Logan D'Auria 10/18/2019

Gaussian Elimination Project Description

Approach

The approach used to solve the parallel gaussian elimination followed a striped mapping of threads. This method considers each row in the matrix as a task. Rows are assigned to threads, depending on how many available processors there are. Each thread will be responsible for n/p rows.

For example, with 4 processors and an 8x8 matrix, each thread would process 2 rows

4	5	9	2	4	5	9	2
1	6	0	2	1	6	0	2
2	4	7	5	2	4	7	5
6	8	7	9	6	8	7	9
4	5	9	2	4	5	9	2
1	6	0	2	1	6	0	2
2	4	7	5	2	4	7	5
6	8	7	9	6	8	7	9

The threads share memory access to the matrix and run concurrently to solve the gaussian elimination with higher efficiency.

Matrix Representation

$$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 3 & -1 & 0 & 0 \\ 5 & 2 & 7 & 0 \\ 4 & 0 & -2 & 8 \end{bmatrix} \quad \begin{bmatrix} 8 & 2 & 5 & -3 \\ 0 & 2 & 9 & 1 \\ 0 & 0 & -4 & 2 \\ 0 & 0 & 0 & 7 \end{bmatrix}$$

Instead of having a square matrix associated with another vector to represent the equation Ax = b, I combined the matrix and the vector to simplify the algorithm. That made the matrices N x N+1 which made pivoting easier, and in turn helped the parallelization of the algorithm. This worked especially well with the striped mapping technique since pivoting and elimination could be done by the same thread.

Solving the System

```
Procedure Back_Substitution (U, x, y) begin
        for k:= N down to 1 do begin
                x[k] := y[k]
                for i := k-1 down to 1 do begin
                        y[i] := y[i] - x[k]×U[i,k]
                endfor
        endfor
end
```

Once a triangular matrix is formed, the next step in the algorithm is back substitution to find the solution vector of the matrix. The back substitution starts in the last row and works its way up the triangular matrix to create a vector of solutions. This process was done serially as there is no easy way to parallelize back substitution. All of the parallelization was done in the elimination phase.

After the vector solution is found, it is rerun through the original matrix using a checkSolution( ) function to verify that the solutions are correct.

Design

The GaussianElimination class contains most of the utility and functionality to execute the program. It also contains the main function. Worker is a class that is used to make runnable objects that act as threads. GaussianElimination spawns these threads to work through the matrix concurrently while using its utility. The elimination logic resides in the run( ) function of the Worker class.