CS633 Assignment-1

Submission by: Aarchie(200004), Udit Prasad (201055)

Analyzing the performance of blocking P2P

We have used following files :

- send.c : Contains the MPI program for P2P send and receive.
- run.py : Contains script to run MPI program on diff nodes
- get_hosts.sh : Contains script to get 6 working nodes
- analysis.py : Contains code to plot graph.

Compilation and Execution command :

python3 run.py

CODE

code.c

```
#include <stdio.h>
#include <stdlib.h>
#include "mpi.h"

int main( int argc, char *argv[])
{
  int myrank, size;
  MPI_Status status;
  double sTime, eTime, time, maxTime;

  MPI_Init(&argc, &argv);

  int count = atoi (argv[1])/4; // Size of int is 4 bytes, so we make an array of size/4 elememts
  int buf[count];

  MPI_Comm_rank(MPI_COMM_WORLD, &myrank) ;
  MPI_Comm_size(MPI_COMM_WORLD, &size);

  for (int i=0; i<count; i++)
    buf[i] = myrank+i;
```

```
  sTime = MPI_Wtime();
  if (myrank == 0)
    MPI_Send (buf, count, MPI_INT, 1, 1, MPI_COMM_WORLD);
  else
    if (myrank == 1)
          MPI_Recv (buf, count, MPI_INT, 0, 1, MPI_COMM_WORLD, &status);
  eTime = MPI_Wtime();
  time = eTime - sTime;
  MPI_Reduce (&time, &maxTime, 1, MPI_DOUBLE, MPI_MAX, 0, MPI_COMM_WORLD);
  if (!myrank) printf ("%lf\n", maxTime);

  MPI_Finalize();
  return 0;
}
```

Script

run.py

```
import numpy as np
import os
from cmath import log, log10

os.system("chmod +x get_hosts.sh")
os.system("./get_hosts.sh")

hosts=[]
with open('hosts.txt') as my_file:
          hosts = my_file.readlines()

for i in range(len(hosts)):
          hosts[i] = hosts[i][:len(hosts[i] )-1]

data_size =
np.array([1024,102400,1048576,8388608,33554432,67108864,134217728])

os.system("mpicc -o hello code.c")

os.system("rm output.txt")
os.system("touch output.txt")
for i in range(len(data_size)):
   for j in range(0,len(hosts),2):
          for k in range(4):
```

```
        os.system("mpirun -np 2 -hosts {0},{1} ./hello {2} 2> error.txt >> output.txt".format(hosts[j],hosts[j+1],data_size[i]))

os.system("python3 analysis.py")
```

Experimentation Methodology

- **Export the MPI PATH**
  First, we need to export the MPI Path as Environment variable:
  For example,
  export PATH=/users/btech/aarchie/mpich-install/bin:$PATH

Script file(run.py) takes care of the following things.

- **Getting the working nodes**
  We have used *ping -c 1 &>/dev/null* command to get 6 nodes which are responding at the moment.
  What this command does is send a ECHO request to the host and then wait for the response, if it gets the response then that node is working otherwise not.

  We are then storing them into an list and using them as 3 pairs for our analysis.

- **Compiling the MPI Program**

- **Executing each data size on each pair of nodes 4 times .**
  We are executing each configuration 12 times (3 x 4) and directing the output (time) to a output file which we will use for analysis.

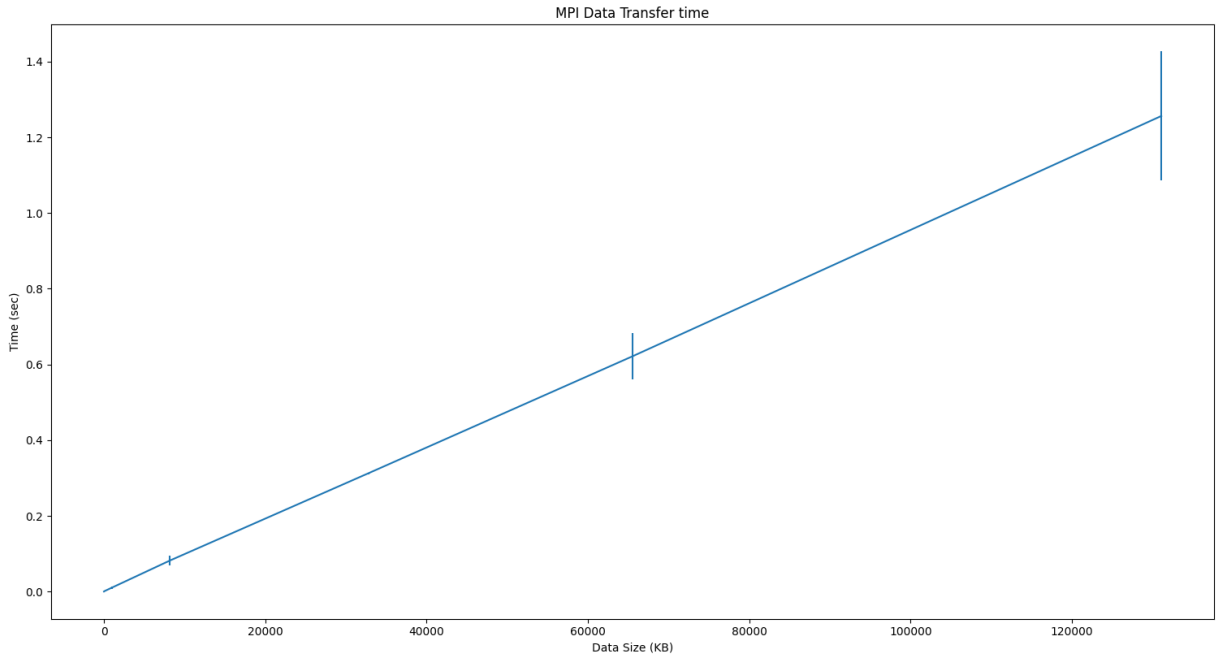
- **Executing the analysis file (analysis.py)**

