Multicore Computing
Lecture09 - Matrix Multiplication & Gaussian Elimination

SUNG KYUN KWAN UNIVERSITY

남 범 석
bnam@skku.edu

Some Applications of Matrix

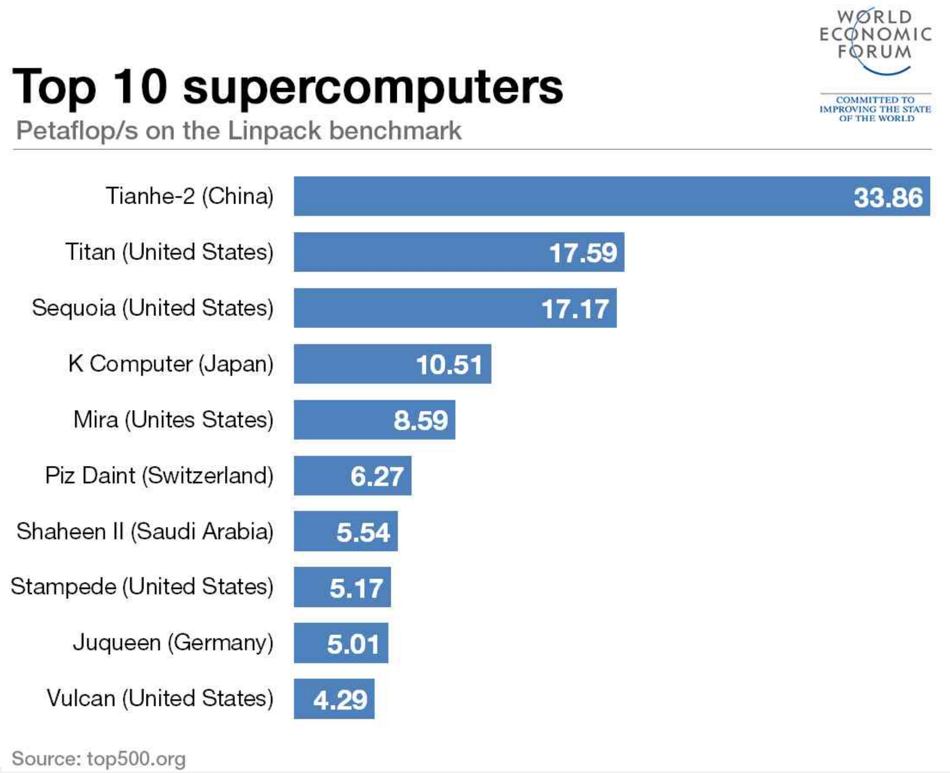
- In Physics related scientific domains, matrices are used in Electrical circuits, Quantum mechanics, Optics, and many more.
- Stochasic matrics and Eigen vector solvers are used in the page rank algorithms
- Encryption of message codes (cryptocurrency, etc)
- 3D modeling in Computer Graphics and Vision
- In robotics and automation, matrices are the base elements for the robot movements

Linear Algebra Packages

- Basic Linear Algebra Subroutines (BLAS)
  - Level 1 (*vector-vector*): vectorization
  - Level 2 (*matrix-vector*): vectorization, parallelization
  - Level 3 (*matrix-matrix*): parallelization
- LINPACK (Fortran)
  - Linear equations and linear least-squares
- EISPACK (Fortran)
  - Eigenvalues and eigenvectors for matrix classes
- LAPACK (Fortran, C) (LINPACK + EISPACK)
  - Use BLAS internally
- ScaLAPACK (Fortran, C, MPI) (scalable LAPACK)

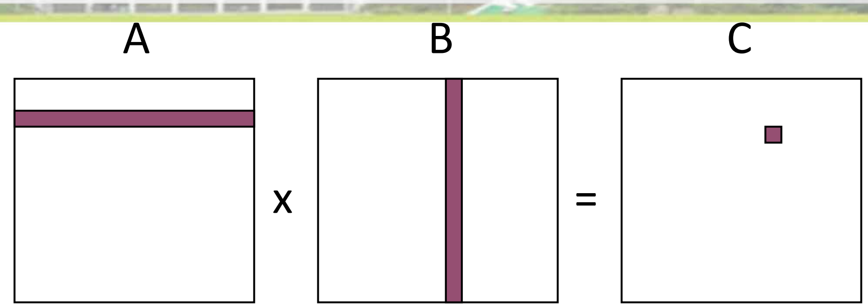
Supercomputers

- Supercomputer: a computer with a high computing performance
- Top500.org
  - measures how fast a computer solves a dense N by N linear equations Ax = b, which is a common task in various domains.
  - HPL (High Performance LINPACK benchmark)

