# *Plan*

- Domain decomposition applied to matrix operations
- Gaussian elimination

Domain Decomposition:

- the data associated with the problem is decomposed, tasks work on portions of the data
- common for matrix like operations
  - assign rows, columns or submatrices of a matrix to different PEs

## *PRAM matrix-matrix multiply*

$$C = A \cdot B \Rightarrow c(i,j) \leftarrow \sum_{k=1}^{n} a(i,k)b(k,j)$$

For a PRAM with "infinite" number of PEs:

- cost of a single "dot" product - $\log_2 n$
- total cost - $\log_2 n$.

For a PRAM with $P$ PEs:

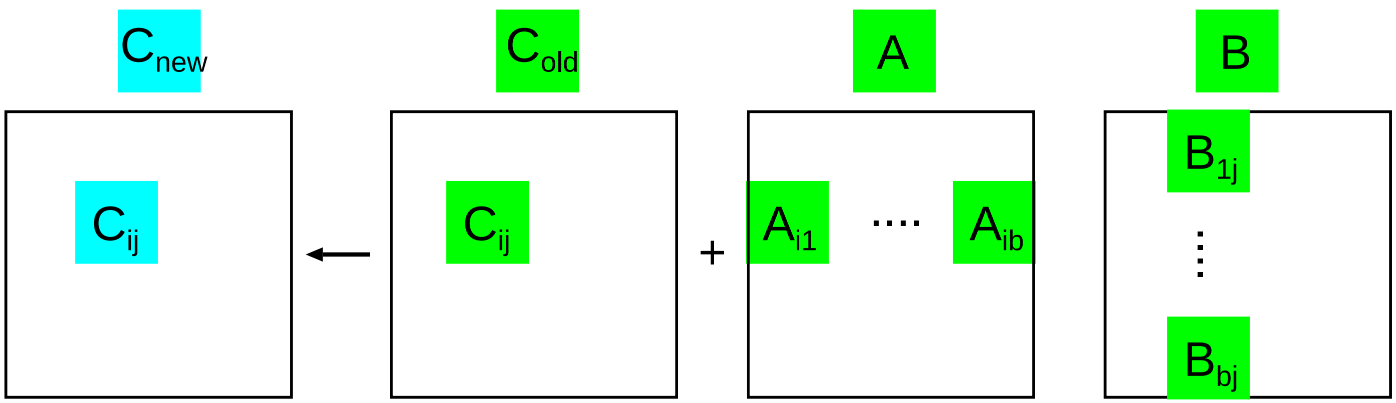
- each PE computes $\frac{n^2}{P}$ elements $c(i,j)$
- total cost - $2\frac{n^3}{P}$

## *Block matrix-matrix multiply*

Shared memory with $P$ PEs, each with individual cache.

Split $A, B, C$ into blocks of size $b \times b$ (thus $n = b \cdot p$).

$$C(i,j) \leftarrow C(i,j) + \sum_{k=1}^{p} A(i,k)B(k,j)$$



# *Block matrix-matrix multiply*

Assume $P = p^2$ $C(i,j) \leftarrow C(i,j) + \sum_{k=1}^{p} A(i,k)B(k,j)$

for $1 \leq i, j \leq p$

  load block $C(i,j)$ into fast memory

  for $k = 1 : p$

    load block $A(i,k)$ into fast memory

    load block $B(k,j)$ into fast memory

    $C(i,j) \leftarrow C(i,j) + A(i,k)B(k,j)$

  store $C(i,j)$ into slow memory

Note that for all $(i,j)$

- $A(i,k)$ is needed by all $C(i,u)$, $u = 1, .., p$
- $B(k,j)$ is needed by all $C(v,j)$, $v = 1, .., p$

## *All node shortest path*

- Bad: Simultaneous reads may create congestion on the interconnect.
- Good: only $(i, j)$ modifies $C(i, j)$ so no cache coherence is required.
- Good: no need for synchronization.
- Cost (time) will depend on the bandwidth of the interconnect between shared memory and caches.

