Introduction to Gaussian Elimination algorithm

Gaussian Elimination method is a numerical method for solving linear system $Ax = b$, where we assume that A is a square $n \times n$ matrix, x and b are both n dimensional vectors. In the process, the system of equations $Ax = b$ is reduced (by Gaussian elimination) to an upper triangular system $Ux = y$ to be solved through backward substitution.

In the numerical program, vector b is stored as $(n+1)-$th column of matrix A. Let loop k control the elimination step, loop i control $i-$th row accessing and loop j control $j-$th column accessing, the sequential Gaussian Elimination algorithm is described as follows:

Forward Elimination:

for $k = 1$ to $n-1$
  for $i = k+1$ to n
    $a_{ik} = a_{ik}/a_{kk}$;
    for $j = k+1$ to $n+1$
      $a_{ij} = a_{ij} - a_{ik} * a_{kj}$;
    endfor
  endfor
endfor

Note that since the entries in the lower triangular matrix vanish after elimination, their space is used to store multipliers $a_{ik} = a_{ik}/a_{kk}$.

Backward Substitution:

for $i = n$ to 1
  for $j = i+1$ to n
    $x_i = x_i - a_{ij} * x_j$;
  endfor
  $x_i = x_i/a_{ii}$;
endfor

Note that x_i is stored in the space of $a_{i,n+1}$.

We now describe a parallel Gaussian elimination algorithm.

In Forward elimination part, the following task (denoted by E_k^i) can be parallelized:

E_k^i:

$a_{ik} = a_{ik}/a_{kk}$;
for $j = k+1$ to $n+1$
  $a_{ij} = a_{ij} - a_{ik} * a_{kj}$;
endfor

The data access pattern of each task is: Read rows A_k, A_i, then write row A_i. Thus for each step k, tasks E_k^{k+1}, E_k^{k+2}, $\cdots E_k^n$ are independent, which leads to the following parallel algorithm:

Parallel Forward Elimination:

for $k = 1$ to $n-1$
  Do E_k^{k+1}, E_k^{k+2}, $\cdots E_k^n$ in parallel on p processors.
endfor

Group tasks into a set of clusters such that each task E_k^i with the same row i is in the same cluster C_i. Row i and cluster C_i will be mapped to the same processor. Cyclic mapping should be used to achieve a balanced load among processors: processor number = $\mathrm{mod}(k-1, p)$.

In Backward substitution part, denote the following task by S_i^j:
S_i^j:

for $j = i+1$ to n
  $x_i = x_i - a_{ij} * x_j$;
endfor
$x_i = x_i/a_{ii}$;

The dependence is: $S_n^j \to S_{n-1}^j \to \cdots \to S_1^j$.

Parallel Backward Substitution:

for $i = n$ to 1
  Execute S_i^j gradually on processor that owns j.
endfor