并行计算
Parallel Computing

主讲 孙经纬

2024年 春季学期

概要

- 第二篇 并行算法
  - **第九章 稠密矩阵运算**
  - 第十章 线性方程组的求解
  - 第十一章 快速傅里叶变换
  - 第十二章 数值计算的基本支撑技术

第九章 稠密矩阵运算

- **9.1 矩阵的划分**
  - 9.1.1 带状划分
  - 9.1.2 棋盘划分
- 9.2 矩阵转置
- 9.3 矩阵-向量乘法
- 9.4 矩阵乘法

划分方法

- 带状划分(striped partitioning):

  one dimensional, row or column,

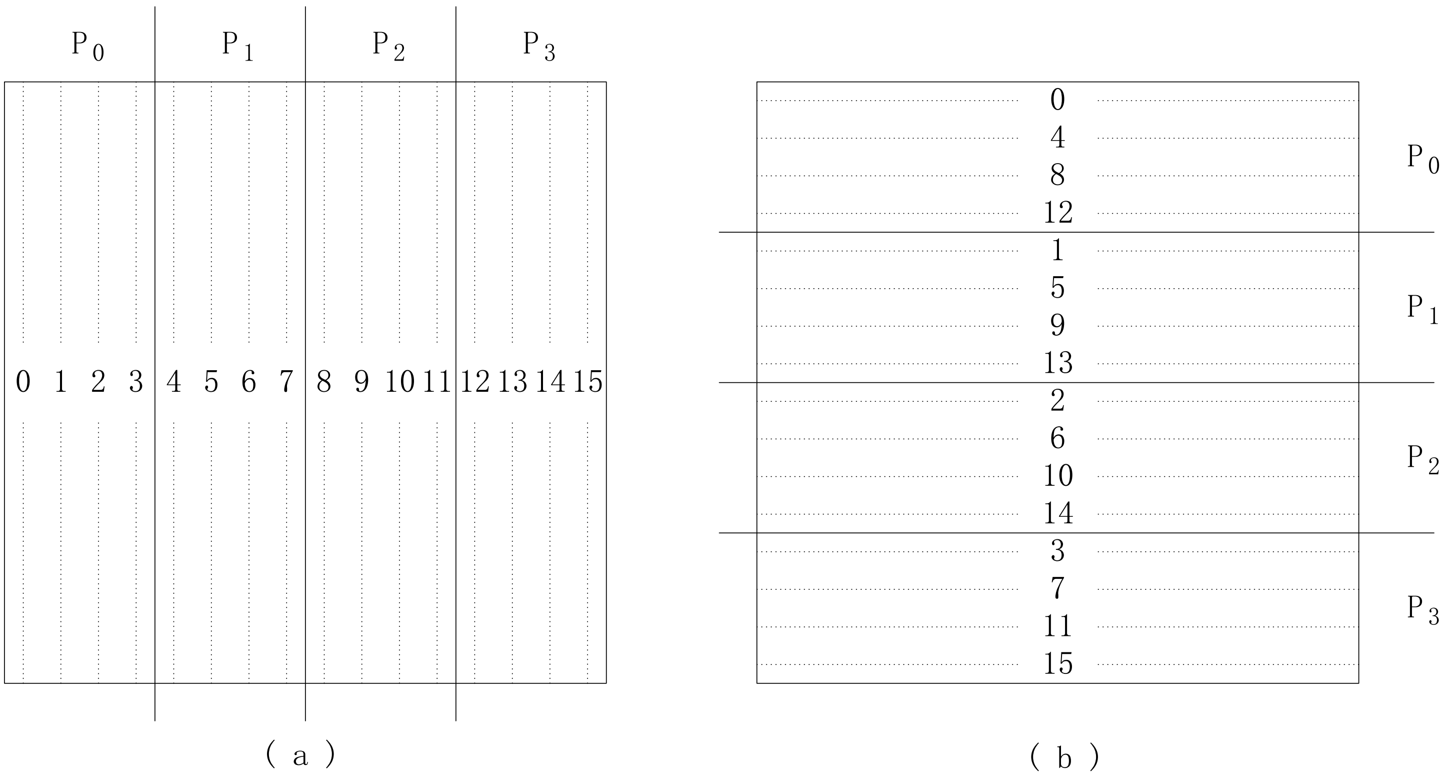
  block or cyclic

- 棋盘划分(checkerboard partitioning):

  two dimensional, block or cyclic

带状划分

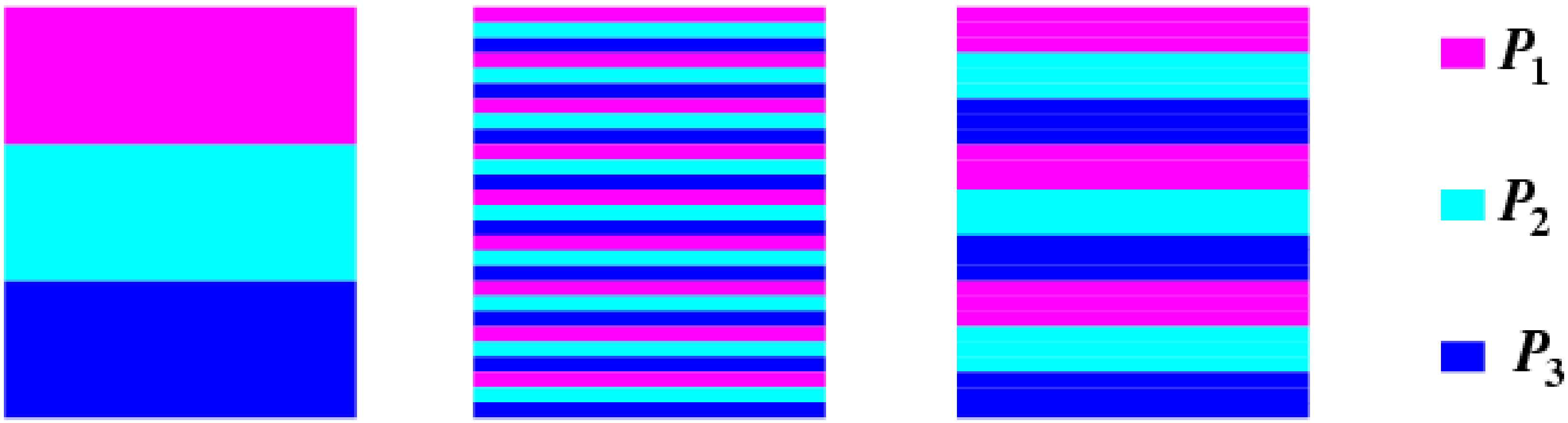
- 16×16阶矩阵，p=4



列块带状划分　　图9.1　　行循环带状划分

带状划分

- 示例：p = 3，27× 27矩阵的3种带状划分



(a) block (b) cyclic (c) block-cyclic

Striped row-major mapping of a 27×27 matrix on $p = 3$ processors.

棋盘划分

- 8×8阶矩阵，p=16

(0, 0) (0, 1) P_0 (1, 0) (1, 1)	(0, 2) (0, 3) P_1 (1, 2) (1, 3)	(0, 4) (0, 5) P_2 (1, 4) (1, 5)	(0, 6) (0, 7) P_3 (1, 6) (1, 7)
(2, 0) (2, 1) P_4 (3, 0) (3, 1)	(2, 2) (2, 3) P_5 (3, 2) (3, 3)	(2, 4) (2, 5) P_6 (3, 4) (3, 5)	(2, 6) (2, 7) P_7 (3, 6) (3, 7)
(4, 0) (4, 1) P_8 (5, 0) (5, 1)	(4, 2) (4, 3) P_9 (5, 2) (5, 3)	(4, 4) (4, 5) P_{10} (5, 4) (5, 5)	(4, 6) (4, 7) P_{11} (5, 6) (5, 7)
(6, 0) (6, 1) P_{12} (7, 0) (7, 1)	(6, 2) (6, 3) P_{13} (7, 2) (7, 3)	(6, 4) (6, 5) P_{14} (7, 4) (7, 5)	(6, 6) (6, 7) P_{15} (7, 6) (7, 7)

（ a ）

块棋盘划分

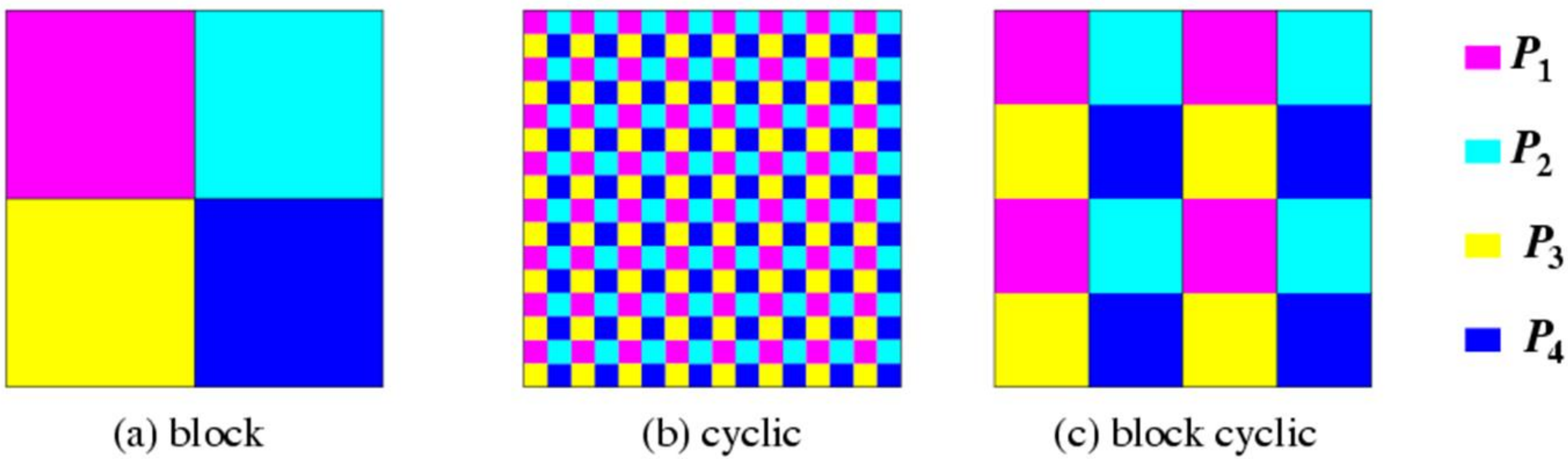
(0, 0) (0, 4) P_0 (4, 0) (4, 4)	(0, 1) (0, 5) P_1 (4, 1) (4, 5)	(0, 2) (0, 6) P_2 (4, 2) (4, 6)	(0, 3) (0, 7) P_3 (4, 3) (4, 7)
(1, 0) (1, 4) P_4 (5, 0) (5, 4)	(1, 1) (1, 5) P_5 (5, 1) (5, 5)	(1, 2) (1, 6) P_6 (5, 2) (5, 6)	(1, 3) (1, 7) P_7 (5, 3) (5, 7)
(2, 0) (2, 4) P_8 (6, 0) (6, 4)	(2, 1) (2, 5) P_9 (6, 1) (6, 5)	(2, 2) (2, 6) P_{10} (6, 2) (6, 6)	(2, 3) (2, 7) P_{11} (6, 3) (6, 7)
(3, 0) (3, 4) P_{12} (7, 0) (7, 4)	(3, 1) (3, 5) P_{13} (7, 1) (7, 5)	(3, 2) (3, 6) P_{14} (7, 2) (7, 6)	(3, 3) (3, 7) P_{15} (7, 3) (7, 7)

（ b ）

图9. 2 循环棋盘划分

棋盘划分

- 示例：p = 4，16×16矩阵的3种棋盘划分



Checkerboard mapping of a 16×16 matrix on $p = 2 \times 2$ processors.

第九章 稠密矩阵运算

- 9.1 矩阵的划分
- **9.2 矩阵转置**
  - 9.2.1 棋盘划分的矩阵转置
  - 9.2.2 带状划分的矩阵转置
- 9.3 矩阵-向量乘法
- 9.4 矩阵乘法

复习——通信时间

启动时间t_s
节点延迟时间t_h
字传输时间t_w
链路数l
信包大小m

基本公式

$$T_{comm}(SF) = t_s + (mt_w + t_h)\cdot l \approx t_s + mt_w \cdot l = O(m \cdot l)$$

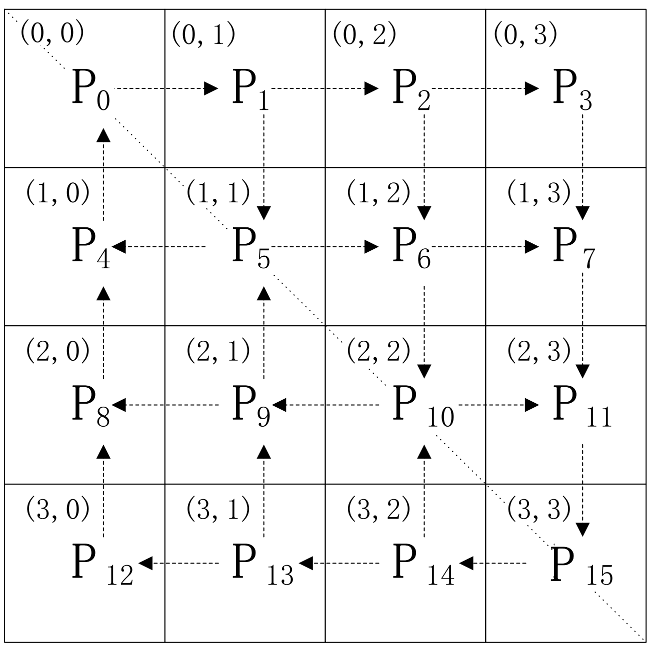
$$T_{comm}(CT) = t_s + mt_w + lt_h \approx t_s + mt_w = O(m + l)$$