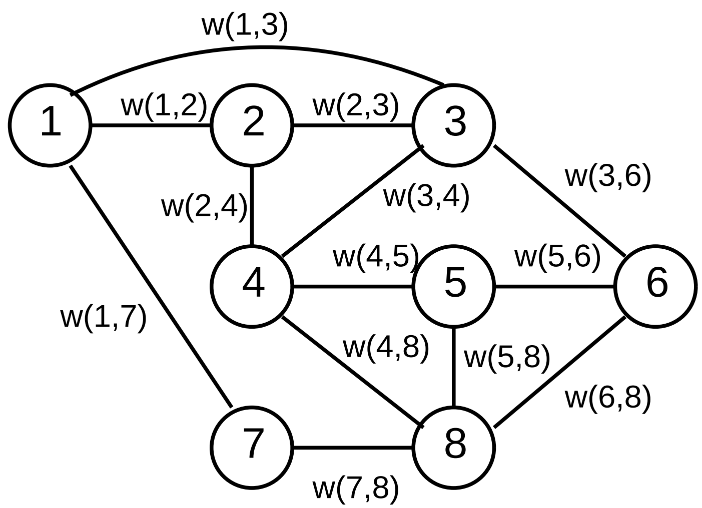
## *All node shortest path*

Other algorithms have structure analogues to matrix-matrix multiply

- FloydWarshall algorithm for all pairs shortest paths.

## *All node shortest path*

$G = (V, E)$, $V = \{1, 2, ..., n\}$.




Edge $(i, j) \in E$ is assigned a non-negative distance $w(i, j)$.

Find shortest paths joining any pair of nodes.

Set $W = \{w_{ij}\}$ where

$$w_{ij} = \begin{cases} 0 & i = j \\ w_{ij} & i \neq j, \ \ (i,j) \in E \\ \infty & i \neq j, \ \ (i,j) \notin E \end{cases}$$

Set $D^{(m)} = \{d_{ij}^{(m)}\}$ where

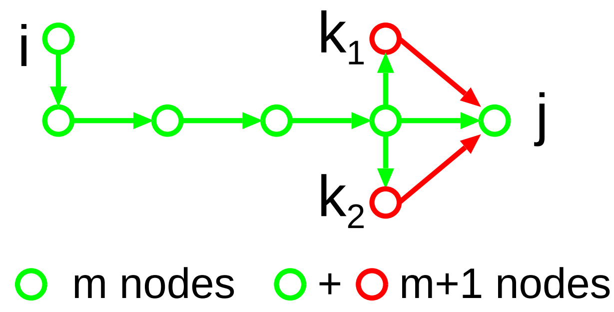
- $d_{ij}^{(m)}$ shortest path from $i$ to $j$ containing at most $m$ nodes

Assume $D^{(m)}$ is known. We want to compute $D^{(m+1)}$.

When $m = n - 1$ then $D^{(m+1)}$ gives us all node shortest path.

How can we get $D^{(m+1)}$ from $D^{(m)}$ ?

## *All node shortest path*




Consider a new path from $i$ to $j$ through $k$ with $m+1$ nodes. We have

$$d_{ij}^{(m+1)} = \min_{1 \le k \le n} (d_{ik}^{(m)} + w_{kj})$$

Note, $d_{ik}^{(m)}$ can be $\infty$.

## *All node shortest path*

Notice the correspondence

| | | | | |
|---|---|---|---|---|
| | matrix multiply | $\Leftrightarrow$ | | minimum sum |
| L: | $c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$ | $\Leftrightarrow$ | R: | $d_{ij}^{(m+1)} = \min_{1 \le k \le n}(d_{ik}^{(m)} + w_{kj})$ |

"$\cdot$" on the left corersponds to "$+$" on the right

"$+$" on the left corresponds to "min" on the right

Computation of $D^{(m+1)}$ can be viewed as multiplication of $D^{(m+1)}$ with $W$ with appropriate interpretation of elementary operations involved.