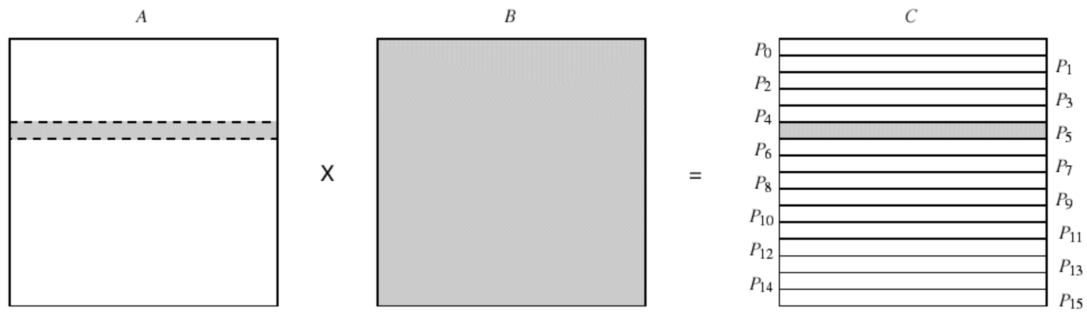


Matrix Multiplication

- A x B = C
- A[i,:] • B[:,j] = C[i,j]

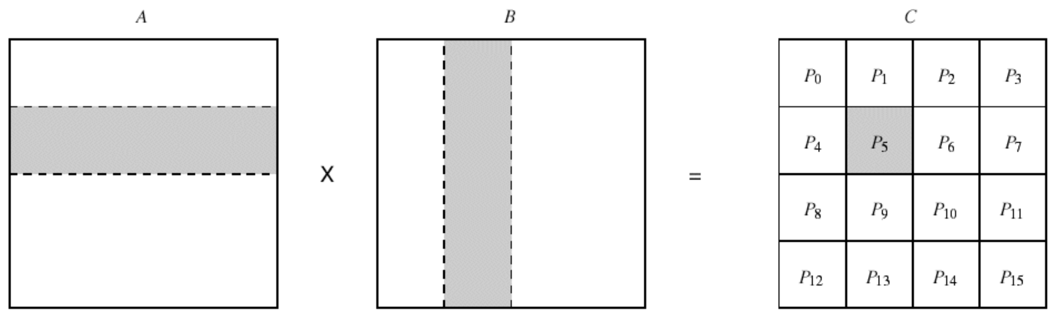


- Row partitioning
  - N tasks



(a)

- Block partitioning
  - N*N/B tasks



- Shading shows data sharing in B matrix

(b)

Matrix Multiplication

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} \times \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \end{pmatrix}$$

m x r r x n m x n

- The number of operations required = O(m x n x r)
- For simplicity, we analyze *square* matrices of order n. So, O(n3)

- Sequential algorithm
  - Square Matrix Multiplication

```
for (i=0;i<n; i++)
{
    for (j=0; j<n; j++)
    {
        c[i][j] = 0;
        for (k=0; k<n; k++)
        {
            c[i][j] += a[i][k] * b[k][j];
        }
    }
}
```

- Assumption:
  - The number of processors available in parallel machine is p
  - The processing nodes are homogeneous
    - Homogeneity make it possible to achieve load balancing
- Parallelization
  - Step 1) Partition the two matrices into p square blocks
  - Step 2) Each square block of A and B are assigned to each process
    - Initial alignment is needed (shown in the next slide)
  - Step 3) p processors process p blocks.
    - Each processor multiplies blocks and add the results to partial results in C.
  - Step 4) The A blocks are rolled one step to the left
and B blocks are rolled one step upward
  - Repeat step 3 and 4 $p^{1/2}$ times

Matrix Multiplication

$$A = \begin{bmatrix} 2 & 1 & 5 & 3 & 7 & 3 & \cdots \\ 0 & 7 & 1 & 6 & 2 & 1 & \cdots \\ 9 & 2 & 4 & 4 & 3 & 2 & \cdots \\ 3 & 6 & 7 & 2 & 5 & 9 & \cdots \\ 1 & 3 & 2 & 0 & 3 & 1 & \cdots \\ 4 & 2 & 0 & 1 & 2 & 0 & \cdots \\ & & & \cdots & & & \end{bmatrix} \qquad B = \begin{bmatrix} 6 & 1 & 2 & 3 & 0 & 2 & \cdots \\ 4 & 5 & 6 & 5 & 2 & 1 & \cdots \\ 1 & 9 & 8 & -8 & 1 & 2 & \cdots \\ 4 & 0 & -8 & 5 & 0 & 8 & \cdots \\ 2 & 3 & 0 & 1 & 1 & 2 & \cdots \\ 0 & 1 & 2 & 3 & 4 & 0 & \cdots \\ & & & \cdots & & & \end{bmatrix}$$

