**Q-1** In this problem you will analyze the performance of distributed memory message communication (lecture 6). Given a 6\*6 processor array which are numbered as

(i,j) (i,j=1,2,3,4,5,6) for a processor in row i and column j, arranged as a **2-dimensional torus**, and assuming **cut-through routing**, *show the* *steps* in the following operations and obtain an estimate for the time taken s assuming the following parameters:

Startup time ts = 10 microseconds

Per hop time th =2 microseconds

Per byte time tw = 0.2 microseconds

1. **One-to-all broadcast** of 1000 bytes of data from processor (2, 2) to all the processors.

**Answer:**

Here, we need to do one-to-all broadcast for Cut-through routing. So, first we need to count number of steps for row operations. It would be 3 as its 6\*6 matrix and we are also counting the step from (2, 2).

The formula for One-to-all broadcast CT routing is,

3

Σ (Ts + mTw) +T hL

i=1

= (3\*10+1000\*0.2\*3) + (2\*5)

= (640)\*2

= **1280** This 2 is multiplied for column operation which is same as row operation for this CT routing.

1. **All-to-all scatter** of sections of 1000 bytes from each processor to every other processor (i.e.

A total of 6\*6\*1000 bytes of data exchanged overall)

**Answer:**

For all-to-all scatter the Cut-through routing does not provide any benefits over store-and-forward for multimode scatters on rings and meshes.

So the formula for all-to-all scatter Cut-through Routing is

√p/2

Σ (Ts + mTw) +T hL

i=1

= ((Ts\*√p/2) + (√p/2 \* Tw \* m) + ( Th \* L))

= ((Ts\*√p/2) + (√p/2 \* Tw \* (14)) + ( Th \* (6\*3))

= (10\*3) + (3\*0.2\*14\*1000) + (2\*6\*3)

= (30+8400 + 36)

= 2(8466)

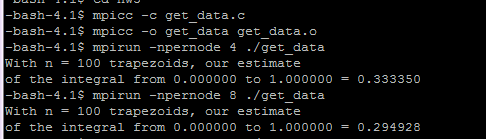
= 16932 This 2 is multiplied for column operation which is same as row operation for this CT routing.

**Q-2** In program get\_data.c, it is process0 who is going to collect data from other processes. **You are**

**Required to modify the program** to make the process with the largest rank to collect data and output the final result.

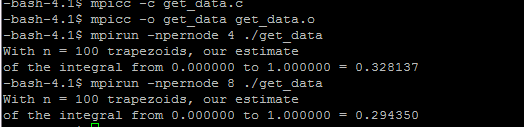
**Answer:**

When I run the program at Rank 0 I will get result as below .



|  |  |  |
| --- | --- | --- |
| Number of Process per node | 4 | 8 |
| Result | 0.333350 | 0.294928 |

When I change the processer rank to highest to collect data then I got below result.



|  |  |  |
| --- | --- | --- |
| Number of Process per node | 4 | 8 |
| Result | 0.328137 | 0.294350 |

There are also 3 bash file named run\_get\_data.basho17618 ,run\_get\_data.bashe17618

,run\_get\_data.bashpo17618 which are attached in the zip folder.

**Q-3 B**) Report running times for 1, 2, 4, 8, 12 and 16 processors. Evaluate the performance of the

Algorithm for a given matrix size (2000\*2000) on the cluster. USE THE MPI TIMERSFOR

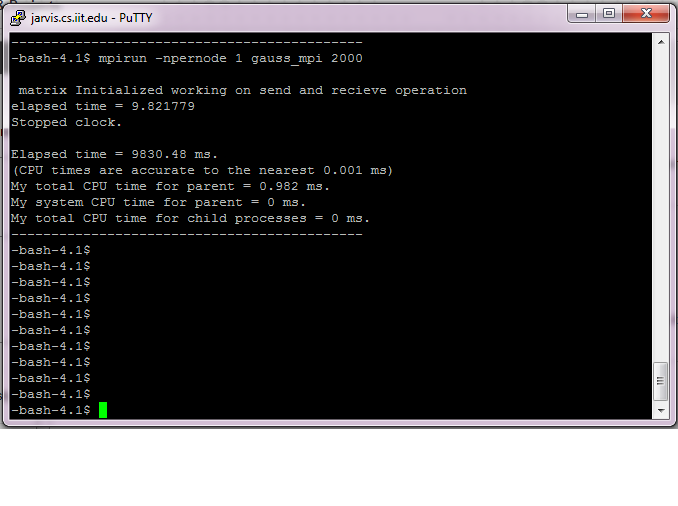
PORTABILITY, i.e. the MPI\_Wtime () calls even though this measures elapsed times and not CPU times.

