

Lecture 8: Fast Linear Solvers (Part 2)

Naive Parallel Backward Substitution Algorithm

After elimination, we obtain upper triangular $U\mathbf{x} = \mathbf{b}^{(n-1)}$.

Assume that U is stored by rows.

$$x_n = \frac{b_n^{(n-1)}}{u_{nn}}$$

for $i = n - 1$ **to** 1

$$x_i = \frac{b_i^{(n-1)} - u_{i,i+1}x_{i+1} - u_{i,i+2}x_{i+2} - \cdots - u_{i,n}x_n}{u_{ii}}$$

for $k = n$ **to** 1

$$x_k = b_k$$

for $i = k + 1$ **to** n

$$x_k = x_k - u_{ki}x_i$$

end;

$$x_k = x_k / u_{kk}$$

broadcast x_k to all rows

end;

Naive Parallel Forward Substitution Algorithm

Consider to solve lower triangular $Lx = b$.

for $i = 1$ **to** n

$$x_i = \frac{b_i - \sum_{j=1}^{i-1} l_{ij}x_j}{l_{ii}}$$

for $i = 1$ **to** n

for $j = 1$ **to** $i - 1$

$$b_i = b_i - l_{ij}x_j$$

end;

$$x_i = b_i/l_{ii}$$

 broadcast x_i to all rows

end;

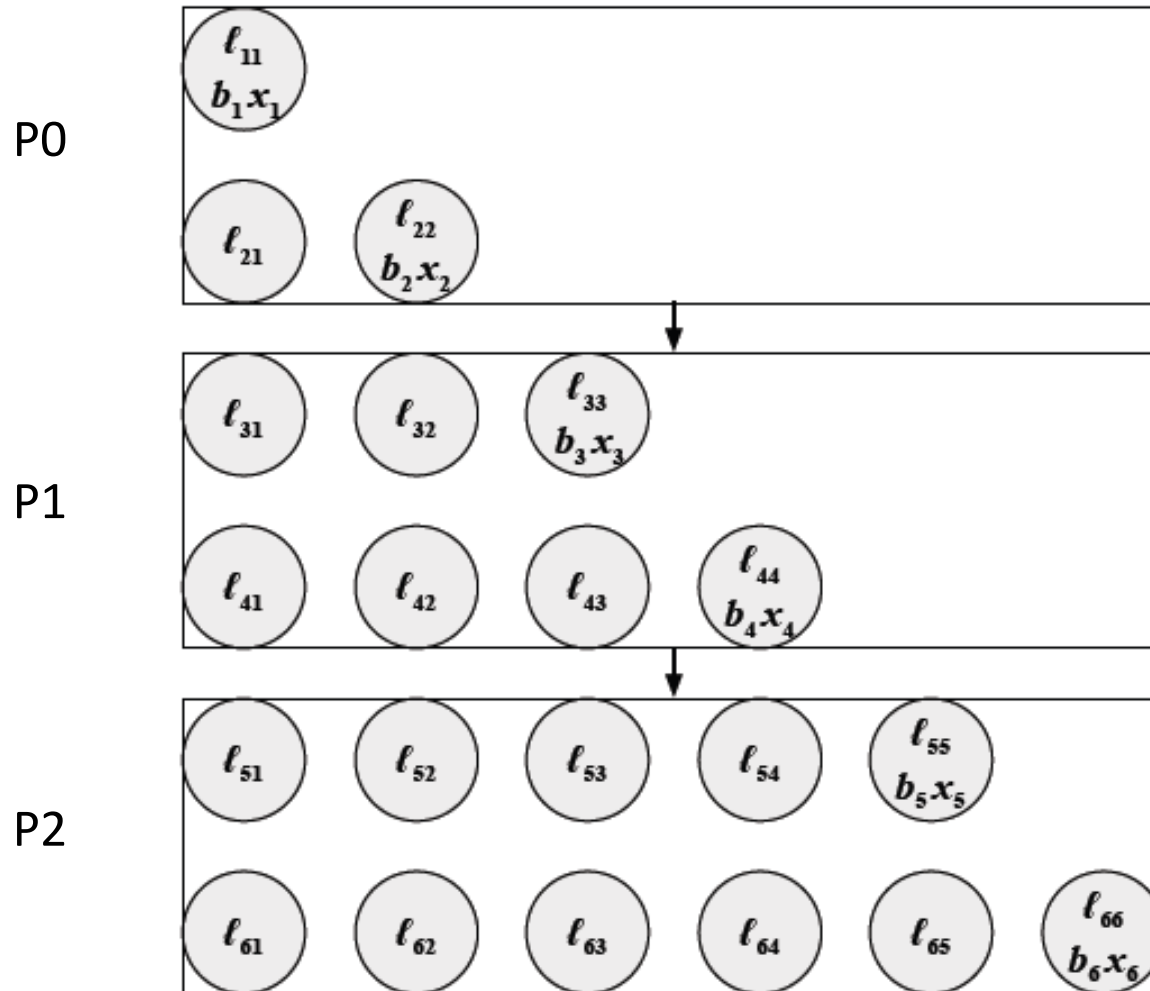
Revised Forward Substitution Algorithm

```
// immediate-update of right hand side
for  $j = 1$  to  $n$ 
     $x_j = b_j / l_{jj}$  // compute solution
    for  $i = j + 1$  to  $n$ 
         $b_i = b_i - l_{ij}x_j$  //update right hand side
    end;
end;
```

