Parallel Scientific Computing

- Matrix-vector multiplication.
- Matrix-matrix multiplication.
- Direct method for solving a linear equation. Gaussian Elimination.
- Iterative method for solving a linear equation. Jacobi, Gauss-Seidel.
- Sparse linear systems and differential equations.

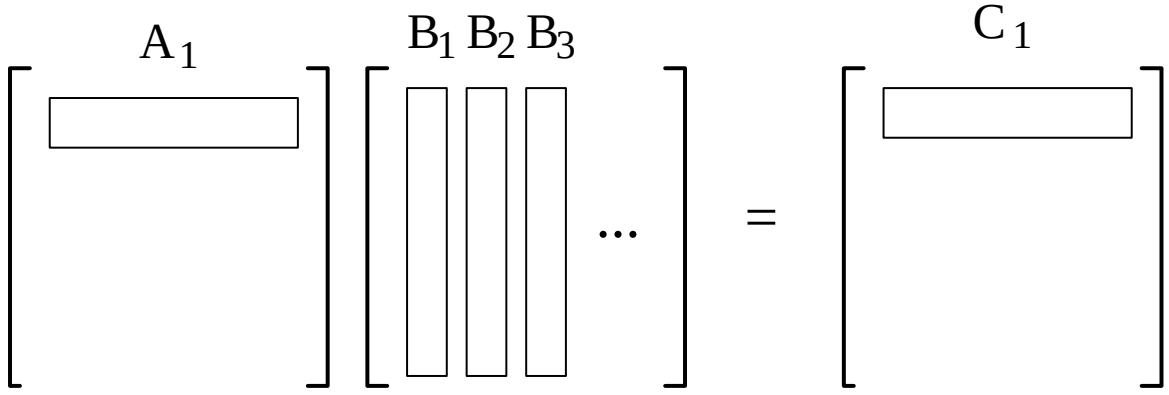
Matrix-Matrix Multiplication

Problem: $C = A * B$ where A and B are $n \times n$ matrices.

Sequential code:

```
for i = 1 to n do
    for j = 1 to n do
        sum = 0;
        for k = 1 to n do
              sum = sum + a[i,k] * b[k,j];
        c[i,j] = sum;
      endfor
   endfor
endfor
```

An example of $A * B$



$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} * \begin{pmatrix} 5 & 7 \\ 6 & 8 \end{pmatrix}$$

$$= \begin{pmatrix} 1*5+2*6 & 1*7+2*8 \\ 3*5+4*6 & 3*7+4*8 \end{pmatrix}$$

$$= \begin{pmatrix} 17 & 23 \\ 39 & 53 \end{pmatrix}$$

Task graph of $C = A * B$

Partitioned code:

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

T_i :

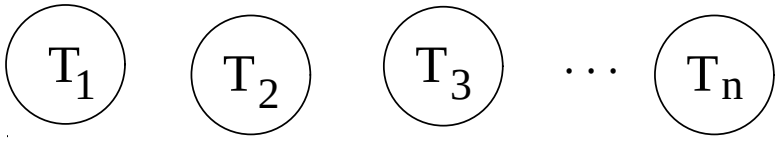
```
    for j = 1 to n do
        sum = 0;
        for k = 1 to n do
            sum = sum + a[i,k] * b[k,j];
        endfor
        c[i,j] = sum;
    endfor
```

endfor

T_i : Read row A_i and matrix B.

Write row C_i

Task graph:



Task and data mapping for $C = A * B$

SPMD code:

for $i = 1$ **to** n
    if proc_map(i)=me do T_i

Data mapping:

A is partitioned using rowwise block mapping

C is partitioned using rowwise block mapping

B is duplicated to all processors

Changes in T_i's code:

$$a_{ik} \longrightarrow a_{local(i)k}$$

$$c_{ij} \longrightarrow c_{local(i)j}$$

Parallel SPMD code of $C = A * B$

for $i = 1$ **to** n **do**
  if proc_map(i)=me do
    for $j = 1$ **to** n **do**
      $sum = 0$;
      for $k = 1$ to n **do**
        $sum = sum + a[local(i), k] * b[k, j]$;
      endfor
      $c[local(i), j] = sum$;
    endfor
  endif
endfor

Parallel algorithm with 1D partitioning

Partitioned code:

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

  for $j = 1$ **to** n **do**

  $T_{i,j}$:

```
sum = 0;
for k = 1 to n do
    sum = sum + a(i,k) * b(k,j);
Endfor
c(i,j) = sum;
```

  Endfor

Endfor

Data access: Each task $T_{i,j}$ reads row A_i and column B_j to write data element $c_{i,j}$.

Task graph: n^2 independent tasks:

$T_{1,1}$ $T_{1,2}$ $\cdots T_{1,n}$

$T_{2,1}$ $T_{2,2}$ $\cdots T_{2,n}$

$\cdots$

$T_{n,1}$ $T_{n,2}$ $\cdots T_{n,n}$

Mapping.