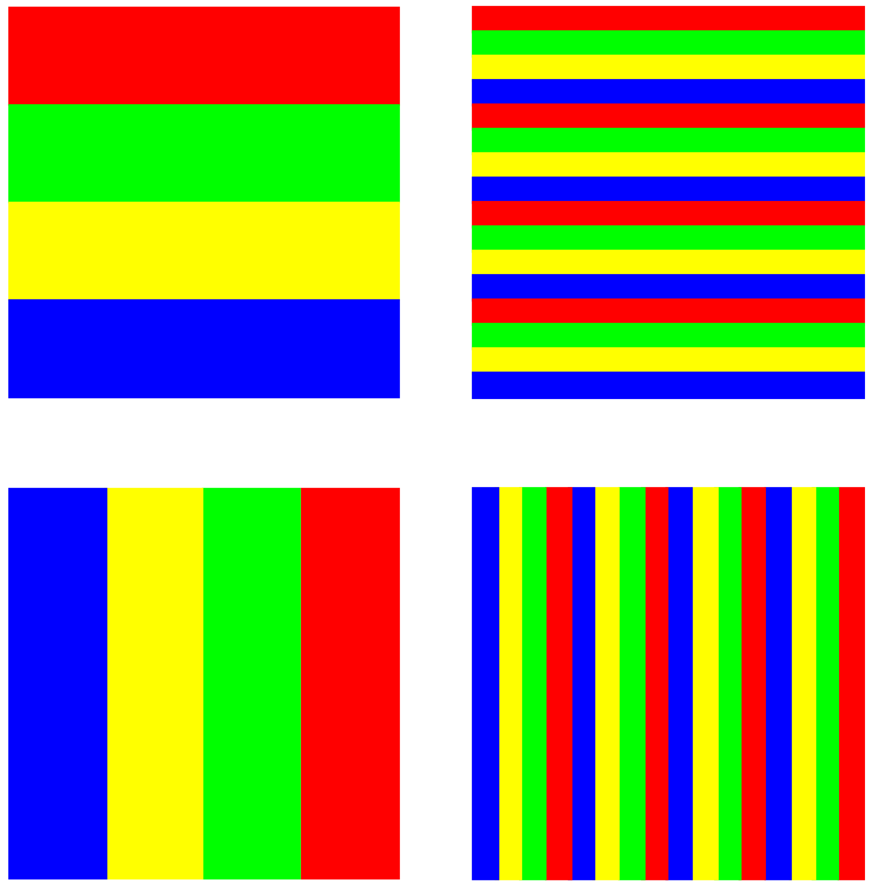
Cost ?

Note that

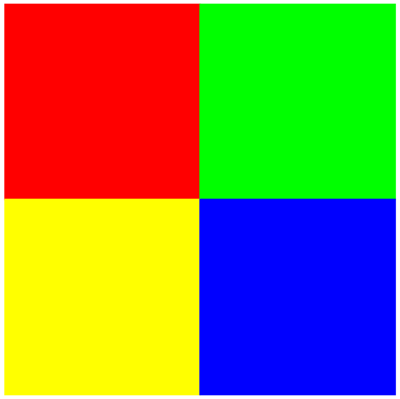
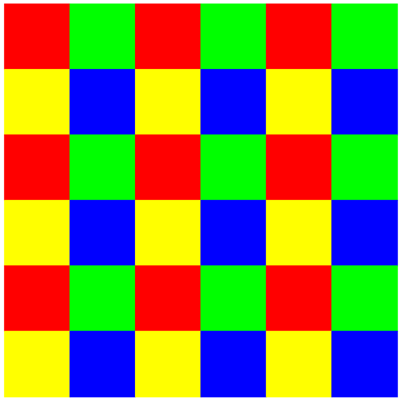
$$D^{(2)} = ?, \quad D^{(m)} = ?$$

Good matrix-matrix multiplication algorithm needed.

## *Matrix distributions*



## *Matrix distributions*

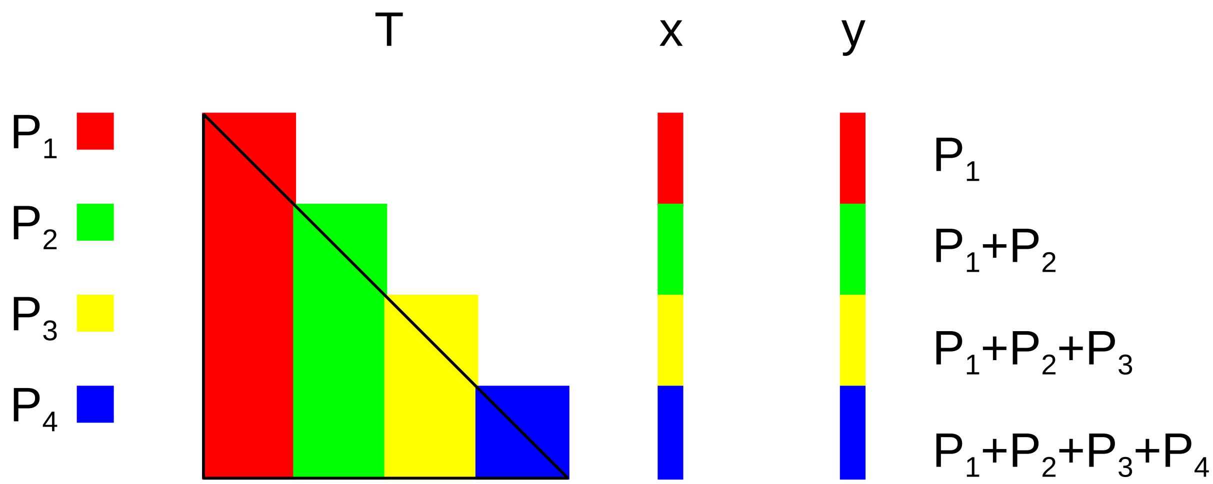
## *Triangular matrix multiply*

Matrices that are not "uniform".

