Parallel dense linear algebra computations (2)

Prof. Richard Vuduc

Georgia Institute of Technology

CSE/CS 8803 PNA, Spring 2008

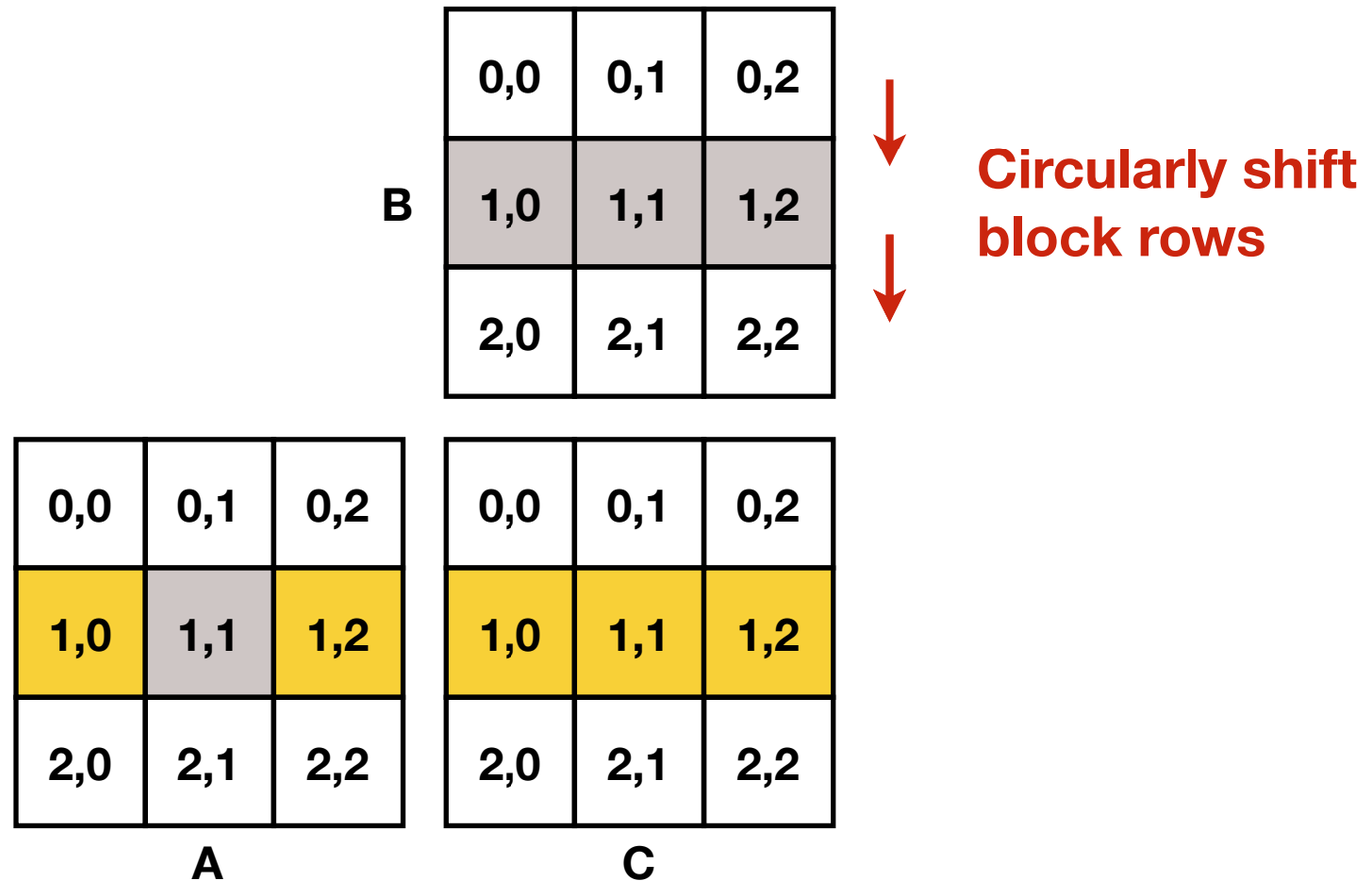
[L.08] Thursday, January 31, 2008

Sources for today's material

- Mike Heath at UIUC
- CS 267 (Yelick & Demmel, UCB)

Review: Parallel matrix multiply

1-D block row-based algorithm



Time, speedup, and efficiency of the 1-D block row-based algorithm

$$T_p = \frac{2n^3}{p} + \alpha p + \frac{n^2}{\beta}$$

⇒ **Not latency tolerant**

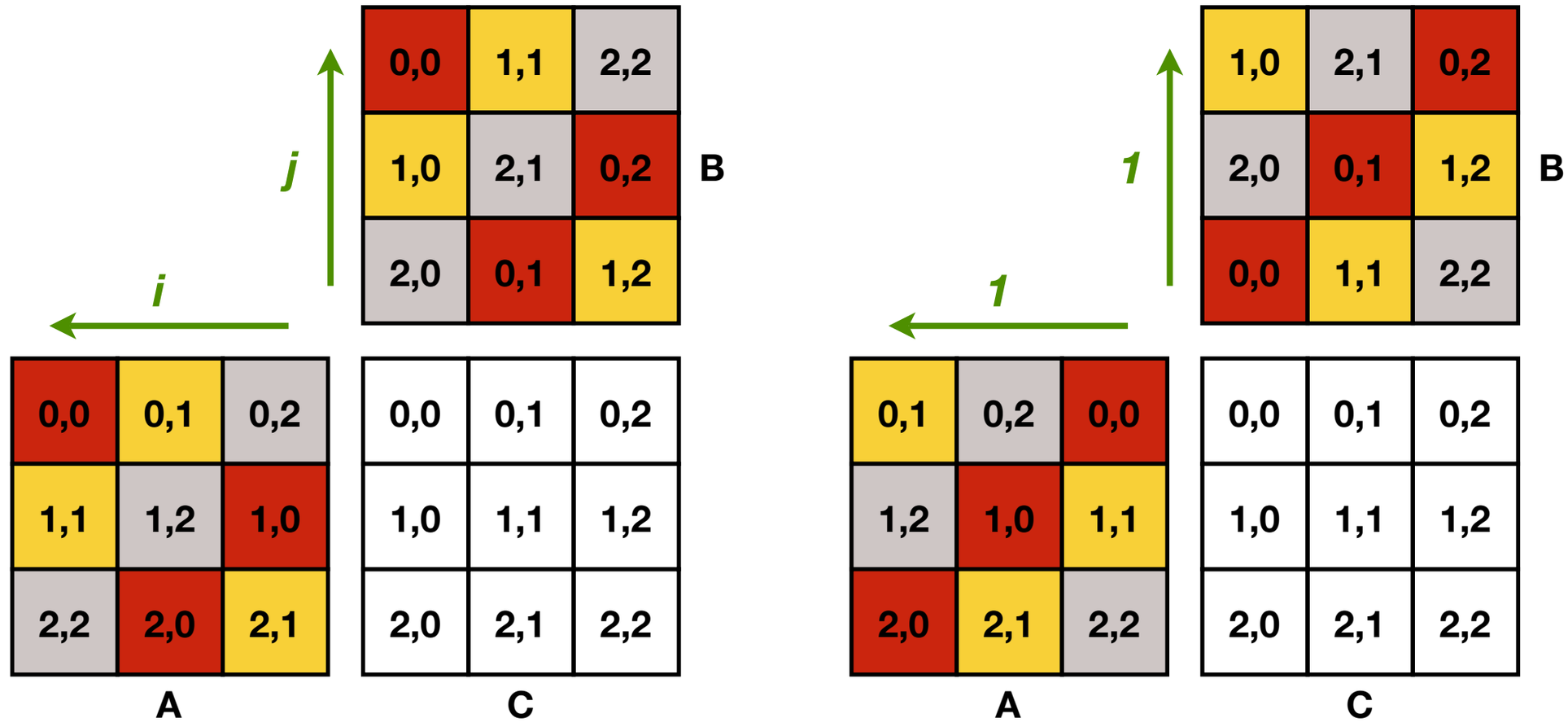
$$S(n) = \frac{1}{\frac{1}{p} + \frac{\alpha}{2}\frac{p}{n^3} + \frac{1}{2\beta}\frac{1}{n}}$$