- Matrix A is partitioned using row-wise block mapping
- Matrix C is partitioned using row-wise block mapping
- Matrix B is partitioned using column-wise block mapping
- Task $T_{i,j}$ is mapped to the processor of row i in matrix A.

  Cluster 1: $\boxed{T_{1,1}\ T_{1,2}\ \cdots T_{1,n}}$

  Cluster 2: $\boxed{T_{2,1}\ T_{2,2}\ \cdots T_{2,n}}$

  $\cdots$

  Cluster n: $\boxed{T_{n,1}\ T_{n,2}\ \cdots T_{n,n}}$

Parallel algorithm:

For $j = 1$ **to** n

Broadcast column B_j to all processors

Do tasks $T_{1,j}, T_{2,j}, \cdots, T_{n,j}$ in parallel.

Endfor

Evaluation:

- Each multiplication or addition counts one time unit ω.
- Each task $T_{i,j}$ costs $2n\omega$.
- Assume that each broadcast costs $(\alpha + \beta n) \log p$.

$$PT = \sum_{j=1}^{n} ((\alpha + \beta n) \log p + \frac{n}{p} 2n\omega)$$

$$= n(\alpha + \beta n) \log p + \frac{2n^3\omega}{p}.$$

Gaussian Elimination

-Direct Method for Solving Linear System-

$$
\begin{aligned}
(1) \quad & 4x_1 - 9x_2 + 2x_3 = 2 \\
(2) \quad & 2x_1 - 4x_2 + 4x_3 = 3 \\
(3) \quad & -x_1 + 2x_2 + 2x_3 = 1
\end{aligned}
$$

$$
\begin{aligned}
(2)\text{-}(1)*\tfrac{2}{4} \quad & 0.5x_2 + 3x_3 = 2 \quad (4) \\
(3)\text{-}(1)*\text{-}\tfrac{1}{4} \quad & -\tfrac{1}{4}x_2 + \tfrac{5}{2}x_3 = \tfrac{3}{2} \quad (5) \\
(5)\text{-}(4)*\text{-}\tfrac{1}{2} \quad & 4x_3 = \tfrac{5}{2}
\end{aligned}
$$

$$
\begin{aligned}
4x_1 - 9x_2 + 2x_3 &= 2 \\
\tfrac{1}{2}x_2 + 3x_3 &= 2 \\
4x_3 &= \tfrac{5}{2}
\end{aligned}
$$

Backward substitution:

$$\begin{aligned} x_3 &= \frac{5}{8} \\ x_2 &= \frac{2-3x_3}{\frac{1}{2}} = \frac{1}{4} \\ x_1 &= \frac{2+9x_2-2x_3}{4} = \frac{3}{4} \end{aligned}$$

GE on Augmented Matrices

Use an augmented matrix to express elimination process for solving $Ax = b$.

Augmented matrix: $(A \mid b)$.

$$\begin{pmatrix} 4 & -9 & 2 & 2 \\ 2 & -4 & 4 & 3 \\ -1 & 2 & 2 & 1 \end{pmatrix} \overset{\substack{(2)=(2)-(1)*\frac{2}{4} \\ (3)=(3)-(1)*\frac{-1}{4}}}{\Longrightarrow} \begin{pmatrix} 4 & -9 & 2 & 2 \\ 0 & \frac{1}{2} & 3 & 2 \\ 0 & \frac{-1}{4} & \frac{5}{2} & \frac{3}{2} \end{pmatrix}$$

$$\longrightarrow \begin{pmatrix} 4 & -9 & 2 & 2 \\ 0 & 1/2 & 3 & 2 \\ 0 & 0 & 4 & 5/2 \end{pmatrix}$$

Column $n + 1$ of A stores column b!

Gaussian Elimination Algorithm

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

Loop k controls the elimination steps. Loop i controls i-th row accessing and loop j controls j-th column accessing.

Backward Substitution

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

For $i = n$ **to** 1

  For $j = i + 1$ **to** n

    $x_i = x_i - a_{i,j} * x_j$;

  Endfor

  $x_i = x_i / a_{i,i}$;

Endfor

Algorithm Complexity

Each division, multiplication, subtraction counts one time unit ω. Ignore loop overhead.

#Operations in forward elimination:

$$\sum_{k=1}^{n-1} \sum_{i=k+1}^{n} \left(1 + \sum_{j=k+1}^{n} 2 + 2\right) \omega$$

$$= \sum_{k=1}^{n-1} \sum_{i=k+1}^{n} (2(n-k)+3)\omega \approx 2\omega \sum_{k=1}^{n-1} (n-k)^2 \approx \frac{2n^3}{3}\omega$$

#Operations in backward substitution:

$$\sum_{k=1}^{n} (1 + \sum_{i=k+1}^{n} 2)\omega \approx 2\omega \sum_{k=1}^{n} (n-k) \approx n^2\omega$$

Total #Operations: $\approx \frac{2n^3}{3}\omega$.

Total space: $\approx n^2$ double-precision numbers.