$$y = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} t_{11} & & & & \\ t_{21} & t_{22} & & & \\ t_{31} & t_{32} & t_{33} & & \\ \vdots & & & \ddots & \\ t_{n1} & t_{n2} & & \cdots & t_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = Tx$$

- How do we distribut work among PEs ?
- Do we need synchronization?

1D block-column distribution:



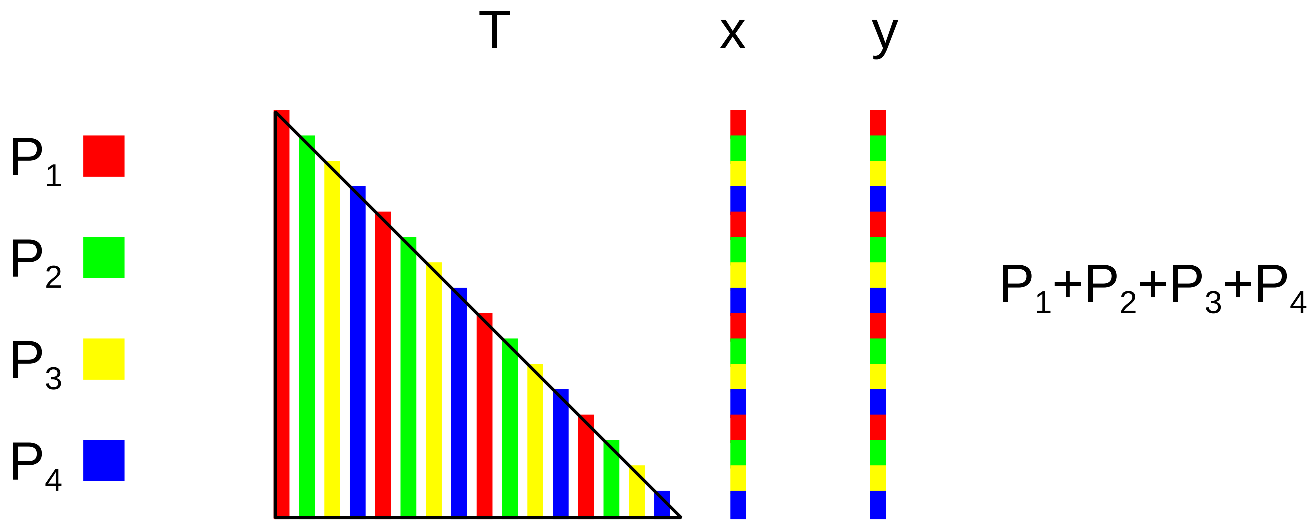

- load imbalance - P4 has the most work while P1 the least
  - about $2n(n-p)$ ops for P4 and $p^2$ for P1

## *Triangular matrix multiply*

- each Pi computes only partial "dot" products
  - P1 computes $\sum_{i=1}^{\frac{n}{p}} t_{ij}x_j$, $i = 1, ..., n$.
- which Pi should accumulate partial "dot" products to get the final "dot" product?
- when should Pi start the final accumulation?

Pi needs to be synchronized. Cache coherence needed?

1D cyclic distribution:

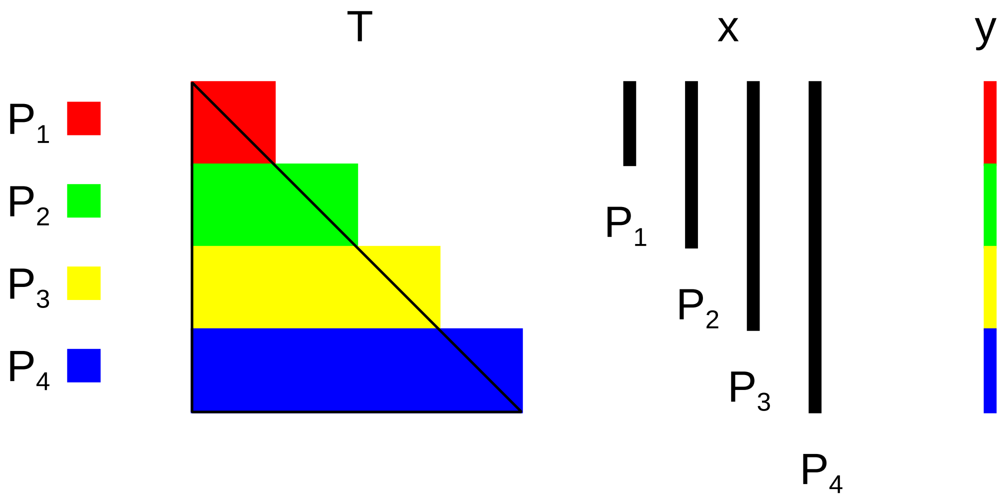



- P1 performs about $n + (n-p) + \cdots + p =?$ ops when computing $x_{1+jp} t_{:,jp+1}$, $j = 1, ..., \frac{n}{p}$

## *Triangular matrix multiply*

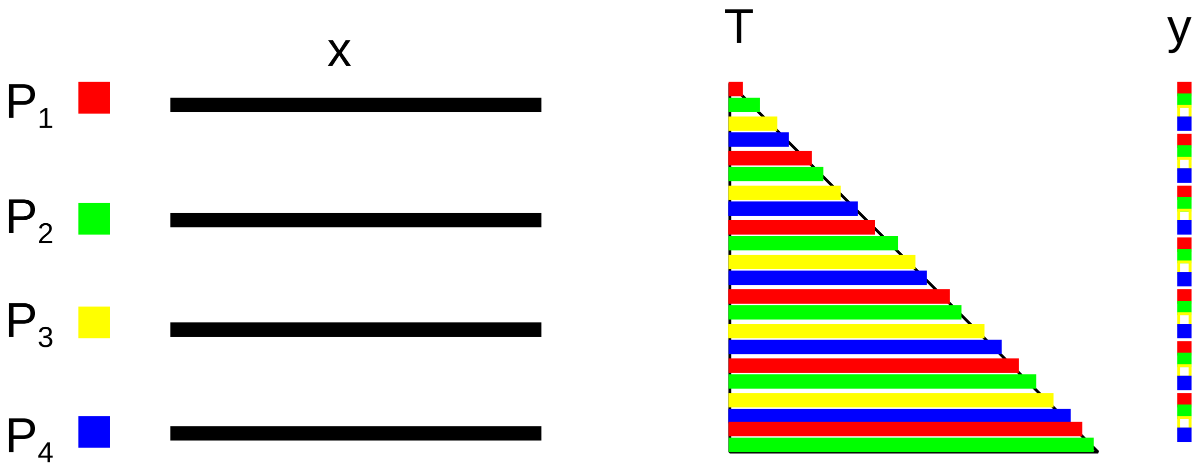
- load balance approximately maintained
- how are those modified columns added?
- we need to synchronize work
- cache coherence?

1D block-row distribution:




- P4 computes $\frac{n}{p}$ dot products of length $\approx n$ for $\frac{n}{p}(2n) = 2\frac{n^2}{p}$ ops.
- P1 computes $\frac{n}{p}$ dot products of length (about) $\frac{n}{p}$ for $\left(\frac{n}{p}\right)^2$ ops
- load imbalance
- no need for synchronization

1D cyclic distribution:




- no need to synchronize work
- load (almost) balanced

## *Solution of triangular systems*

$$T^{(0)}x = \begin{pmatrix} t_{11} & & & & \\ t_{21} & t_{22} & & & \\ t_{31} & t_{32} & t_{33} & & \\ \vdots & & & \ddots & \\ t_{n1} & t_{n2} & t_{n3} & \cdots & t_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{pmatrix} = b^{(0)}$$

## *Solution of triangular systems*

$$x_1 = \frac{b_1}{t_{11}},\ b^{(1)} = b^{(0)} - t_{:,1} \cdot x_1,\ \begin{pmatrix} t_{22} & & & \\ t_{32} & t_{33} & & \\ \vdots & & \ddots & \\ t_{n2} & t_{n3} & \cdots & t_{nn} \end{pmatrix} \begin{pmatrix} x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = b^{(1)}_{2:n}$$

$$x_2 = \frac{b^{(1)}_2}{t_{22}},\ b^{(2)} = b^{(1)} - t_{:,2} \cdot x_1,\ \begin{pmatrix} t_{33} & & & \\ t_{43} & t_{44} & & \\ \vdots & & \ddots & \\ t_{n3} & t_{n4} & \cdots & t_{nn} \end{pmatrix} \begin{pmatrix} x_3 \\ x_4 \\ \vdots \\ x_n \end{pmatrix} = b^{(2)}_{3:n}$$