⇒ **“Perfect” speedup**

$$E_p = \frac{1}{1 + \frac{\alpha}{2}\frac{p^2}{n^3} + \frac{1}{2\beta}\frac{p}{n}}$$

⇒ **n ~ p, so large flop and memory req’d to scale**

Cannon's algorithm, iteration step: Local multiply + circular shift (1)



Time, speedup, and efficiency of Cannon's algorithm

$$T_p = \frac{2n^3}{p} + 4\alpha\sqrt{p} + \frac{4n^2}{\beta\sqrt{p}}$$

⇒ **Better latency term**

$$S(n) = \frac{1}{\frac{1}{p} + 2\alpha\frac{\sqrt{p}}{n^3} + \frac{2}{\beta}\frac{1}{\sqrt{p}}}$$

⇒ **"Perfect" speedup**

$$E_p = \frac{1}{1 + 2\alpha\left(\frac{\sqrt{p}}{n}\right)^3 + \frac{2}{\beta}\frac{\sqrt{p}}{n}}$$

⇒ **n ~ sqrt(p)**

Cannon's algorithm: Pros and cons

- Pro: Local matrix-multiply "optimize-able" with better flop-to-memory ratio
- Con: Hard to generalize for…
  - p not a perfect square
  - A & B not square
  - dim(A) & dim(B) not divisible by s=sqrt(p)
  - A & B not "aligned" in their storage

- Can we do better?

SUMMA:
Scalable Universal Matrix Multiply

- By van de Geijn and Watts: http://www.netlib.org/lapack/lawns/lawn96.ps
- Used in the Parallel BLAS: http://www.netlib.org/lapack/lawns/lawn100.ps

Equivalent formulations of matrix-matrix multiplication

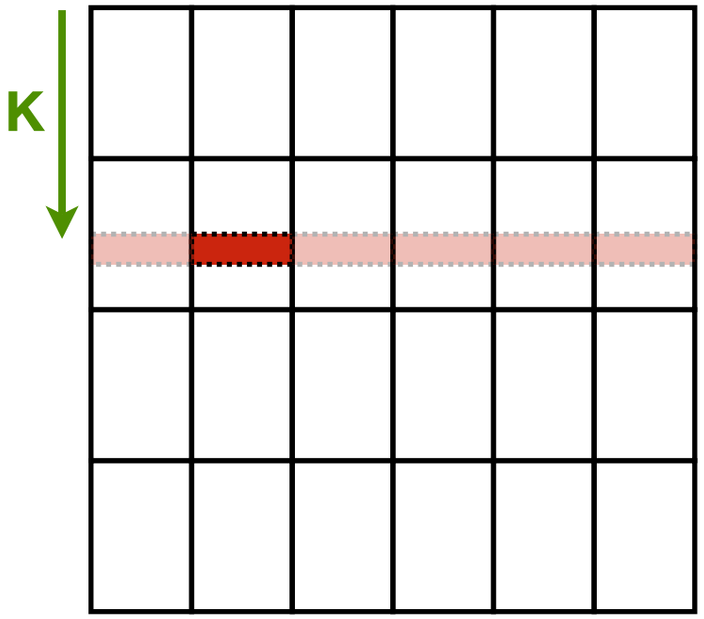
Dot products

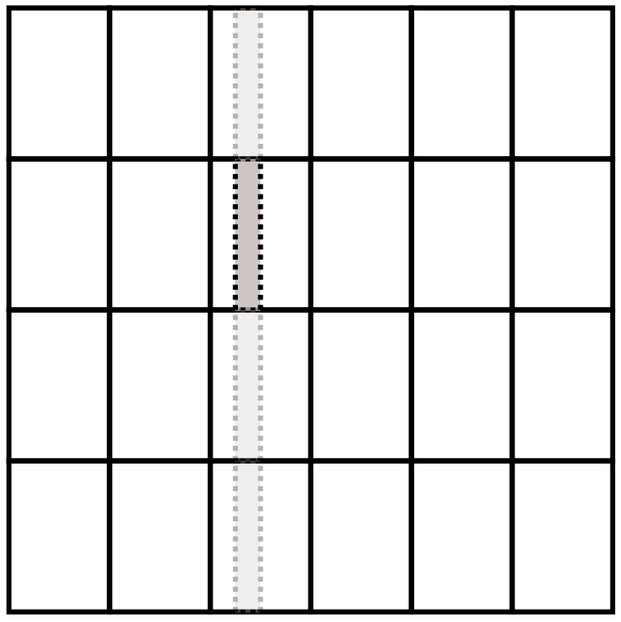
$$C = A \cdot B = \begin{pmatrix} \hat{a}_1^T \\ \vdots \\ \hat{a}_m^T \end{pmatrix} \cdot \begin{pmatrix} b_1 & \cdots & b_n \end{pmatrix} = \begin{pmatrix} \hat{a}_1^T \cdot b_1 & \cdots & \hat{a}_1^T \cdot b_n \\ \vdots & & \vdots \\ \hat{a}_m^T \cdot b_1 & \cdots & \hat{a}_m^T \cdot b_n \end{pmatrix}$$

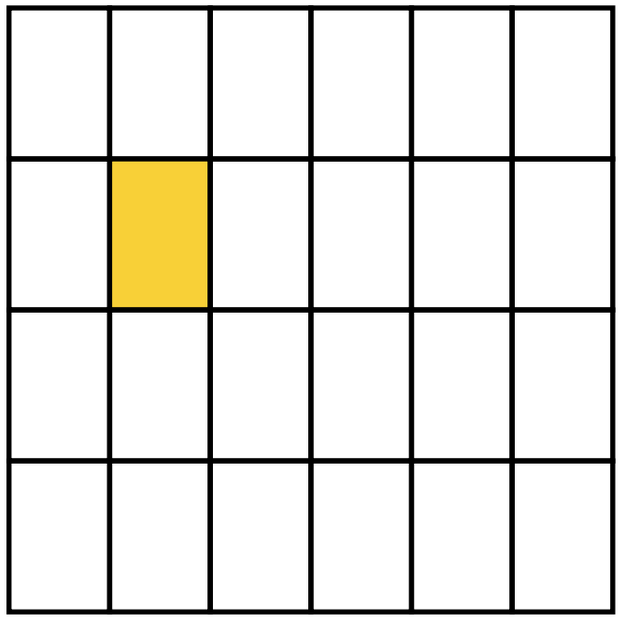
$$\implies c_{ij} = \hat{a}_i^T \cdot b_j$$