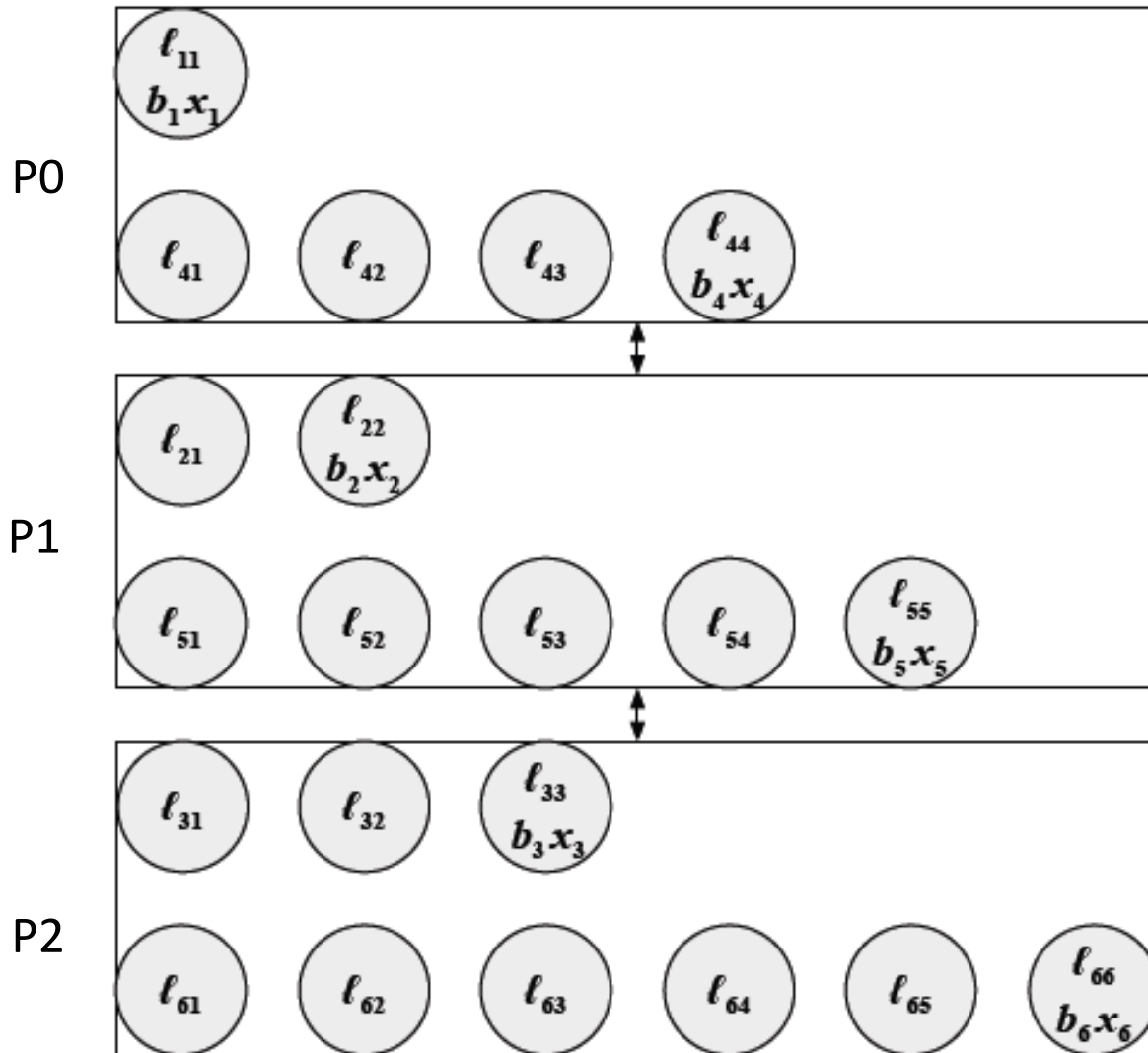
Parallel Forward Substitution Algorithm

- Assume a fine-grained decomposition in which process p_{ij} stores l_{ij} and compute $l_{ij}x_j$ for $i = 2, \dots, n, j = 1, \dots, i - 1$
- Assume p_{ii} stores l_{ii} and b_i , collects $\sum_{j=1}^{i-1} l_{ij}x_j$ and computes $x_i = \frac{b_i - \sum_{j=1}^{i-1} l_{ij}x_j}{l_{ii}}$ for $i = 1, \dots, n$

