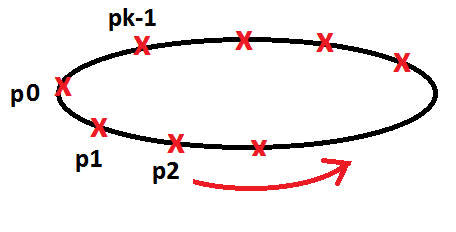
Trung Le 670-451-575

ECE 566 Parallel Processing

LAB 1 REPORT

1. RING OF K PROCESSORS METHOD

Part 1 of the lab is to implement ring of k processors topology to solve FIND\_SUM.



My method involves the master processor (p0) to create n-elements array with every element set ==1 and distribute it evenly across other k-1 processors. However, to distribute it evenly using a ring of processor, the process has to do it for k iterations. For each iteration i from 0 – (k-1), the processor with ID == I will distribute total of (n – neighbor\_id \* (n/number\_processors)) to its neighbor. Also locally calculate its portion of the array’s sum and send it to neighbor as well.

For example: Let’s say n=800 elements and p=2 and k=4 (Which means 8 total available processors).

For iter i=0, P0 sends (800 – 1\*(800/8)) = 700 elements of array to P1

P0 sends sum from index 0-99 to P1

For iter i=1, P1 sends (800 – 2\*(800/8)) = 600 elements of array to P2

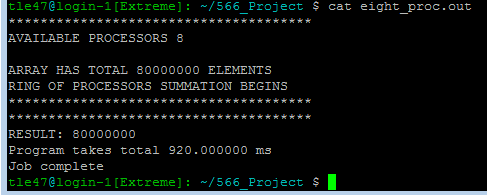
P1 sends sum from index 100-199 to P2.

Etc

This will repeat for total of 8 iterations to distribute evenly all the array elements and locally computed sum from each processor. At 8th Iteration, the master receives final sum of whole array and print out result, while calculate total time it takes to calculate the sum across all 8 processors.

**MPI Calls Used:** MPI\_SEND(), MPI\_RECV(), MPI\_Init(), MPI\_COMM\_WORLD,

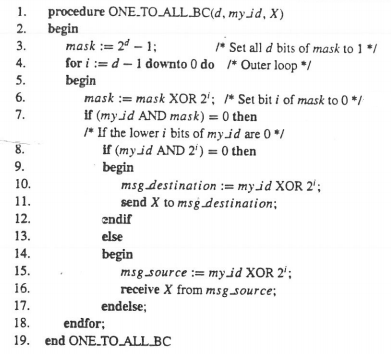
MPI\_Comm\_size(), MPI\_Comm\_rank()



1. HYPER CUBE OF 2d PROCESSORS METHOD

Part 2 of the lab requires implementing a hypercube topology to calculate FIND\_SUM for total of n-element array

For this part, I used the algorithm in the book to implement passing array elements throughout all k processors. The algorithm is described as follows:

(From Lecture slide 2)

Using this algorithm, the master processor 0 passes evenly distributed array elements throughout all k processors , with every elements in array set =1. However, the chunk size of each array elements being passed is different from part 1 of the lab. For each iteration i from d-1:0 , where d is the dimension of hypercube, the current active processors will send chunk size of n/pow(2,d-i).

The reason is that for the next iterations, the current active processors will be able to send substantial amount of array elements. Let’s say n=800 elements with d=3,p=2 and k=3 (total of 8 available processors).

For iter i=2, P0 sends (800/pow(2,1))=400 elements to P4

For iter i=1, P0 sends (800/pow(2,2))=200 elements to P2

P4 sends (800/pow(2,2))=200 elements to P6

For iter i=0, P0 sends (800/pow(2,3))=100 elements to P1

P2 sends (800/pow(2,3))=100 elements to P3

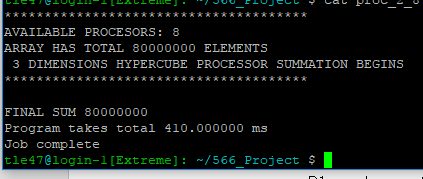
P4 sends (800/pow(2,3))=100 elements to P5

P6 sends (800/pow(2,3))=100 elements to P7

While sending the array elements, each processors calculate the sum locally and send it towards its neighbor until the master processor gets final sum.