Initial alignment

$$\begin{matrix} \begin{bmatrix} 2 & 1 \\ 0 & 7 \end{bmatrix} & \begin{bmatrix} 5 & 3 \\ 1 & 6 \end{bmatrix} & \begin{bmatrix} 7 & 3 \\ 2 & 1 \end{bmatrix} & \cdots \\ \begin{bmatrix} 4 & 4 \\ 7 & 2 \end{bmatrix} & \begin{bmatrix} 3 & 2 \\ 5 & 9 \end{bmatrix} \cdots & \begin{bmatrix} 9 & 2 \\ 3 & 6 \end{bmatrix} & \leftarrow \\ \begin{bmatrix} 3 & 1 \\ 2 & 0 \end{bmatrix} \cdots & \begin{bmatrix} 1 & 3 \\ 4 & 2 \end{bmatrix} & \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} & \leftarrow \leftarrow \\ \cdots & & & \end{matrix} \qquad \begin{matrix} \begin{bmatrix} 6 & 1 \\ 4 & 5 \end{bmatrix} & \begin{bmatrix} 8 & -8 \\ -8 & 5 \end{bmatrix} & \begin{bmatrix} 1 & 2 \\ 4 & 0 \end{bmatrix} \\ & & \cdots \\ \begin{bmatrix} 1 & 9 \\ 4 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} & \begin{bmatrix} 0 & 2 \\ 2 & 1 \end{bmatrix} \\ & \cdots & \\ \begin{bmatrix} 2 & 3 \\ 0 & 1 \end{bmatrix} & \begin{bmatrix} 2 & 3 \\ 6 & 5 \end{bmatrix} & \begin{bmatrix} 1 & 2 \\ 0 & 8 \end{bmatrix} \\ \cdots & \uparrow & \uparrow\uparrow \end{matrix}$$

Matrix Multiplication

- 2x2 blocks

$$A = \begin{pmatrix} 2 & 1 & 5 & 3 \\ 0 & 7 & 1 & 6 \\ 9 & 2 & 4 & 4 \\ 3 & 6 & 7 & 2 \end{pmatrix} \qquad B = \begin{pmatrix} 6 & 1 & 2 & 3 \\ 4 & 5 & 6 & 5 \\ 1 & 9 & 8 & -8 \\ 4 & 0 & -8 & 5 \end{pmatrix}$$

$$\begin{matrix} \begin{pmatrix} 2 & 1 \\ 0 & 7 \end{pmatrix} & \begin{pmatrix} 5 & 3 \\ 1 & 6 \end{pmatrix} \\ \begin{pmatrix} 4 & 4 \\ 7 & 2 \end{pmatrix} & \begin{pmatrix} 9 & 2 \\ 3 & 6 \end{pmatrix} \end{matrix} \qquad \begin{matrix} \begin{pmatrix} 6 & 1 \\ 4 & 5 \end{pmatrix} & \begin{pmatrix} 8 & -8 \\ -8 & 5 \end{pmatrix} \\ \begin{pmatrix} 1 & 9 \\ 4 & 0 \end{pmatrix} & \begin{pmatrix} 2 & 3 \\ 6 & 5 \end{pmatrix} \end{matrix}$$

Initial alignment

Matrix Multiplication

$$A = \begin{pmatrix} 2 & 1 & 5 & 3 \\ 0 & 7 & 1 & 6 \\ 9 & 2 & 4 & 4 \\ 3 & 6 & 7 & 2 \end{pmatrix} \qquad B = \begin{pmatrix} 6 & 1 & 2 & 3 \\ 4 & 5 & 6 & 5 \\ 1 & 9 & 8 & -8 \\ 4 & 0 & -8 & 5 \end{pmatrix}$$

$P_{0,0}$

$$\begin{pmatrix} 2 & 1 \\ 0 & 7 \end{pmatrix} \times \begin{pmatrix} 6 & 1 \\ 4 & 5 \end{pmatrix}$$

$P_{0,1}$

$$\begin{pmatrix} 5 & 3 \\ 1 & 6 \end{pmatrix} \times \begin{pmatrix} 8 & -8 \\ -8 & 5 \end{pmatrix}$$

$P_{1,0}$

$$\begin{pmatrix} 4 & 4 \\ 7 & 2 \end{pmatrix} \times \begin{pmatrix} 1 & 9 \\ 4 & 0 \end{pmatrix}$$

$P_{1,1}$

$$\begin{pmatrix} 9 & 2 \\ 5 & 3 \end{pmatrix} \times \begin{pmatrix} 2 & 3 \\ 6 & 5 \end{pmatrix}$$

