UYBHM Yaz Çalıştayı
15 – 26 Haziran 2009

ULUSAL YÜKSEK BAŞARIMLI HESAPLAMA MERKEZİ
İSTANBUL TEKNİK ÜNİVERSİTESİ

Parallel Programming Application: Matrix Multiplication

(senol@be.itu.edu.tr)

Outline

- Matrix Multiplication (BLAS3 Operation)
- Cannon Algorithm
- Fox Algorithm

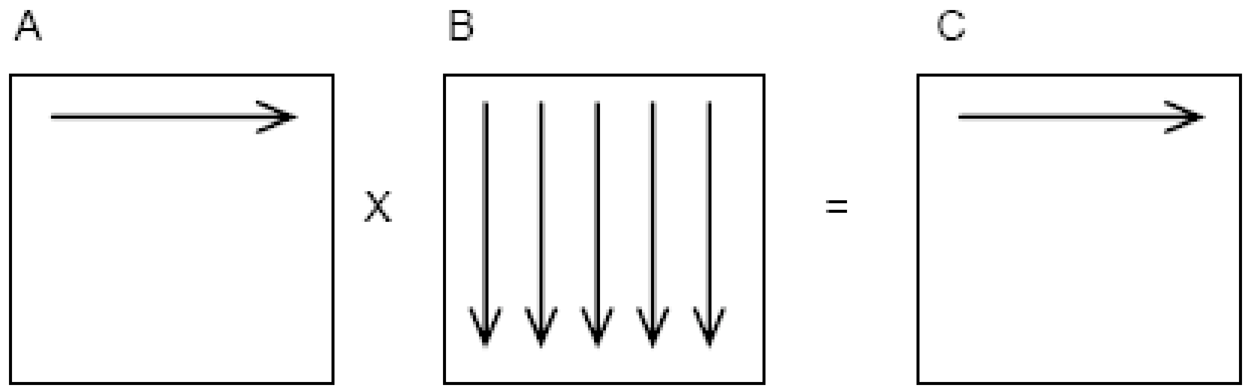
Matrix Multiplication (BLAS3)

A and B nxn matrices => C=AxB

```
void Serial_mat_mult(mt_A,mt_B,mt_C,int n)
{
  int i,j,k;
for (i=0;i<n;i++){
    for (j=0;j<n;j++){
       C[i][j]=0.0;
       for (k=0;k<n;k++)
           C[i][j]= C[i][j]+A[i][k]*B[k][j];
   }
}
```

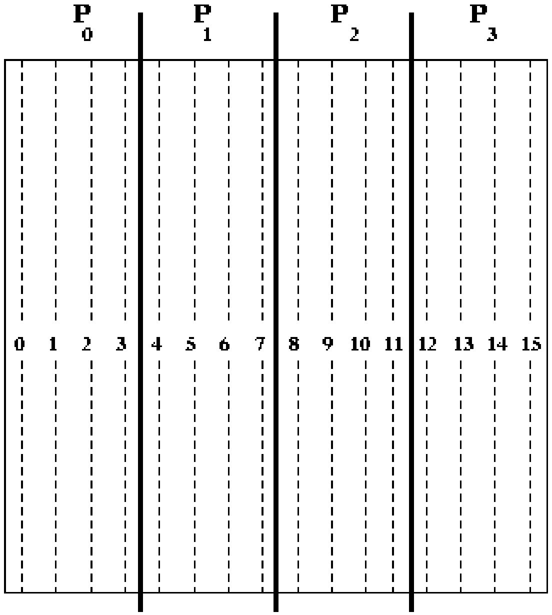
- This algorithm requires n^3 multiplications and n^3 additions, leading to a sequential time complexity of $O(n^3)$

Serial Matrix Multiplication

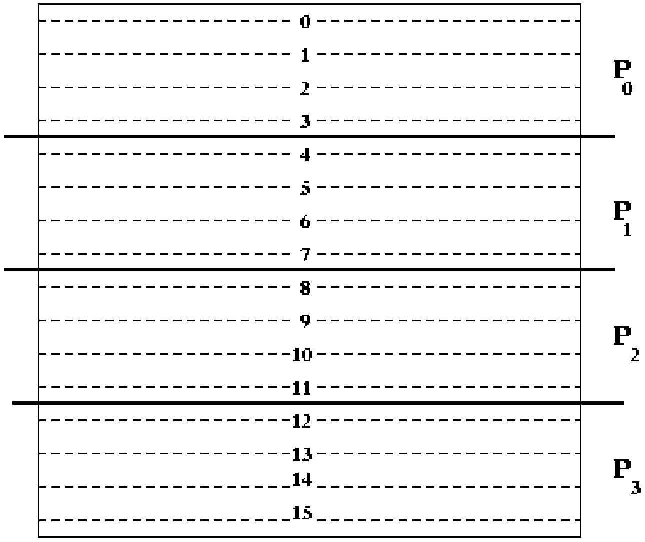


- During the first iteration of loop variable *i* the first matrix *A* row and all the columns of matrix *B* are used to compute the elements of the first result matrix *C* row