**MPI Calls Used:** MPI\_SEND(), MPI\_RECV(), MPI\_Init(), MPI\_COMM\_WORLD,

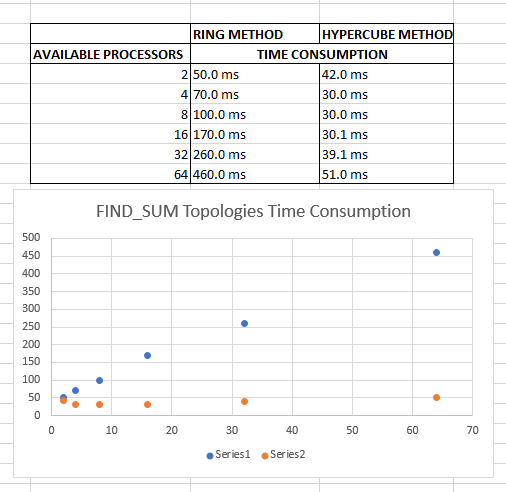
MPI\_Comm\_size(), MPI\_Comm\_rank()



3. EXPERIMENTS AND DATA RESULTS.

I ran the experiments with a lot of different unique input of available processors and the results are the followings:

For each experiments, n = 8,000,000 elements.



4. ANALYSIS

Since the ring of processors will always take same iterations as number of processors. For large amount of number of processors, it will take much longer time just to pass all the required parameters to next neighbors. Hence the ring method is not good in any of the situation where hypercube method is an alternative.

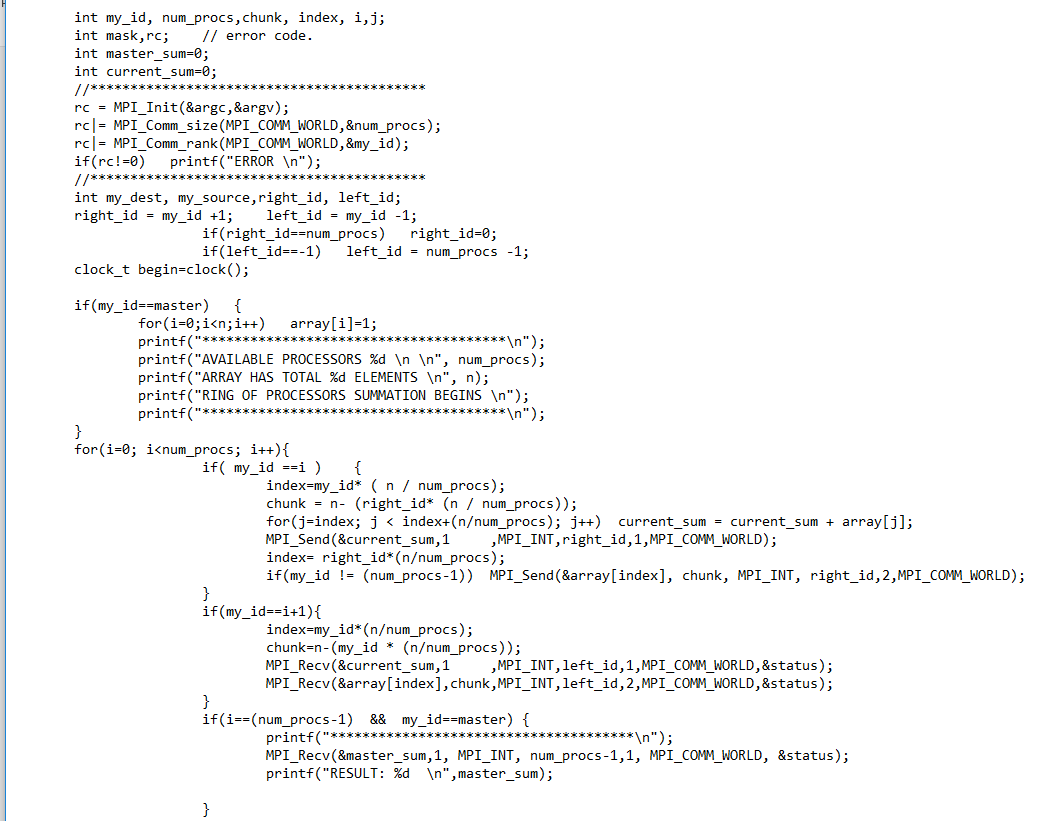
For the hypercube method, it will always perform as good (if not much better) than the alternative of ring k processors. In the algorithm i implemented, the hypercube only takes d-dimension iterations to completely pass all the required array elements and sum.