	$p-$环	$\sqrt{p}\times\sqrt{p}-$环绕网孔	$p-$超立方
$T_{one-to-one}(SF)$:	$t_s + mt_w\lfloor p/2 \rfloor$	$t_s + 2mt_w\lfloor \sqrt{p}/2 \rfloor$	$t_s + mt_w \log p$
$T_{one-to-one}(CT)$:	$t_s + mt_w$	$t_s + mt_w$	$t_s + mt_w$
$T_{one-to-all}(SF)$:	$(t_s + mt_w)\lceil p/2 \rceil$	$2(t_s + mt_w)\lceil \sqrt{p}/2 \rceil$	$(t_s + mt_w)\log p$
$T_{one-to-all}(CT)$:	$(t_s + mt_w)\log p$	$(t_s + mt_w)\log p$	$(t_s + mt_w)\log p$
$T_{all-to-all}(SF)$:	$(t_s + mt_w)(p-1)$	$2t_s(\sqrt{p}-1) + mt_w(p-1)$	$t_s \log p + mt_w(p-1)$
$T_{all-to-all}(CT)$:	同上	同上	同上

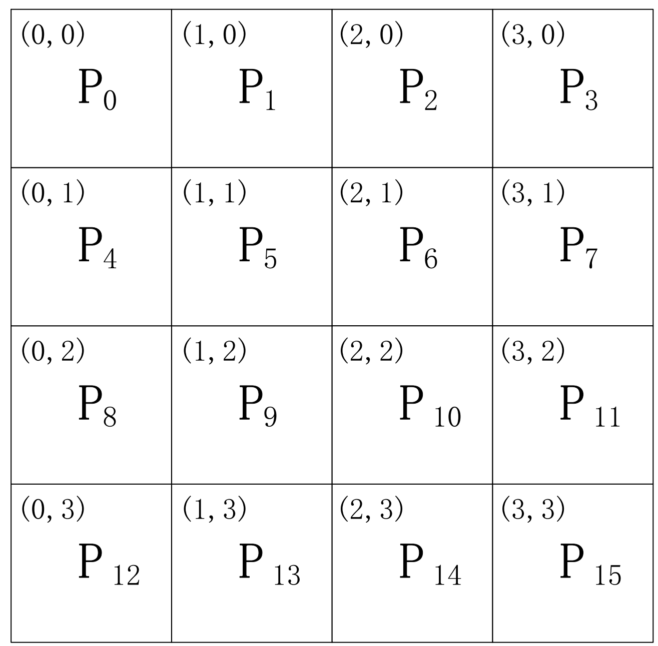
棋盘划分的矩阵转置

- 网孔连接 —— 情形1: $p=n^2$

下三角的元素：先上移至对角线，再右移至目的地址
上三角的元素：先下移至对角线，再左移至目的地址



（ a ）



（ b ）

图9. 3

棋盘划分的矩阵转置

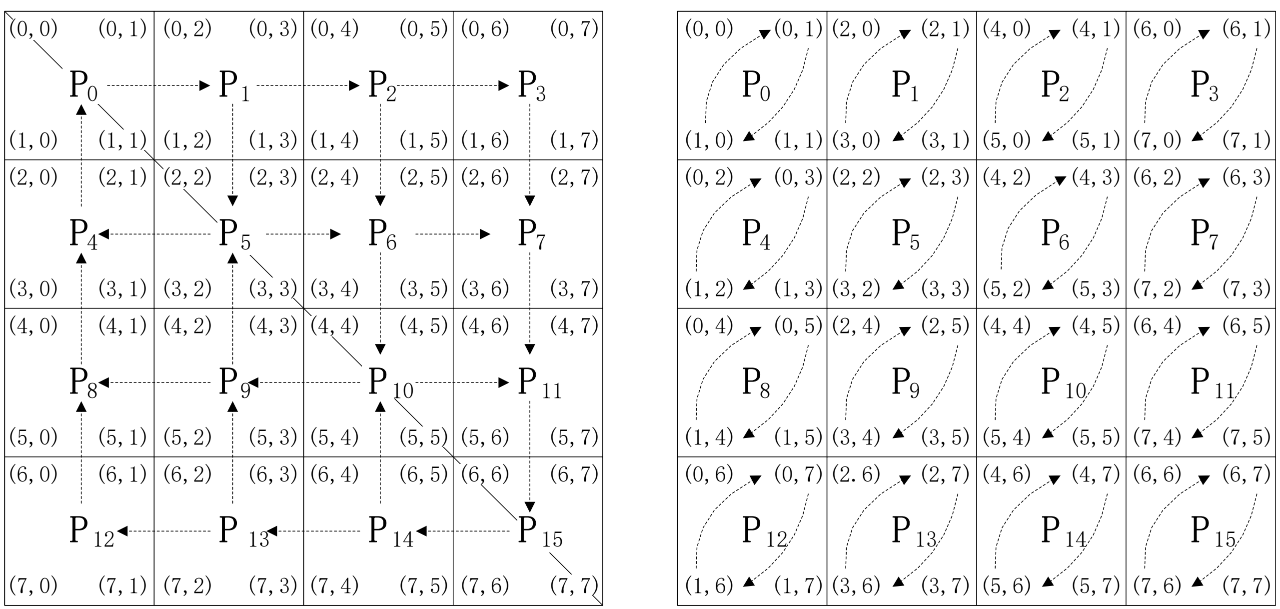
- 网孔连接 — 情形2: p<n^2
  - 划分: $A_{n\times n}$*划分成p个大小为* $\frac{n}{\sqrt{p}}\times\frac{n}{\sqrt{p}}$ *子块*
  - 算法: ①将子块视为一个元素转置(不同处理器间)
    ②进行块内转置(同一处理器内)

最长移动距离

移动单个子块耗时

// $\boxed{2\sqrt{p}}\boxed{(t_s+t_w n^2/p)}$ …通讯

// $\frac{n^2}{2p}$ ……计算



(a) (b)

图9. 4

总运行时间:

$$T_p=\frac{n^2}{2p}+2t_s\sqrt{p}+2t_w n^2/\sqrt{p}$$

棋盘划分的矩阵转置

- 超立方连接
  - 划分: $A_{n\times n}$*划分成p个大小为*$\frac{n}{\sqrt{p}}\times\frac{n}{\sqrt{p}}$*子块*
  - 算法:

①将$\boldsymbol{A}=\begin{pmatrix}\boldsymbol{A}_{11} & \boldsymbol{A}_{12}\\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22}\end{pmatrix}$转置为$\begin{pmatrix}\boldsymbol{A}_{11} & \boldsymbol{A}_{21}\\ \boldsymbol{A}_{12} & \boldsymbol{A}_{22}\end{pmatrix}$

②对A_{ij}**递归**应用①进行转置，直至分块矩阵的元素处于**同一处理器**；

③进行同一处理器的内部转置。

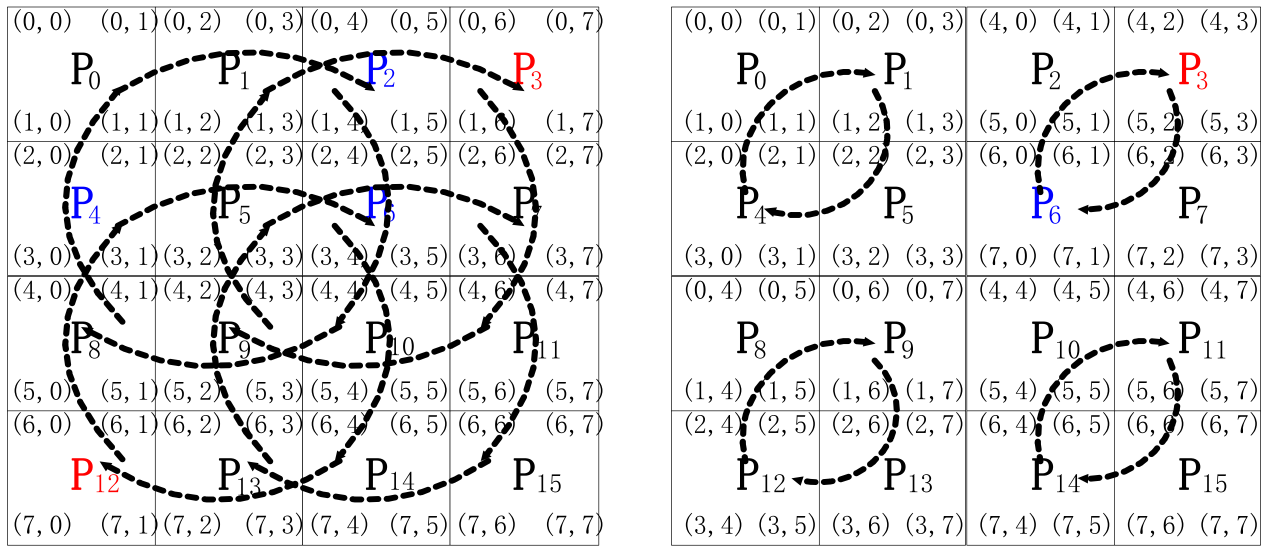
- 运行时间:

$$T_p=\frac{n^2}{2p}+2(t_s+t_w\frac{n^2}{p})\log\sqrt{p}\quad //\text{内部转置}\ \frac{n^2}{2p},\ \text{选路}:2(t_s+t_w\frac{n^2}{p}),\ \text{递归步}:\log\sqrt{p}$$

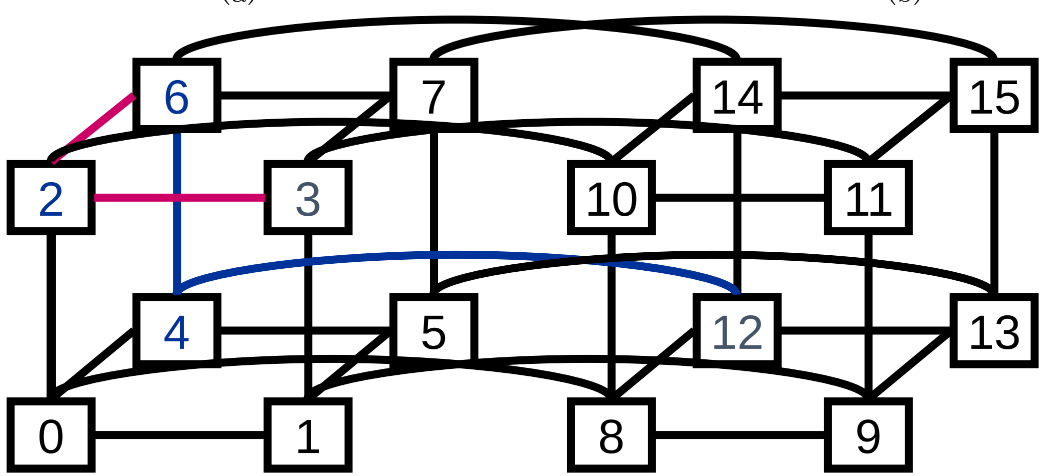
$$=\frac{n^2}{2p}+(t_s+t_w\frac{n^2}{p})\log p$$

棋盘划分的矩阵转置

超立方示例:

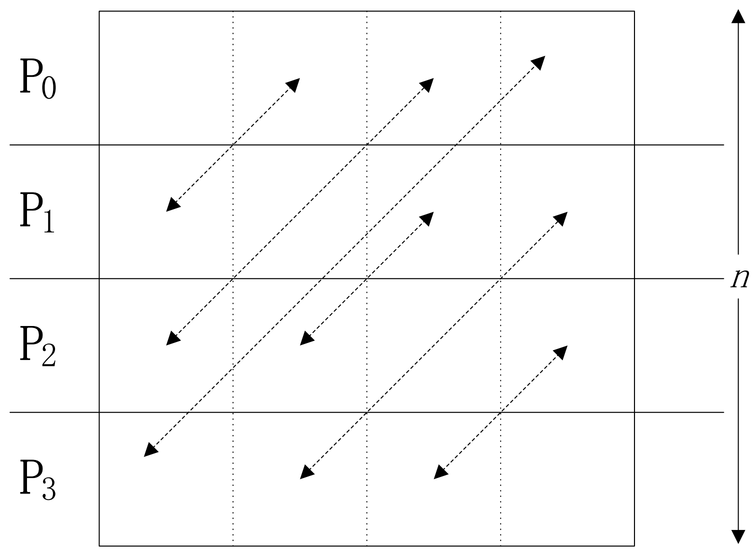


(a) (b)



带状划分的矩阵转置

- 划分：$A_{n\times n}$分成p个(n/p)×n大小的带



- 算法：

图9.7

①P_i有p-1个(n/p)×(n/p)大小子块发送到另外p-1个处理器中；

②每个处理器本地交换相应的元素；

③时间分析？

第九章 稠密矩阵运算

- 9.1 矩阵的划分
- 9.2 矩阵转置
- **9.3 矩阵-向量乘法**
  - 9.3.1 带状划分的矩阵-向量乘法
  - 9.3.2 棋盘划分的矩阵-向量乘法
  - 9.3.3 矩阵-向量乘法的脉动算法
- 9.4 矩阵乘法

矩阵-向量乘法

- 求Y=AX

$$\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,n-1} \\ a_{1,0} & a_{1,1} & & a_{1,n-1} \\ \vdots & \vdots & & \vdots \\ a_{n-1,0} & a_{n-1,1} & \cdots & a_{n-1,n-1} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{pmatrix}$$

$$y_i = \sum_{j=0}^{n-1} a_{ij} \cdot x_j$$

串行算法计算时间t(n)=O(n^2)

带状划分的矩阵-向量乘法

- 划分(行带状划分): P_i存放x_i和$a_{i,0},a_{i,1},\cdots,a_{i,n-1}$, 并输出$y_i$
- 算法: 对p=n情形

  ①每个P_i向其他处理器播送x_i(多到多播送);

  ②每个P_i做相应计算;
- 注: 对p<n情形,算法中P_i要播送X中相应的n/p个分量

  (1)超立方连接的计算时间

$$T_p = \frac{n^2}{p} + t_s \log p + \frac{n}{p} t_w (p-1) \quad \text{// 前1项是乘法时间，后2项是多到多的播送时间}$$

$$= \frac{n^2}{p} + t_s \log p + n t_w \quad \text{// } p\text{充分大时}$$

	$\sqrt{p}\times\sqrt{p}$–环绕网孔	p–超立方
$T_{all\text{-}to\text{-}all}(CT)$:	$2t_s(\sqrt{p}-1)+mt_w(p-1)$	$t_s \log p + mt_w(p-1)$