Parallel Row-Oriented GE

For $k = 1$ **to** $n - 1$

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

T_k^i : $a_{ik} = a_{ik}/a_{kk}$

For $j = k + 1$ **to** $n + 1$

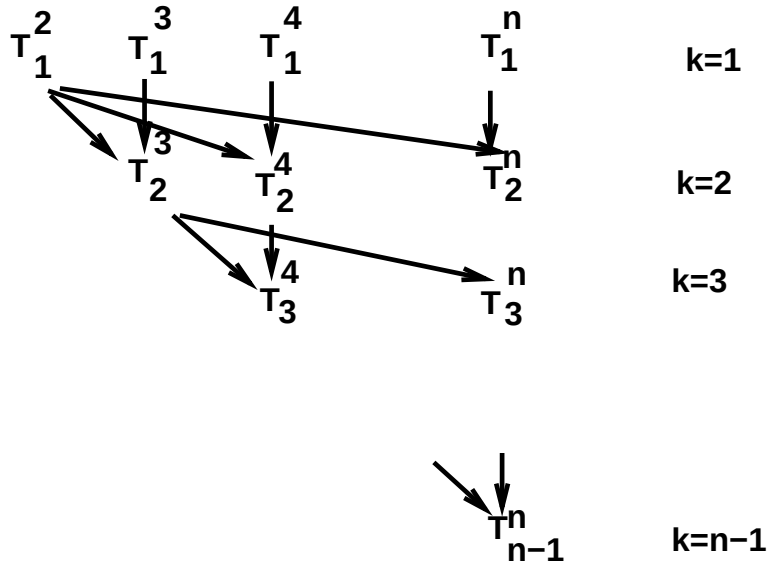
$a_{ij} = a_{ij} - a_{ik} * a_{kj}$

EndFor

T_k^i : Read rows A_k, A_i

Write row A_i

Dependence Graph



Parallelism and Scheduling

Parallelism:

Tasks T_k^{k+1} T_k^{k+2} $\ldots$ T_k^n are independent.

Parallel Algorithm(Basic idea)

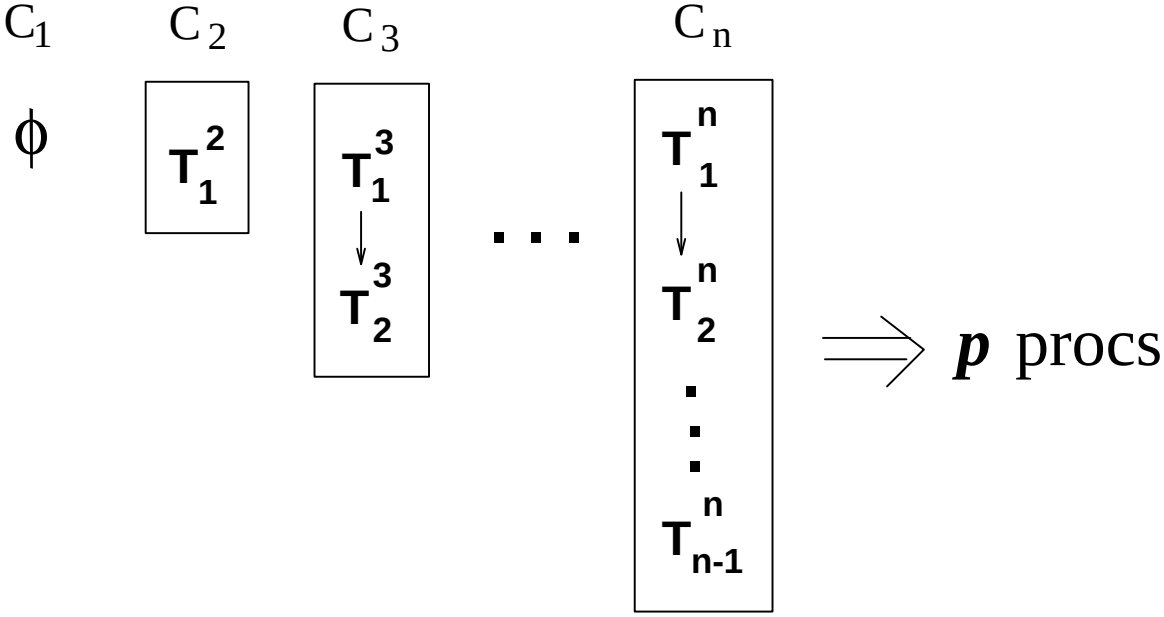
For $k = 1$ **to** $n - 1$

Do T_k^{k+1} T_k^{k+2} $\ldots$ T_k^n in parallel

on p processors.

