Goethe-University
Frankfurt am Main

Lab Parallelization · Summer Semester 2018

Prof. Dr. Ulrich Meyer
Dipl. Inf. Gerhard Leuck

Assignment 3

Hand out: 29.05.2018
Hand in the tasks at ppva-tut@informatik.uni-frankfurt.de
Task 1 and Task 2: 14.06.2018

Task 1

Solving a linear system of equations with the Jacobi Method

In numerical linear algebra, the Jacobi method is an algorithm for determining the solutions of a diagonally dominant system of linear equations. We seek the solution to a set of linear equations, written in matrix terms as $Ax = b$ with $A = (a_{ij})$, b=(b_i), i=1...n, j=1...n and $|a_{ii}| > \sum_{j=1,j\neq i}^{n} |a_{ij}|$.

Let $A = D + (L + U)$ where D, L, and U represent the diagonal, lower triangular, and upper triangular parts of the coefficient matrix A. Then the equation can be rephrased as:

$$Ax = Dx + (L + U)\,x = b.$$

Moreover,

$$x = D^{-1}\,[b - (L + U)\,x]\,,$$

if $a_{i,i} \neq 0$ for all i. By iterative rule, the definition of the Jacobi method can be expressed as:

$$x^{(k+1)} = D^{-1}\left[b - (L + U)\,x^{(k)}\right]$$

where k is the iteration count. Often an element-based approach is used:

$$x_i^{(k+1)} = \frac{1}{a_{i,i}}\left(b_i - \sum_{j=1}^{i-1} a_{i,j}x_j^{(k)} - \sum_{j=i+1}^{n} a_{i,j}x_j^{(k)}\right), i = 1, 2, ..., n.$$

Write an MPI program that solves a set of linear equations $Ax = b$ with the Jacobi method that converges if the distance between the vectors $x^{(k)}$ and $x^{(k+1)}$ is small enough.

$$\sum_{i=1}^{n} \left|x_i^{(k+1)} - x_i^{(k)}\right| < \epsilon$$

You can assume that the number of rows of the matrix can be devided by the number of processes. Each process reads its part of the matrix A as a block by rows from a file using MPI_File_read_ordered

For reading the vector b use MPI_File_read. Use collective communication and global reduce operations. The file names and ϵ have to be specified by the user as parameters. Test if the condition diagonally dominant is fulfil. After calculation each process writes the part of the result vector to the same file using MPI_File_writed_ordered. You can find example files in:
/home/lab/2018/src/3
For updating the iteration vector x use a colletive communication operation.

Task 2

Modify the program from Task 1: The root process reads matrix A and the vector b from files.
Then the root scatters the parts of the matrix A as a block of columns and the parts of vector b to the other processes with colletive communication operations. In each iteration the process who knows $a_{i,i}$ needs to calculate x_i the partial results of all processes. After the final iteration calculation the root process gathers the parts of the results and writes the result vector to a file.