  (2)网孔连接的计算时间

$$T_p = \frac{n^2}{p} + 2(\sqrt{p}-1)t_s + \frac{n}{p} t_w (p-1) \quad \text{// 前1项是乘法时间，后2项是多到多的播送时间}$$

$$= \frac{n^2}{p} + 2t_s(\sqrt{p}-1) + n t_w \quad \text{// } p\text{充分大时}$$

带状划分的矩阵-向量乘法

- 示例

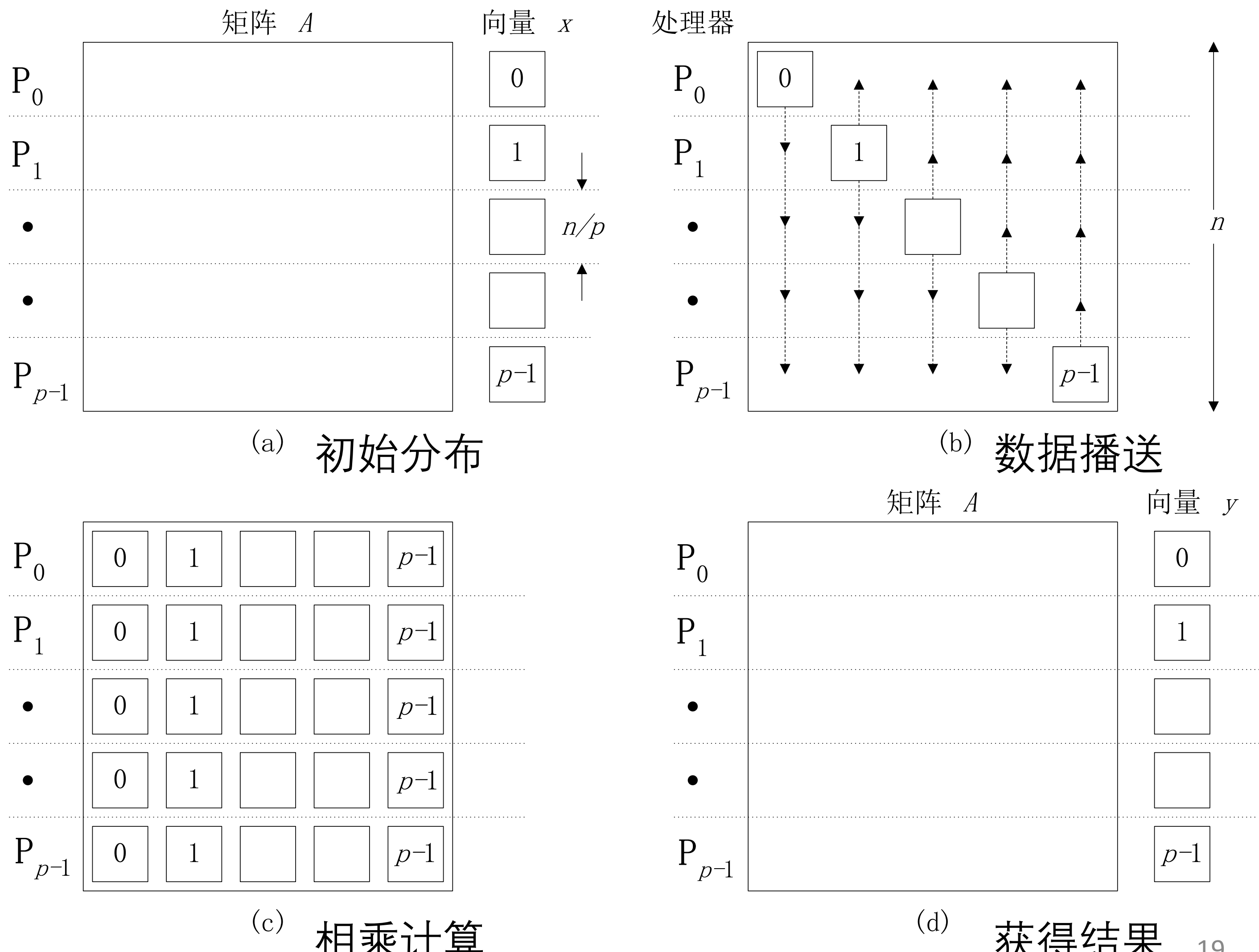


图9.8

棋盘划分的矩阵-向量乘法

- 划分(块棋盘划分): P_{ij}存放$a_{i,j}$, x_i置入$P_{i,i}$中
- 算法: 对$p=n^2$情形

①每个$P_{i,i}$向$P_{j,i}$播送x_i(列方向一到多播送);

②按行方向进行乘-加与积累运算，最后一列$P_{i,n-1}$收集的结果为y_i;

- 对$p<n^2$情形，p个处理器排成$\sqrt{p}\times\sqrt{p}$的二维网孔，

算法中$P_{i,i}$向$P_{j,i}$播送X中相应的$n/\sqrt{p}$个分量

网孔连接上的运行时间T_p(CT): $T_p \approx \frac{n^2}{p} + t_s \log p + \frac{n}{\sqrt{p}} t_w \log p + 3t_h\sqrt{p}$

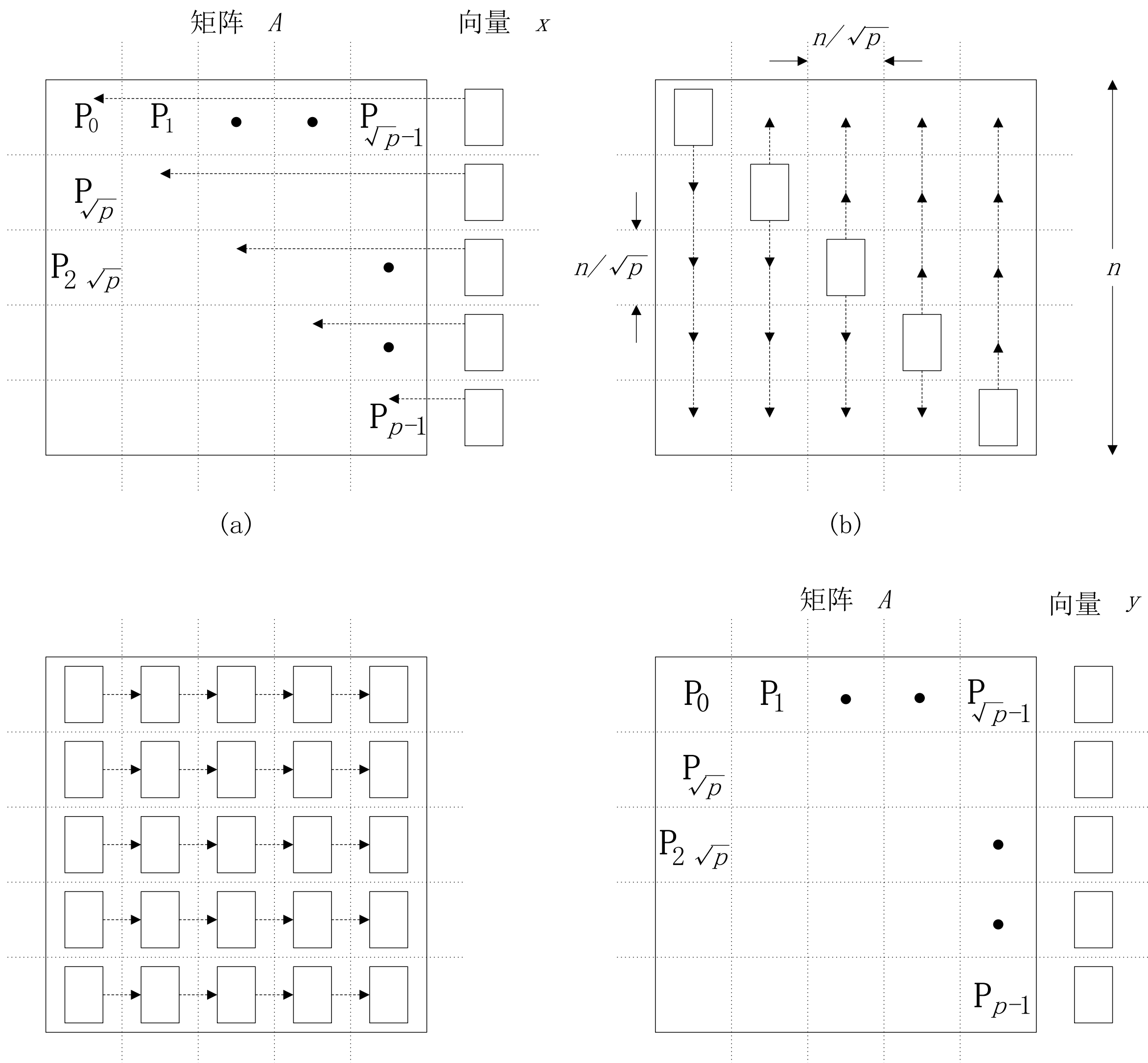
X中相应分量置入$P_{i,i}$的通讯时间: $t_s + \frac{n}{\sqrt{p}} t_w + t_h\sqrt{p}$

按列一到多播送时间: $(t_s + \frac{n}{\sqrt{p}} t_w)\log\sqrt{p} + t_h(\sqrt{p}-1)$

按行单点积累的时间: $(t_s + \frac{n}{\sqrt{p}} t_w)\log\sqrt{p} + t_h(\sqrt{p}-1)$

棋盘划分的矩阵-向量乘法

- 示例



(a)
(b)
(c)
(d)

带状与棋盘划分比较

以网孔连接为例

- 网孔上带状划分的运行时间

$$T_p = \frac{n^2}{p} + 2t_s(\sqrt{p} - 1) + nt_w \tag{9.5}$$

- 网孔上棋盘划分的运行时间

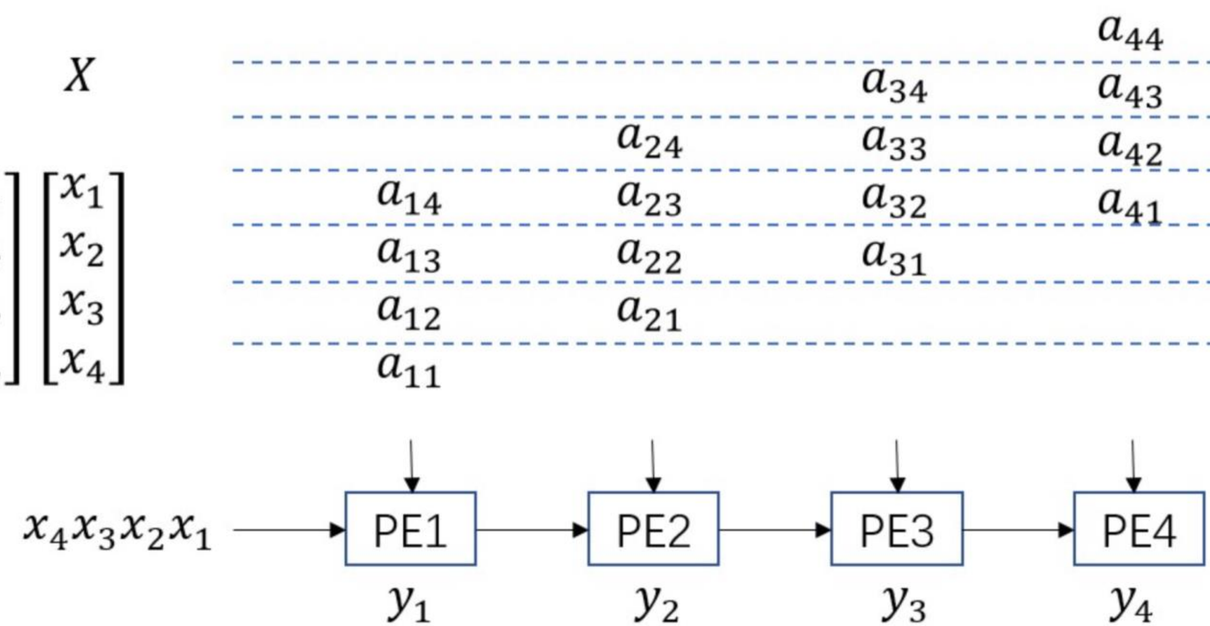
$$T_p \approx \frac{n^2}{p} + t_s \log p + \frac{n}{\sqrt{p}} t_w \log p + 3t_h\sqrt{p} \tag{9.6}$$

- 棋盘划分要比带状划分快（在超立方上也是）

矩阵-向量乘法的脉动算法

- 一维阵列示例

$$
\underset{Y}{\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}} = \underset{A}{\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}} \underset{X}{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}}
$$



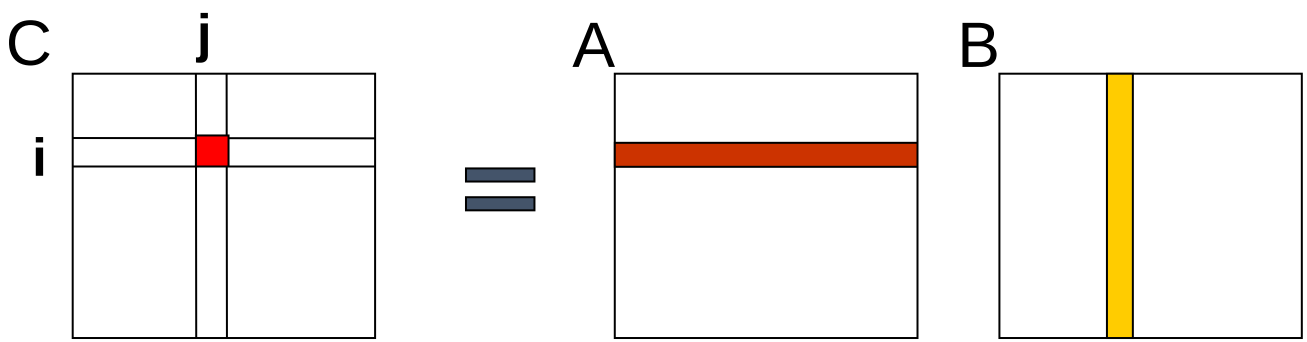
第九章 稠密矩阵运算

- 9.1 矩阵的划分
- 9.2 矩阵转置
- 9.3 矩阵-向量乘法
- **9.4 矩阵乘法**
  - 9.4.1 简单并行分块乘法
  - 9.4.2 Cannon乘法
  - 9.4.3 Fox乘法
  - 9.4.4 Systolic乘法
  - 9.4.5 DNS乘法