Task Mapping

Define n clusters:



Cluster $T_1^2 \longrightarrow C_2$

Cluster $T_1^3 T_2^3 \longrightarrow C_3$

$\ldots$

Map n clusters to p processors:

$C_k \implies$ `proc_map(k)`

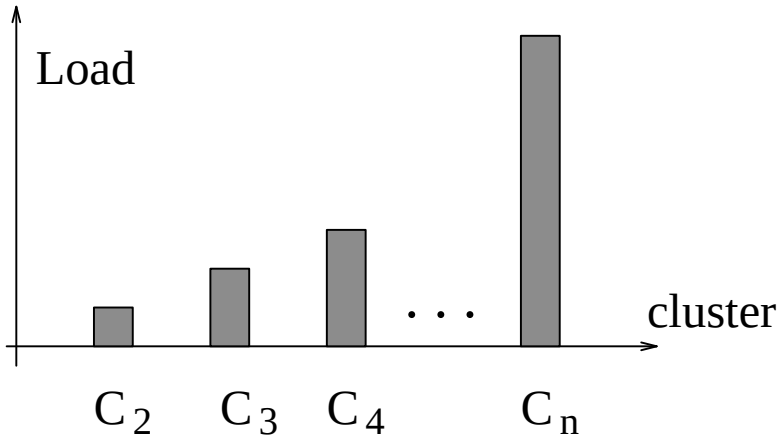
cyclic

block

Block vs. Cyclic Mapping

- **If block mapping is used**

  Profile the computation load of $C_2, C_3, \ldots, C_n$.



  Then

$$Load(P_0) \ll Load(P_1) \ll \ldots \ll Load(P_{n-1})$$

  Load is **NOT** balanced among processors!

- **If cyclic mapping is used.**

  Load is balanced.

Parallel Algorithm:

Proc 0 broadcasts Row 1

For $k = 1$ **to** $n - 1$

Do $T_k^{k+1} \ldots T_k^n$ in parallel

($T_k^i \rightarrow$ `proc_map(i)`).

Broadcast row $k + 1$.

endfor

SPMD Code:

me=mynode();

For $i = 1$ **to** n

if proc_map(i)==me, initialize Row i;

If proc_map(1)==me, broadcast Row 1 **else** receive it;

For $k = 1$ **to** $n - 1$

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

If proc_map(i)==me, do T_k^i

If proc_map(k+1)==me, then broadcast Row $k + 1$ **else** receive it.

Column-Oriented GE

Interchange loops i and j of the row-oriented GE.

For $k = 1$ **to** $n - 1$
  For $i = k + 1$ **to** n
    $a_{ik} = a_{ik}/a_{kk}$
  EndFor
  For $j = k + 1$ **to** $n + 1$
    For $i = k + 1$ **to** n
      $a_{ij} = a_{ij} - a_{ik} * a_{kj}$
    EndFor
  EndFor
EndFor

Impact on data accessing patterns

Example

$$\begin{pmatrix} 4 & -9 & 2 & 2 \\ 2 & -4 & 4 & 3 \\ -1 & 2 & 2 & 1 \end{pmatrix} \overset{\substack{(2)=(2)-(1)*\frac{2}{4} \\ (3)=(3)-(1)*\frac{-1}{4}}}{\Longrightarrow} \begin{pmatrix} 4 & -9 & 2 & 2 \\ 0 & \frac{1}{2} & 3 & 2 \\ 0 & \frac{-1}{4} & \frac{5}{2} & \frac{3}{2} \end{pmatrix}$$

Data access (writing) sequence for row-oriented GE:

$$\begin{pmatrix} \boxed{1} & \boxed{2} & \boxed{3} & \boxed{4} \\ \boxed{5} & \boxed{6} & \boxed{7} & \boxed{8} \end{pmatrix}$$

Data writing sequence for column-oriented GE:

$$\begin{pmatrix} \boxed{1} & \boxed{3} & \boxed{5} & \boxed{7} \\ \boxed{2} & \boxed{4} & \boxed{6} & \boxed{8} \end{pmatrix}$$

Column-oriented backward substitution.

Interchange loops i and j in the row-oriented backward substitution code.

For $j = n$ **to** 1

  $x_j = x_j / a_{j,j}$;

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

    $x_i = x_i - a_{i,j} x_j$;

  Endfor

EndFor

For example, given:

$$\begin{aligned} 4x_1 - 9x_2 + 2x_3 &= 2 \\ 0.5x_2 + 3x_3 &= 2 \\ 4x_3 &= \tfrac{5}{2}. \end{aligned}$$

The row-oriented algorithm performs:

$x_3 = \frac{5}{8}$

$x_2 = 2 - 3x_3$

$x_2 = \frac{x_2}{0.5}$

$x_1 = 2 + 9x_2$

$x_1 = x_1 - 2x_3$

$x_1 = \frac{x_1}{4}$.

The column-oriented algorithm performs:

$x_3 = \frac{5}{8}$

$x_2 = 2 - 3x_3$

$x_1 = 2 - 2x_3$

$x_2 = \frac{x_2}{0.5}$

$x_1 = x_1 + 9x_2$

$x_1 = \frac{x_1}{4}$.