Matrix Multiplication

$$A = \begin{pmatrix} 2 & 1 & 5 & 3 \\ 0 & 7 & 1 & 6 \\ 9 & 2 & 4 & 4 \\ 3 & 6 & 7 & 2 \end{pmatrix} \qquad B = \begin{pmatrix} 6 & 1 & 2 & 3 \\ 4 & 5 & 6 & 5 \\ 1 & 9 & 8 & -8 \\ 4 & 0 & -8 & 5 \end{pmatrix}$$

$P_{0,0}$

$$\begin{matrix} 2 & 1 \\ 0 & 7 \end{matrix} \times \begin{matrix} 6 & 1 \\ 4 & 5 \end{matrix} \quad \begin{matrix} 16 & 7 \\ 28 & 35 \end{matrix}$$

$P_{0,1}$

$$\begin{matrix} 5 & 3 \\ 1 & 6 \end{matrix} \times \begin{matrix} 8 & -8 \\ -8 & 5 \end{matrix} \quad \begin{matrix} 16 & -25 \\ -40 & 22 \end{matrix}$$

$P_{1,0}$

$$\begin{matrix} 4 & 4 \\ 7 & 2 \end{matrix} \times \begin{matrix} 1 & 9 \\ 4 & 0 \end{matrix} \quad \begin{matrix} 20 & 36 \\ 15 & 63 \end{matrix}$$

$P_{1,1}$

$$\begin{matrix} 9 & 2 \\ 5 & 3 \end{matrix} \times \begin{matrix} 2 & 3 \\ 6 & 5 \end{matrix} \quad \begin{matrix} 30 & 37 \\ 42 & 39 \end{matrix}$$

Matrix Multiplication

- Shift A one step to left
- Shift B one step up

$P_{0,0}$

$$\begin{bmatrix} 2 & 1 \\ 0 & 7 \end{bmatrix} \times \begin{bmatrix} 6 & 1 \\ 4 & 5 \end{bmatrix} \quad \begin{bmatrix} 16 & 7 \\ 28 & 35 \end{bmatrix}$$

$P_{0,1}$

$$\begin{bmatrix} 5 & 3 \\ 1 & 6 \end{bmatrix} \times \begin{bmatrix} 8 & -8 \\ -8 & 5 \end{bmatrix} \quad \begin{bmatrix} 16 & -25 \\ -40 & 22 \end{bmatrix}$$

$P_{1,0}$

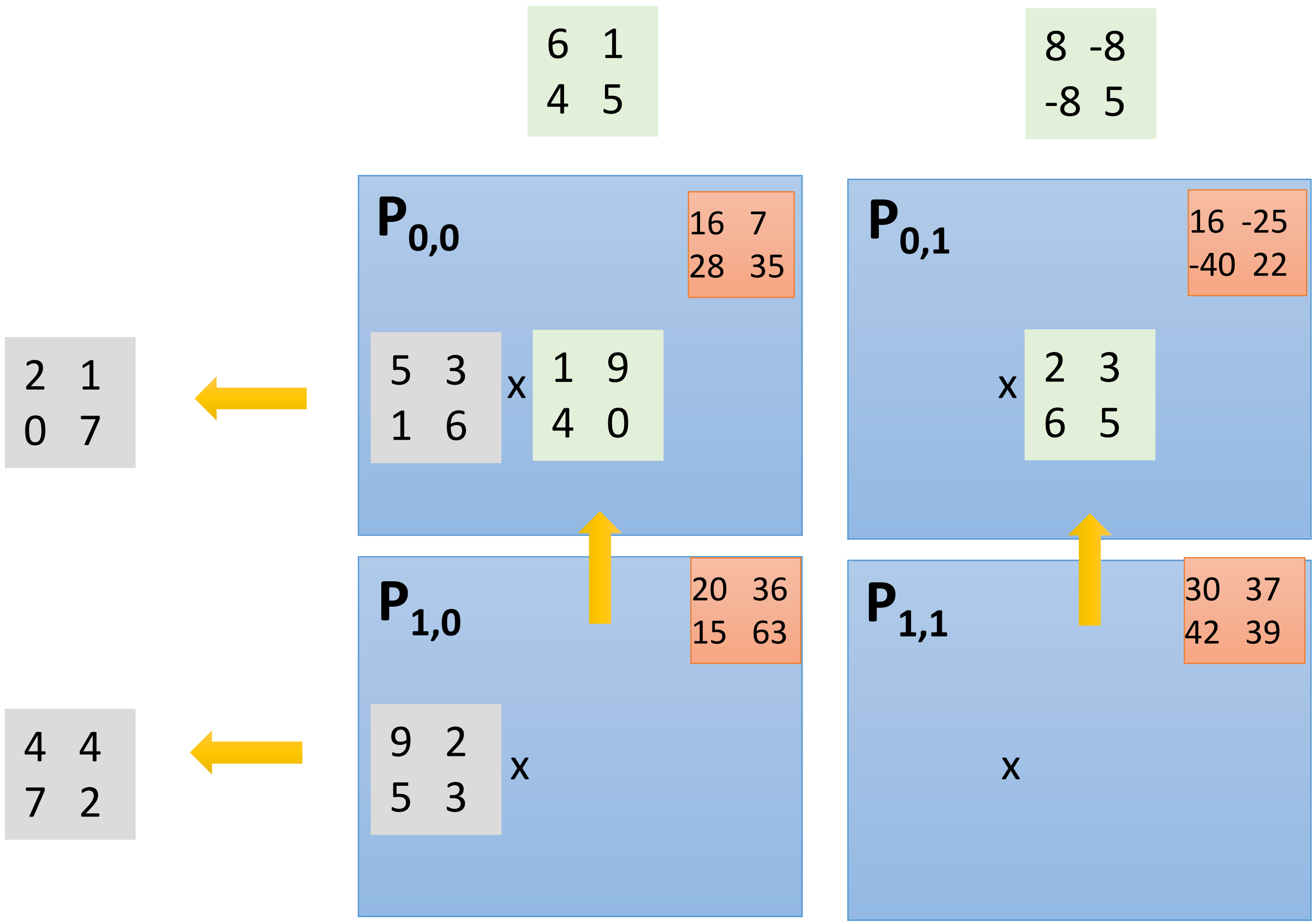
$$\begin{bmatrix} 4 & 4 \\ 7 & 2 \end{bmatrix} \times \begin{bmatrix} 1 & 9 \\ 4 & 0 \end{bmatrix} \quad \begin{bmatrix} 20 & 36 \\ 15 & 63 \end{bmatrix}$$