矩阵乘法符号及定义

设$A=(a_{ij})_{n\times n}$　$B=(b_{ij})_{n\times n}$　$C=(c_{ij})_{n\times n}$，　$C=A\times B$

$$\begin{pmatrix} c_{0,0} & c_{0,1} & \cdots & c_{0,n-1} \\ c_{1,0} & c_{1,1} & & c_{1,n-1} \\ \vdots & \vdots & & \vdots \\ c_{n-1,0} & c_{n-1,1} & \cdots & c_{n-1,n-1} \end{pmatrix} = \begin{pmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,n-1} \\ a_{1,0} & a_{1,1} & & a_{1,n-1} \\ \vdots & \vdots & & \vdots \\ a_{n-1,0} & a_{n-1,1} & \cdots & a_{n-1,n-1} \end{pmatrix} \cdot \begin{pmatrix} b_{0,0} & b_{0,1} & \cdots & b_{0,n-1} \\ b_{1,0} & b_{1,1} & & b_{1,n-1} \\ \vdots & \vdots & & \vdots \\ b_{n-1,0} & b_{n-1,1} & \cdots & b_{n-1,n-1} \end{pmatrix}$$



$$c_{ij}=\sum_{k=0}^{n-1} a_{ik} b_{kj}$$

A中元素的第2下标与

B中元素的第1下标相一致**（对准）**

矩阵乘法并行实现方法

- 空间对准(元素已加载到阵列中)

  Cannon’s， Fox’s， DNS

- 时间对准(元素未加载到阵列中)

  Systolic

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

简单并行分块乘法

- 分块: A、B和C分成 $p = \sqrt{p} \times \sqrt{p}$ 的方块阵 $A_{i,j}$、$B_{i,j}$和$C_{i,j}$, 大小均为 $\frac{n}{\sqrt{p}} \times \frac{n}{\sqrt{p}}$

  p个处理器编号为$(P_{0,0},\ldots,P_{0,\sqrt{p}-1},\ldots,P_{\sqrt{p}-1,\sqrt{p}-1})$, $P_{i,j}$存放$A_{i,j}$、$B_{i,j}$和$C_{i,j}$

- 算法:

  ①通讯:每行处理器进行A矩阵块的多到多播送(得到$A_{i,k}$, k=0~$\sqrt{p}-1$)

  每列处理器进行B矩阵块的多到多播送(得到$B_{k,j}$, k=0~ $\sqrt{p}-1$)

  ②乘-加运算: $P_{i,j}$计算$C_{ij} = \sum_{k=0}^{\sqrt{p}-1} A_{ik} \cdot B_{kj}$

- 运行时间

  (1)超立方连接:

  ①的时间 $t_1 = 2(t_s \log\sqrt{p} + t_w \frac{n^2}{p}(\sqrt{p}-1))$

  ②的时间 $t_2 = \sqrt{p} \times (\frac{n}{\sqrt{p}})^3 = n^3 / p$

  $$\therefore \quad T_p = \frac{n^3}{p} + t_s \log p + 2t_w \frac{n^2}{\sqrt{p}}$$

简单并行分块乘法

- 运行时间

  (1)超立方连接:

  (2)二维环绕网孔连接:

  $\sqrt{p}-$环

  ①的时间: $t_1 = 2(t_s + \frac{n^2}{p}t_w)(\sqrt{p}-1) = 2t_s\sqrt{p} + 2t_w\frac{n^2}{\sqrt{p}}$

  ②的时间: $t_2=n^3/p$

  $$\therefore\quad T_p = \frac{n^3}{p} + 2t_s\sqrt{p} + 2t_w\frac{n^2}{\sqrt{p}}$$

- 注

  (1)本算法的缺点是对处理器的**存储要求过大**

  每个处理器有 $2\sqrt{p}$ 个块, 每块大小为n^2/p,

  所以需要$O(n^2/\sqrt{p})$ 空间, p个处理器共需要$O(n^2\sqrt{p})$,

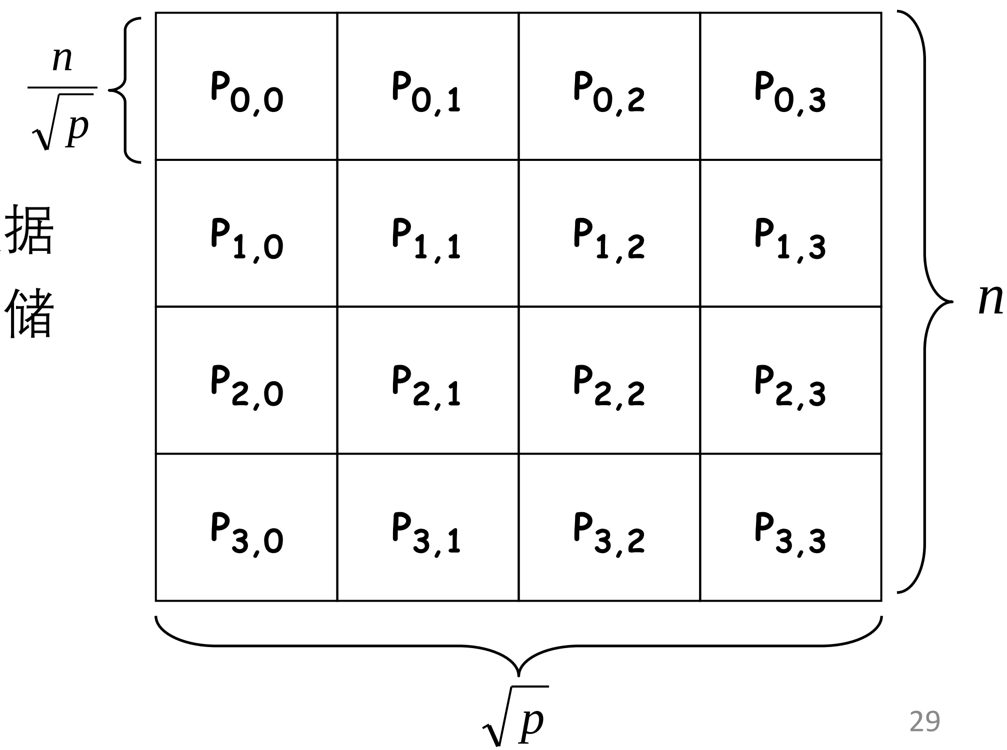
  是串行算法的 $\sqrt{p}$ 倍

  (2)$p=n^2$时, $t(n)=O(n)$, $c(n)=O(n^3)$

Cannon乘法

- 分块: A、B和C分成 $p=\sqrt{p}\times\sqrt{p}$ 的方块阵$A_{i,j}$、$B_{i,j}$和$C_{i,j}$, 大小均为 $\frac{n}{\sqrt{p}}\times\frac{n}{\sqrt{p}}$ ， p个处理器编号为 $(P_{0,0},\ldots,P_{0,\sqrt{p}-1},\ldots,P_{\sqrt{p}-1,\sqrt{p}-1})$ ，$P_{i,j}$存放$A_{i,j}$、$B_{i,j}$和$C_{i,j}$(n > > p)

- 设计原则
  - 避免多到多播送所有数据
  - 通过循环位移，减少存储



Cannon乘法

- 算法步骤 (1969年)

①所有块$A_{i,j}(0\leqslant i,j\leqslant \sqrt{p}-1)$向左循环移动i步(按行移位);

所有块$B_{i,j}(0\leqslant i,j\leqslant \sqrt{p}-1)$向上循环移动j步(按列移位);

②所有处理器$P_{i,j}$做执行$A_{i,j}$和$B_{i,j}$的乘-加运算;

③A的每个块向左循环移动一步;

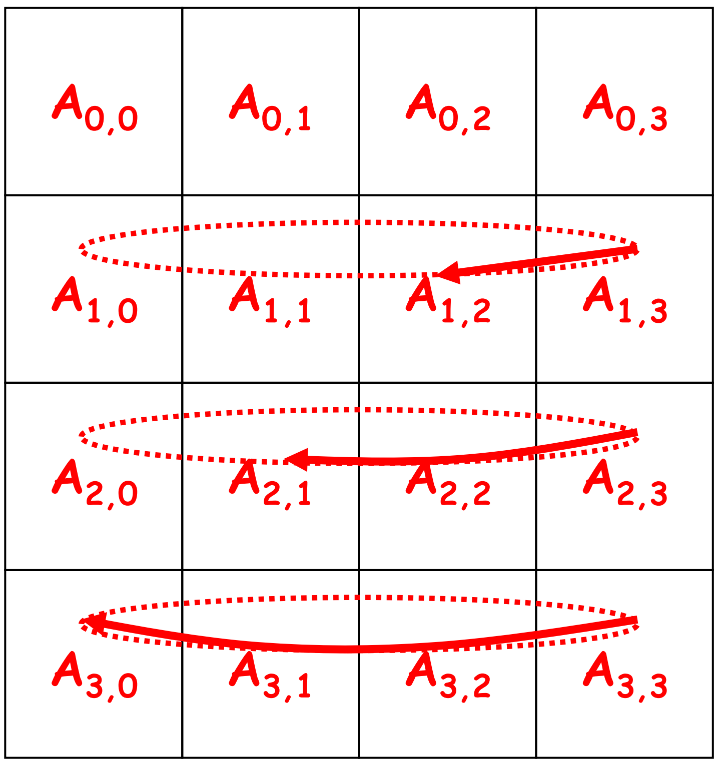
B的每个块向上循环移动一步;

④转②执行$\sqrt{p}-1$次;

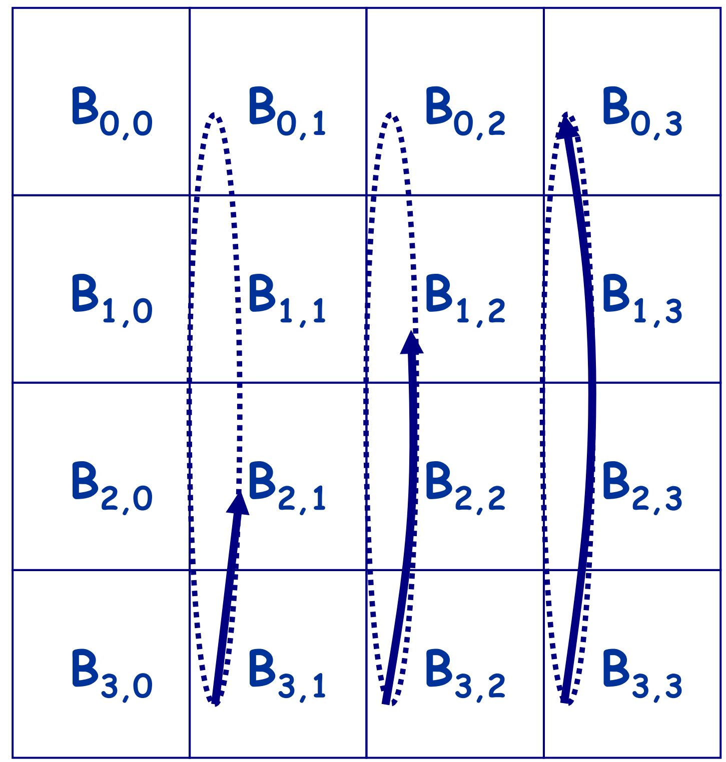
Cannon乘法

- 示例：$A_{4\times4}$，$B_{4\times4}$，p=16

Initial alignment of A



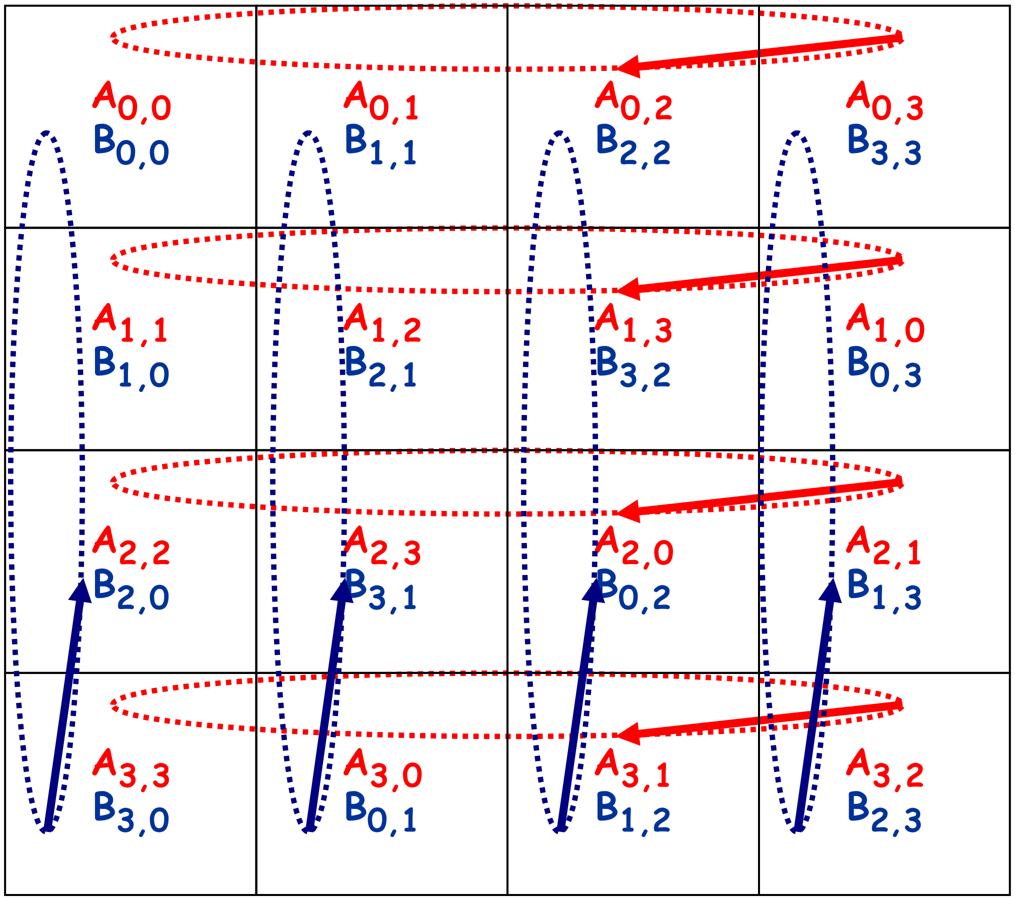
Initial alignment of B



Cannon乘法

- 示例：$A_{4\times4}$，$B_{4\times4}$，p=16

A and B after initial alignment and shifts after every step



Cannon乘法

- 示例：$A_{4\times4}$，$B_{4\times4}$，p=16

After first shift

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

After second shift

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

After third shift

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

Cannon乘法

- 算法描述

```
//输入: A_{n×n}, B_{n×n};     输出: C_{n×n}
  Begin
    (1)for k=0 to √p−1  do
         for all P_{i,j} par-do
           (i) if i>k then
                 A_{i,j} ← A_{i,(j+1)mod √p}
               endif
           (ii)if j>k then
                 B_{i,j} ← B_{(i+1)mod√p ,j}
               endif
         endfor
       endfor
    (2)for all P_{i,j} par-do C_{i,j}=0 endfor
    (3)for k=0 to √p−1  do
         for all P_{i,j} par-do
           (i)  C_{i,j}=C_{i,j}+A_{i,j}B_{i,j}
           (ii) A_{i,j} ← A_{i,(j+1)mod √p}
           (iii)B_{i,j} ← B_{(i+1)mod √p ,j}
         endfor
       endfor
  End
```