Parallel Column-Oriented GE

Partitioned code:

For $k = 1$ **to** $n - 1$

T_k^k : **For** $i = k + 1$ **to** n

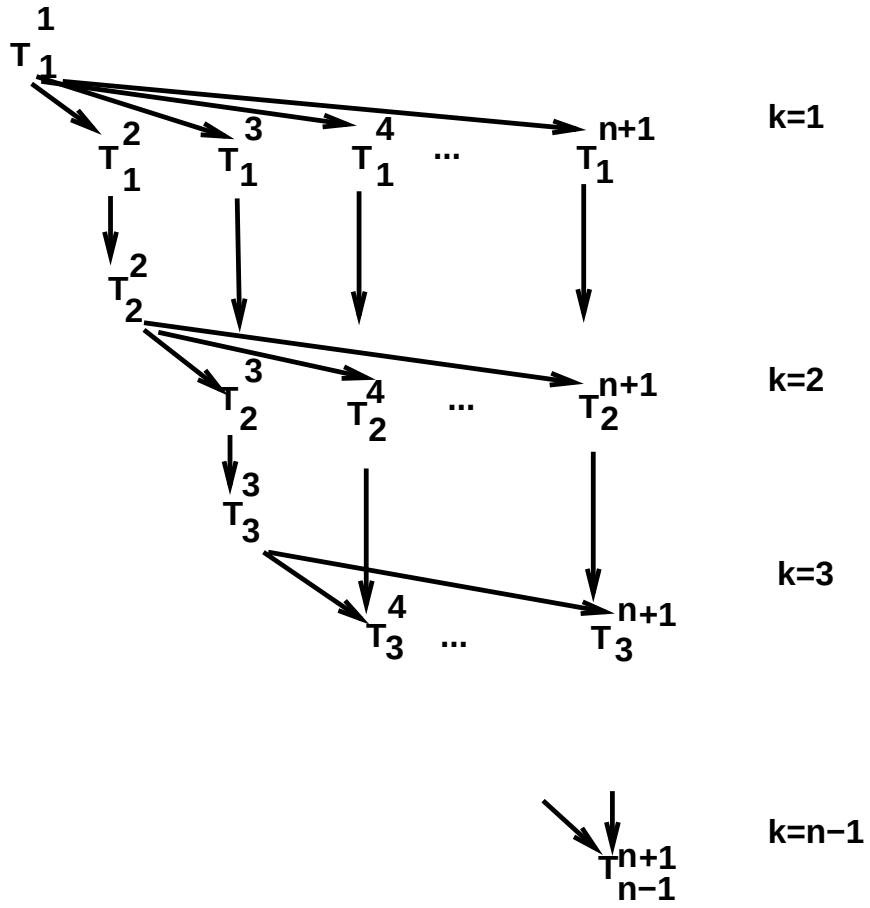
$a_{ik} = a_{ik}/a_{kk}$

For $j = k + 1$ **to** $n + 1$

T_k^j : **For** $i = k + 1$ **to** n

$a_{ij} = a_{ij} - a_{ik} * a_{kj}$

Task graph:



Schedule: ?

SPMD code: ?

Column-oriented backward substitution

Partitioning:

For $j = n$ **to** 1

S_j^x

$x_j = x_j / a_{j,j}$;
For $i = j - 1$ **to** 1
$x_i = x_i - a_{i,j} x_j$;
Endfor

EndFor

Dependence:

$S_n^x \longrightarrow S_{n-1}^x \longrightarrow \cdots \longrightarrow S_1^x$.

Parallel Algorithm:

Execute all these tasks ($S_j^x,\ j = n, \cdots, 1$) gradually on the processor that owns x (column $n+1$).

For $j = n$ **to** 1

  If owner(column x)==me then

    Receive column j if not available.

    Do S_j^x.

  Else If owner(column j)==me, send column j to the owner of column x.

EndFor

Problems with the GE Method

Problem 1: $a_{k,k} = 0$.

$$\begin{array}{lllll} (1) & 0 + x_2 + x_3 & = & 2 \\ (2) & 3x_1 + 2x_2 - 3x_3 & = & 2 \\ (3) & x_1 + 5x_2 - x_3 & = & 5 \end{array}$$

$$\begin{pmatrix} 0 & 1 & 1 \\ 3 & 2 & -3 \\ 1 & 5 & -1 \end{pmatrix} x = \begin{pmatrix} 2 \\ 2 \\ 5 \end{pmatrix}$$

Using Gaussian elimination:

$$Eq(2) - (1) * \frac{3}{0} \quad Eq(3) - (1) * \frac{1}{0}$$

Solution: At stage k, interchange rows such that $a_{k,k}$ is the maximum in the lower portion of the column k.