Outer products

$$C = A \cdot B = \begin{pmatrix} a_1 \cdots a_k \end{pmatrix} \cdot \begin{pmatrix} \hat{b}_1^T \\ \vdots \\ \hat{b}_k^T \end{pmatrix} = a_1 \hat{b}_1^T + \cdots + a_k \hat{b}_k^T = \sum_k a_k \hat{b}_k^T$$





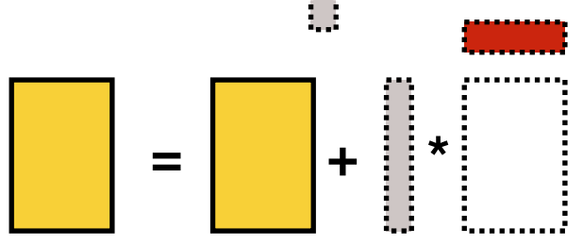


SUMMA:

for-all **K**:

Broadcast

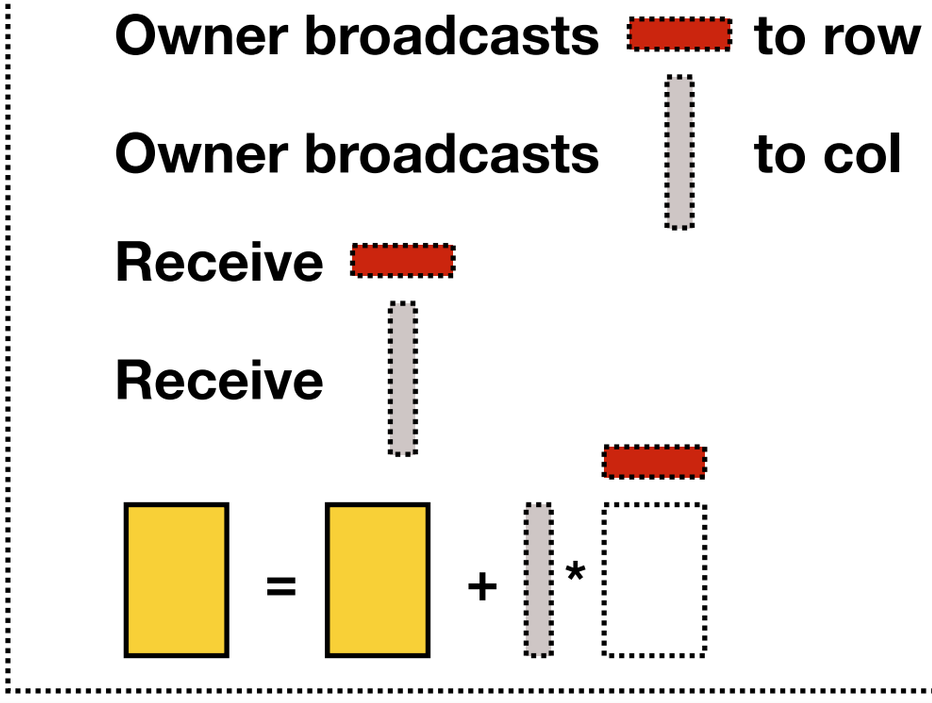
Broadcast



Cost of SUMMA

SUMMA:

for-all **K**, with panel block size ***b***:



- Assume:
  - Square matrix multiply, dim = ***n***
  - sqrt(***p***) x sqrt(***p***) processor grid
  - Panel block size ***b***
  - Tree-based broadcast

Time, efficiency, and space of SUMMA

$$\text{For } 1 \leq b \leq \frac{n}{\sqrt{p}}:$$

$$T_p = \frac{2n^3}{p} + \alpha \frac{n \log p}{b} + \frac{1}{\beta} \frac{n^2 \log p}{\sqrt{p}}$$

$$E_p = \frac{1}{1 + \frac{\alpha}{2} \frac{p \log p}{n^2 b} + \frac{1}{2\beta} \frac{\sqrt{p} \log p}{n}}$$

$$\frac{M(n,b)}{p} = 3\frac{n^2}{p} + 2\frac{nb}{\sqrt{p}}$$

⇒ Use **b** to trade latency and storage

Parallel dense Gaussian elimination

Algorithms for 2-D (3-D) Poisson, $N=n^2$ ($=n^3$)

Algorithm	Serial	PRAM	Memory	# procs
Dense LU	**N^3**	**N**	**N^2**	**N^2**
Band LU	N^2 ($N^{7/3}$)	N	$N^{3/2}$ ($N^{5/3}$)	N ($N^{4/3}$)
Jacobi	**N^2 ($N^{5/3}$)**	**N ($N^{2/3}$)**	**N**	**N**
Explicit inverse	N^2	log N	N^2	N^2
Conj. grad.	**$N^{3/2}$ ($N^{4/3}$)**	**$N^{1/2}$(1/3) log N**	**N**	**N**
RB SOR	**$N^{3/2}$ ($N^{4/3}$)**	**$N^{1/2}$ ($N^{1/3}$)**	**N**	**N**
Sparse LU	$N^{3/2}$ (N^2)	$N^{1/2}$	N log N ($N^{4/3}$)	N
FFT	N log N	log N	N	N
Multigrid	N	$\log^2$ N	N	N
Lower bound	**N**	**log N**	**N**	

PRAM = idealized parallel model with zero communication cost.

Source: Demmel (1997)

Main problems in linear algebra

- Solving linear systems

$$Ax = b$$

- Least squares problems

$$\min_x ||Ax - b||_2$$

- Eigenvalue problems (including SVD)

$$Ax = \lambda x \qquad A^T Ax = \sigma^2 x$$

Why consider dense algorithms?