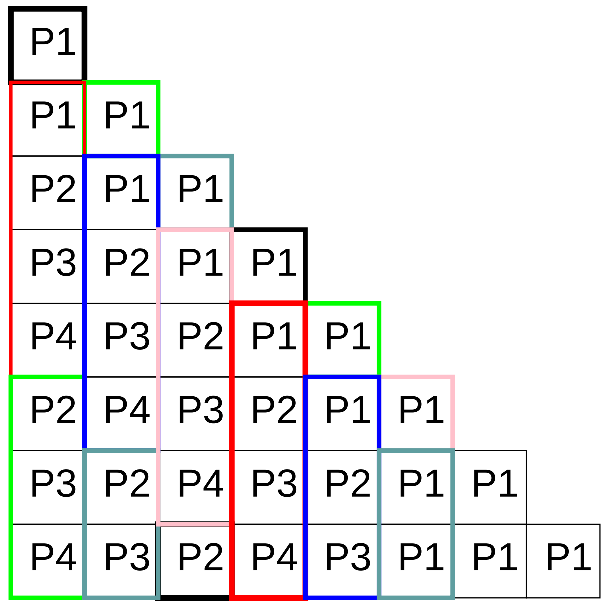
How do we distribute work among PEs ?

## *Solution of triangular systems*

$$T^{(0)}x = \left(\begin{array}{cc|ccc} t_{11} & & & & \\ t_{21} & t_{22} & & & \\ \hline t_{31} & t_{32} & t_{33} & & \\ \vdots & & & \ddots & \\ t_{n1} & t_{n2} & t_{n3} & \cdots & t_{nn} \end{array}\right) \left(\begin{array}{c} x_1 \\ x_2 \\ \hline x_3 \\ \vdots \\ x_n \end{array}\right) = \left(\begin{array}{c} b_1 \\ b_2 \\ \hline b_3 \\ \vdots \\ b_n \end{array}\right) = b^{(0)}$$

- Get $[x_1 \;\; x_2]^T$ (sequentially?) and
- modify $b^{(0)}$ to $b^{(2)}$ (here $p = 2$) where all PEs can operate independently
- assign to each PE $\frac{n-p}{p}$ rows of the "active" submatrics
- repeat until done

Can we "pipeline" a bit?




- dynamic assignment of submatrices
- need to synchronize after each triangular solve and update

What if the matrix is upper triangular ?

$$T^{(0)}x = \begin{pmatrix} t_{11} & t_{12} & t_{13} & \cdots & t_{1n} \\ & t_{22} & t_{23} & \cdots & t_{2n} \\ & & t_{33} & \cdots & t_{3n} \\ & & & \ddots & \vdots \\ & & & & t_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{pmatrix} = b^{(0)}$$

- Model problem: Gaussian elimination

$$Ax = b$$

Gaussian elimination has two stages:

1. Transform the matrix of the system to upper triangular form
2. Solve the traingular system by backsubstitution

Issues to resolve:

- Distribution of work over PEs.
- Synchronization.

# *Solution of linear systems*

$$Ax = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{pmatrix}$$

Guassian elimination: set $i = 1$ and repeat:

1. divide row $i$ of $A^{(i)}$ and $b^{(i)}$ by $a_{ii}^{(i)}$
2. for $j > i$ multiply new row $\hat{a}_{i,:}^{(i)}$ by $a_{ji}^{(i)}$ and subtract from row $a_{j,:}^{(i)}$ to get $a_{j,:}^{(i+1)}$
3. repeat (2) for the rhs to get $b^{(i+1)}$
4. repeat (1) to (3) until $i = n - 1$

## *Solution of linear systems*

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}$$

1. Set $i = 1$, (here $n = 4$)
2. Divide row i of $A$ and $b$ by $a_{ii}$ and store as $\hat{a}_{i,i:n}$, $\hat{b}_i$.
3. For rows $j = i + 1, .., n$, multiply $(\hat{a}_{i,i:n},\ \hat{b}_i)$ by $a_{ji}$ and subtract from $(\hat{a}_{j,i:n},\ \hat{b}_j)$
4. Set $i := i + 1$ and repeat from (2) until done.