Partitioning Matrices:Block Stripping



(a) Columnwise block stripping



(b) Rowwise block stripping

- Uniform block-striped partitioning of 16 x 16 matrix on 4 processors

Partitioning Matrices: Checkerboard

(0, 0) (0, 1) P_{00} (1, 0) (1, 1)	(0, 2) (0, 3) P_{01} (1, 2) (1, 3)	(0, 4) (0, 5) P_{02} (1, 4) (1, 5)	(0, 6) (0, 7) P_{03} (1, 6) (1, 7)
(2, 0) (2, 1) P_{10} (3, 0) (3, 1)	(2, 2) (2, 3) P_{11} (3, 2) (3, 3)	(2, 4) (2, 5) P_{12} (3, 4) (3, 5)	(2, 6) (2, 7) P_{13} (3, 6) (3, 7)
(4, 0) (4, 1) P_{20} (5, 0) (5, 1)	(4, 2) (4, 3) P_{21} (5, 2) (5, 3)	(4, 4) (4, 5) P_{22} (5, 4) (5, 5)	(4, 6) (4, 7) P_{23} (5, 6) (5, 7)
(6, 0) (6, 1) P_{30} (7, 0) (7, 1)	(6, 2) (6, 3) P_{31} (7, 2) (7, 3)	(6, 4) (6, 5) P_{32} (7, 4) (7, 5)	(6, 6) (6, 7) P_{33} (7, 6) (7, 7)

(a) Block-checkboard mapping

(0, 0) (0, 4) P_{00} (4, 0) (4, 4)	(0, 1) (0, 5) P_{01} (4, 1) (4, 5)	(0, 2) (0, 6) P_{02} (4, 2) (4, 6)	(0, 3) (0, 7) P_{03} (4, 3) (4, 7)
(1, 0) (1, 4) P_{10} (5, 0) (5, 4)	(1, 1) (1, 5) P_{11} (5, 1) (5, 5)	(1, 2) (1, 6) P_{12} (5, 2) (5, 6)	(1, 3) (1, 7) P_{13} (5, 3) (5, 7)
(2, 0) (2, 4) P_{20} (6, 0) (6, 4)	(2, 1) (2, 5) P_{21} (6, 1) (6, 5)	(2, 2) (2, 6) P_{22} (6, 2) (6, 6)	(2, 3) (2, 7) P_{23} (6, 3) (6, 7)
(3, 0) (3, 4) P_{30} (7, 0) (7, 4)	(3, 1) (3, 5) P_{31} (7, 1) (7, 5)	(3, 2) (3, 6) P_{32} (7, 2) (7, 6)	(3, 3) (3, 7) P_{33} (7, 3) (7, 7)

(b) Cyclic-checkboard partitioning

Parallel Matrix Multiplication

– **Partitioning into Submatrices**

- Suppose the matrix is divided into s^2 submatrices. Each submatrix has n/s x n/s elements. Using the notation Ap,q as the submatrix in submatrix row p and submatrix column q:

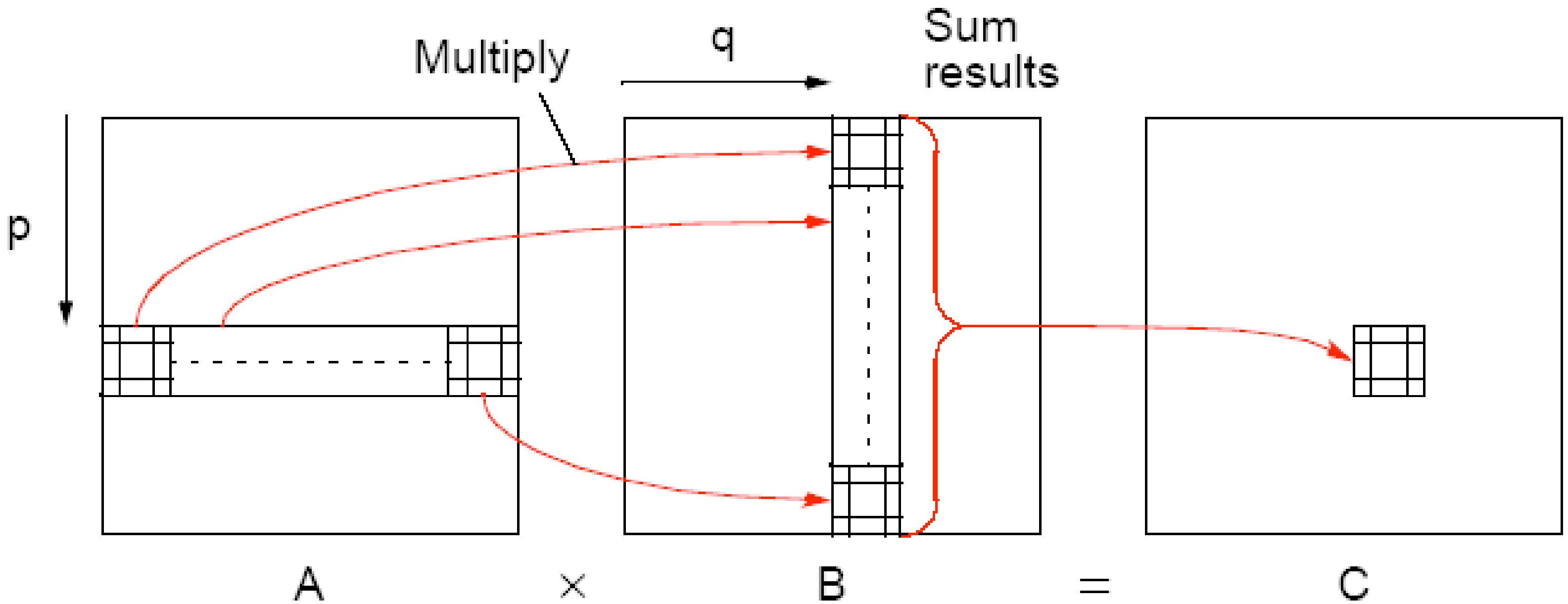
```
for (p = 0; p < s; p++)
    for (q = 0; q < s; q++) {
        Cp,q = 0; /* clear elements of submatrix */
        for (r = 0; r < m; r++) /* submatrix multiplication and */
            Cp,q = Cp,q + Ap,r * Br,q;/* add to accumulating
          submatrix */
    }
```

- The line

  ```
  Cp,q = Cp,q + Ap,r * Br,q;
  ```

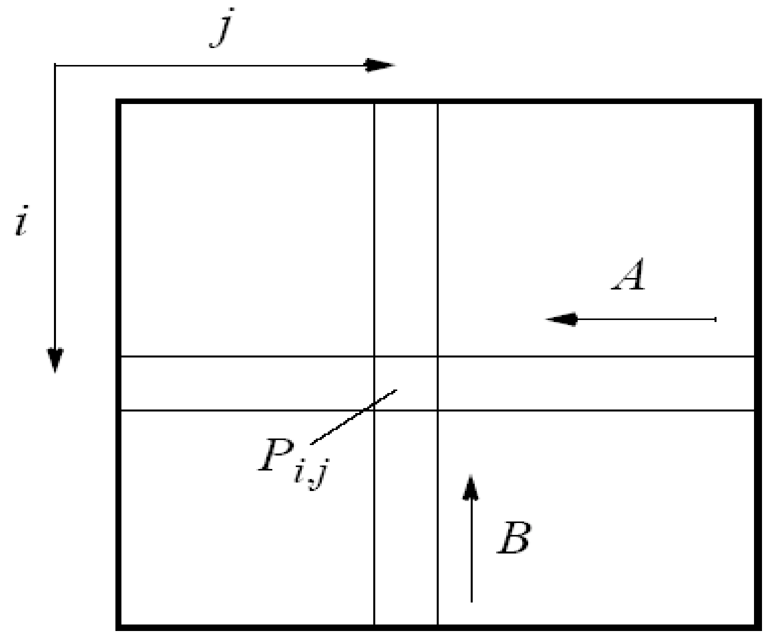
  means multiply submatrix Ap,r and Br,q using matrix multiplication and add to submatrix Cp,q using matrix addition.