- Dense problems arise in practice
  - Computational electromagnetics
  - Quantum chemistry
  - Statistics
- LINPACK, for better or worse
- Large sparse problem may yield smaller (but still large) dense subproblems

Solving linear systems

$$Ax = b$$

- Gaussian elimination (LU factorization)

$$\begin{array}{rcl} A & = & L \cdot U \\ & \Downarrow & \\ Ax & = & LUx = b \\ Ly & = & b \\ Ux & = & y \end{array}$$

$$\begin{pmatrix} l_{11} & & \\ \vdots & \ddots & \\ l_{n1} & \cdots & l_{nn} \end{pmatrix} \cdot \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$$

⇐ **Forward-substitution**

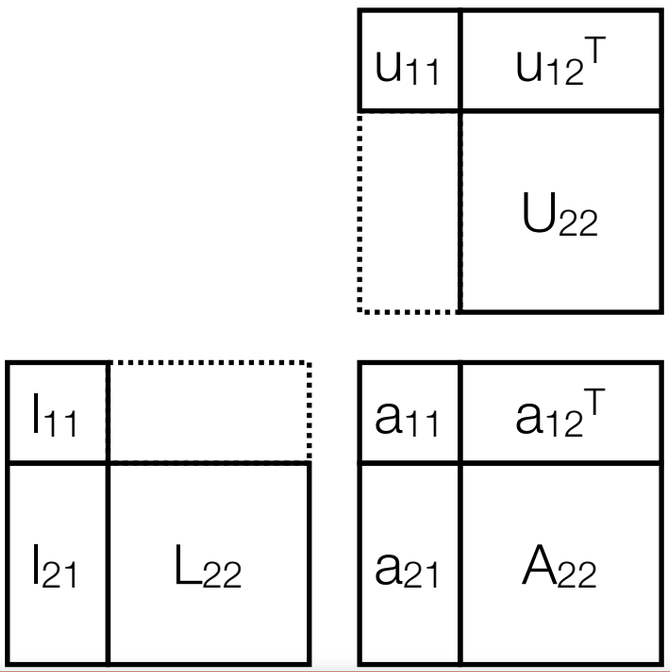
⇐ **Back-substitution**

An LU factorization algorithm

$$\begin{pmatrix} l_{11} & \\ l_{21} & L_{22} \end{pmatrix} \cdot \begin{pmatrix} u_{11} & \hat{u}_{12}^T \\ & U_{22} \end{pmatrix} = \begin{pmatrix} a_{11} & \hat{a}_{12}^T \\ a_{21} & A_{22} \end{pmatrix}$$

$$\Downarrow$$

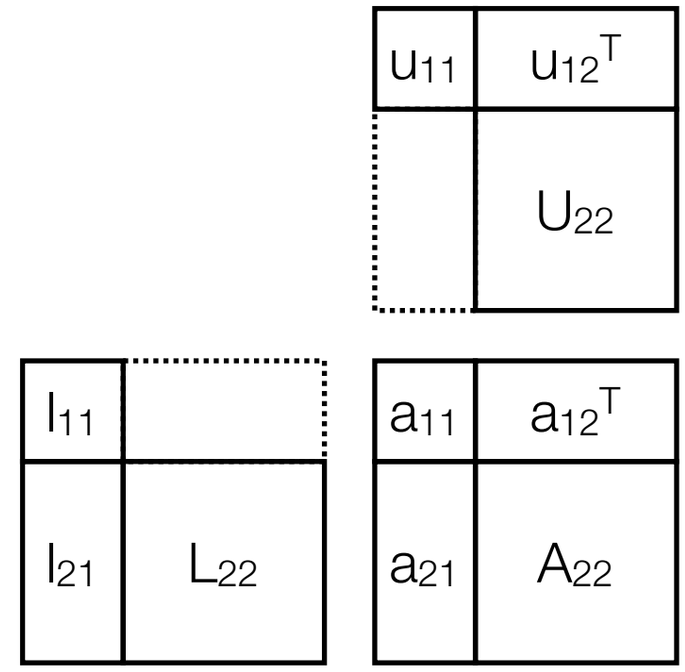
$$\begin{aligned} l_{11} &= 1 \\ u_{11} &= a_{11} \\ \hat{u}_{12}^T &= \hat{a}_{12}^T \\ l_{21} &= \frac{a_{21}}{u_{11}} = \frac{a_{21}}{a_{11}} \\ L_{22}U_{22} &= A_{22} - l_{21}\hat{u}_{12}^T \end{aligned}$$



An LU factorization algorithm (Overwrites A)

$$
\begin{aligned}
l_{11} &= 1 \\
u_{11} &= a_{11} \\
\hat{u}_{12}^T &= \hat{a}_{12}^T \\
l_{21} &= \frac{a_{21}}{a_{11}} \\
L_{22}U_{22} &= A_{22} - l_{21}\hat{u}_{12}^T
\end{aligned}
$$

```
for k = 1 to n-1 do
  for i = k+1 to n do
    a(i,k) ← a(i,k) / a(k, k)
  for i = k+1 to n do
    for j = k+1 to n do
      a(i,j) ← a(i,j) - a(i,k)*a(k,j)
```