粗略的时间分析:

$$T_p(n) = T_1 + T_2 + T_3$$
$$= O(\sqrt{p}) + O(1) + O(\sqrt{p}\cdot(n/\sqrt{p})^3)$$
$$= O(n^3/p)$$

Cannon乘法

时间分析

- 超立方连接，CT选路模式:

$$t_1 = 2(t_s + t_w \frac{n^2}{p} + t_h \log \sqrt{p})，\quad t_2 = (n/\sqrt{p})^3，\quad t_3 = 2(t_s + t_w \frac{n^2}{p})$$

②和③执行$\sqrt{p}-1$次，所以运行时间为

$$T_p = t_1 + \sqrt{p}(t_2 + t_3) = \frac{n^3}{p} + 2\sqrt{p}t_s + 2t_w \frac{n^2}{\sqrt{p}}$$

- 二维网孔连接:

$$t_1 = 2(t_s + t_w \frac{n^2}{p})\sqrt{p}，\quad t_2 = (n/\sqrt{p})^3，\quad t_3 = 2(t_s + t_w \frac{n^2}{p})$$

②和③执行 $\sqrt{p}-1$次，所以运行时间为

$$T_p = t_1 + \sqrt{p}(t_2 + t_3) = \frac{n^3}{p} + 4\sqrt{p}t_s + 4t_w \frac{n^2}{\sqrt{p}}$$

Fox乘法

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

- 算法步骤（1987年）

①$A_{i,i}$向所在行的其他处理器进行一到多播送;

②各处理器将收到的A块与原有的B块进行乘-加运算;

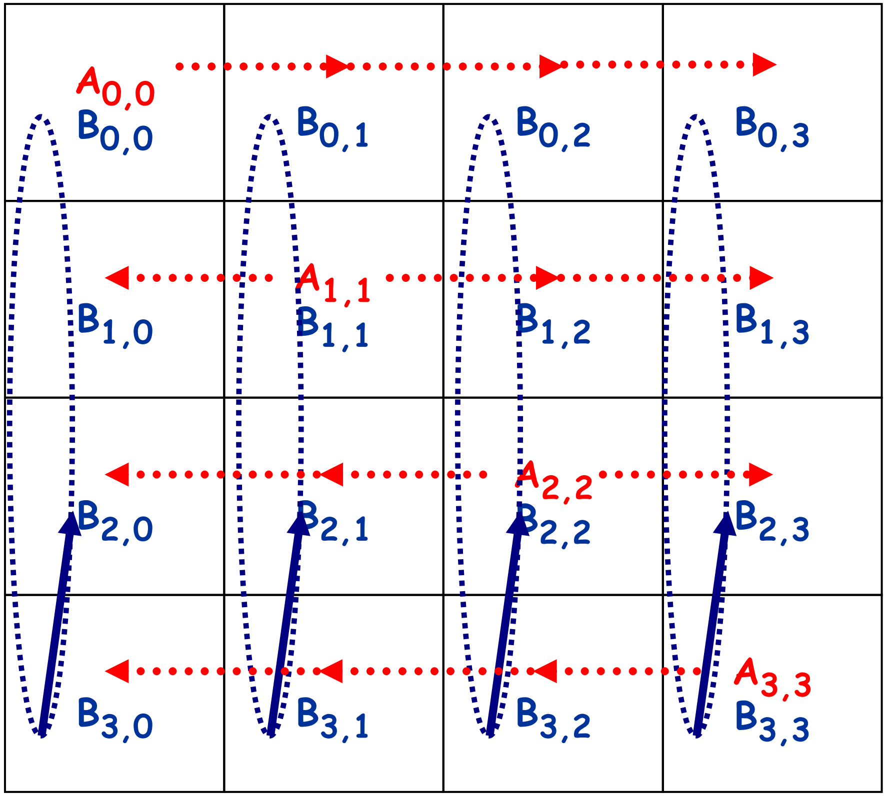
③B块向上循环移动一步;

④如果$A_{i,j}$是上次第i行播送的块，本次选择$A_{i,(j+1)\bmod\sqrt{p}}$向所在行的其他处理器进行一到多播送;

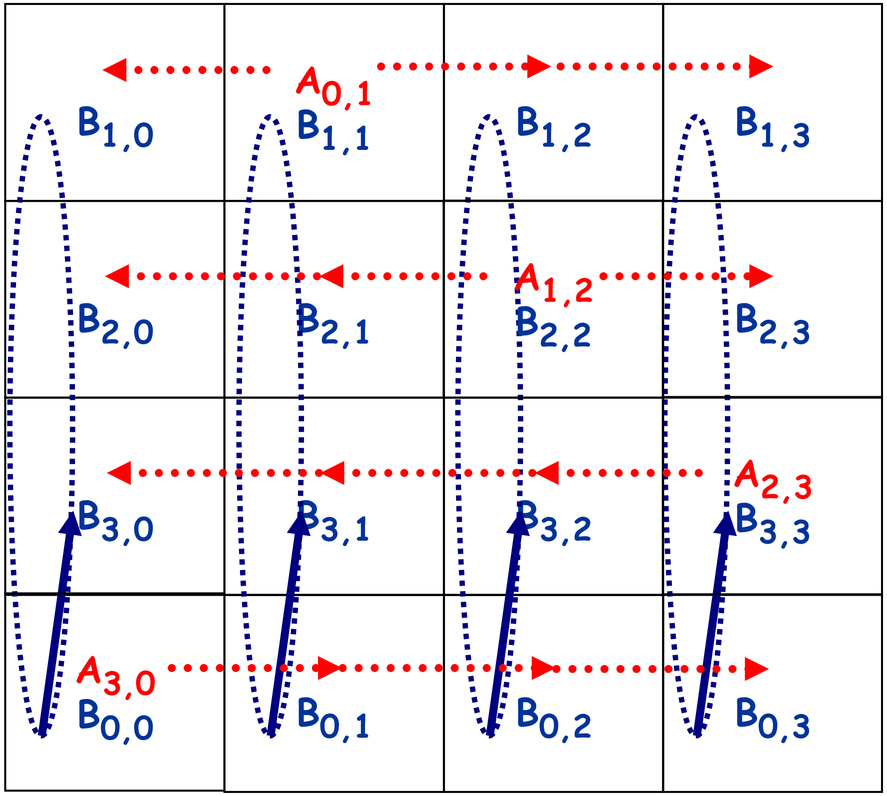
⑤转②执行$\sqrt{p}-1$次;

Fox乘法

- 示例： $A_{4\times4}$， $B_{4\times4}$， p=16



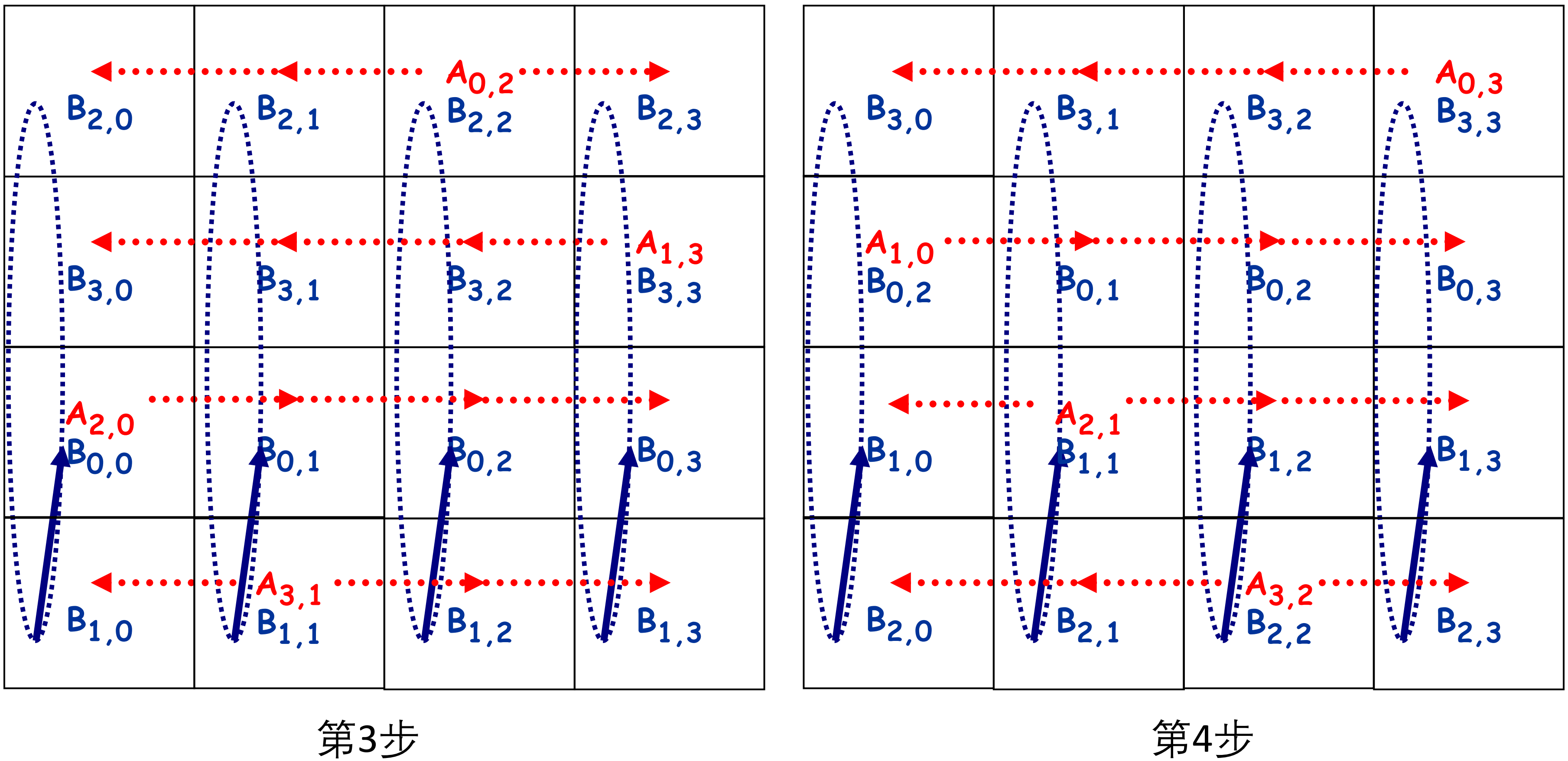
第1步



第2步

Fox乘法

- 示例：$A_{4\times4}$，$B_{4\times4}$，p=16



Fox乘法

时间分析

- 超立方连接，CT选路模式:

$$t_1 = t_4 = (t_s + t_w \frac{n^2}{p}) \log \sqrt{p}, \quad t_2 = (n/\sqrt{p})^3, \quad t_3 = t_s + t_w \frac{n^2}{p}$$

②、③和④(含①)执行$\sqrt{p}$次，所以运行时间为

$$T_p = \sqrt{p}(t_2 + t_3 + t_4) = \frac{n^3}{p} + \frac{1}{2}(t_s + t_w \frac{n^2}{p})\sqrt{p} \log p$$

当$p=n^2$时, t(n)=O(nlogn)

- 二维网孔连接 (思考?)