Algorithm



Cannon Algorithm

1. Initially processor $P_{i,j}$ has elements $a_{i,j}$ and $b_{i,j}$ $(0 \leq i < n,\ 0 \leq k < n)$.

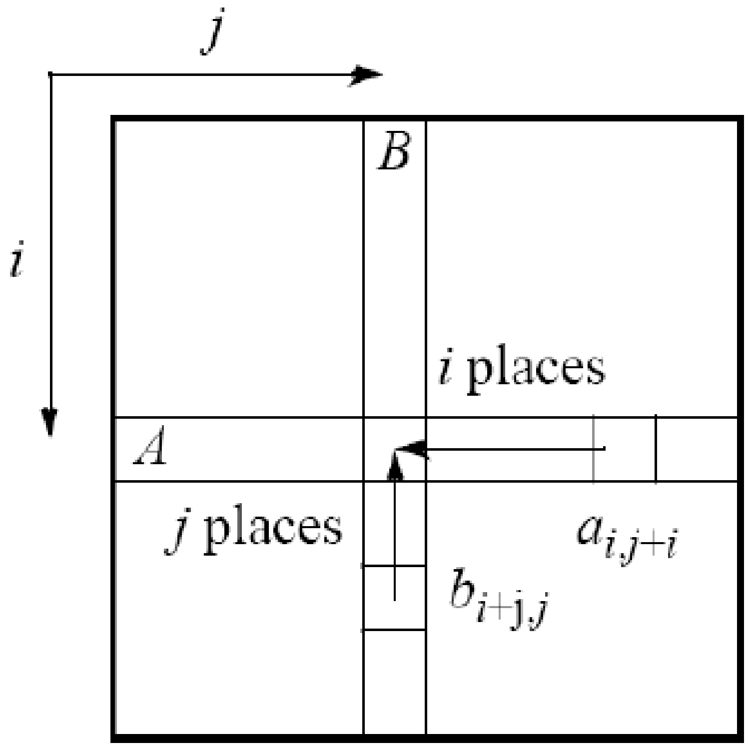


Cannon Algorithm

2. Elements are moved from their initial position to an "alligned" position.

The complete *i*th row of A is shifted *i* places left and the complete *j*th column of B is shifted *j* places upward.

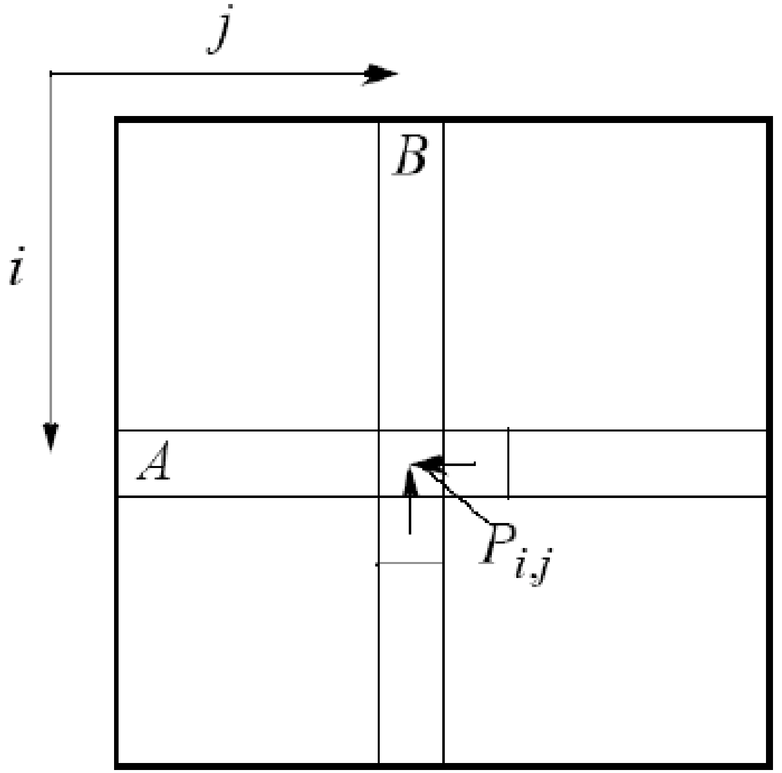
Cannon Algorithm



Cannon Algorithm

3. Each processor $P_{i,j}$ muliply its elements.
4. The *i*th row of A is shifted one place right, and the *j*th column of B is shifted one place upward.

Cannon Algorithm



Cannon Algorithm

5. Each processor $P_{i,j}$ muliplies its elements brought to it and adds the results to the accumulating sum.

6. Step 4 and 5 are repeated until the final result is obtained ($n-1$ shifts with n rows and n columns of elements).

Cannon Algorithm

- Initially the matrix A:

- Row0 is unchanged.
- Row1 is shifted 1 place left.
- Row2 is shifted 2 places left.
- Row3 is shifted 3 places left.

