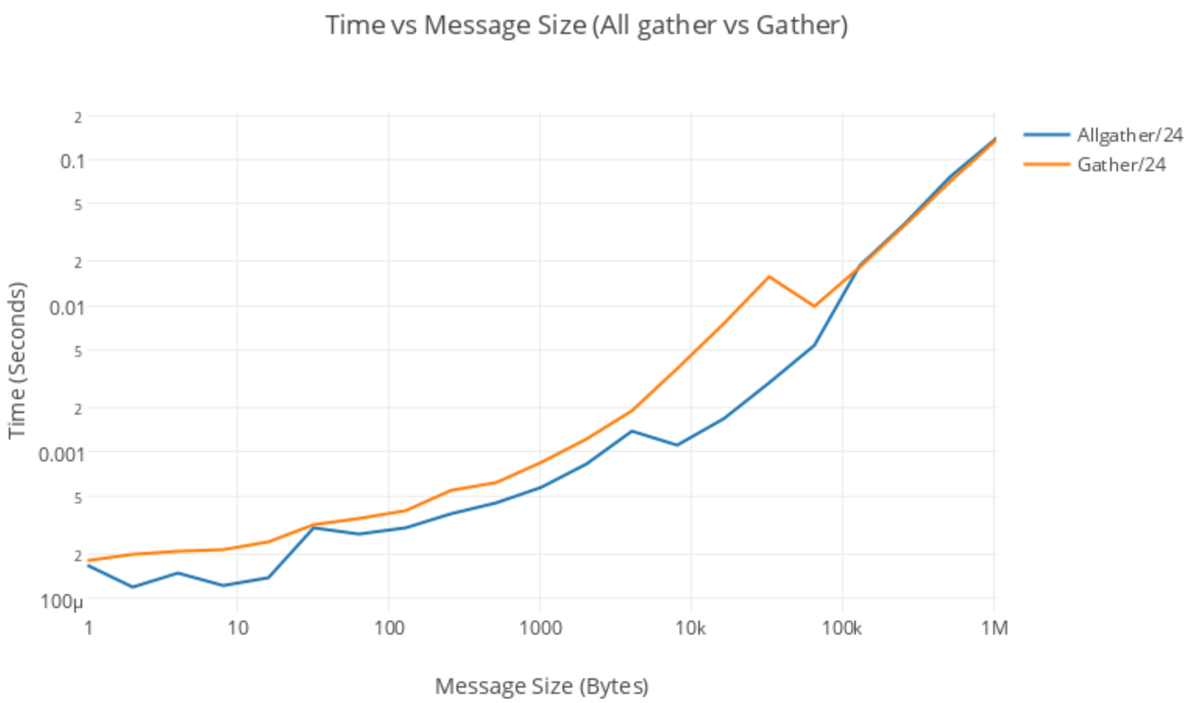
This graphs measures the bandwidth when sending a message from one process to another. As expected the bandwidth for sending messages to a node inside of the same CPU is bigger than when sending a message to a process in a separate physical node. This is probably caused the extra time needed for the information to travel through wires and likely a slower connection speed.

Question 2-a


Time vs Message Size (All gather vs Gather)
Allgather/4
Gather/4
Time (Seconds)
0.01
5
2
0.001
5
2
100μ
5
2
10μ
5
2
1
10
100
1000
10k
100k
1M
Message Size (Bytes)