Gaussian Elimination with Pivoting

Row-oriented Forward Elimination

For $k = 1$ **to** $n - 1$

Find m such that

$|a_{m,k}| = \max_{i \geq k}\{|a_{i,k}|\}$;

If $a_{m,k} = 0$, No unique solution, stop;

Swap row(k) with row(m);

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

$a_{ik} = a_{ik}/a_{kk}$;

For $j = k + 1$ to n

$a_{ij} = a_{ij} - a_{ik} * a_{kj}$;

endfor

$b_i = b_i - a_{ik} * b_k$;

endfor

endfor

An example of GE with Pivoting

$$\left(\begin{array}{ccc|c} 0 & 1 & 1 & 2 \\ 3 & 2 & -3 & 2 \\ 1 & 5 & -1 & 5 \end{array}\right) \overset{(1)\leftrightarrow(2)}{\Longrightarrow} \left(\begin{array}{ccc|c} 3 & 2 & -3 & 2 \\ 0 & 1 & 1 & 2 \\ 1 & 5 & -1 & 5 \end{array}\right)$$

$$\overset{(3)-(1)*\frac{1}{3}}{\Longrightarrow} \left(\begin{array}{ccc|c} 3 & 2 & -3 & 2 \\ 0 & 1 & 1 & 2 \\ 0 & \frac{13}{3} & 0 & \frac{13}{3} \end{array}\right)$$

$$\overset{(2)\leftrightarrow(3)}{\Longrightarrow} \left(\begin{array}{ccc|c} 3 & 2 & -3 & 2 \\ 0 & \frac{13}{3} & 0 & \frac{13}{3} \\ 0 & 1 & 1 & 2 \end{array}\right)$$

$$\overset{(3)-(2)*\frac{3}{13}}{\Longrightarrow} \left(\begin{array}{ccc|c} 3 & 2 & -3 & 2 \\ 0 & \frac{13}{3} & 0 & \frac{13}{3} \\ 0 & 0 & 1 & 1 \end{array}\right) \qquad \begin{array}{l} x_1 = 1 \\ x_2 = 1 \\ x_3 = 1 \end{array}$$

Column-Oriented GE with Pivoting

For $k = 1$ **to** $n - 1$

Find m such that

$|a_{m,k}| = \max_{i \geq k}\{|a_{i,k}|\}$;

If $a_{m,k} = 0$, No unique solution, stop.

Swap row(k) with row(m);

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

$a_{ik} = a_{ik}/a_{kk}$

EndFor

For $j = k + 1$ **to** $n + 1$

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

$a_{ij} = a_{ij} - a_{ik} * a_{kj}$

EndFor

EndFor

EndFor

Parallel column-oriented GE with pivoting

Partitioned forward elimination:

For $k = 1$ **to** $n - 1$

P_k^k Find m such that $|a_{m,k}| = \max_{i \geq k}\{|a_{i,k}|\}$;
If $a_{m,k} = 0$, No unique solution, stop.

For $j = k$ **to** $n + 1$

S_k^j : Swap $a_{k,j}$ with $a_{m,j}$;

Endfor

T_k^k: **For** $i = k + 1$ **to** n

$a_{i,k} = a_{i,k}/a_{k,k}$

endfor

For $j = k+1$ **to** $n+1$

T_k^j

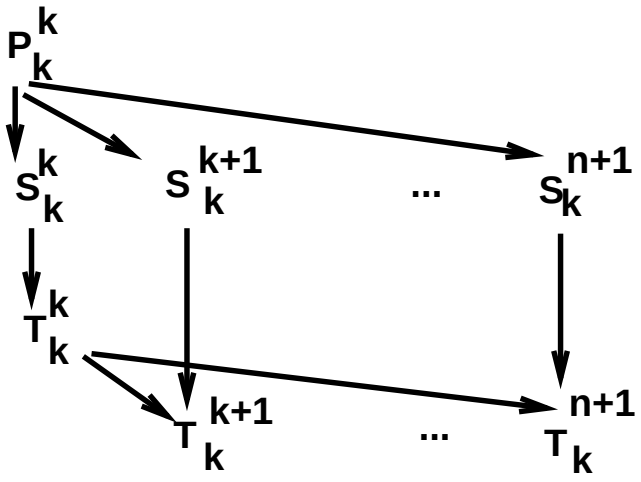
For $i = k+1$ **to** n

$a_{i,j} = a_{i,j} - a_{i,k} * a_{k,j}$

endfor

endfor

Dependence structure for iteration k



Find the maximum element.