- Initially the matrix B:

- Column 0 is unchanged.
- Column1 is shifted 1 place up.
- Column2 is shifted 2 places up.
- Column3 is shifted 3 places up.

Cannon Algorithm

$A_{0,0}$ $B_{0,0}$	$A_{0,1}$ $B_{0,1}$	$A_{0,2}$ $B_{0,2}$	$A_{0,3}$ $B_{0,3}$
$A_{1,0}$ $B_{1,0}$	$A_{1,1}$ $B_{1,1}$	$A_{1,2}$ $B_{1,2}$	$A_{1,3}$ $B_{1,3}$
$A_{2,0}$ $B_{2,0}$	$A_{2,1}$ $B_{2,1}$	$A_{2,2}$ $B_{2,2}$	$A_{2,3}$ $B_{2,3}$
$A_{3,0}$ $B_{3,0}$	$A_{3,1}$ $B_{3,1}$	$A_{3,2}$ $B_{3,2}$	$A_{3,3}$ $B_{3,3}$

$A_{0,0}$ $B_{0,0}$	$A_{0,1}$ $B_{1,1}$	$A_{0,2}$ $B_{2,2}$	$A_{0,3}$ $B_{3,3}$
$A_{1,1}$ $B_{1,0}$	$A_{1,2}$ $B_{2,1}$	$A_{1,3}$ $B_{3,2}$	$A_{1,0}$ $B_{0,3}$
$A_{2,2}$ $B_{2,0}$	$A_{2,3}$ $B_{3,1}$	$A_{2,0}$ $B_{0,2}$	$A_{2,1}$ $B_{1,3}$
$A_{3,3}$ $B_{3,0}$	$A_{3,0}$ $B_{0,1}$	$A_{3,1}$ $B_{1,2}$	$A_{3,2}$ $B_{2,3}$

Cannon Algorithm

$$\underbrace{\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix}}_{A} \times \underbrace{\begin{bmatrix} b_{00} & b_{01} & b_{02} & b_{03} \\ b_{10} & b_{11} & b_{12} & b_{13} \\ b_{20} & b_{21} & b_{22} & b_{23} \\ b_{30} & b_{31} & b_{32} & b_{33} \end{bmatrix}}_{B} =$$

$$\underbrace{\begin{bmatrix} c_{00} & c_{01} & c_{02} & c_{03} \\ c_{10} & c_{11} & c_{12} & c_{13} \\ c_{20} & c_{21} & c_{22} & c_{23} \\ c_{30} & c_{31} & c_{32} & c_{33} \end{bmatrix}}_{C}$$

Cannon Algorithm (0)

a_{00} b_{00}	a_{01} b_{11}	a_{02} b_{22}	a_{03} b_{33}
a_{11} b_{10}	a_{12} b_{21}	a_{13} b_{32}	a_{10} b_{03}
a_{22} b_{20}	a_{23} b_{31}	a_{20} b_{02}	a_{21} b_{13}
a_{33} b_{30}	a_{30} b_{01}	a_{31} b_{12}	a_{32} b_{23}

$$c_{00} = a_{00} \quad b_{00} \qquad c_{10} = a_{11} \quad b_{10}$$
$$c_{01} = a_{01} \quad b_{11} \qquad c_{11} = a_{12} \quad b_{21}$$
$$c_{02} = a_{02} \quad b_{22} \qquad c_{12} = a_{13} \quad b_{32}$$
$$c_{03} = a_{03} \quad b_{33} \qquad c_{13} = a_{10} \quad b_{03}$$

$$c_{20} = a_{22} \quad b_{20} \qquad c_{30} = a_{33} \quad b_{30}$$
$$c_{21} = a_{23} \quad b_{31} \qquad c_{31} = a_{30} \quad b_{01}$$
$$c_{22} = a_{20} \quad b_{02} \qquad c_{32} = a_{31} \quad b_{12}$$
$$c_{23} = a_{21} \quad b_{13} \qquad c_{33} = a_{32} \quad b_{23}$$

