INDIAN INSTITUTE OF TECHNOLOGY KANPUR

CS 633A Semester 2020–2021-II

Assignment II

Members
Hirak Mondal
Roll-20111022

Chandan Kumar
Roll-20111015

Professor:
Dr. Preeti Malakar
Department of Computer Science

CODE EXPLANATION

MPI_BCast Optimization

We have optimised this using Binomial tree method i.e In round one, Root will send its data to its $\frac{n}{2}^{th}$ neighbouring process. This will now divide logical processes into 2 subsets i.e root to $\frac{n}{2}$ - 1 and $\frac{n}{2}$ to n. Now In 2nd round root and the $\frac{n}{2}^{th}$ process will send its data to their $\frac{n}{2}^{th}$ neighbouring process in their respective partition. This will create 4 subsets. Again In the 3rd round the subsequent processes will send the data to their $\frac{n}{2}^{th}$ neighbouring process creating 8 subsets. This process goes on till all the processes gets the desired data.
We have used the code for receiving process :

```
while (mask < num_procs) {
    if (relative_rank & mask) {
      src = rank - mask;
      if (src < 0)
        src += num_procs;
      MPI_Recv(buff, count, MPI_DOUBLE , src, 99, MPI_COMM_WORLD,
   MPI_STATUS_IGNORE);
      break;
    }
    mask <<= 1;
  }
```

Listing 1: Getting exact source ranks for MPI_RECV().

Now for sending process the code is :

```
mask >>= 1;
  while (mask > 0) {
    if (relative_rank + mask < num_procs) {
      dst = rank + mask;
      if (dst >= num_procs)
        dst -= num_procs;
      MPI_Send(buff, count, MPI_DOUBLE , dst, 99, MPI_COMM_WORLD);
    }
    mask >>= 1;
  }
```

Listing 2: Getting exact destination rank for MPI_SEND()

MPI_Reduce Optimization

We have optimised this by splitting the MPI_COMM_WORLD into sub-communicators and after that we have performed reduce operation on them. In reduce we have performed the following MPI_SUM as the reduce operation.

```
    if(color==0)
  MPI_Reduce(sendval, maxval,count, MPI_DOUBLE, MPI_SUM, 0, newcomm
   ); // find sum
  else
  MPI_Reduce(sendval, maxval,count, MPI_DOUBLE, MPI_SUM, 0, newcomm
   ); // find sum
```

Listing 3: Using MPI_Reduce along with MPI_SUM on each sub-communicator

after that we have send the reduced value from one sub communicator to the other and calculated the total sum from it.

```
if(myrank==1)
  MPI_Send (maxval,count, MPI_DOUBLE, 0, 1, MPI_COMM_WORLD);
  if(myrank==0){
  MPI_Recv (maxval2,count, MPI_DOUBLE, 1, 1, MPI_COMM_WORLD, &
   status);
```

Listing 4: Using MPI_Send and MPI_Receive to send value from one sub-communicator to other

MPI_Gather Optimization

For this also we have used the concept of sub communicators, we have divided the MPI_COMM_WORLD into 2 sub communicators based on their colors. Now we have gathered the values on each of those sub communicators.

```
  if(color == 0)
   MPI_Gather(message, arrSize, MPI_DOUBLE, recvMessage, arrSize,
   MPI_DOUBLE, 0, newcomm);
  else
   MPI_Gather(message, arrSize, MPI_DOUBLE, recvMessage, arrSize,
   MPI_DOUBLE, 0, newcomm);
```

Listing 5: Performing Gather on Each sub-communicator

And after that we are sending it from one sub-communicator to the other and gathering at that point only.

```
 if(myrank == 1)
    MPI_Send (recvMessage,(arrSize*offset), MPI_DOUBLE, 0, 1,
   MPI_COMM_WORLD);

  if(myrank == 0)
  {
    MPI_Recv(recvMessage + (offset*arrSize),offset*arrSize,
  MPI_DOUBLE, 1, 1, MPI_COMM_WORLD, &status);

}
```

Listing 6: Combining the gathered values at a single sub-communicator

MPI_Alltoallv

We were unable to execute code MPI_Alltoallv, and get any speed up from it. So instead we have worked with MPI_Alltoall, as MPI_Alltoall is also a subset of MPI_Alltoallv . We know that it is a combination of MPI_Scatter() followed by MPI_Gather(), so we have tried using that. First we have used MPI_Scatter() followed by MPI_Gather()

```

// scatter to all processes
MPI_Scatter (message, 1, MPI_DOUBLE, newMessage, 1, MPI_DOUBLE,
  0, MPI_COMM_WORLD);


// gather at root
double *recvMessage = (double *) malloc(totalSize * sizeof(double
  )); //significant at the root process


//running MPI GATHER at all other processes.
for(int i=0;i<numtasks;i++)
MPI_Gather (message, arrSize, MPI_DOUBLE, recvMessage, arrSize,
  MPI_DOUBLE, i, MPI_COMM_WORLD);
```