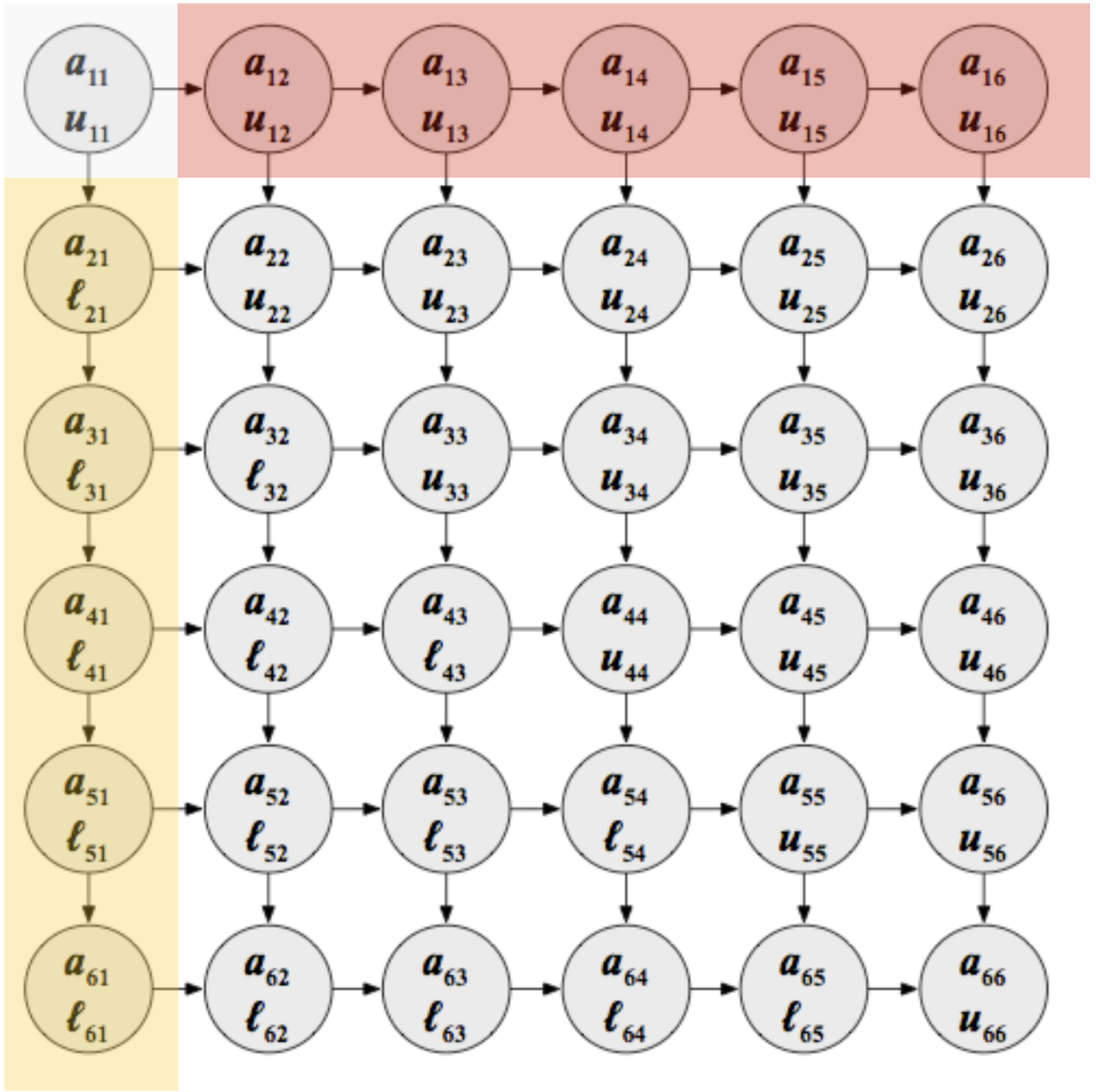
Serial costs of the algorithm

$$T_1 \approx \frac{2}{3}n^3 + \frac{1}{2}n^2$$

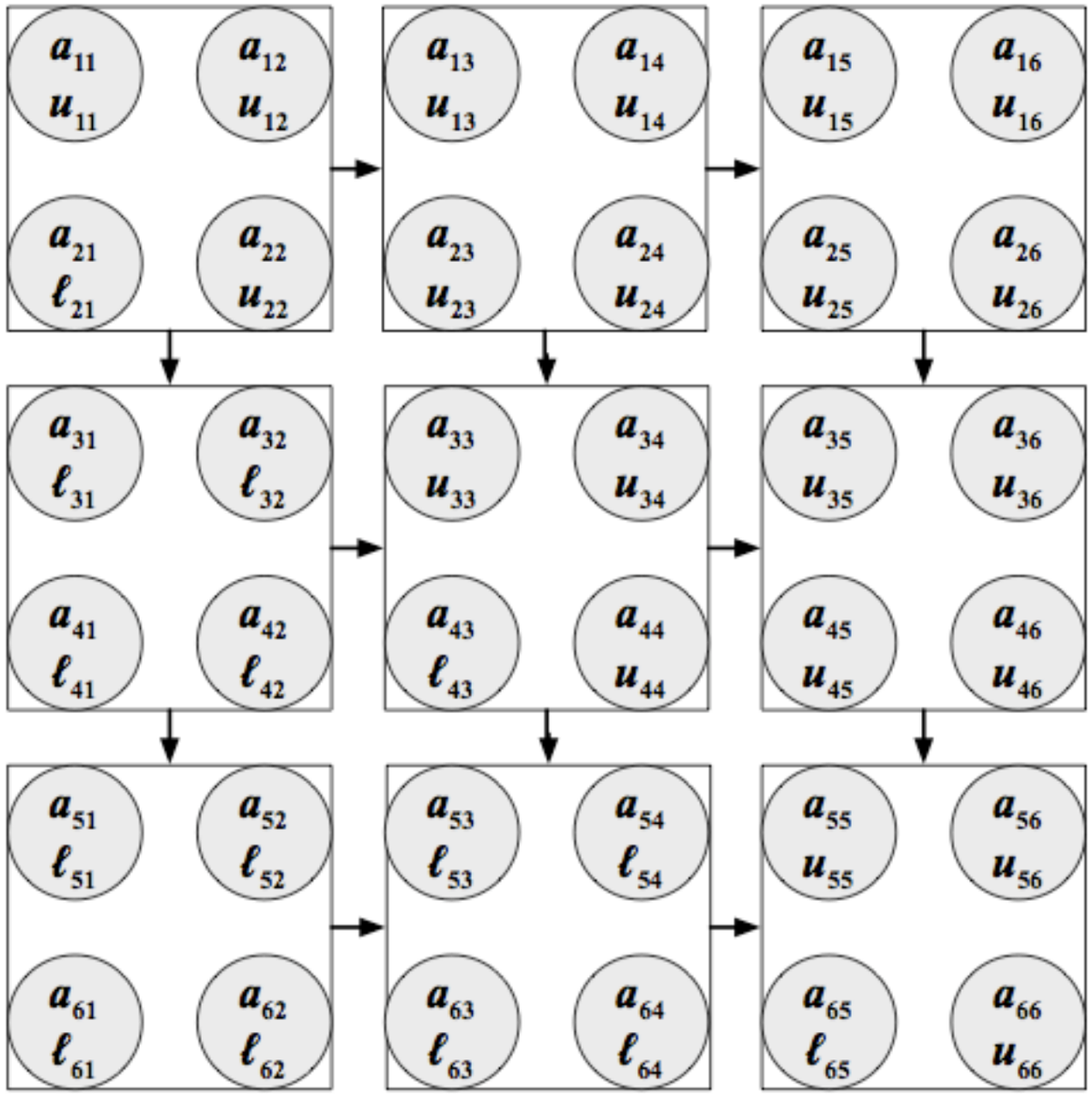
Several loop orders are possible

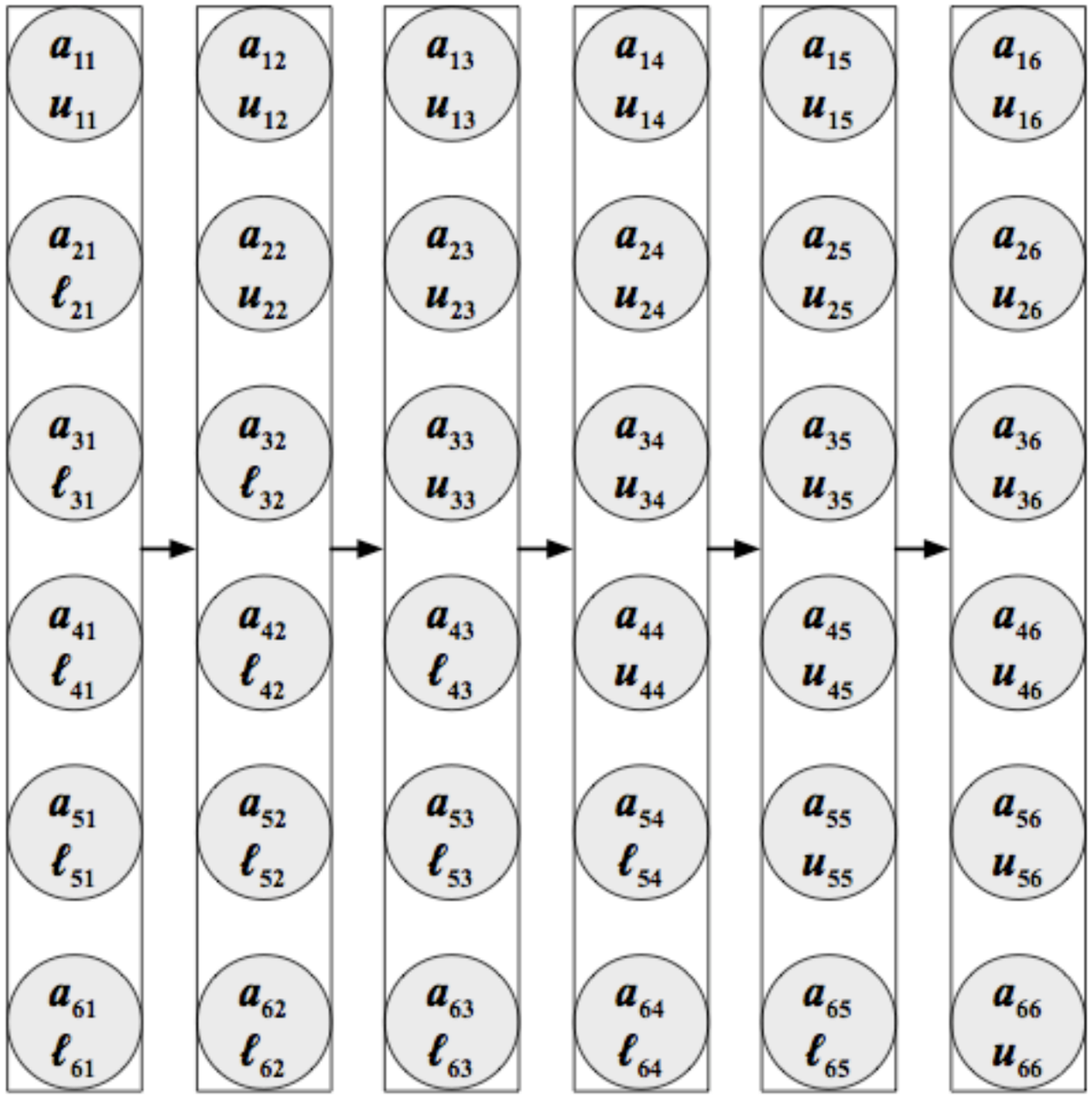
```
for _______ do
  for _______ do
    for _______ do
      a(i,j) ← a(i,j) - a(i,k)/a(k,k)*a(k,j)
```