## *Solution of linear systems*

After step 1:

$$\left(\begin{array}{c|ccc} a_{11} & a_{12} & a_{13} & a_{14} \\ \hline 0 & a_{22}^{(1)} & a_{23}^{(1)} & a_{24}^{(1)} \\ 0 & a_{32}^{(1)} & a_{33}^{(1)} & a_{34}^{(1)} \\ 0 & a_{42}^{(1)} & a_{43}^{(1)} & a_{44}^{(1)} \end{array}\right) \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \left(\begin{array}{c} b_1 \\ \hline b_2^{(1)} \\ b_3^{(1)} \\ b_4^{(1)} \end{array}\right)$$

After step 2:

$$\left(\begin{array}{c|ccc} a_{11} & a_{12} & a_{13} & a_{14} \\ \hline 0 & a_{22}^{(1)} & a_{23}^{(1)} & a_{24}^{(1)} \\ 0 & & a_{33}^{(2)} & a_{34}^{(2)} \\ 0 & & a_{43}^{(2)} & a_{44}^{(2)} \end{array}\right) \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \left(\begin{array}{c} b_1 \\ \hline b_2^{(1)} \\ b_3^{(2)} \\ b_4^{(2)} \end{array}\right)$$

After step $n-1$

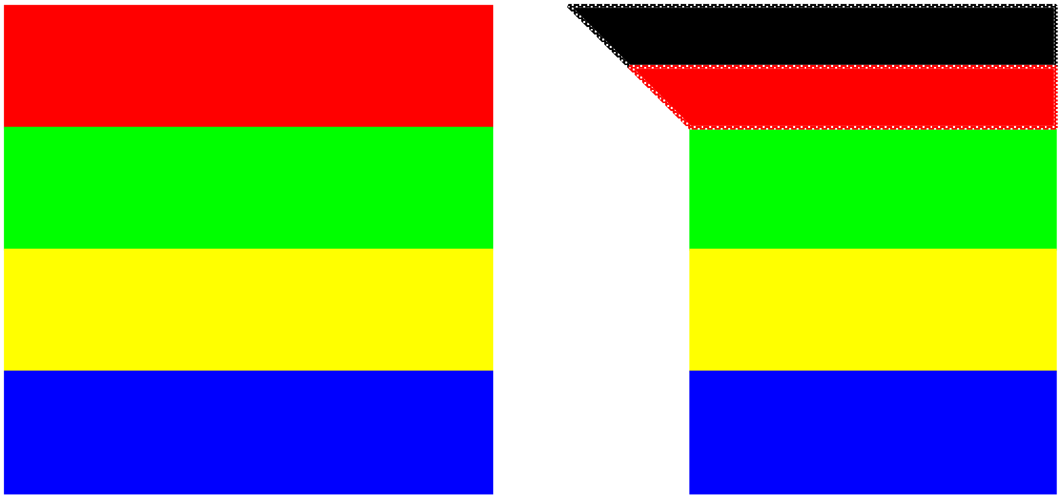
$$\underbrace{\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & a_{24}^{(1)} \\ 0 & 0 & a_{33}^{(2)} & a_{34}^{(2)} \\ 0 & 0 & 0 & a_{44}^{(3)} \end{pmatrix}}_{U} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \underbrace{\begin{pmatrix} b_1 \\ b_2^{(1)} \\ b_3^{(2)} \\ b_4^{(3)} \end{pmatrix}}_{d}$$

Questions:

- What if $a_{ii}^{(i)} = 0$ ?
- What if $|a_{ii}^{(i)}|$ is very small ?
- Pivoting: find $k = \text{argmax}_{j \geq i} |a_{ji}|$.
- swap rows $i$ and $k$
- How do we distribute $A$ among PEs ?