$P_{1,1}$

$$\begin{bmatrix} 9 & 2 \\ 5 & 3 \end{bmatrix} \times \begin{bmatrix} 2 & 3 \\ 6 & 5 \end{bmatrix} \quad \begin{bmatrix} 30 & 37 \\ 42 & 39 \end{bmatrix}$$

Matrix Multiplication

- Shift A one step to left
- Shift B one step up



Matrix Multiplication

- Shift A one step to left
- Shift B one step up

$P_{0,0}$

$\begin{matrix} 16 & 7 \\ 28 & 35 \end{matrix}$

$\begin{matrix} 5 & 3 \\ 1 & 6 \end{matrix} \times \begin{matrix} 1 & 9 \\ 4 & 0 \end{matrix} = \begin{matrix} 17 & 45 \\ 25 & 9 \end{matrix}$

$P_{0,1}$

$\begin{matrix} 16 & -25 \\ -40 & 22 \end{matrix}$

$\begin{matrix} 2 & 1 \\ 0 & 7 \end{matrix} \times \begin{matrix} 2 & 3 \\ 6 & 5 \end{matrix} = \begin{matrix} 10 & 11 \\ 42 & 35 \end{matrix}$

$P_{1,0}$

$\begin{matrix} 20 & 36 \\ 15 & 63 \end{matrix}$

$\begin{matrix} 9 & 2 \\ 5 & 3 \end{matrix} \times \begin{matrix} 6 & 1 \\ 4 & 5 \end{matrix} = \begin{matrix} 62 & 19 \\ 42 & 33 \end{matrix}$

$P_{1,1}$

$\begin{matrix} 30 & 37 \\ 42 & 39 \end{matrix}$

$\begin{matrix} 4 & 4 \\ 7 & 2 \end{matrix} \times \begin{matrix} 8 & -8 \\ -8 & 5 \end{matrix} = \begin{matrix} 0 & -12 \\ 40 & -46 \end{matrix}$

Matrix Multiplication

- Done

$P_{0,0}$

$$\begin{matrix}16 & 7\\ 28 & 35\end{matrix} + \begin{matrix}17 & 45\\ 25 & 9\end{matrix} = \begin{matrix}33 & 52\\ 53 & 44\end{matrix}$$

$P_{0,1}$

$$\begin{matrix}16 & -25\\ -40 & 22\end{matrix} + \begin{matrix}10 & 11\\ 42 & 35\end{matrix} = \begin{matrix}26 & -14\\ 2 & 57\end{matrix}$$

$P_{1,0}$

$$\begin{matrix}20 & 36\\ 15 & 63\end{matrix} + \begin{matrix}62 & 19\\ 42 & 33\end{matrix} = \begin{matrix}82 & 55\\ 57 & 96\end{matrix}$$

$P_{1,1}$

$$\begin{matrix}30 & 37\\ 42 & 39\end{matrix} + \begin{matrix}0 & -12\\ 40 & -46\end{matrix} = \begin{matrix}30 & 25\\ 82 & -7\end{matrix}$$

HPL (High-Performance Linpack Benchmark)