Broadcast swapping positions

Swap each column

Scaling column k

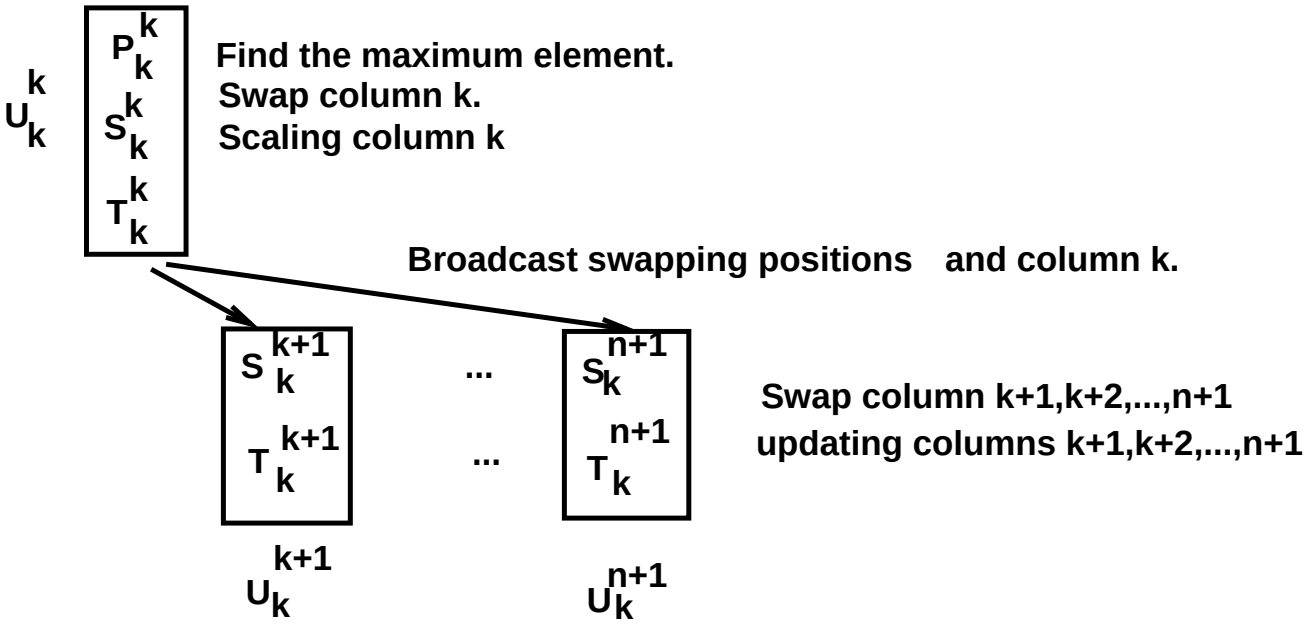
Broadcast column k

updating columns k+1,k+2,...,n+1

Combining messages and merging tasks

Define task U_k^k as performing P_k^k, S_k^k, and T_k^k.

Define task U_k^j as performing S_k^j, and T_k^j
$(k+1 \leq j \leq n+1)$.



Parallel algorithm for pivoting

For $k = 1$ **to** $n - 1$

The owner of column k does U_k^k and broadcasts the swapping positions and column k.

Do $U_k^{k+1} \ldots U_k^n$ in parallel

endfor

Iterative Methods for Solving Ax = b

Ex:

$$
\begin{array}{lrcl}
(1) & 6x_1 - 2x_2 + x_3 & = & 11 \\
(2) & -2x_1 + 7x_2 + 2x_3 & = & 5 \\
(3) & x_1 + 2x_2 - 5x_3 & = & \text{-1}
\end{array}
$$

$\Longrightarrow$

$$
\begin{array}{lcl}
x_1 & = & \frac{11}{6} - \frac{1}{6}(-2x_2 + x_3) \\
x_2 & = & \frac{5}{7} - \frac{1}{7}(-2x_1 + 2x_3) \\
x_3 & = & \frac{1}{5} - \frac{1}{-5}(x_1 + 2x_2)
\end{array}
$$

$\Longrightarrow$

$$
\begin{array}{lcl}
x_1^{(k+1)} & = & \frac{1}{6}(11 - (-2x_2^{(k)} + x_3^{(k)})) \\
x_2^{(k+1)} & = & \frac{1}{7}(5 - (-2x_1^{(k)} + 2x_3^{(k)})) \\
x_3^{(k+1)} & = & \frac{1}{-5}(-1 - (x_1^{(k)} + 2x_2^{(k)}))
\end{array}
$$

Initial Approximation: $x_1 = 0, x_2 = 0, x_3 = 0$

Iter	0	1	2	3	4	$\cdots$	8
x_1	0	1.833	2.038	2.085	2.004	$\cdots$	2.000
x_2	0	0.714	1.181	1.053	1.001	$\cdots$	1.000
x_3	0	0.2	0.852	1.080	1.038	$\cdots$	1.000

Stop when $\| \vec{x}^{(k+1)} - \vec{x}^{(k)} \| < 10^{-4}$

Need to define **norm** $\| \vec{x}^{(k+1)} - \vec{x}^{(k)} \|$.

Iterative methods in a matrix format

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}^{k+1} = \begin{bmatrix} 0 & \frac{2}{6} & -\frac{1}{6} \\ \frac{2}{7} & 0 & -\frac{2}{7} \\ \frac{1}{5} & \frac{2}{5} & 0 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}^{k} + \begin{pmatrix} \frac{11}{6} \\ \frac{5}{7} \\ \frac{1}{5} \end{pmatrix}$$

General iterative method:

Assign an initial value to $\vec{x}^{(0)}$

k=0

Do

$$\vec{x}^{(k+1)} = H * \vec{x}^{(k)} + d$$

until $\| \vec{x}^{(k+1)} - \vec{x}^{(k)} \| < \varepsilon$

Norm of a Vector

Given $x = (x_1, x_2, \cdots x_n)$:

$$\| x \|_1 = \sum_{i=1}^{n} | x_i |$$

$$\| x \|_2 = \sqrt{\sum | x_i |^2}$$

$$\| x \|_\infty = max | x_i |$$

Example:

$$x = (-1, 1, 2)$$

$$\| x \|_1 = 4$$

$$\| x \|_2 = \sqrt{1 + 1 + 2^2} = \sqrt{6}$$

$$\| x \|_\infty = 2$$

Applications:

$$\| Error \| \leq \varepsilon$$

Jacobi Method for Ax = b

$$x_i^{k+1} = \frac{1}{a_{ii}}(b_i - \sum_{j \neq i} a_{ij} x_j^k) \quad i = 1, \cdots n$$