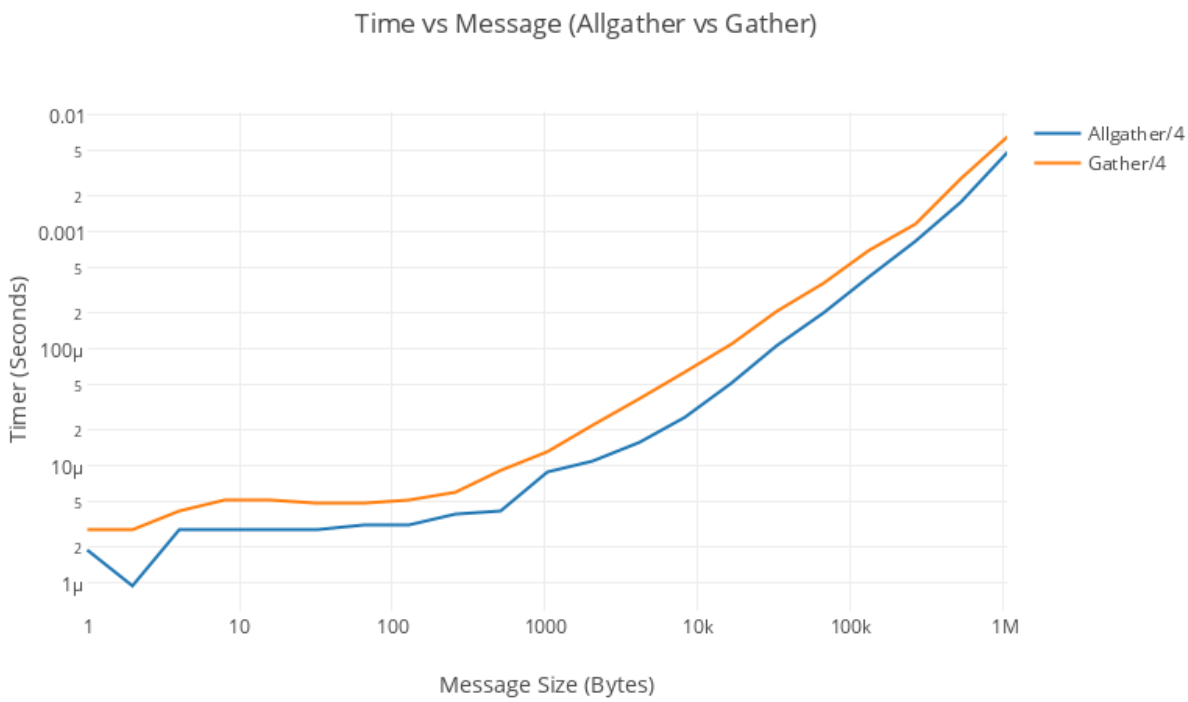

Time vs Message Size (All gather vs Gather)
Allgather/8
Gather/8
Time (Seconds)
0.1
5
2
0.01
5
2
0.001
5
2
100μ
5
1
10
100
1000
10k
100k
1M
Message Size (Bytes)