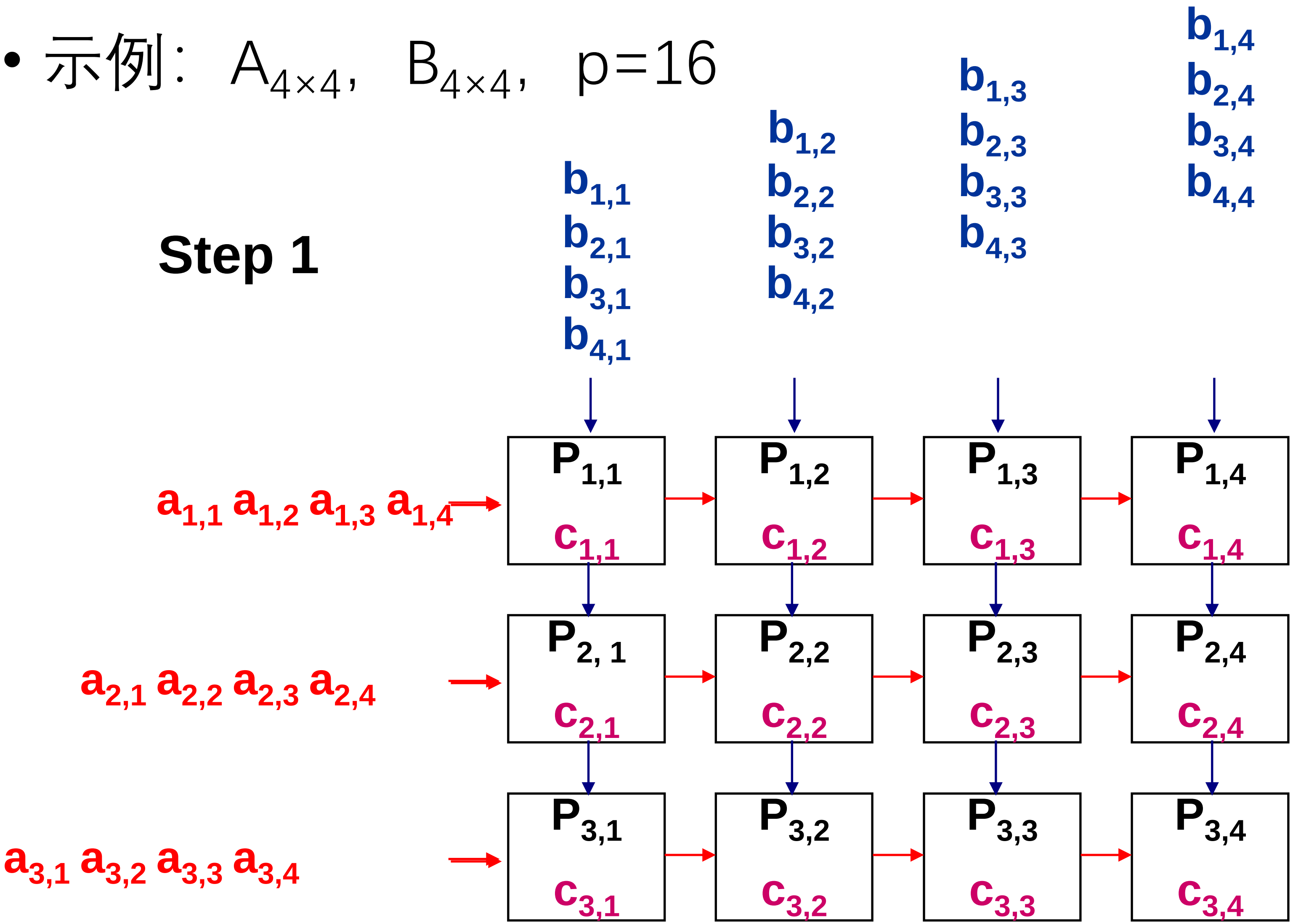
Systolic乘法

- Systolic算法（H.T. Kung）

```
//输入: A_{m×n}, B_{n×k};     输出: C_{m×k}
 Begin
     for i=1 to m   par- do
            for j=1 to k  par-do
             (i) c_{i,j} = 0
             (ii) while P_{i,j} 收到a和b时 do
                     c_{i,j} = c_{i,j} +ab
                     if  i < m then 发送b给P_{i+1,j} endif //传给右一列
                     if  j < k  then 发送a给P_{i,j+1} endif //传给下一行
                  endwhile
             endfor
        endfor
 End
```