Cannon Algorithm (1)

a_{01} b_{10}	a_{02} b_{21}	a_{03} b_{32}	a_{00} b_{03}
a_{12} b_{20}	a_{13} b_{31}	a_{10} b_{02}	a_{11} b_{13}
a_{23} b_{30}	a_{20} b_{01}	a_{21} b_{12}	a_{22} b_{23}
a_{30} b_{00}	a_{31} b_{11}	a_{32} b_{22}	a_{33} b_{33}

$$c_{00} += a_{01}\ b_{10} \qquad c_{10} += a_{12}\ b_{20}$$
$$c_{01} += a_{02}\ b_{21} \qquad c_{11} += a_{13}\ b_{31}$$
$$c_{02} += a_{03}\ b_{32} \qquad c_{12} += a_{10}\ b_{02}$$
$$c_{03} += a_{00}\ b_{03} \qquad c_{13} += a_{11}\ b_{13}$$

$$c_{20} += a_{23}\ b_{30} \qquad c_{30} += a_{30}\ b_{00}$$
$$c_{21} += a_{20}\ b_{01} \qquad c_{31} += a_{31}\ b_{11}$$
$$c_{22} += a_{21}\ b_{12} \qquad c_{32} += a_{32}\ b_{22}$$
$$c_{23} += a_{22}\ b_{23} \qquad c_{33} += a_{33}\ b_{33}$$

Cannon Algorithm (2)

a_{02} b_{20}	a_{03} b_{31}	a_{00} b_{02}	a_{01} b_{13}
a_{13} b_{30}	a_{10} b_{01}	a_{11} b_{12}	a_{12} b_{23}
a_{20} b_{00}	a_{21} b_{11}	a_{22} b_{22}	a_{23} b_{33}
a_{31} b_{10}	a_{32} b_{21}	a_{33} b_{32}	a_{30} b_{03}

$$c_{00} += a_{02}\ b_{20} \qquad c_{10} += a_{13}\ b_{30}$$
$$c_{01} += a_{03}\ b_{31} \qquad c_{11} += a_{10}\ b_{01}$$
$$c_{02} += a_{00}\ b_{02} \qquad c_{12} += a_{11}\ b_{12}$$
$$c_{03} += a_{01}\ b_{13} \qquad c_{13} += a_{12}\ b_{23}$$

$$c_{20} += a_{20}\ b_{00} \qquad c_{30} += a_{31}\ b_{10}$$
$$c_{21} += a_{21}\ b_{11} \qquad c_{31} += a_{32}\ b_{21}$$
$$c_{22} += a_{22}\ b_{22} \qquad c_{32} += a_{33}\ b_{32}$$
$$c_{23} += a_{23}\ b_{33} \qquad c_{33} += a_{30}\ b_{03}$$

Cannon Algorithm (3)

a_{03} b_{30}	a_{00} b_{01}	a_{01} b_{12}	a_{02} b_{23}
a_{10} b_{00}	a_{11} b_{11}	a_{12} b_{22}	a_{13} b_{33}
a_{21} b_{10}	a_{22} b_{21}	a_{23} b_{32}	a_{20} b_{03}
a_{32} b_{20}	a_{33} b_{31}	a_{30} b_{02}	a_{31} b_{13}

$$c_{00} += a_{03} \; b_{30} \qquad c_{10} += a_{10} \; b_{00}$$
$$c_{01} += a_{00} \; b_{01} \qquad c_{11} += a_{11} \; b_{11}$$
$$c_{02} += a_{01} \; b_{12} \qquad c_{12} += a_{12} \; b_{22}$$
$$c_{03} += a_{02} \; b_{23} \qquad c_{13} += a_{13} \; b_{33}$$

$$c_{20} += a_{21} \; b_{10} \qquad c_{30} += a_{32} \; b_{20}$$
$$c_{21} += a_{22} \; b_{21} \qquad c_{31} += a_{33} \; b_{31}$$
$$c_{22} += a_{23} \; b_{32} \qquad c_{32} += a_{30} \; b_{02}$$
$$c_{23} += a_{20} \; b_{03} \qquad c_{33} += a_{31} \; b_{13}$$

1D-2D-3D Topologies

Interconnection Networks

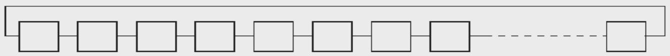


Figure 1.10 Ring.

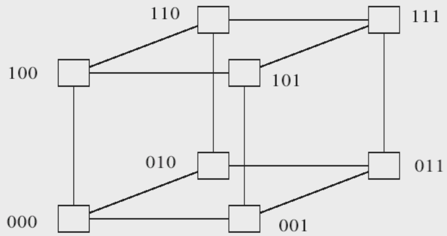


Figure 1.13 Three-dimensional hypercube.