Analysis file (analysis.py) will take care of the following things

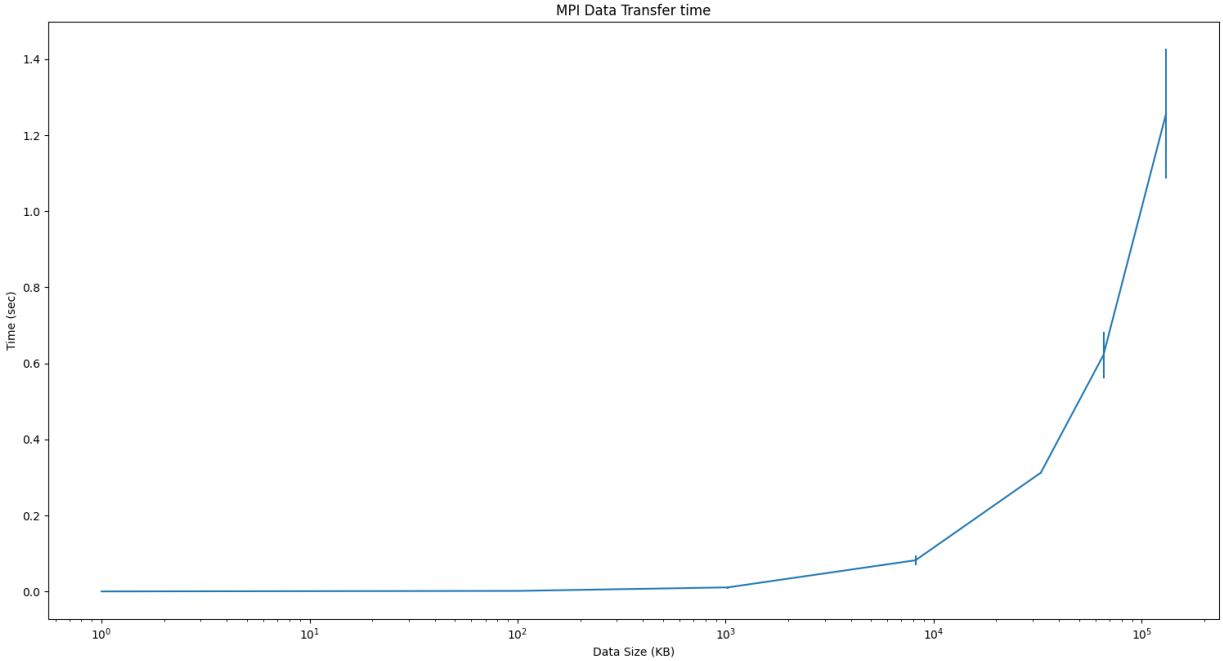
- **Reading the data(time taken)**

- **Plot the graph**
  We are using matplotlib.pyplot.errorbar to plot the graph with error bars
  Where error bar is the range (max of 12 times, min of 12 times) and data point is the average of 12 outputs at a given data size.

PLOTS



(a) Linear Scale



(b) Log Scale

OBSERVATION

- Time taken to transmit data is increasing with data size but not similarly for all data sizes.
  Here bandwidth comes into the play,
  Eff. Bandwidth = Data transmitted / time taken.

Data Size	Effective Bandwidth
1 KB	3 MB/s
100 KB	58 MB/s
1 MB	91 MB/s
8 MB	97 MB/s
32 MB	101 MB/s
64 MB	102 MB/s
128 MB	102 MB/s

  - ☐ From the above table, we can infer that for small data sizes, effective bandwidth is too low.
    This can be due to:
    1. Small data cannot utilize the available bandwidth to its fullest as explained in the class with the example of traffic on road.
  - ☐ When the data size is increasing, effective bandwidth is saturating which indicates that now the available bandwidth is being used to its fullest.

- From the log plot, we can see that when data is small then increase in data size does not increase the time as such but when data is large, then increase in data size increases the time in linear fashion.
  Possible reasons for this can be:
  1) Effective bandwidth
  2) Small data i.e. < Eager limit will not block the send call.
  3) For larger data, data will be fragmented and transmitted in fragments which will increase time due to increase in Startup time and Per hop time. (Hockney model)

- From the plots, we can observe that error bars are also increasing with the data size which is quite natural, but sometimes error bar can be larger for smaller data set due to an outlier which i saw in some of my observations.