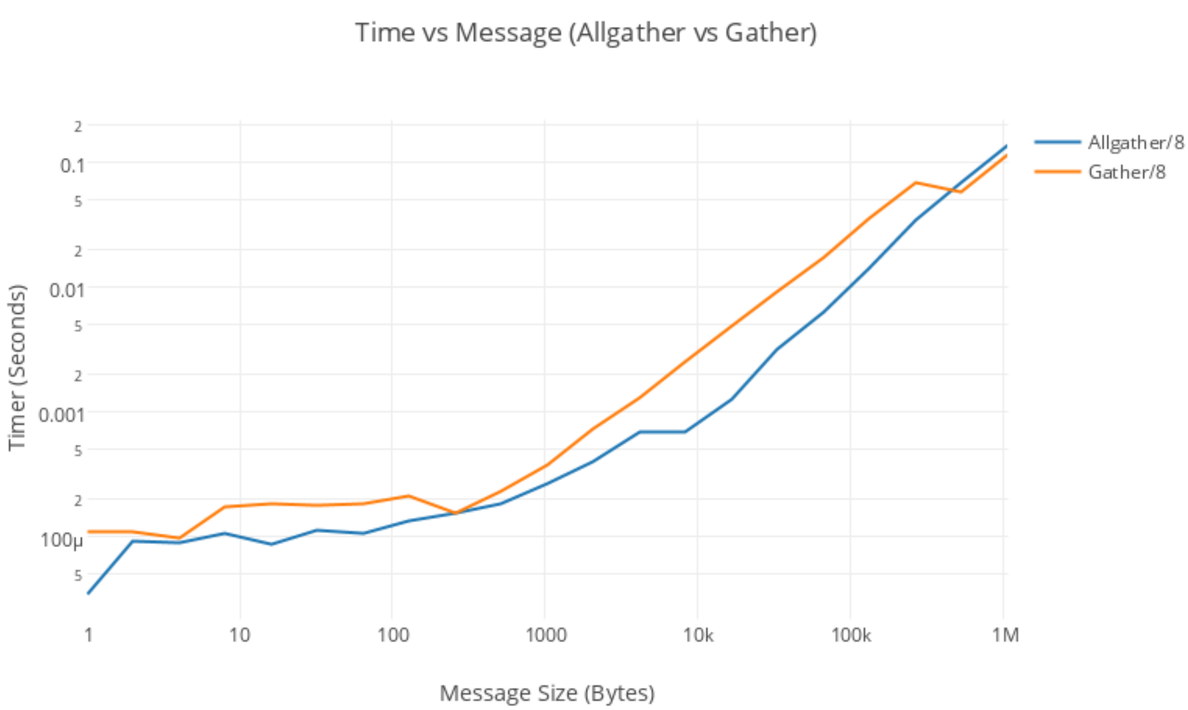

Time vs Message Size (All gather vs Gather)
Allgather/16
Gather/16
Time (Seconds)
Message Size (Bytes)
2
0.1
5
2
0.01
5
2
0.001
5
2
100μ
1
10
100
1000
10k
100k
1M

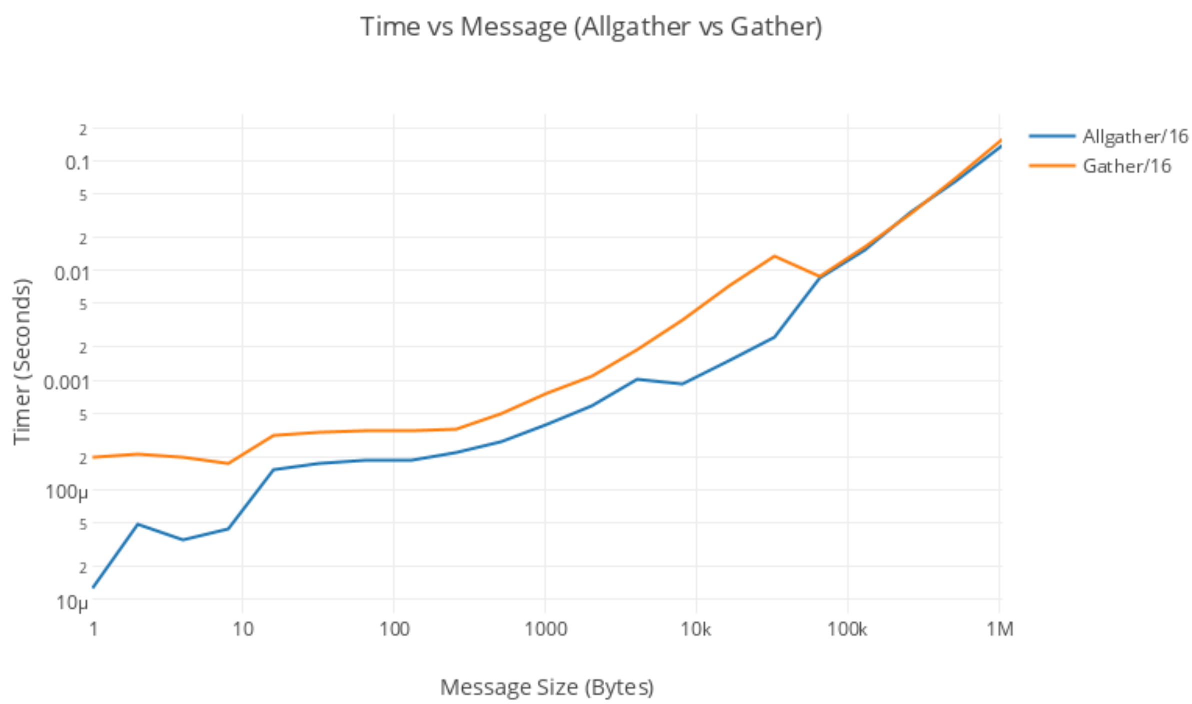

Time vs Message Size (All gather vs Gather)
Allgather/24
Gather/24
Time (Seconds)
Message Size (Bytes)
2
0.1
5
2
0.01
5
2
0.001
5
2
100μ
1
10
100
1000
10k
100k
1M

