

Project 2: CS621, Fall 2007

Due Date: 12:15PM, November 8, 2007

1.) Prefix Sum (50 points) This exercise is to use one-to-all or all-to-all broadcast to write your own parallel program for computing the prefix sum. Even though you will be programming and running your program using MPI on a distributed memory machine, assume that the program will run on a hypercube interconnection network. The hypercube algorithm for prefix sum is described in your text book on page 168.

Let the dimension of the hypercube be a parameter in your code. It is not necessary to pass this parameter to the program on the command line, it is sufficient to define a constant in one spot in your code, representing the dimension of the target machine. This means that your code must be able to work with the correct number of processes.

In order to run your code, you will have to number your processes from 0 to $p - 1$, where p is the total number of nodes in the presumed hypercube. You can identify your processes in any way you like, an example of how to do this is given in Project 1. But once these numbers are chosen, assume that they form a standard numbering system for a hypercube, as shown in the text.

For output, let the dimension of your target machine be 4, giving a total of 16 nodes in your hypercube. For data, let each node use its identification number plus 5, so that node 0 uses the number 5, node 1 uses the number 6, etc. Print out, in ascending order of node numbers, the result of the prefix sum in decimal followed by the task (id) of the process. Each set of the data should be printed on a single line by itself so that 16 and only 16 lines of the data are printed out. You can format your data for readability but make sure that your data is printed out in order of node numbers, starting with node number 0 and ending with node number $p - 1$.

Note that IO can only be done with process 0. You should have a safeguard procedure against improper data, such as "file cannot be found".

2.) Matrix-Vector Product (50 points) You will be creating your programs to run efficiently on a 2D mesh architecture without the wraparound. You may assume that the target machine has p processors in every row and q processors in every column and that the dimensions of the data that you will work on is evenly divisible by q . Keep in mind that MPI provides special routines for mesh architectures and you are encouraged to use them.

Process 0 will read the data from a data file. This data must strictly adhere to the following format. Lines one and two consist of two integers m and n which represent the number of rows and columns of the matrix A . Line 3 and the next m lines contain single rows of the matrix A , with each number separated by at least one blank space. The next one line contains the elements of the vector b . For example, if the data below come from a file,

```
4
4
4.1 4.2 4.3 4.4
3.1 3.2 3.3 3.4
2.1 2.2 2.3 2.4
1.1 1.2 1.3 1.4
9.0 8.1 7.2 6.3
```

then the 4×4 matrix has 16 elements and the vector has 4 elements. Note that the vector is considered as a column vector.

Process 0 will output the result of the computation of $A * b$ with the output vector in one line. You are expected to check for errors in the input, any warnings or error messages should be printed to standard output.

Preprocessing: Assume for simplicity that you have $p \times q$ processors and the dimensions of the matrix A is $p \times q$. Since only process 0 does the input and output, the matrix should be distributed to the other processes. Each process should have one element of A . You also need to distribute the vector b to the first row of the mesh.

When you compute $A * b$, the first row of the processes need to broadcast their values to other processes in their individual columns. The results are summed to the first column of the mesh and then accumulated to process 0 for printing out.

For hand-in, assume the dimensions of the matrix A are 4×4 and the length of the vector b is 4.

Hand-In. Turn in the **hardcopy** of all your source code, and data file and outputs.

Please be reminded that you should work on your project independently. The minimum penalty for plagiarism is a fail grade for this course.