- HPL is a parallel, blocked, LU decomposition solver.
- Suppose you want to solve these equations

$$5x + 3y - 2z = 23$$
$$7x + 9y + 3z = 102$$
$$8x + 8y - 8z = 8$$

You could put them into the form:

$$Ax = b$$

$$\begin{bmatrix} 5 & 3 & -2 \\ 7 & 9 & 3 \\ 8 & 8 & -8 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 23 \\ 102 \\ 8 \end{bmatrix}$$

Taking advantage of Properties of A

$$Ax = b$$

- A is a given n x n matrix
- b is a given n-vector
- x is unknown solution n-vector

- Properties of A can make this equation easier to solve.
  - If A = lower triangular matrix → forward substitution.
  - If A = upper triangular matrix → backward substitution.
- How to change the matrix to upper or lower triangular?

- $Ax=b$

$$
\begin{array}{lllll}
a_{0,0}x_0 & + a_{0,1}x_1 & + \ldots + & a_{0,n-1}x_{n-1} & = b_0 \\
a_{1,0}x_0 & + a_{1,1}x_1 & + \ldots + & a_{1,n-1}x_{n-1} & = b_1 \\
\ldots & & & & \\
A_{n-1,0}x_0 & + a_{n-1,1}x_1 & + \ldots + & a_{n-1,n-1}x_{n-1} & = b_{n-1}
\end{array}
$$

- Gaussian elimination (classic algorithm)
  - Forward elimination to $Ux=y$ (*U* is upper triangular)
    - without or with partial pivoting
  - Back substitution to solve for *x*
  - Parallel algorithms based on partitioning of *A*

Gaussian Elimination for a System of Linear Equations

$$\left[\begin{array}{ccc|c} 1 & 3 & 1 & 9 \\ 1 & 1 & -1 & 1 \\ 3 & 11 & 5 & 35 \end{array}\right] \to \left[\begin{array}{ccc|c} 1 & 3 & 1 & 9 \\ 0 & -2 & -2 & -8 \\ 0 & 2 & 2 & 8 \end{array}\right] \to \left[\begin{array}{ccc|c} 1 & 3 & 1 & 9 \\ 0 & -2 & -2 & -8 \\ 0 & 0 & 0 & 0 \end{array}\right] \to \left[\begin{array}{ccc|c} 1 & 0 & -2 & -3 \\ 0 & 1 & 1 & 4 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

- Use Elementary Row Operations
  - Swapping two rows,
  - Multiplying a row by a nonzero number,
  - Adding a multiple of one row to another row.
- Forward Elimination: Upper Triangular Matrix
  - row1 → row2 → row3
  - Then, Backward Substitution (x3 → x2 → x1)

Inverse Matrix using Gaussian Elimination

$$
\left(\; I \quad | \quad M \; \right) \rightarrow \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & -1 & 1 & 2 \\ 0 & 1 & 0 & 3 & -1 & 1 \\ 0 & 0 & 1 & -1 & 3 & 4 \end{array}\right)
$$

$$
\rightarrow \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & -1 & 1 & 2 \\ 3 & 1 & 0 & 0 & 2 & 7 \\ -1 & 0 & 1 & 0 & 2 & 2 \end{array}\right)
$$

$$
\rightarrow \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & -1 & 1 & 2 \\ 3 & 1 & 0 & 0 & 2 & 7 \\ -4 & -1 & 1 & 0 & 0 & -5 \end{array}\right)
$$

$$
\rightarrow \left(\begin{array}{ccc|ccc} -1 & 0 & 0 & 1 & -1 & -2 \\ 1.5 & 0.5 & 0 & 0 & 1 & 3.5 \\ 0.8 & 0.2 & -0.2 & 0 & 0 & 1 \end{array}\right)
$$

$$
\rightarrow \left(\begin{array}{ccc|ccc} 0.6 & 0.4 & -0.4 & 1 & -1 & 0 \\ -1.3 & -0.2 & 0.7 & 0 & 1 & 0 \\ 0.8 & 0.2 & -0.2 & 0 & 0 & 1 \end{array}\right)
$$

$$
\rightarrow \left(\begin{array}{ccc|ccc} -0.7 & 0.2 & 0.3 & 1 & 0 & 0 \\ -1.3 & -0.2 & 0.7 & 0 & 1 & 0 \\ 0.8 & 0.2 & -0.2 & 0 & 0 & 1 \end{array}\right)
$$