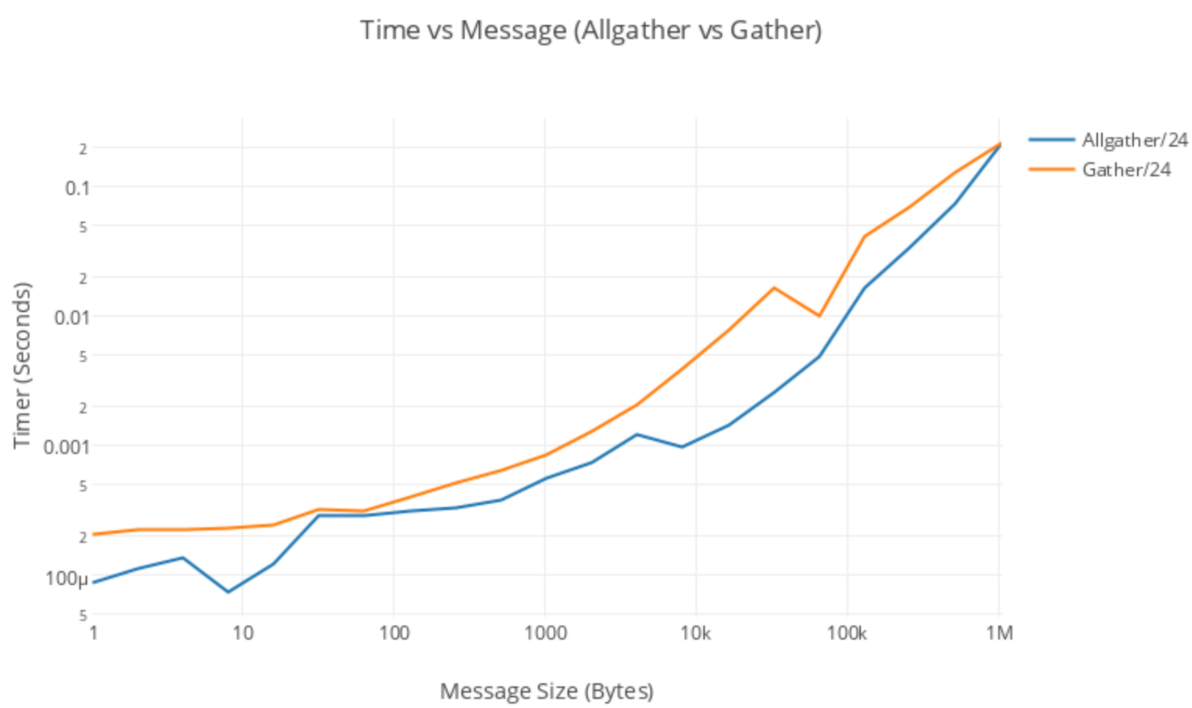
These graphs measure the time it takes for a collection of processes to gather a message, then broadcast it to everyone and finally add up the vector inside the message. The graphs indicate that the performance of Allgather is better than gather and broadcast regardless of message size. Another interesting feature of the graphs is that after a certain point it seems that the performance of both methods merge, effectively taking the same amount of time to execute.