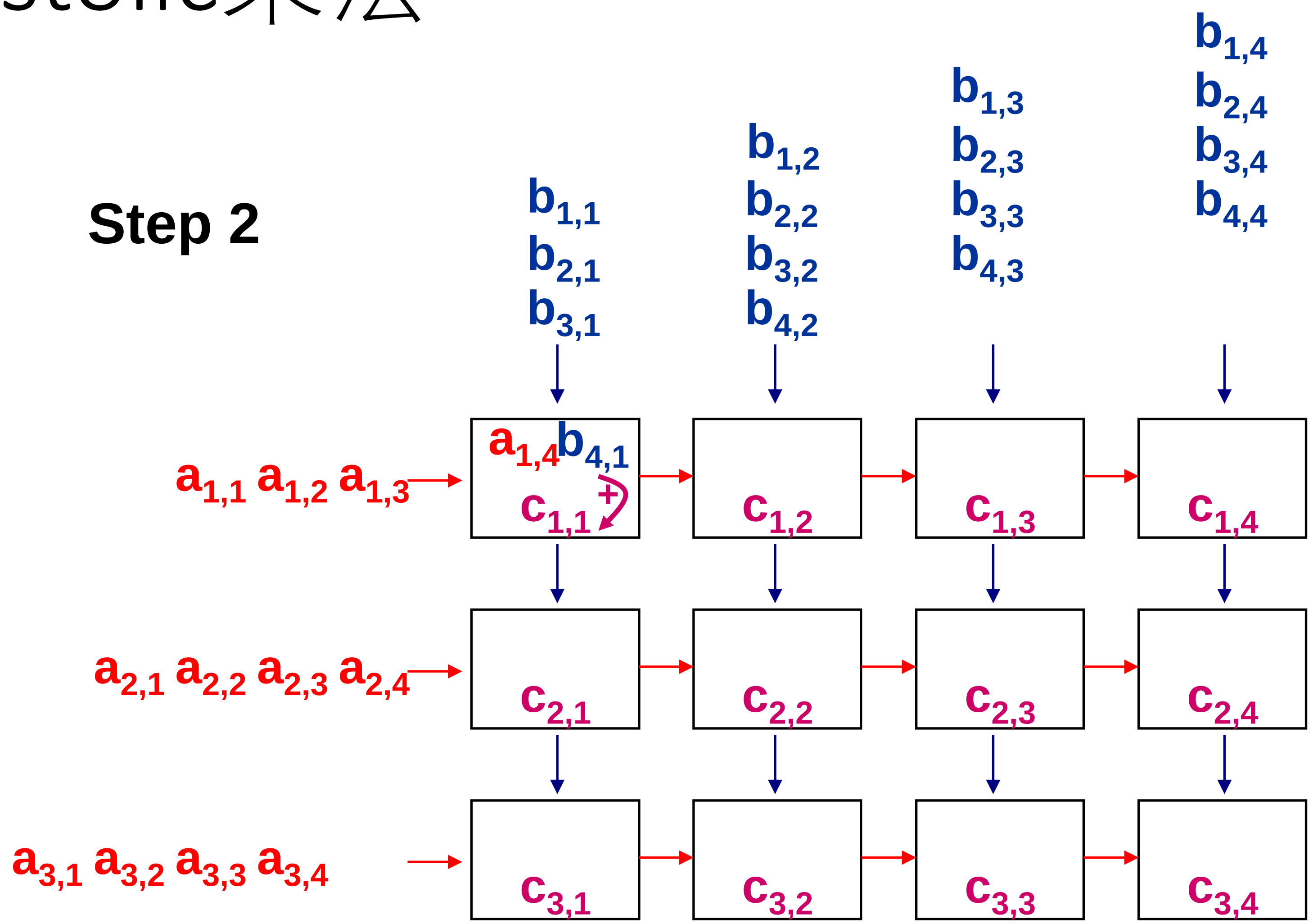
Systolic乘法

- 示例： $A_{4\times4}$， $B_{4\times4}$， $p=16$



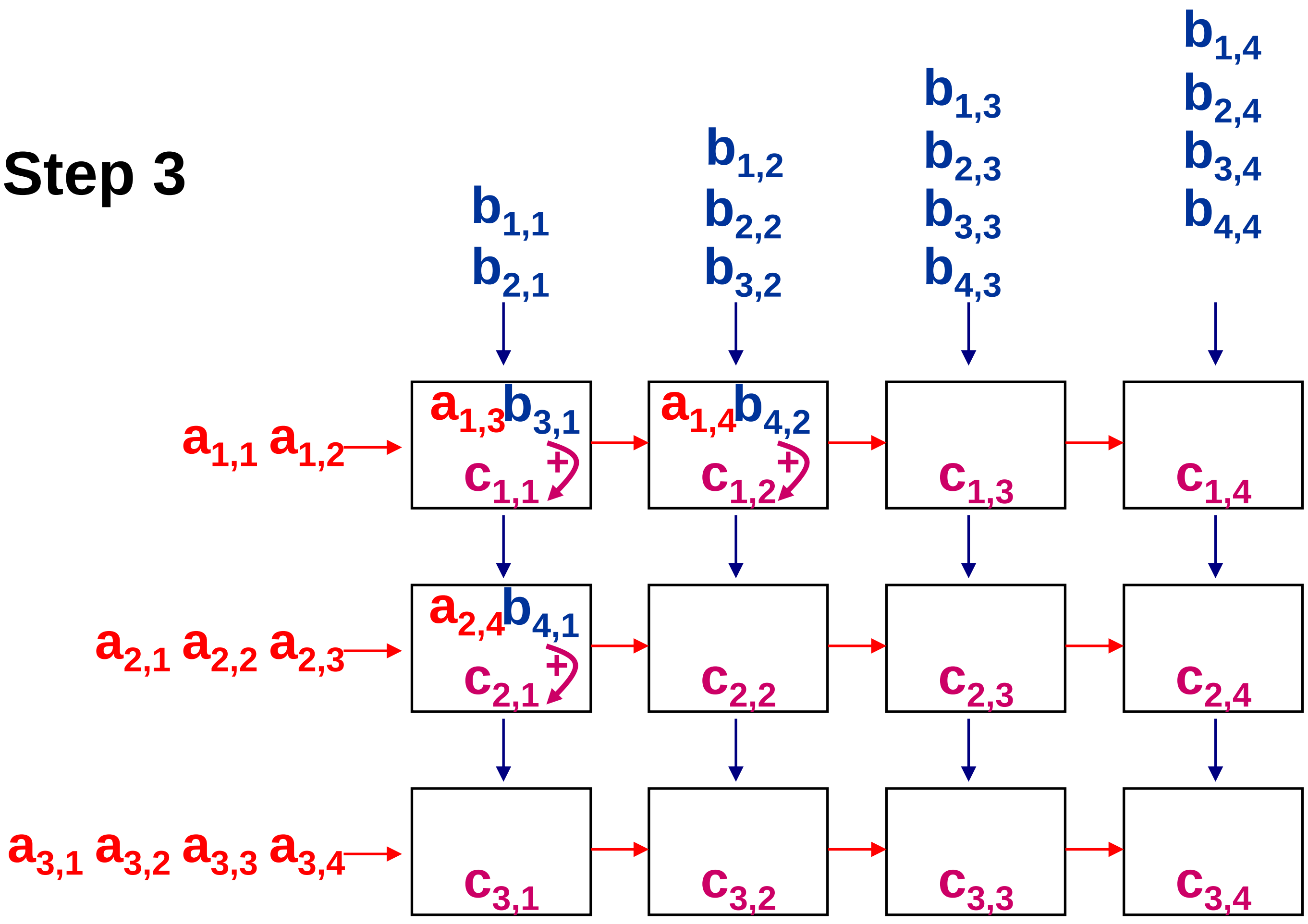
Systolic乘法

Step 2



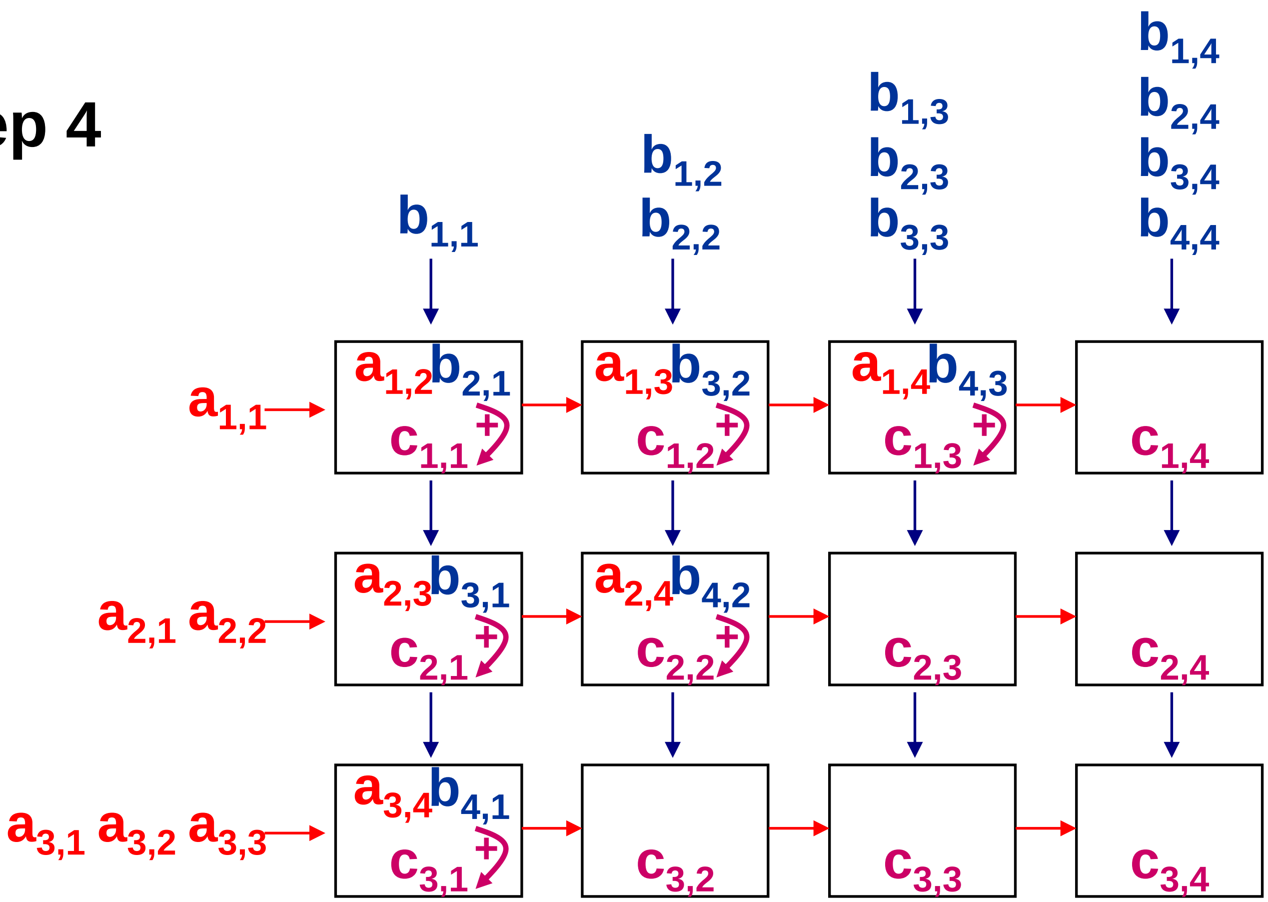
Systolic乘法

Step 3



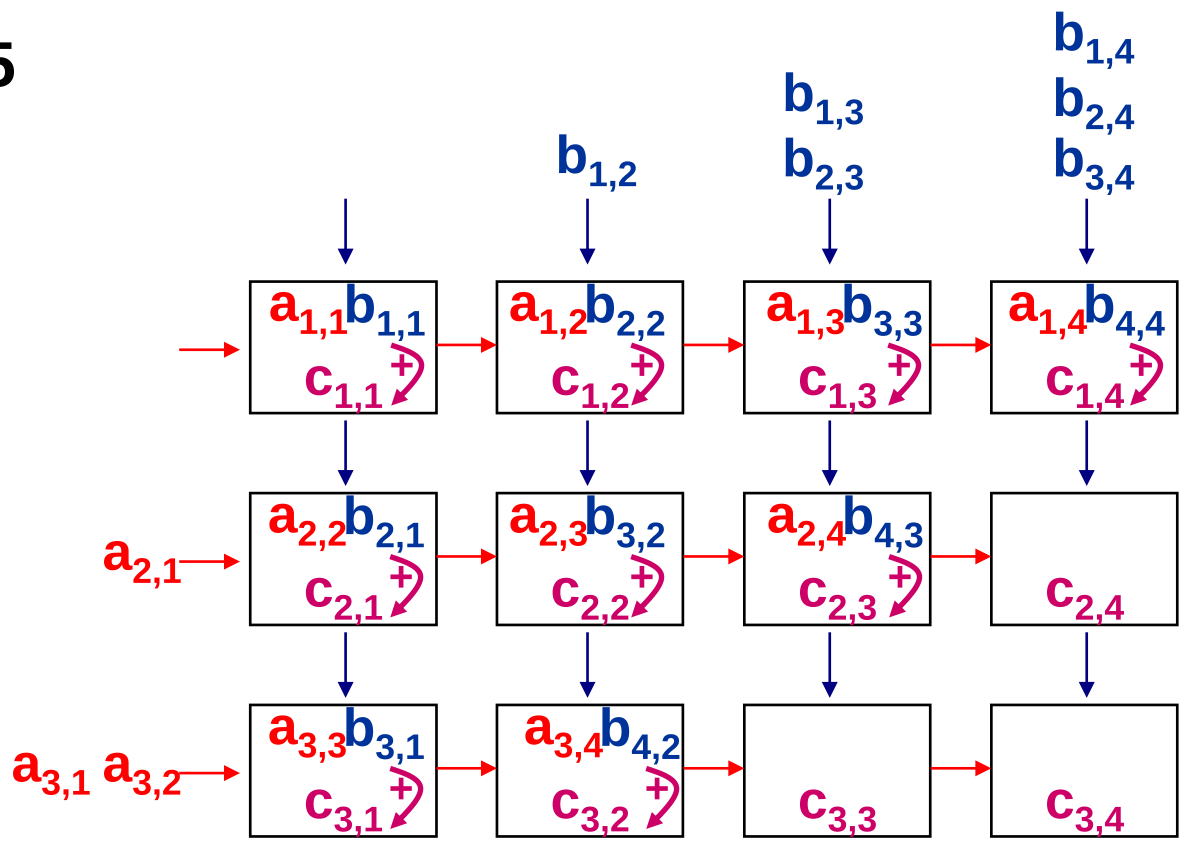
Systolic乘法

Step 4



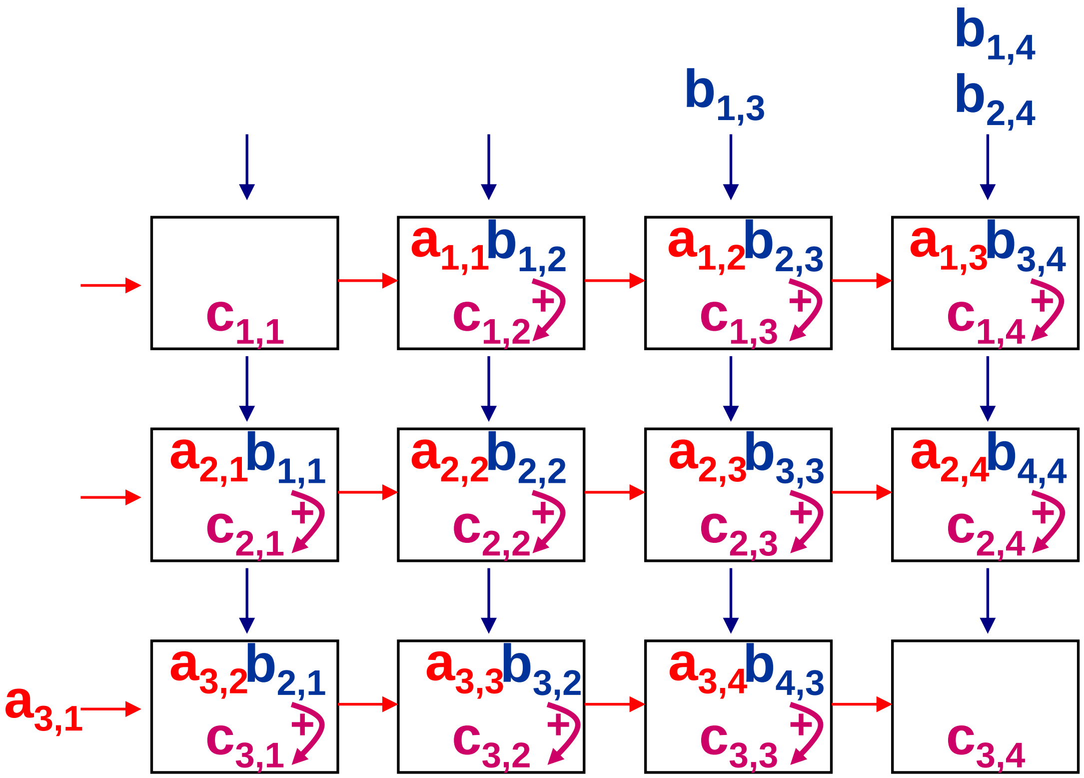
Systolic乘法

Step 5



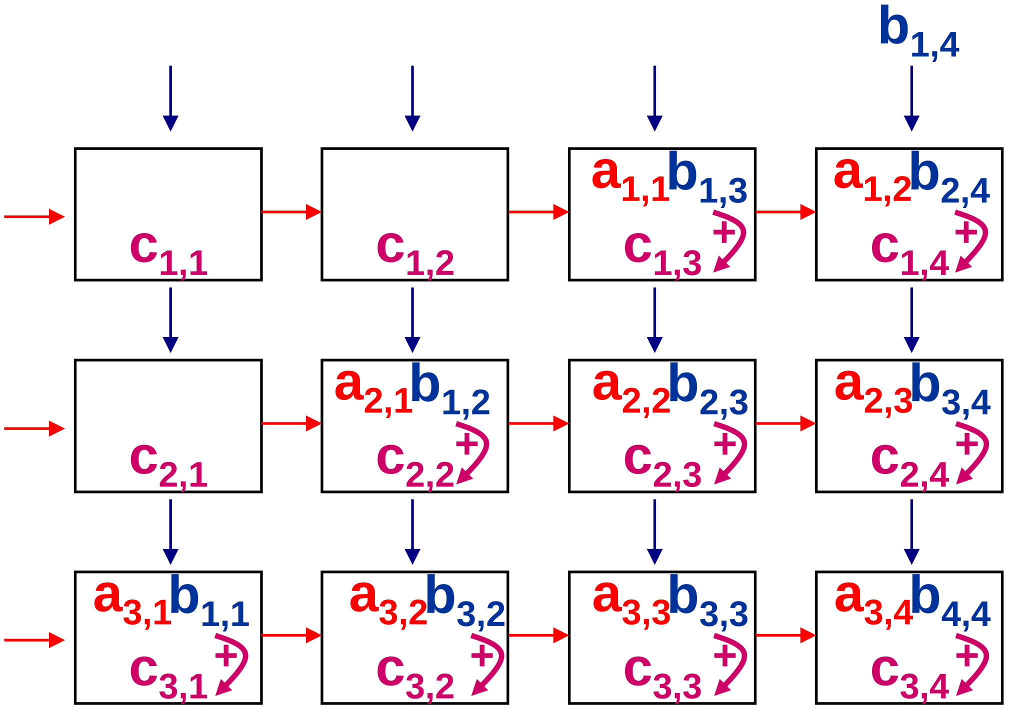
Systolic乘法

Step 6



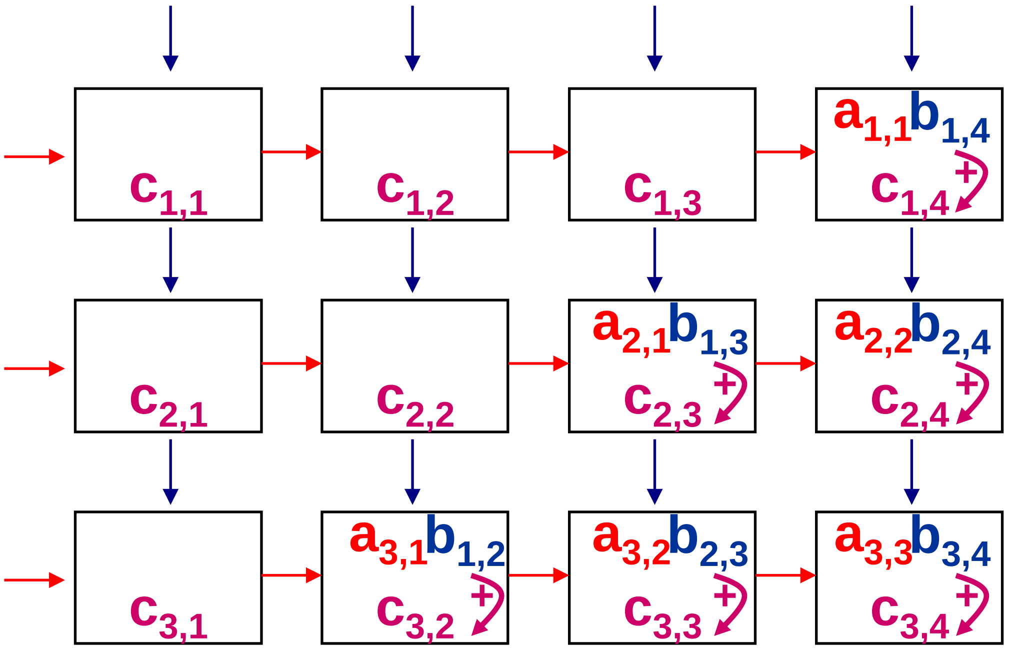
Systolic乘法

Step 7



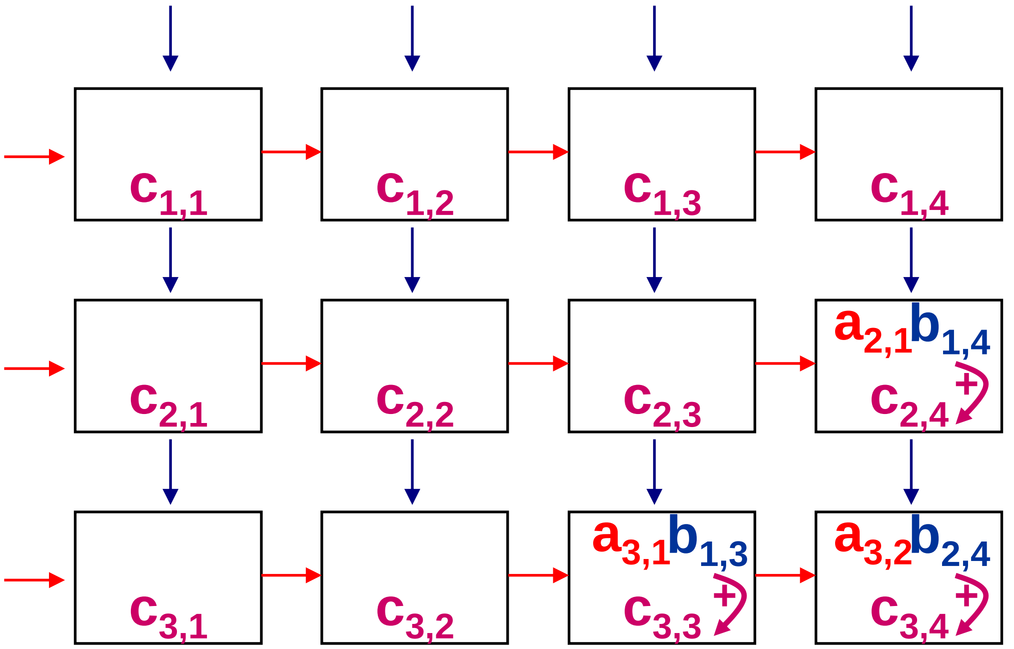
Systolic乘法

Step 8



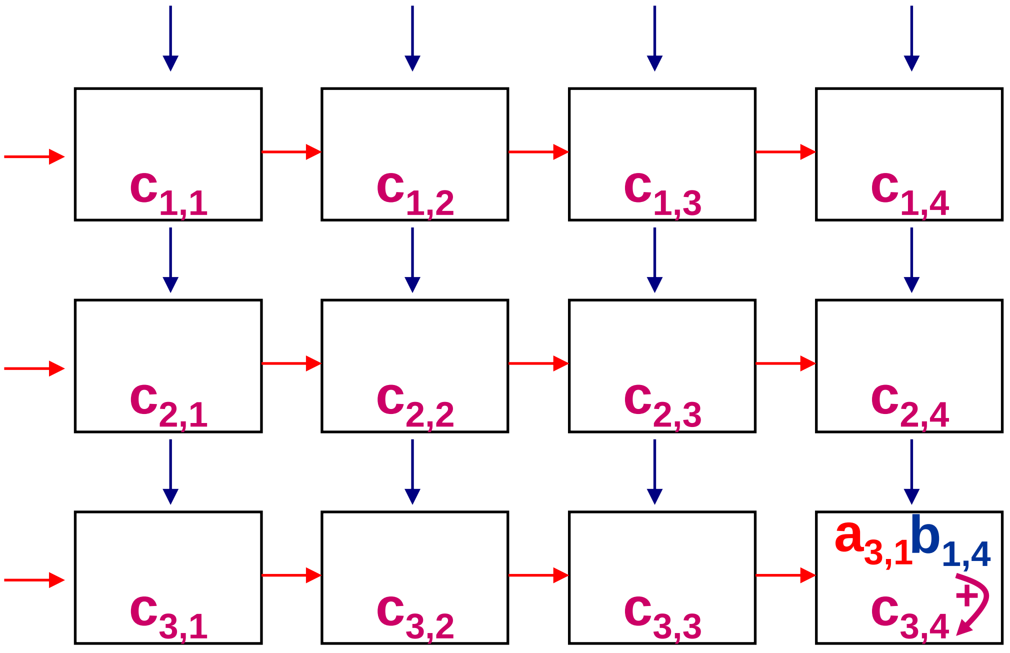
Systolic乘法

Step 9



Systolic乘法

Step 10



Systolic乘法

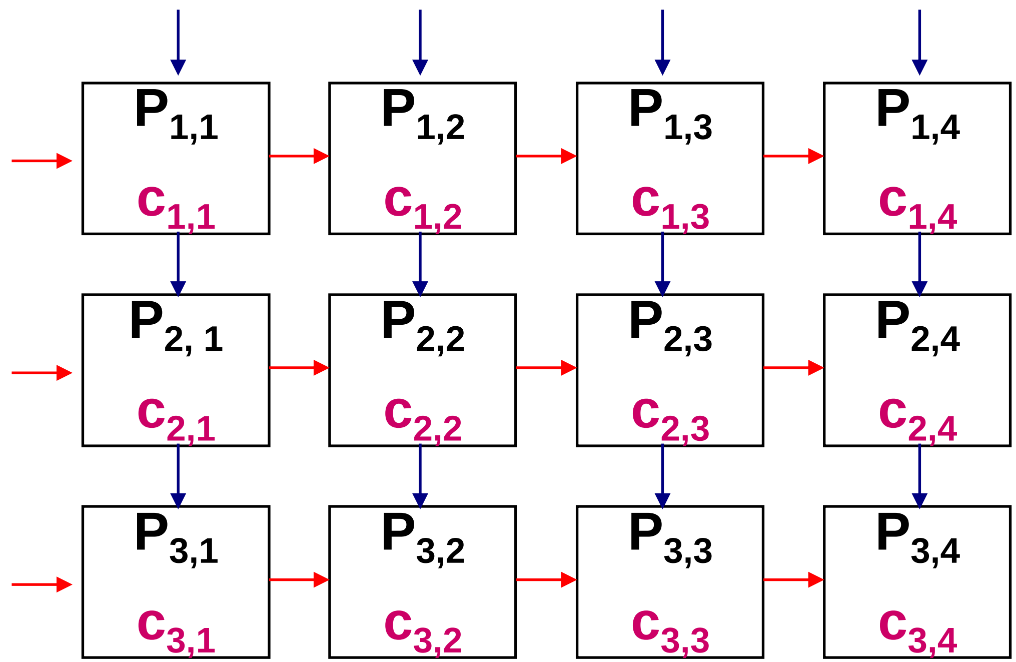
$$c_{1,1} = a_{1,1}\, b_{1,1} + a_{1,2}\, b_{2,1} + a_{1,3}\, b_{3,1} + a_{1,4}\, b_{4,1}$$

$$c_{1,2} = a_{1,1}\, b_{1,2} + a_{1,2}\, b_{2,2} + a_{1,3}\, b_{3,2} + a_{1,4}\, b_{4,2}$$

Over

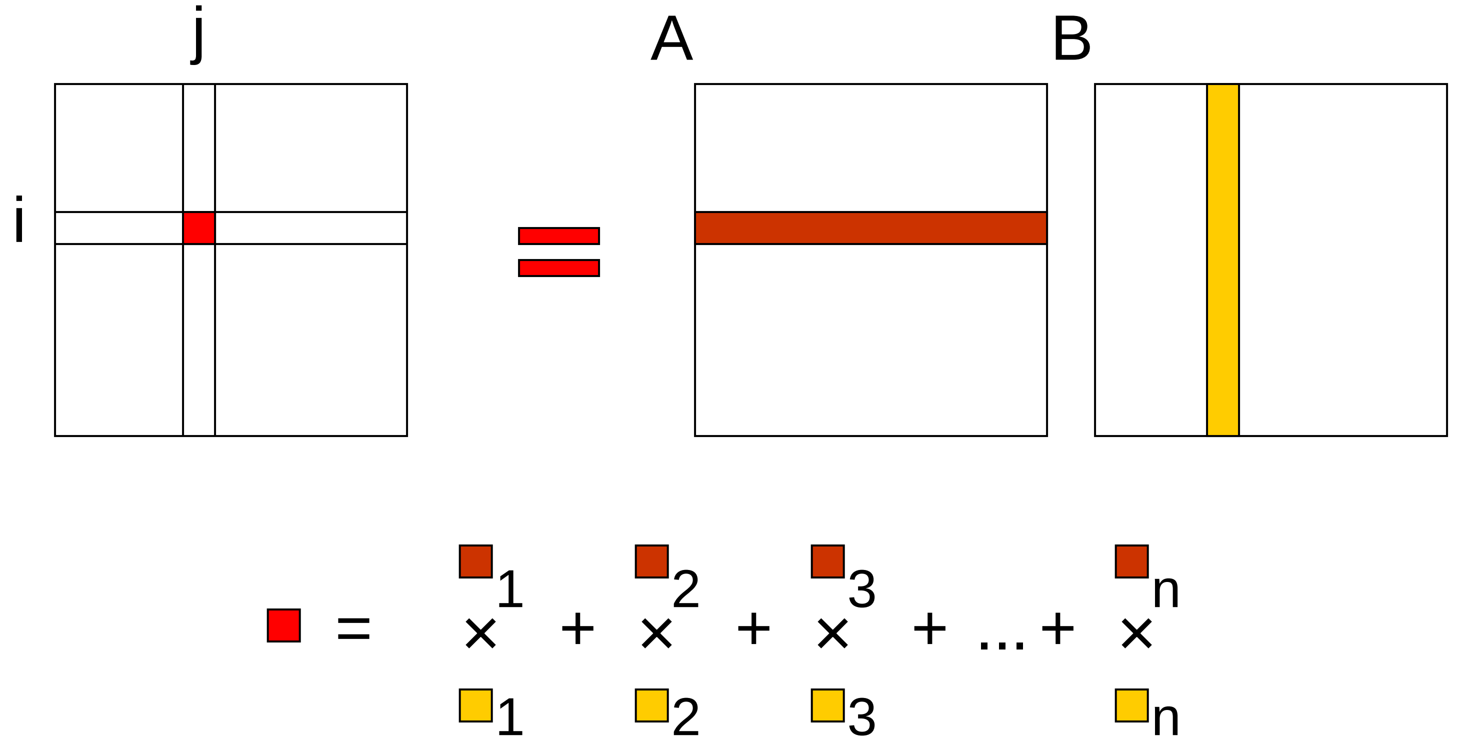
....

$$c_{3,4} = a_{3,1}\, b_{1,4} + a_{3,2}\, b_{2,4} + a_{3,3}\, b_{3,4} + a_{3,4}\, b_{4,4}$$