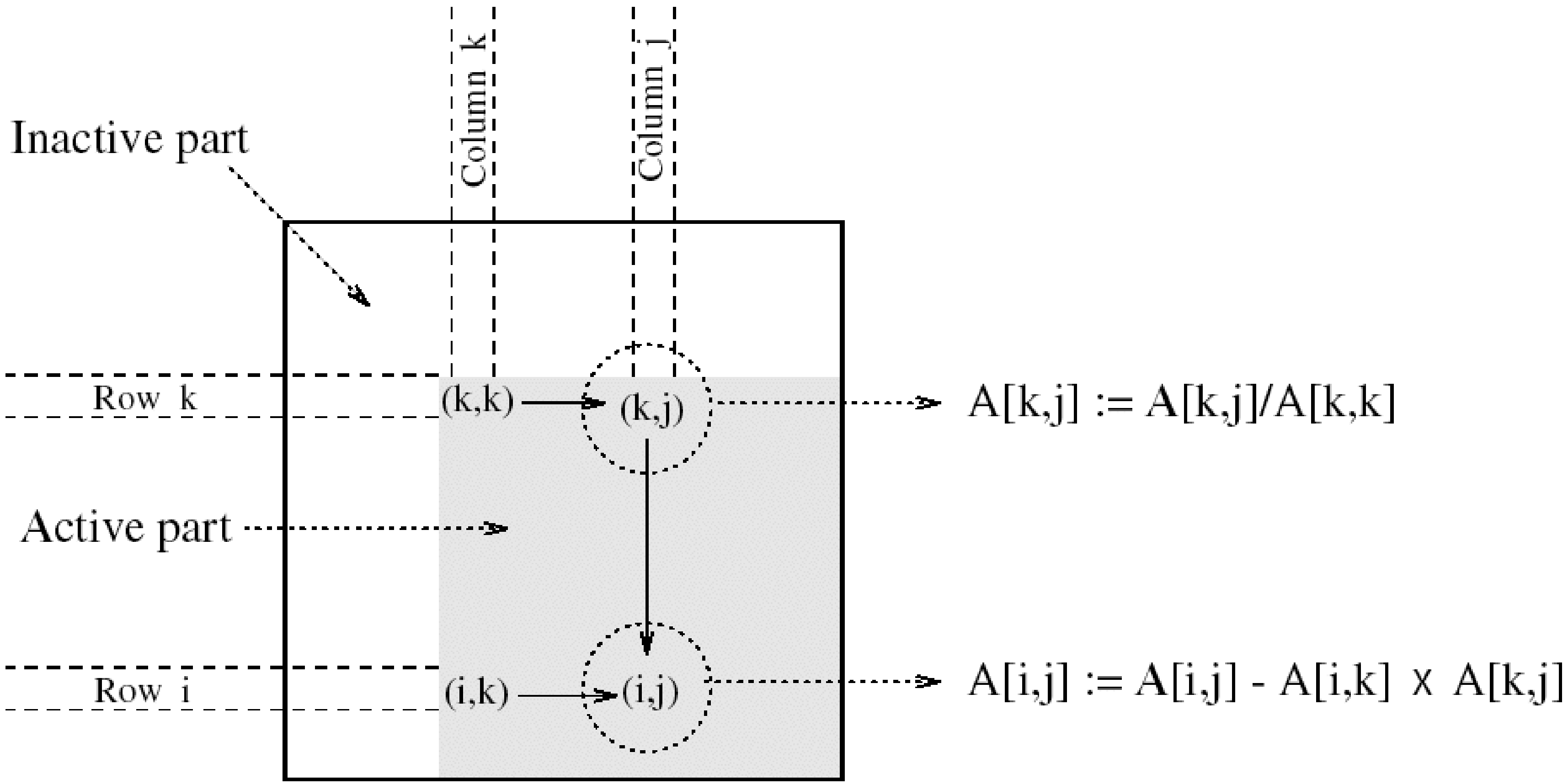
Sequential Gaussian Elimination

```
procedure GAUSSIAN ELIMINATION (A, b, y)
begin
   for k := 0 to n - 1 do /* Outer loop */
   begin
         for j := k + 1 to n - 1 do
               A[k, j] := A[k, j]/A[k, k];
         y[k] := b[k]/A[k, k];
         A[k, k] := 1;
         for i := k + 1 to n - 1 do
         begin
             for j := k + 1 to n - 1 do
               A[i, j] := A[i, j] - A[i, k] x A[k, j ];
             b[i] := b[i] - A[i, k] x y[k];
             A[i, k] := 0;
         endfor;
   endfor;
end GAUSSIAN ELIMINATION
```