1D Row Block Mapping



1D Row Cyclic Mapping



Forward Substitution Parallel Algorithm Based on 1D Row Mapping

```
// immediate-update of right hand side
for  $j = 1$  to  $n$ 
  for process holding  $j$ th row
     $x_j = b_j / l_{jj}$  // compute solution
  end;
  broadcast  $x_j$  to all processes
  for process holding  $i$ th row  $i > j$ 
     $b_i = b_i - l_{ij}x_j$  //update right hand side
  end;
end;
```

References

- G. Li and T. F. Coleman. A new method for solving triangular systems on distributed-memory message-passing multiprocessors, *SIAM J. Sci. Stat. Comput.* 10:382-396, 1989
- E. E. Santos. On designing optimal parallel triangular solvers, *Information and Computation* 161:72-210, 2000

Gaussian Elimination and Sparse System

Consider to solve the tridiagonal system

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = F_i, \quad i = 1, \dots, n$$

for unknowns $x_1, \dots, x_i, \dots, x_n$ (So $x_0 = x_{n+1} = 0$).

Here a_i, b_i, c_i , and F_i are given.

- *Let m be the band width.*

```
for  $k = 1$  to  $n - 1$                                 // loop over columns
  for  $i = k + 1$  to  $\min(k + m, n)$ 
     $l_{ik} = a_{ik} / a_{ii}$                                 // multipliers for  $k$ th column
     $b_i = b_i - l_{ik} b_k$ 
  end;
  for  $j = k + 1$  to  $\min(k + m, n)$ 
    for  $i = k + 1$  to  $\min(k + m, n)$ 
       $a_{ij} = a_{ij} - l_{ik} a_{kj}$                         // elimination step
    end;
  end;
end;
```

Parallel Cyclic Reduction for Tridiagonal System

- When $m < p$, neither row-cyclic nor column-cyclic decomposition is efficient. Because only m processors are actively used.
- Assume that $n = 2^p - 1$, where p is the number of processors. If $n \neq 2^p - 1$, then add a trivial equation $x_i = 0, i = n + 1, \dots, 2^p - 1$.

- Consider the case when $n = 7 = 2^3 - 1$.
- Key idea:
 - Combine linearly equations to eliminate the odd-numbered unknowns x_1, x_3, x_5, \dots in the first stage.
 - Adding a multiple of $(i - 1)th$ equation and a multiple of $(i + 1)th$ equation to ith equation to eliminate x_{i-1} and x_{i+1} from the ith equation for $i = 2, 4, \dots$
 - Then renumber unknowns and repeat this process till there is a single equation with one unknown.
 - Solve backward to obtain the rest of the unknowns.

Remark: Cyclic Reduction is a divide-and-conquer method

- Multiply parameters $\alpha_2, \beta_2, \gamma_2$ to the first three equations respectively to get:

$$\alpha_2 b_1 x_1 + \alpha_2 c_1 x_2 = \alpha_2 F_1$$

$$\beta_2 a_2 x_1 + \beta_2 b_2 x_2 + \beta_2 c_2 x_3 = \beta_2 F_2$$

$$\gamma_2 a_3 x_2 + \gamma_2 b_3 x_3 + \gamma_2 c_3 x_4 = \gamma_2 F_3$$

To eliminate x_1 and x_3 , add above three equations and let

$$\beta_2 = 1$$

$$\alpha_2 b_1 + \beta_2 a_2 = 0$$

$$\beta_2 c_2 + \gamma_2 b_3 = 0$$

$$\Rightarrow \hat{b}_2 x_2 + \hat{c}_2 x_4 = \hat{F}_2$$

Where

$$\hat{b}_2 = \alpha_2 c_1 + \beta_2 b_2 + \gamma_2 a_3$$

$$\hat{c}_2 = \gamma_2 c_3$$

$$\hat{F}_2 = \alpha_2 F_1 + \beta_2 F_2 + \gamma_2 F_3$$

- Multiply parameters $\alpha_4, \beta_4, \gamma_4$ to the third, fourth and fifth equations respectively and add to eliminate x_3 and x_5 :