Example:

$$\begin{array}{lrcl} (1) & 6x_1 - 2x_2 + x_3 & = & 11 \\ (2) & -2x_1 + 7x_2 + 2x_3 & = & 5 \\ (3) & x_1 + 2x_2 - 5x_3 & = & \text{-1} \end{array}$$

$\Longrightarrow$

$$\begin{array}{lcl} x_1 & = & \frac{11}{6} - \frac{1}{6}(-2x_2 + x_3) \\ x_2 & = & \frac{5}{7} - \frac{1}{7}(-2x_1 + 2x_3) \\ x_3 & = & \frac{1}{5} - \frac{1}{-5}(x_1 + 2x_2) \end{array}$$

Jacobi method in a matrix-vector form

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}^{k+1} = \begin{bmatrix} 0 & \frac{2}{6} & -\frac{1}{6} \\ \frac{2}{7} & 0 & -\frac{2}{7} \\ \frac{1}{5} & \frac{2}{5} & 0 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}^{k} + \begin{pmatrix} \frac{11}{6} \\ \frac{5}{7} \\ \frac{1}{5} \end{pmatrix}$$

Parallel Jacobi Method

$$x^{k+1} = D^{-1}Bx^k + D^{-1}b$$

or in general

$$x^{k+1} = Hx^k + d.$$

Parallel solution:

- Distribute rows of H to processors.
- Perform computation based on owner-computes rule.
- Perform all-all broadcasting after each iteration.

If the iterative matrix is sparse

If it contains a lot of zeros, the code design should take advantage of this:

- Not store too many known zeros.
- Code should explicitly skip those operations applied to zero elements.

Example: $y_0 = y_{n+1} = 0$.

$$y_0 - 2y_1 + y_2 = h^2$$

$$y_1 - 2y_2 + y_3 = h^2$$

$$\vdots$$

$$y_{n-1} - 2y_n + y_{n+1} = h^2$$

This set of equations can be rewritten as:

$$\begin{pmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & & \ddots & 1 & \\ & & & 1 & -2 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{pmatrix} = \begin{pmatrix} h^2 \\ h^2 \\ \vdots \\ h^2 \\ h^2 \end{pmatrix}$$

The Jacobi method in a matrix format (right side):

$$0.5* \begin{pmatrix} 0 & 1 & & & & \\ 1 & 0 & 1 & & & \\ & 1 & 0 & 1 & & \\ & & & \ddots & 1 & \\ & & & 1 & 0 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{pmatrix}^k -0.5* \begin{pmatrix} h^2 \\ h^2 \\ \vdots \\ h^2 \\ h^2 \end{pmatrix}$$

Too time and space consuming if you multiply using the entire iterative matrix!

Correct solution: write the Jacobi method as:

Repeat

For i= 1 to n

$y_i^{new} = 0.5(y_{i-1}^{old} + y_{i+1}^{old} - h^2)$

Endfor

Until $\| \vec{y}^{new} - \vec{y}^{old} \| < \varepsilon$

Gauss-Seidel Method

Utilize new solutions as soon as they are available.

$$
\begin{array}{lrcl}
(1) & 6x_1 - 2x_2 + x_3 & = & 11 \\
(2) & -2x_1 + 7x_2 + 2x_3 & = & 5 \\
(3) & x_1 + 2x_2 - 5x_3 & = & \text{-1}
\end{array}
$$

$\Longrightarrow$ Jacobi method.

$$
\begin{aligned}
x_1^{k+1} &= \tfrac{1}{6}(11 - (-2x_2^k + x_3^k)) \\
x_2^{k+1} &= \tfrac{1}{7}(5 - (-2x_1^k + 2x_3^k)) \\
x_3^{k+1} &= \tfrac{1}{-5}(-1 - (x_1^k + 2x_2^k))
\end{aligned}
$$

$\Longrightarrow$ Gauss-Seidel method.

$$
\begin{aligned}
x_1^{k+1} &= \tfrac{1}{6}(11 - (-2x_2^k + x_3^k)) \\
x_2^{k+1} &= \tfrac{1}{7}(5 - (-2x_1^{k+1} + 2x_3^k)) \\
x_3^{k+1} &= \tfrac{1}{-5}(-1 - (x_1^{k+1} + 2x_2^{k+1}))
\end{aligned}
$$

$$\varepsilon = 10^{-4}$$

	0	1	2	3	4	5
x_1	0	1.833	2.069	1.998	1.999	2.000
x_2	0	1.238	1.002	0.995	1.000	1.000
x_3	0	1.062	1.015	0.998	1.000	1.000

It converges faster than Jacobi's method.