- Gaussian elimination : triple-nested loop
- ```
  for k
      for i
          for j
              a_ij = a_ij - (a_ik/a_kk) a_kj
          end
      end
  end
  ```

$$a_{ij} = a_{ij} - (a_{ik}/a_{kk})\, a_{kj}$$
  ```

Computation Step in Gaussian Elimination



$$5x + 3y = 22$$
$$8x + 2y = 13$$

$$x = (22 - 3y) / 5$$
$$8(22 - 3y)/5 + 2y = 13$$

$$x = (22 - 3y) / 5$$
$$y = (13 - 176/5) / (24/5 + 2)$$

Row-wise Partitioning on Eight Processes

P_0	1	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)	(0,7)
P_1	0	1	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)
P_2	0	0	1	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)
P_3	0	0	0	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)
P_4	0	0	0	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)
P_5	0	0	0	(5,3)	(5,4)	(5,5)	(5,6)	(5,7)
P_6	0	0	0	(6,3)	(6,4)	(6,5)	(6,6)	(6,7)
P_7	0	0	0	(7,3)	(7,4)	(7,5)	(7,6)	(7,7)

(a) Computation:

(i) A[k,j] := A[k,j]/A[k,k]

(ii) A[k,k] := 1

P_0	1	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)	(0,7)
P_1	0	1	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)
P_2	0	0	1	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)
P_3	0	0	0	1	(3,4)	(3,5)	(3,6)	(3,7)
P_4	0	0	0	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)
P_5	0	0	0	(5,3)	(5,4)	(5,5)	(5,6)	(5,7)
P_6	0	0	0	(6,3)	(6,4)	(6,5)	(6,6)	(6,7)
P_7	0	0	0	(7,3)	(7,4)	(7,5)	(7,6)	(7,7)

(b) Communication:

One–to–all broadcast of row A[k,*]

Row-wise Partitioning on Eight Processes

P_0	1	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)	(0,7)
P_1	0	1	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)
P_2	0	0	1	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)
P_3	0	0	0	1	(3,4)	(3,5)	(3,6)	(3,7)
P_4	0	0	0	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)
P_5	0	0	0	(5,3)	(5,4)	(5,5)	(5,6)	(5,7)
P_6	0	0	0	(6,3)	(6,4)	(6,5)	(6,6)	(6,7)
P_7	0	0	0	(7,3)	(7,4)	(7,5)	(7,6)	(7,7)

(c) Computation:

(i) $A[i,j] := A[i,j] - A[i,k] \times A[k,j]$
for $k < i < n$ and $k < j < n$

(ii) $A[i,k] := 0$ for $k < i < n$

- P4 ~ P7 : parallel computation

2D Partitioning on 64 Processes

1	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)	(0,7)
0	1	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)
0	0	1	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)
0	0	0	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)
0	0	0	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)
0	0	0	(5,3)	(5,4)	(5,5)	(5,6)	(5,7)
0	0	0	(6,3)	(6,4)	(6,5)	(6,6)	(6,7)
0	0	0	(7,3)	(7,4)	(7,5)	(7,6)	(7,7)

(a) Rowwise broadcast of A[i,k]
for (k - 1) < i < n

1	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)	(0,7)
0	1	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)
0	0	1	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)
0	0	0	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)
0	0	0	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)
0	0	0	(5,3)	(5,4)	(5,5)	(5,6)	(5,7)
0	0	0	(6,3)	(6,4)	(6,5)	(6,6)	(6,7)
0	0	0	(7,3)	(7,4)	(7,5)	(7,6)	(7,7)

(b) A[k,j] := A[k,j]/A[k,k]
for k < j < n

1	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)	(0,7)
0	1	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)
0	0	1	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)
0	0	0	1	(3,4)	(3,5)	(3,6)	(3,7)
0	0	0	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)
0	0	0	(5,3)	(5,4)	(5,5)	(5,6)	(5,7)
0	0	0	(6,3)	(6,4)	(6,5)	(6,6)	(6,7)
0	0	0	(7,3)	(7,4)	(7,5)	(7,6)	(7,7)

(c) Columnwise broadcast of A[k,j]
for k < j < n

1	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)	(0,7)
0	1	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)
0	0	1	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)
0	0	0	1	(3,4)	(3,5)	(3,6)	(3,7)
0	0	0	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)
0	0	0	(5,3)	(5,4)	(5,5)	(5,6)	(5,7)
0	0	0	(6,3)	(6,4)	(6,5)	(6,6)	(6,7)
0	0	0	(7,3)	(7,4)	(7,5)	(7,6)	(7,7)

(d) A[i,j] := A[i,j]-A[i,k] x A[k,j]
for k < i < n and k < j < n

Back Substitution to Find Solution

```
procedure BACK SUBSTITUTION (U, x, y)
begin
    for k := n - 1 downto 0 do /* Main loop */
    begin
            x[k] := y[k];
            for i := k - 1 downto 0 do
                    y[i] := y[i] - x[k] * U[i, k];
    endfor;
end BACK SUBSTITUTION
```

Drawback of Gaussian Elimination

- Drawback of Gaussian Elimination → lots of computations
  - $n^3/3$ additions and multiplications
  - $n^2/2$ divisions.
  - Equations, especially {b}, have to be changed in each step
  - What if we want to solve the equation for a different b?
  - Can we do better?