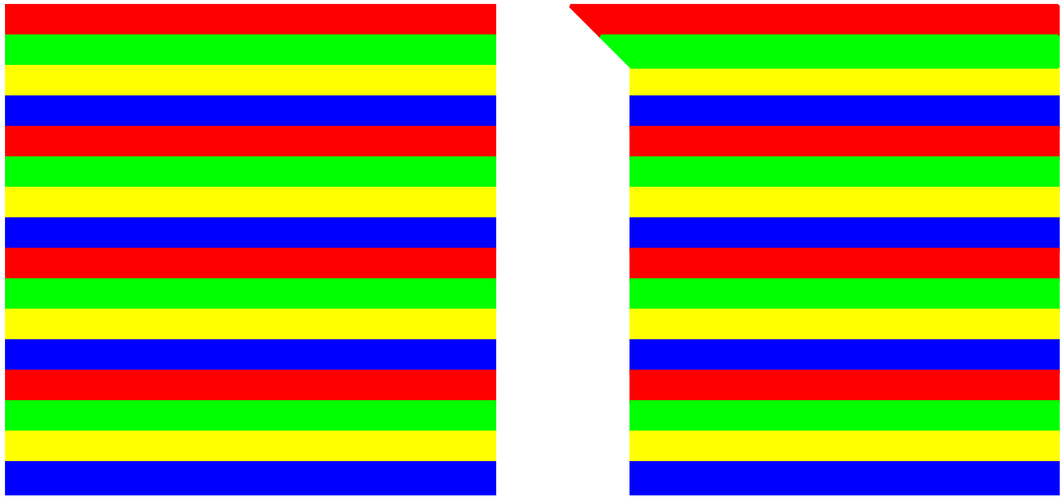
## *Block row distribution*

Distribute data.



- load imbalance
- need to synchronize (where ?)

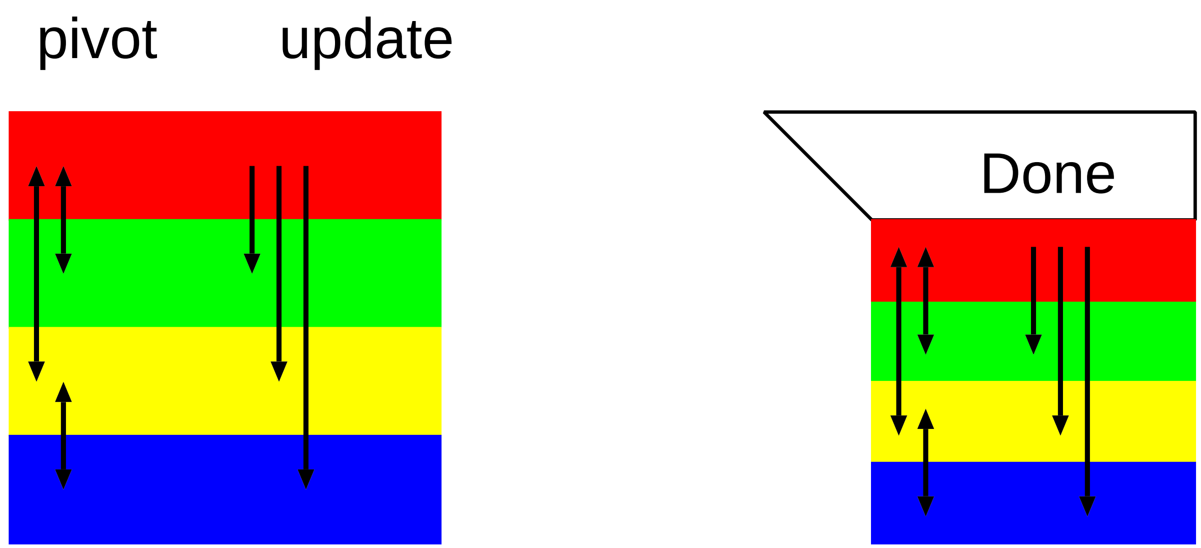
Distribute data.



- cyclic distributes work more evenly
- need to synchronize (where ?)

# *Block row dynamic distribution*

Another option:




- Divide evenly amount of work left.
- At some point switch from multiple PEs to a single PE.

## Steps

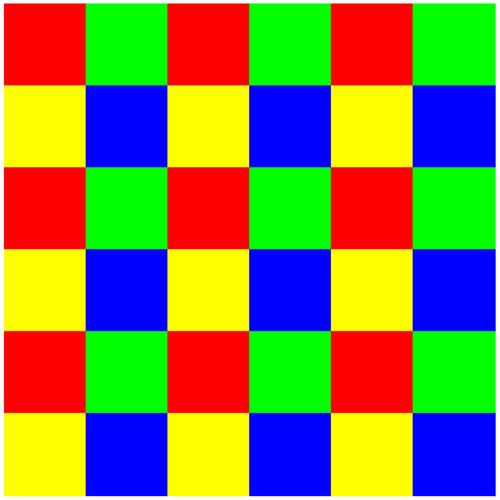
1. local pivot (binary tree)
2. synchronize
3. global pivot
4. synchronize
5. get a copy of the pivot row (all or part ?)
6. update
7. synchronize
8. repeat

## *2D block cyclic distribution*

Many other ways to organize Gaussian elimination.

Each new architecture may need a new kind of reorganization.

A good project.



## *Next time*

Introduction to Pthreads