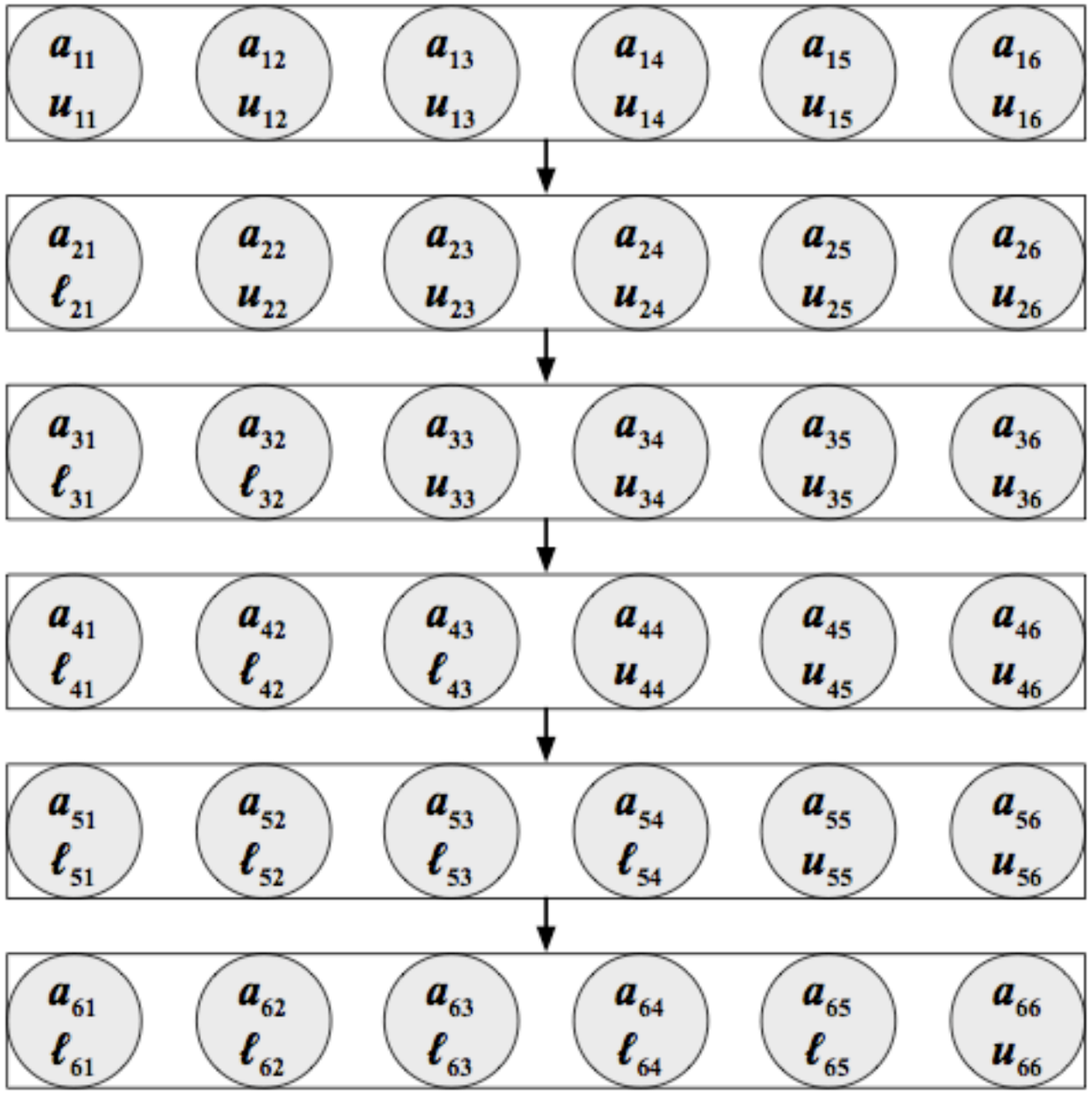
```
for k = 1 to n-1 do
  for i = k+1 to n do
    a(i,k) ← a(i,k) / a(k,k)
  for j = k+1 to n do
    for i = k+1 to n do
      a(i,j) ← a(i,j) - a(i,k)*a(k,j)
```



$$
\begin{aligned}
l_{11} &= 1 \\
u_{11} &= a_{11} \\
\hat{u}_{12}^T &= \hat{a}_{12}^T \\
l_{21} &= \frac{a_{21}}{a_{11}} \\
L_{22}U_{22} &= A_{22} - l_{21}\hat{u}_{12}^T
\end{aligned}
$$