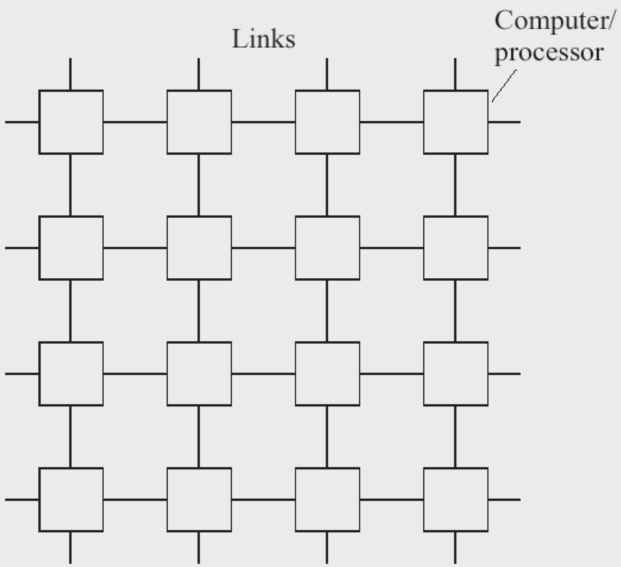


Figure 1.11 Two-dimensional array (mesh).

Creating Topology

- Grid Elements:
  - the dimension: 1, 2, 3 etc
  - the sizes of each dimension
  - the periodicity if the extreme are adjacent
- MPI Methods:
  - **MPI_Cart_create() t**o create the grid
  - **MPI_Card_coords()** to get the coordinates
  - **MPI_Card_rank()** to find the rank

Computation Analysis

- Each submatrix multiplication requires m^3 multiplications and m^3 additions.

- Hence, with s - 1 shifts,

  $$\boldsymbol{t}\mathbf{comp} = \mathbf{2}\boldsymbol{s}\ \boldsymbol{m}^{\mathbf{3}} = \mathbf{2}\ \boldsymbol{m}^{\mathbf{2}}\ \boldsymbol{n}$$

  or a computational time complexity of $\mathbf{O}(\boldsymbol{m}^{\mathbf{2}}\ \boldsymbol{n})$

Communication Analysis

- Given s^2 submatrices, each of size *mxm*, the initial alignment requires a maximum of s - 1 shift (communication) operations. After that, there will be s - 1 shift operations
- Each shift operation involves *mxm* elements.

$$t_{\mathrm{comm}} = 2(s - 1)(t_{\mathrm{startup}} + m^2 t_{\mathrm{data}})$$

or a communication time complexity of $O(sm^2)$ or $O(mn)$

Fox Algorithm

- Both n matrices A and B are partitioned among p processors so that each processor initially stores

$$(n/\sqrt{p}) \times (n/\sqrt{p})$$

- The algorithm uses one to all broadcasts of the blocks of matrix A in processor rows, and single-step circular upwards shifts of the blocks of matrix B along processor columns
- Initially, each diagonal block Aii is selected for broadcast

Fox Algorithm

- Steps (repeated $\sqrt{p}$ times)
  - Broadcast $A_{i,i}$ to all processors in the row
  - Multiply block of A received with resident block of B
  - Send the block of B up one step (with wraparound)
  - Select block $A_{i,(j+1)\bmod\sqrt{p}}$ (where $A_{i,j}$ is the block broadcast in the previous step) and broadcast to all processors in row. Go to 2

Fox Algorithm (0)

- Initially broadcast the diagonal elements of A

a_{00} b_{00}	b_{01}	b_{02}	b_{03}
b_{10}	a_{11} b_{11}	b_{12}	b_{13}
b_{20}	b_{21}	a_{22} b_{22}	b_{23}
b_{30}	b_{31}	b_{32}	a_{33} b_{33}

$$
\begin{array}{ll}
c_{00} = a_{00}\; b_{00} & c_{10} = a_{11}\; b_{10} \\
c_{01} = a_{00}\; b_{01} & c_{11} = a_{11}\; b_{11} \\
c_{02} = a_{00}\; b_{02} & c_{12} = a_{11}\; b_{12} \\
c_{03} = a_{00}\; b_{03} & c_{13} = a_{11}\; b_{13}
\end{array}
$$

$$
\begin{array}{ll}
c_{20} = a_{22}\; b_{20} & c_{30} = a_{33}\; b_{30} \\
c_{21} = a_{22}\; b_{21} & c_{31} = a_{33}\; b_{31} \\
c_{22} = a_{22}\; b_{22} & c_{32} = a_{33}\; b_{32} \\
c_{23} = a_{22}\; b_{23} & c_{33} = a_{33}\; b_{33}
\end{array}
$$

