Team :

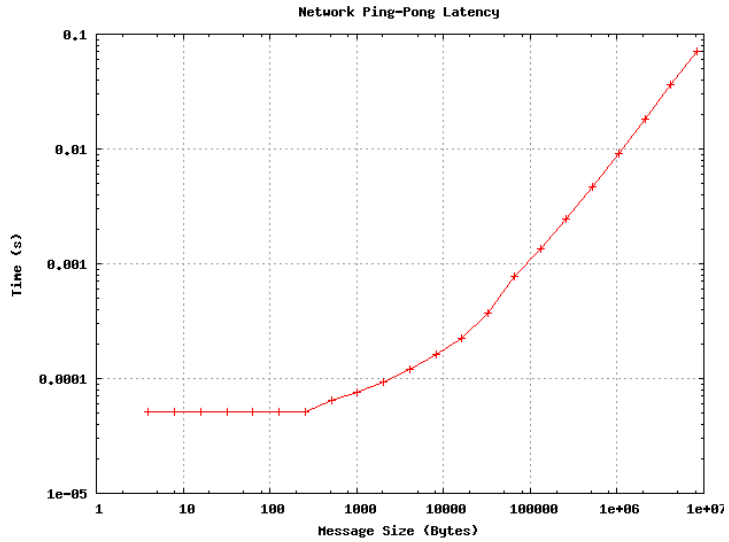
Sriram Vamsi Ilapakurthy(78605368)

Vivek Narayanamurthy (23123110)

Part1

• Take a look at netplot.png. What is the minimum time to send a message of any size?

Ans: Graph is below:



The minimum time is almost constant for the first few messages which is ~5.09884x 10^{-4} seconds.

• When the message size is large, estimate the slope of the line. What does the slope represent in this case?

Ans: For the last two data points the slope is

Slope (m) = 8.500 x 10^{-9}

The slope represents the per byte transmission cost due to the network bandwidth. This represented by β(1/bandwidth).

• Using your answers to the above questions, come up with a model (i.e., formula) that would allow someone to estimate T(m), the time to send a message of size m bytes.

T(m) = α which is a onetime cost to transmit the data in the network. It is due to the latency. This is the dominant term at lower message sizes.

T(m) = m β for higher message sizes. This is the dominant term at higher message sizes.

Overall the model can be described as

T(m) = $\alpha + m\beta$

• Take a look at the function pingpong() in pingpong.c; for your convenience, the code for this function appears below. Go line-by-line and do your best to explain what the code is doing. Make a note of anything you don't understand as well.

```
/* Pingpong test  */

void pingpong (int* msgbuf, const int len)

{

 /* Define at two tags for communication */

 const int MSG_PING = 1;

 const int MSG_PONG = 2;

 /*  Define a rank variable for coomunicating processes */

 int rank;

 /*  Determine the rank of the executing process */

 MPI_Comm_rank (MPI_COMM_WORLD, &rank);

 /* Add sepaarte code for different processes   */

 if (rank == 0) {

 /* If this a process with rank ==0 */

 /* Define a status variable  */

 MPI_Status stat;

 /* Send a message of type MPI_INT of length len with tag as MSG_PING to the MPI_COMM_WORLD group */

 MPI_Send (msgbuf, len, MPI_INT, 1, MSG_PING, MPI_COMM_WORLD);

 /* Recive a message to destination 1 of type MPI_INTof length len with tag as MSG_PONG from the MPI_COMM_WORLD group, captures the status in  stat */

 MPI_Recv (msgbuf, len, MPI_INT, 1, MSG_PONG, MPI_COMM_WORLD, &stat);

 } else {

 /* If this is a process with different rank  */

 /* Define a status variable  */

 MPI_Status stat;

  /* Recive a message of type MPI_INTof length len with tag as MSG_PING from the MPI_COMM_WORLD group, captures the status in  stat */

 MPI_Recv (msgbuf, len, MPI_INT, 0, MSG_PING, MPI_COMM_WORLD, &stat);

  /* Send a message to destination 0 of type MPI_INT of length len with tag as MSG_PONG to the MPI_COMM_WORLD group */
```

```
 MPI_Send (msgbuf, len, MPI_INT, 0, MSG_PONG, MPI_COMM_WORLD);
}
}
```

Part2:

Results:

Master slave for image size of 4096x4096

np	Tp	Ts	Speedup	Efficiency
2	14.08	12.2086	0.8670880682	0.4335440341
4	7	12.2086	1.744085714	0.4360214286
8	5.01	12.2086	2.436846307	0.3046057884
16	4.27	12.2086	2.859156909	0.1786973068
32	4.27	12.2086	2.859156909	0.0893486534
64	4.33	12.2086	2.819538106	0.04405528291

Intern Joe: 4096x 4096

np	Tp	Ts	Speedup	Efficiency
1	12.41	12.2086	0.9837711523	0.9837711523
2	6.96	12.2086	1.754109195	0.8770545977
4	6.6	12.2086	1.849787879	0.4624469697
8	4.72	12.2086	2.586567797	0.3233209746
16	3.52	12.2086	3.468352273	0.216772017
32	3.44	12.2086	3.549011628	0.1109066134
64	3.48	12.2086	3.508218391	0.05481591236

Intern Susie: 4096x 4096

np	Tp	Ts	Speedup	Efficiency
1	11.75	12.2086	1.039029787	1.039029787
2	6.68	12.2086	1.827634731	0.9138173653
4	4.17	12.2086	2.927721823	0.7319304556
8	2.95	12.2086	4.138508475	0.5173135593
16	3.58	12.2086	3.410223464	0.2131389665
32	3.37	12.2086	3.62272997	0.1132103116
64	3.61	12.2086	3.381883657	0.05284193213

Q: Compare the master/slave strategy with Susie/Joe's implementation. Which do you think will scale to very large image sizes? Why?

Ans: Even though Master-slave model requires more communication, it performs a better as all the slave processors will always be at work till the completion of work. In other methods, the total time is going to be the worst time performance of a node. Master-slave avoids the equal division of work and provides work only when the slave nodes requests for the work. Apart from the above reason, computations required for each input varies. In this method, the load balancing will be done better as it avoids the scenarios such as high computation inputs going to the same node, slowing that process and thus slowing the entire execution. Hence, this method scales better. The communication cost required can be optimized by minimizing the communication, either by experimentally calculating the optimal block size to allot to each processor at a time or by dynamically calculating this. One of the method for dynamic calculation is doubling the work for each process whenever a new request for work is method.

Q) Which do you think is better? Why? Which intern do you offer a full-time job?

Ans: Among Susie's and Joe's methods, Susie method performs better. Joe's method involves dividing the rows as the blocks. That will cause the nodes that get the earlier row chunks to complete their work faster. Hence these nodes are going to stay idle till other nodes complete the work allotting to them. Whereas Susie's method divides the input more equally, hence it is a better strategy. Not an easy call to say who gets a FT offer! Maybe Susie should get for her better strategy. :)