

CS 240A: Applied Parallel Computing // Homework 3

Assigned February 13, 2006

Due by class time Monday, February 27

The object of this assignment is to write a parallel subroutine to multiply a sparse matrix by a dense vector (yielding another dense vector), and then to use that “matvec” routine in a routine to solve a symmetric positive definite system of linear equations by the conjugate gradient method.

The matvec routine is partly an exercise in data structures for irregular, sparse objects. You’ll use a compressed row data structure that can store any sparse matrix in space proportional to the number of nonzero elements in the matrix. The data generator we supply will give you the matrix distributed over the processors by rows, with n/p rows of an n -by- n matrix on each of p processors. You may rearrange the matrix data if you wish in order to make the sequence of matvecs faster.

The conjugate gradient routine will follow the outline we discussed in class on February 8. There is a sequential Matlab code on the course web site. As usual, the Matlab code includes a harness with a data generator and validator, which you can use to check the correctness of your code. The data generator can produce an identity matrix, or the matrix whose graph is a regular k -by- k -by- k grid corresponding to the discretized Poisson equation in three dimensions. The Matlab routine `cgsolve.m` is the conjugate gradient solver. The sparse matvec is built into Matlab; it is just the line “`Ad = A*d;`”.

The data distribution and communication for this problem are up to you. You will probably want to think about not communicating more of the vector than necessary in the matvec. You may want to experiment with rearranging the rows and columns of the matrix to reduce the amount of communication; for fairly small matrices this is unlikely to help but it might be useful for the biggest matrices you can run. You should be able to do pretty large problems on a parallel machine; on my laptop, the sequential Matlab code takes about 2 minutes for a 1,000,000-by-1,000,000 matrix.

Those of you who did Homework 2 in MPI will do this homework in UPC or CAF, and vice versa. You should do this assignment on the Cray X1 at AHPARC. (If you want to do it on a different machine, talk to the instructor or the t.a. first.)

Grading on this problem will be 50% on correctness and 50% on performance. For performance, we will measure your code running on the sample data from the test harness, on the largest possible matrix sizes. The measure of performance will be a combination of raw speed (on a fixed number of processors and a fixed matrix size) and speedup (ratio of time on one processor to time on p processors).

See the file `hw3/faq.text` for details as needed.