Fox Algorithm (1)

- Broadcast the next element of A in rows, shift B in column and perform multiplication

b_{10}	a_{01} b_{11}	b_{12}	b_{13}
b_{20}	b_{21}	a_{12} b_{22}	b_{23}
b_{30}	b_{31}	b_{32}	a_{23} b_{33}
a_{30} b_{00}	b_{01}	b_{02}	b_{03}

$$c_{00} += a_{01} \quad b_{10} \qquad c_{20} += a_{23} \quad b_{30}$$
$$c_{01} += a_{01} \quad b_{11} \qquad c_{21} += a_{23} \quad b_{31}$$
$$c_{02} += a_{01} \quad b_{12} \qquad c_{22} += a_{23} \quad b_{32}$$
$$c_{03} += a_{01} \quad b_{13} \qquad c_{23} += a_{23} \quad b_{33}$$

$$c_{10} += a_{12} \quad b_{20} \qquad c_{30} += a_{30} \quad b_{00}$$
$$c_{11} += a_{12} \quad b_{21} \qquad c_{31} += a_{30} \quad b_{01}$$
$$c_{12} += a_{12} \quad b_{22} \qquad c_{32} += a_{30} \quad b_{02}$$
$$c_{13} += a_{12} \quad b_{23} \qquad c_{33} += a_{30} \quad b_{03}$$

Fox Algorithm (2)

- Broadcast the next element of A in rows, shift B in column and perform multiplication

b_{20}	b_{21}	a_{02} b_{22}	b_{23}
b_{30}	b_{31}	b_{32}	a_{13} b_{33}
a_{20} b_{00}	b_{01}	b_{02}	b_{03}
b_{10}	a_{31} b_{11}	b_{12}	b_{13}

$$c_{00} += a_{02}\, b_{20} \qquad c_{10} += a_{13}\, b_{30}$$
$$c_{01} += a_{02}\, b_{21} \qquad c_{11} += a_{13}\, b_{31}$$
$$c_{02} += a_{02}\, b_{22} \qquad c_{12} += a_{13}\, b_{32}$$
$$c_{03} += a_{02}\, b_{23} \qquad c_{13} += a_{13}\, b_{33}$$

$$c_{20} += a_{20}\, b_{00} \qquad c_{30} += a_{31}\, b_{10}$$
$$c_{21} += a_{20}\, b_{01} \qquad c_{31} += a_{31}\, b_{11}$$
$$c_{22} += a_{20}\, b_{02} \qquad c_{32} += a_{31}\, b_{12}$$
$$c_{23} += a_{20}\, b_{03} \qquad c_{33} += a_{31}\, b_{13}$$

Fox Algorithm (3)

- Broadcast the next element of A in rows, shift B in column and perform multiplication

b_{30}	b_{31}	b_{32}	a_{03} b_{33}
a_{10} b_{00}	b_{01}	b_{02}	b_{03}
b_{10}	a_{21} b_{11}	b_{12}	b_{13}
b_{20}	b_{21}	a_{32} b_{22}	b_{23}

$$c_{00} += a_{03} \quad b_{30} \qquad c_{10} += a_{10} \quad b_{00}$$
$$c_{01} += a_{03} \quad b_{31} \qquad c_{11} += a_{10} \quad b_{01}$$
$$c_{02} += a_{03} \quad b_{32} \qquad c_{12} += a_{10} \quad b_{02}$$
$$c_{03} += a_{03} \quad b_{33} \qquad c_{13} += a_{10} \quad b_{03}$$

$$c_{20} += a_{21} \quad b_{10} \qquad c_{30} += a_{32} \quad b_{20}$$
$$c_{21} += a_{21} \quad b_{11} \qquad c_{31} += a_{32} \quad b_{21}$$
$$c_{22} += a_{21} \quad b_{12} \qquad c_{32} += a_{32} \quad b_{22}$$
$$c_{23} += a_{21} \quad b_{13} \qquad c_{33} += a_{32} \quad b_{23}$$

Fox Algorithm (4)

- Shifting is over. Stop the ITERATION

a_{00} b_{00}	b_{01}	b_{02}	b_{03}
b_{10}	a_{11} b_{11}	b_{12}	b_{13}
b_{20}	b_{21}	a_{22} b_{22}	b_{23}
b_{30}	b_{31}	b_{32}	a_{33} b_{33}

- Fox algorithm is a memory efficient method.
- Communication overhead is more than cannon algorithm