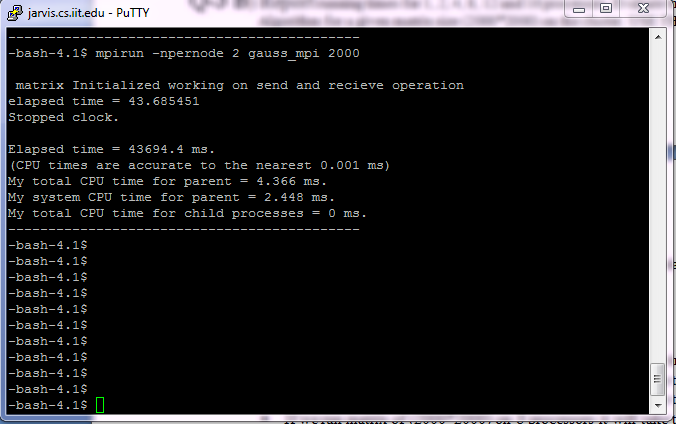
**Answer:**

* If we run matrix of (2000\*2000) on 1 processor it will take time of 9 seconds.
* If we run matrix of (2000\*2000) on 2 processors it will take time of 44 seconds.
* If we run matrix of (2000\*2000) on 4 processors it will take time of 56 seconds.
* If we run matrix of (2000\*2000) on 8 processors it will take time of 155 seconds.
* If we run matrix of (2000\*2000) on 12 processors it will take time of 104 seconds.
* If we run matrix of (2000\*2000) on 16 processors it will take time of 143 seconds.

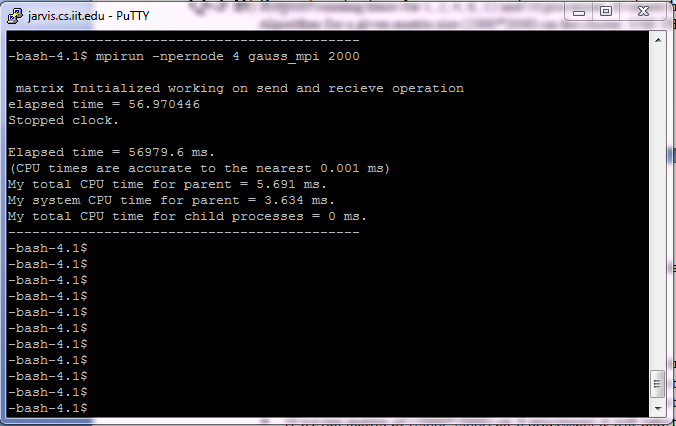
I attached some screenshot below for other timing I count and program execute sequence.

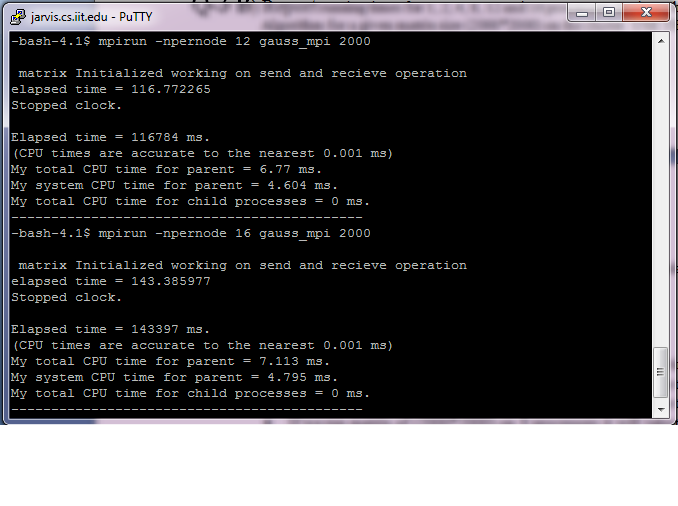
**For 1,**

**For 2,**

****

**For 4,**

****

**For 12 and 16, **