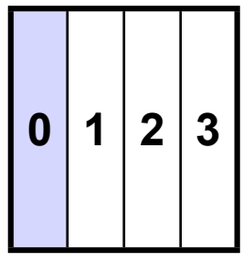

a_{11} u_{11}
a_{12} u_{12}
a_{13} u_{13}
a_{14} u_{14}
a_{15} u_{15}
a_{16} u_{16}
a_{21} ℓ_{21}
a_{22} u_{22}
a_{23} u_{23}
a_{24} u_{24}
a_{25} u_{25}
a_{26} u_{26}
a_{31} ℓ_{31}
a_{32} ℓ_{32}
a_{33} u_{33}
a_{34} u_{34}
a_{35} u_{35}
a_{36} u_{36}
a_{41} ℓ_{41}
a_{42} ℓ_{42}
a_{43} ℓ_{43}
a_{44} u_{44}
a_{45} u_{45}
a_{46} u_{46}
a_{51} ℓ_{51}
a_{52} ℓ_{52}
a_{53} ℓ_{53}
a_{54} ℓ_{54}
a_{55} u_{55}
a_{56} u_{56}
a_{61} ℓ_{61}
a_{62} ℓ_{62}
a_{63} ℓ_{63}
a_{64} ℓ_{64}
a_{65} ℓ_{65}
a_{66} u_{66}


a_{11} u_{11}
a_{12} u_{12}
a_{13} u_{13}
a_{14} u_{14}
a_{15} u_{15}
a_{16} u_{16}
a_{21} ℓ_{21}
a_{22} u_{22}
a_{23} u_{23}
a_{24} u_{24}
a_{25} u_{25}
a_{26} u_{26}
a_{31} ℓ_{31}
a_{32} ℓ_{32}
a_{33} u_{33}
a_{34} u_{34}
a_{35} u_{35}
a_{36} u_{36}
a_{41} ℓ_{41}
a_{42} ℓ_{42}
a_{43} ℓ_{43}
a_{44} u_{44}
a_{45} u_{45}
a_{46} u_{46}
a_{51} ℓ_{51}
a_{52} ℓ_{52}
a_{53} ℓ_{53}
a_{54} ℓ_{54}
a_{55} u_{55}
a_{56} u_{56}
a_{61} ℓ_{61}
a_{62} ℓ_{62}
a_{63} ℓ_{63}
a_{64} ℓ_{64}
a_{65} ℓ_{65}
a_{66} u_{66}

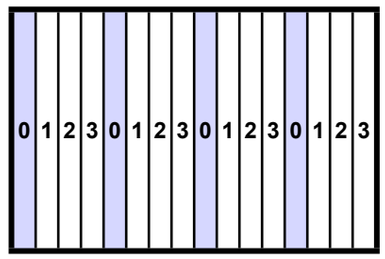

a_{11} u_{11}
a_{12} u_{12}
a_{13} u_{13}
a_{14} u_{14}
a_{15} u_{15}
a_{16} u_{16}
a_{21} ℓ_{21}
a_{22} u_{22}
a_{23} u_{23}
a_{24} u_{24}
a_{25} u_{25}
a_{26} u_{26}
a_{31} ℓ_{31}
a_{32} ℓ_{32}
a_{33} u_{33}
a_{34} u_{34}
a_{35} u_{35}
a_{36} u_{36}
a_{41} ℓ_{41}
a_{42} ℓ_{42}
a_{43} ℓ_{43}
a_{44} u_{44}
a_{45} u_{45}
a_{46} u_{46}
a_{51} ℓ_{51}
a_{52} ℓ_{52}
a_{53} ℓ_{53}
a_{54} ℓ_{54}
a_{55} u_{55}
a_{56} u_{56}
a_{61} ℓ_{61}
a_{62} ℓ_{62}
a_{63} ℓ_{63}
a_{64} ℓ_{64}
a_{65} ℓ_{65}
a_{66} u_{66}

1-D column blocked

Bad load balance



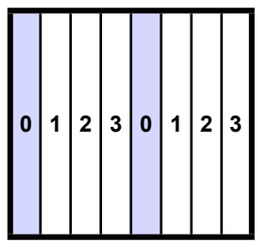
1-D column cyclic



Can't easily use BLAS2/3

1-D column block cyclic

Factorization bottleneck



Block skewed

0	1	2	3
3	0	1	2
2	3	0	1
1	2	3	0

Complicated addressing

2-D row & col blocked

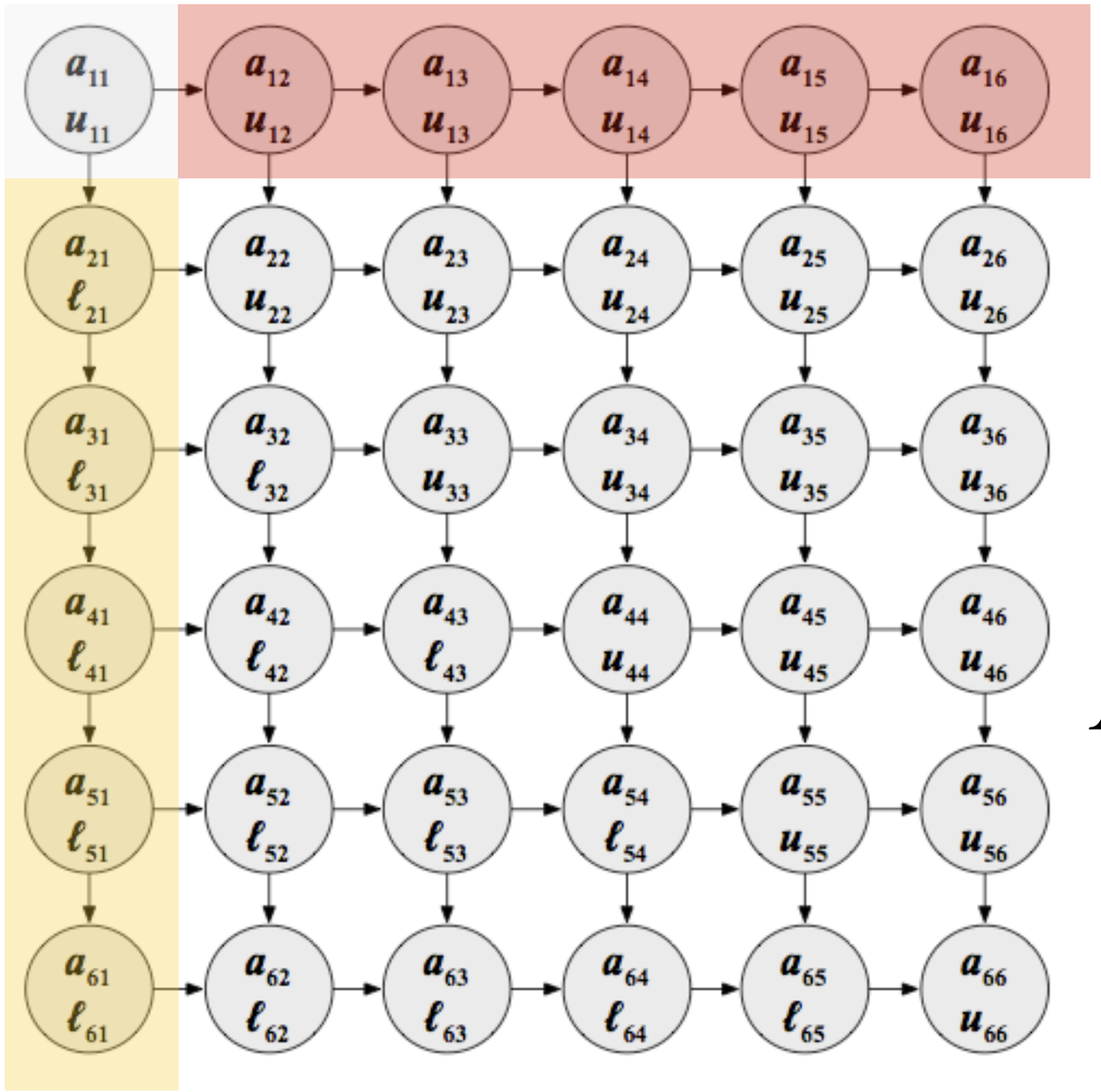
Bad load balance

0	1
2	3

2-D row & col block cyclic

0	1	0	1	0	1	0	1
2	3	2	3	2	3	2	3
0	1	0	1	0	1	0	1
2	3	2	3	2	3	2	3
0	1	0	1	0	1	0	1
2	3	2	3	2	3	2	3
0	1	0	1	0	1	0	1
2	3	2	3	2	3	2	3

