

ECE 493/693
Fall 2004
Assignment 3

Task: Write a parallel matrix/vector multiply

Due: Monday, November 3, 2004

Details:

Write a C/MPI program that reads a binary file that contains an n by m matrix and another binary file with an m by 1 vector, multiply them and write a binary file with an m by 1 binary vector. Each element of the vectors and matrix should be double precision floating point (double in C). The program should take the following arguments:

-n <int> the height of the matrix
-m <int> the width of the matrix and length of the vectors
-A <string> the name of the file with the matrix
-b <string> the name of the file with the vector to multiply
-c <string> the name of the file to write the result vector into

Your program must work for cases where m is not equal to n . For testing you will also need to write a program to either create a binary matrix and vector, or to convert an ASCII file into a binary file. You may also want to write a sequential matrix/vector multiply to use in verifying your results. Note that a parallel multiply WILL NOT generate identical results due to error in the floating point format.

Your program should attempt to use checkerboard (2D) partitioning. Your program should use the MPI facility `MPI_Dims_create` to select a 2D partitioning. If the call returns a 1D partitioning (returns a 1 for one of the values - as it will if `nnodes` is odd) then the program should correctly implement a 1D row-wise partitioning. You cannot guarantee that either n or m will correctly divide by the number of processors in a given dimension. However, if they do, and the partitioning is square you may utilize any optimizations available (as described in the book).

You must use MPI collective communication, communicators, groups, and topologies where appropriate. You should use `getopt` (man 3 getopt) to parse command line arguments.

Turn in:

- Printout of program code, commented so the grader can easily follow your design logic.
- A one page code summary that lists: identification (name, etc.), code purpose, degree of completeness of the code, summarized results of testing, any other relevant comments.
- Timing of executing the program on between 1 and 8 nodes on class cluster for vector sizes of 100, 500, 1000, 5000, each suitably graphed as a speedup chart (processors versus speedup) for presentation.