$$\Rightarrow \hat{a}_4 x_2 + \hat{b}_4 x_4 + \hat{c}_4 x_6 = \hat{F}_4$$

Where

$$\hat{a}_4 = \alpha_4 a_3$$

$$\hat{b}_4 = \alpha_4 c_3 + \beta_4 b_4 + \gamma_4 a_5$$

$$\hat{c}_4 = \gamma_4 c_5$$

$$\hat{F}_4 = \alpha_4 F_3 + \beta_4 F_4 + \gamma_4 F_5$$

$\alpha_4, \beta_4, \gamma_4$ are determined by :

$$\beta_4 = 1$$

$$\alpha_4 b_3 + \beta_4 a_4 = 0$$

$$\beta_4 c_4 + \gamma_4 b_5 = 0$$

- Finally, multiply parameters $\alpha_6, \beta_6, \gamma_6$ to the fifth, sixth and seventh equations respectively and add to eliminate x_5 and x_7 :

$$\Rightarrow \hat{a}_6 x_4 + \hat{b}_6 x_6 = \hat{F}_6$$

Where

$\alpha_6, \beta_6, \gamma_6$ are determined by :

$$\beta_6 = 1$$

$$\alpha_6 b_5 + \beta_6 a_6 = 0$$

$$\beta_6 c_6 + \gamma_6 b_7 = 0$$

- In stage two:

$$\hat{b}_2 x_2 + \hat{c}_2 x_4 = \hat{F}_2$$

$$\hat{a}_4 x_2 + \hat{b}_4 x_4 + \hat{c}_4 x_6 = \hat{F}_4$$

$$\hat{a}_6 x_4 + \hat{b}_6 x_6 = \hat{F}_6$$

- Repeat the same elimination process, which leads to only one equation

$$\alpha_4^* x_4 = F_4^*$$

- Use backward substitution, x_2 and x_6 are solved by $\hat{b}_2 x_2 + \hat{c}_2 x_4 = \hat{F}_2$ and $\hat{a}_6 x_4 + \hat{b}_6 x_6 = \hat{F}_6$
- Use the original equations to solve for x_1, x_3, x_5, x_7 .

Cyclic Reduction Algorithm

```
for(i=0; i < log2(size+1)-1;i++)    // levels of reduction
{
  for(j=2i+1 - 1; j < size; j=j+ 2i+1) // rows that are reduced
  {
    offset = 2i;
    index1 = j - offset; // index of row before the jth row
    index2 = j + offset; // index of row after the jth row
     $\alpha = A[j][\text{index1}]/A[\text{index1}][\text{index1}];$ 
     $\gamma = A[j][\text{index2}]/A[\text{index2}][\text{index2}];$ 

    for(k=0; k < size; k++)
       $A[j][k] -= \alpha A[\text{index1}][k] + \gamma A[\text{index2}][k];$  // do the reduction to have only
                                                                // jth row being active

     $F[j] -= \alpha F[\text{index1}] + \gamma F[\text{index2}];$ 
  }
}
```

Backward Substitution

```
int index = (size-1)/2;
x[index] = F[index]/A[index][index];

for(i=log2(size+1)+2;i>=0; i--)
{
    for(j=2i+1 - 1; j < size; j=j+ 2i+1)
    {
        offset = 2i;
        index1 = j - offset;
        index2 = j + offset;

        x[index1] = F[index1];
        x[index2] = F[index2];
        for(k=0; k < size; k++)
        {
            if(k != index1)
                x[index1] -= A[index1][k]*x[k];
            if(k != index2)
                x[index2] -= A[index2][k]*x[k];
        }
        x[index1] = x[index1]/A[index1][index1];
        x[index2] = x[index2]/A[index2][index2];
    }
}
```

Source of Parallelism

- Simultaneous reduction of equations in the system
- Simultaneous backward substitution to solve for the solution

Row Decomposition

For $i = 1, \dots, n$ of equations

Process i stores i th equation.

Computation

when $\text{modulus}(i, 2) = 0$, do row reduction to yield updated i th equation

Communication

i th equation receive $(i - 1)$ th and $(i + 1)$ th equations for $\text{modulus}(i, 2) = 0$

Reference

- B. Buzbbe, G. Golub, and C. Nielsen.
On direct methods for solving Poisson's equation. *SIAM J. Numer. Anal.*, 7:627-656, 1970
- J. Dongarra and S. Johnsson. Solving banded systems on a parallel processor, *Parallel Computing* 5:219-246, 1987
- M. Hegland. On the parallel solution of tridiagonal systems by wrap-around partitioning and incomplete LU factorization, *Numer. Math.* 59:453-472, 1991
- V. Mehrmann. Divide and conquer methods for block tridiagonal systems, *Parallel Computing* 19:257-280, 1993