For the same number of processors, the hypercube will always outperform ring topology since each processor will handle same amount of array elements, however the passing parameters part will take much longer time in ring topology.

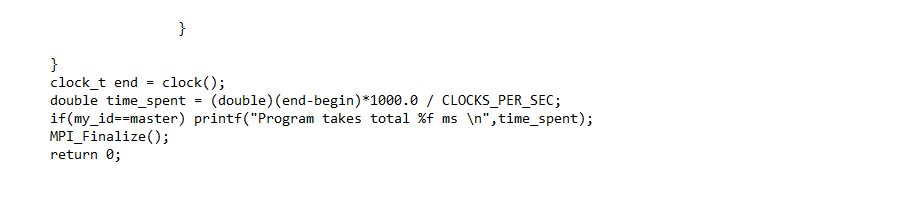
The breakpoint for computation time vs communication time for hypercube topology is at 32+ processors, as shown in the results at p=32 it will take 39.1ms and p=64 takes 51.0ms. There is a trend of increase in time consumption for larger number processors. Best way to handle large amount of data is to keep it under the breakpoint of 32processors

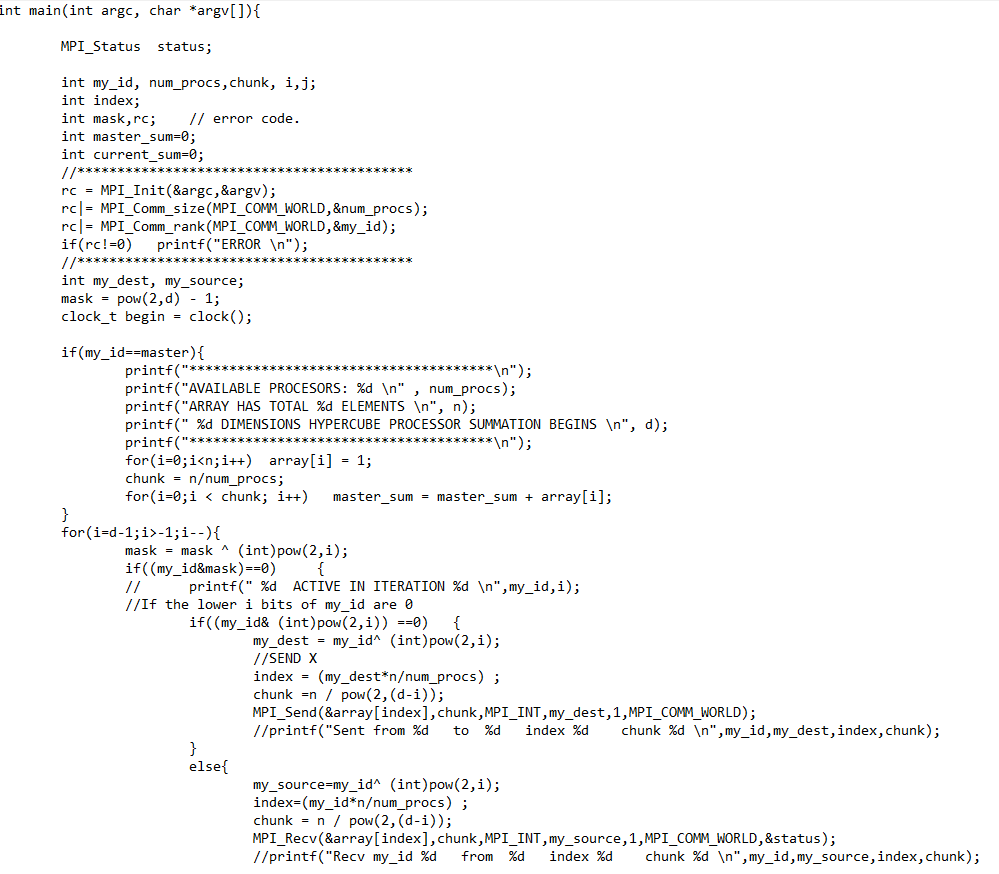
**5.** LESSONS

For this lab, I learned how to program in MPI standard with simple program of FIND\_SUM in C language. Also I get to use the supercomputer EXTREME at UIC which is eye-opening for me.

6. CODE

RING TOPOLOGY



 **HYPERCUBE TOLOPOGY**