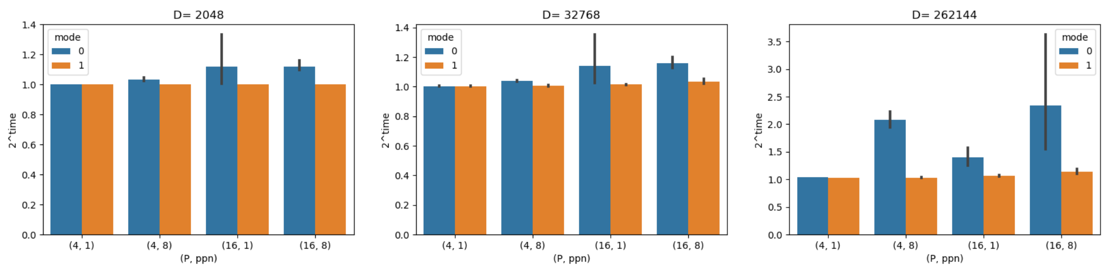
Listing 7: MPI_Scatter() and MPI_Gather() to implement MPI_Alltoall

But in this method we were not able to get speed up unlike the previous methods. We included it show our approach to the problem, which did not provided any speed up as such.

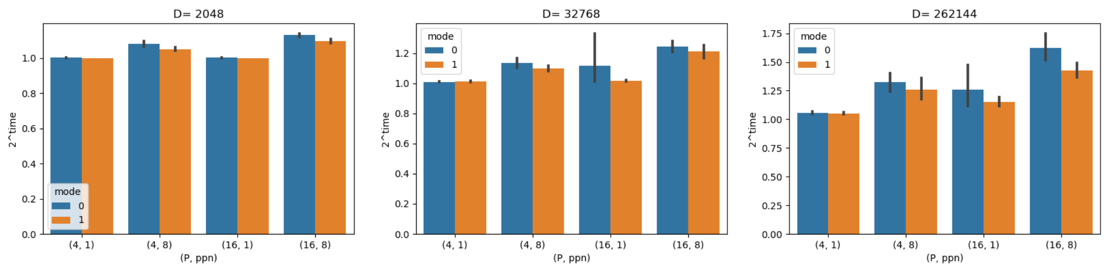
PLOTS

We are getting 240 data lines for each of the output file.The Y-axis in the graph is scaled to 2^{time} format (except MPI_Alltoallv, where time is taken as Y axis), as the time obtained were very less and the variation in such a minute scale was varying a lot. So we have raised it to the power 2, to plot the optimization graph more accurately.

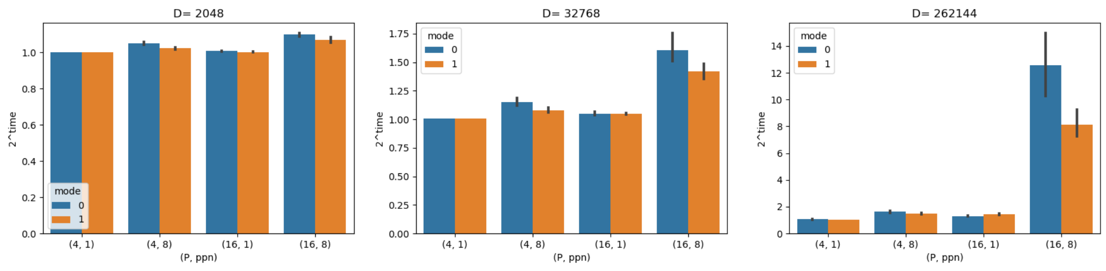
a.Plot for MPI_BCAST



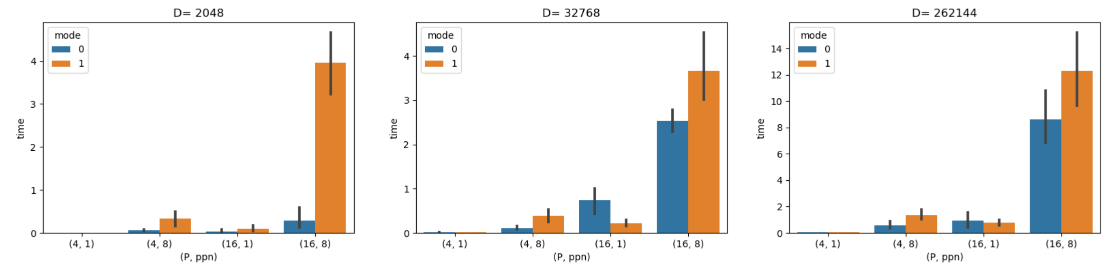
b.Plot for MPI_REDUCE



c.Plot for MPI_GATHER



b.Plot for MPI_Alltoall



Observations Regarding Performance From The Plots

From the plots we can see that, we are able to obtain speedup in the all the methods. So our idea of using Binomial tree for MPI_Bcast and sub-communicators for MPI_Reduce and MPI_Gather has indeed worked. Since sub communicators divide the entire work into 2 parts and rather than dealing with the whole problem data it deals with half of them for each communicator, so the job is performed more quickly, as a result of which we are able to get the speed up.