Best option



$$
\begin{aligned}
l_{11} &= 1 \\
u_{11} &= a_{11} \\
\hat{u}_{12}^T &= \hat{a}_{12}^T \\
l_{21} &= \frac{a_{21}}{a_{11}} \\
L_{22}U_{22} &= A_{22} - l_{21}\hat{u}_{12}^T
\end{aligned}
$$

$$
A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}
$$

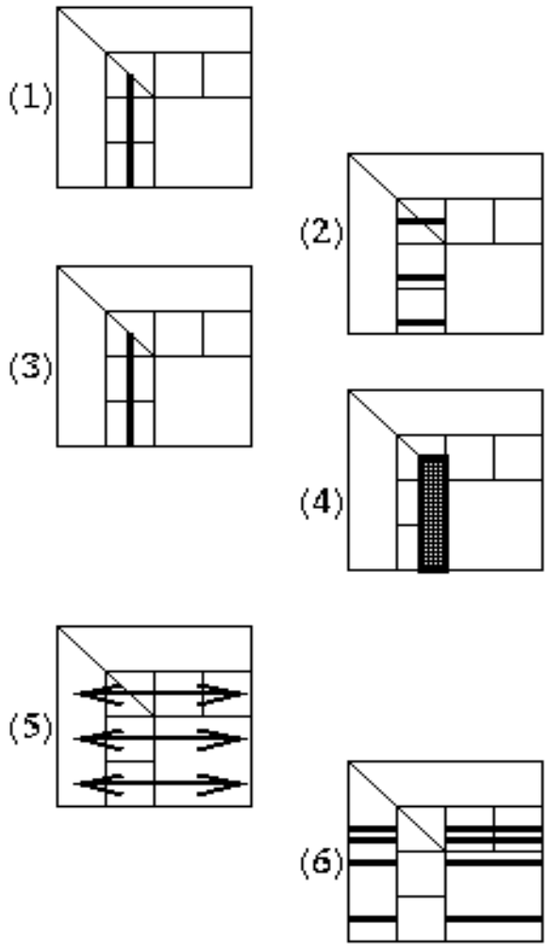
Partial pivoting

- LU factorization as presented so far fails on this matrix

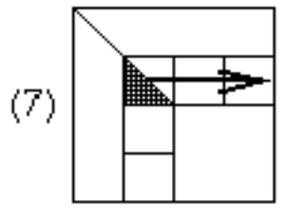
$$A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

- Rows may be processed in any order
- **Partial pivoting**: Swap rows so “top” row has leading entry with largest magnitude

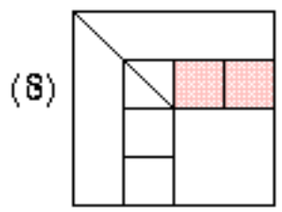
Distributed Gaussian Elimination with a 2D Block Cyclic Layout



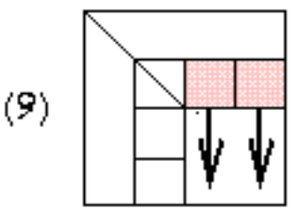

```
for ib = 1 to n−1 step b

    end = min( ib+b−1, n )

    for i = ib to end

    (1)   find pivot row k, column broadcast

    (2)   swap rows k and i in block column, broadcast row k

    (3)   A( i+1:n , i ) = A( i+1:n , i ) / A( i , i )

    (4)   A( i+1:n , i+1:end) −= A( i+1:n , i ) * A( i , i+1:end )

    end for

(5)   broadcast all swap information right and left

(6)   apply all rows swaps to other columns
```



(7) Broadcast LL right

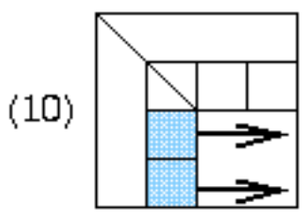


(8) A(ib:end , end+1:n) = LL \ A(ib:end , end+1:n)



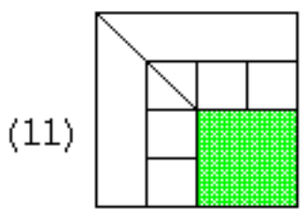
(9) Broadcast A(ib:end , end+1:n) down

Tree: $$\left(\alpha + \frac{b(n-\text{end})}{p_c} \cdot \frac{1}{\beta}\right) \cdot \log p_r$$



(10) Broadcast A(end+1:n , ib:end) right

Ring: $$\left(\alpha + \frac{b(n-\text{end})}{p_r} \cdot \frac{1}{\beta}\right) \cdot 2$$



(11) Eliminate A(end+1:n , end+1:n)

Matmul: $$\frac{2b(n-\text{end})^2}{p}$$

Total costs

- Sum over end = b, 2*b, 3*b, …, n-b, for steps 9, 10, and 11:

$$\frac{2}{3}\frac{n^3}{p} + \alpha\frac{n\log p_r + 2}{b} + \beta n^2\left(\frac{\log p_r}{2p_c} + \frac{1}{p_r}\right)$$

Total costs

$$T_p = \frac{2}{3}\frac{n^3}{p} + \alpha n\left(6 + \log p_r\right) + \beta n^2\left(2\frac{p_r - 1}{p} + \frac{p_c - 1}{p} + \frac{\log p_r}{2p_c}\right)$$

$$E_p = \frac{1}{1 + \frac{3}{2}\alpha(6 + \log p_r)\frac{p}{n^2} + \frac{3}{2}\beta\frac{2p_r + p_c + \frac{1}{2}p_r \log p_r - 3}{n}}$$

Administrivia

Administrative stuff

- **New room** (dumpier, but cozier?): College of Computing Building **(CCB) 101**
- **Accounts**: Apparently, you already have them
  - Front-end login node: **ccil.cc.gatech.edu** (CoC Unix account)
    - We “own” **warp43—warp56**
    - Some docs (**MPI**): http://www-static.cc.gatech.edu/projects/ihpcl/mpi.html
    - **Sign-up** for mailing list: https://mailman.cc.gatech.edu/mailman/listinfo/ihpc-lab

Homework 1:
Parallel conjugate gradients

- Implement a parallel solver for Ax = b (serial C version provided)
  - Evaluate on three matrices: 27-pt stencil, and two application matrices
  - "Simplified:" No preconditioning
  - **Bonus**: Reorder, precondition
- Performance models to understand scalability of your implementation
  - Make measurements
  - Build predictive models
- Collaboration encouraged: Compare programming models or platforms

“In conclusion…”

Backup slides