Question 2-b


Time vs Message (Allgather vs Gather)
Allgather/4
Gather/4
Timer (Seconds)
Message Size (Bytes)
0.01
0.001
100μ
10μ
1μ
1
10
100
1000
10k
100k
1M

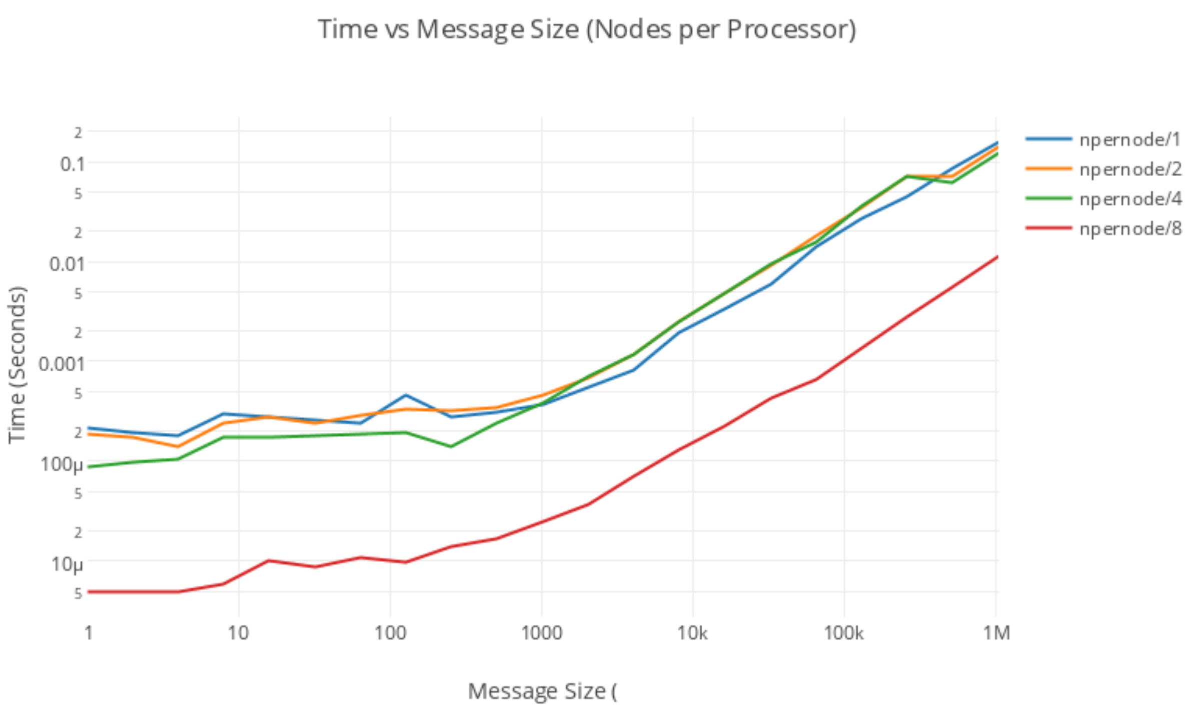

Time vs Message (Allgather vs Gather)
Allgather/8
Gather/8
Timer (Seconds)
Message Size (Bytes)
0.1
0.01
0.001
100μ
1
10
100
1000
10k
100k
1M


Time vs Message (Allgather vs Gather)
Timer (Seconds)
Message Size (Bytes)
Allgather/16
Gather/16
2
0.1
5
2
0.01
5
2
0.001
5
2
100μ
5
2
10μ
1
10
100
1000
10k
100k
1M


Time vs Message (Allgather vs Gather)
Timer (Seconds)
Message Size (Bytes)
Allgather/24
Gather/24
2
0.1
5
2
0.01
5
2
0.001
5
2
100μ
5
1
10
100
1000
10k
100k
1M

Similar to the graphs of question 2-a these graphs show the time it takes for a collection of processes to gather a message and then broadcast it to everything. The difference being that the vector inside of the message is not being added. The graphs show a similar performance to the previous question and exhibit the same traits. This can lead us to believe that the traits of the graphs are given by the gathering and broadcasting of dat and not by the summing of the vector.

Question 2-c



This graph measure the time it takes for a message to be broadcasted across all processes. This particular program was running eight processes with varying number of process per physical node. It is apparent that when at leas two physical nodes are being used the performance is limited by the network speed. However, as soon as only one physical node is being used the time needed drops considerably.