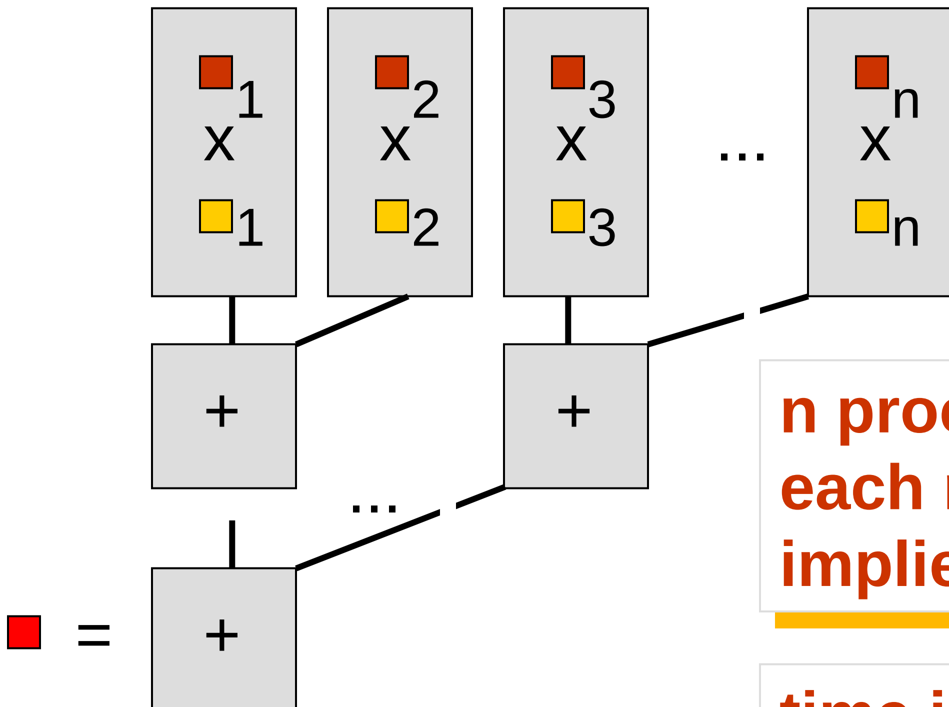
DNS乘法

- Motivation: From a good and common idea



DNS乘法

- Motivation: From a good and common idea



n processors for each result element implies n x n^2 = n^3

time is $\log_2$ n

DNS乘法

- 背景: 1981年由Dekel、Nassimi和Sahni提出的SIMD-CC上的矩阵乘法, 处理器数目为n^3, 运行时间为O(logn), 是一种速度很快的算法。
- 基本思想: 通过一到一和一到多的播送办法，使得处理器(k,i,j)拥有$a_{i,k}$和$b_{k,j}$, 进行本地相乘,再沿k方向进行单点积累求和,结果存储在处理器(0,i,j)中。

DNS乘法

- 处理器编号: 处理器数$p=n^3=(2^q)^3=2^{3q}$, 处理器P_r位于位置(k,i,j), 这里$r=kn^2+in+j$, ($0\leqslant i, j, k\leqslant n-1$)。位于(k,i,j)的处理器$P_r$的三个寄存器$A_r$,$B_r$,$C_r$分别表示为A[k,i,j], B[k,i,j]和C[k,i,j], 初始时均为0。
- 算法: 初始时$a_{i,j}$和$b_{i,j}$存储于寄存器A[0,i,j]和B[0,i,j];

  ①数据复制:A,B同时在k维复制(一到一播送);

  A在j维复制(一到多播送); B在i维复制(一到多播送);

  ②相乘运算:所有处理器的A、B寄存器两两相乘;

  ③求和运算:沿k方向进行单点积累求和;

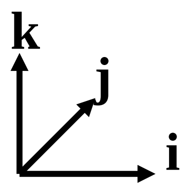
DNS乘法

- 示例

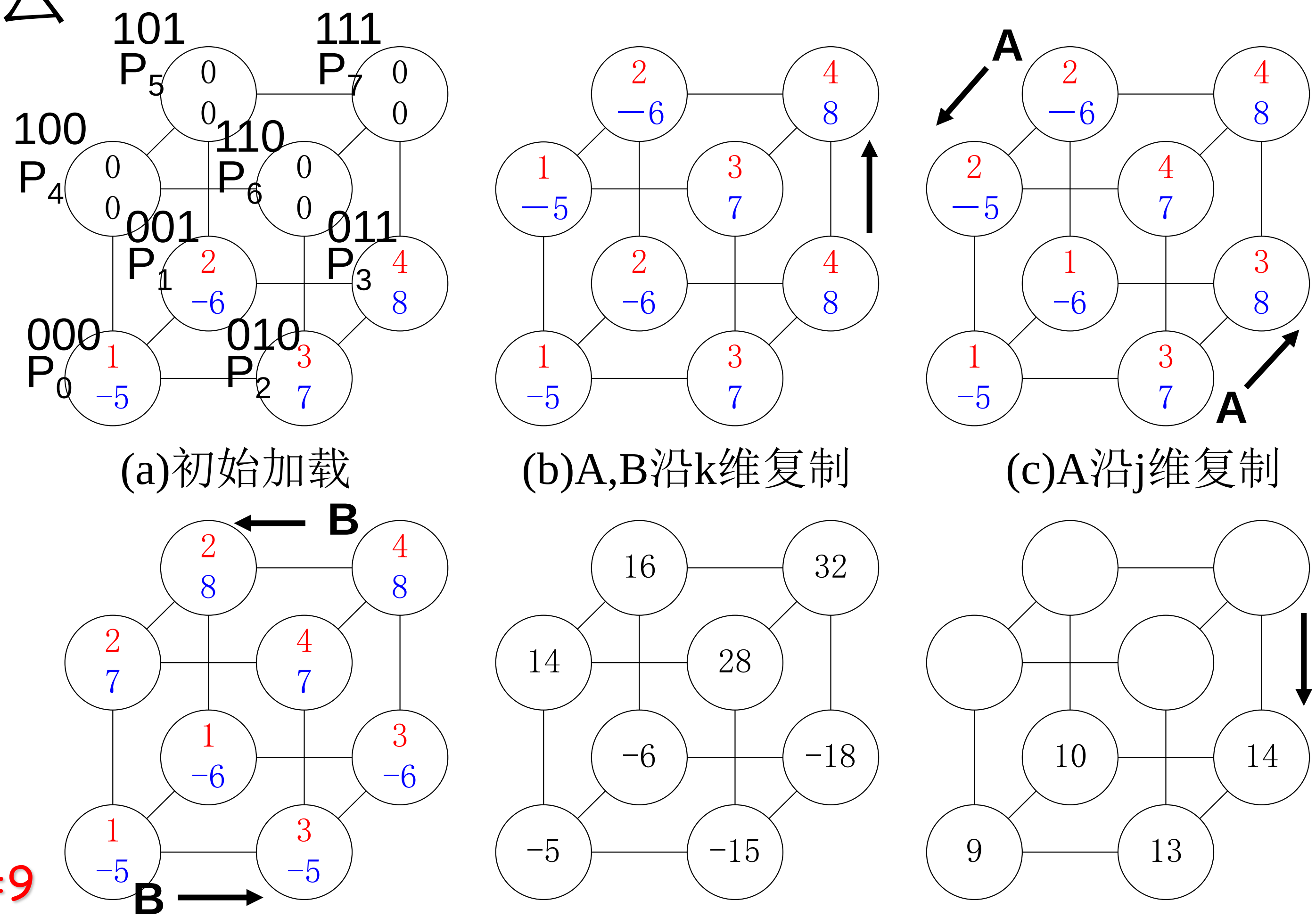
$$A=\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}$$

$$B=\begin{pmatrix}-5 & -6\\ 7 & 8\end{pmatrix}$$

求$C=A\times B$



C00=1×(-5)+2×7=9
C01=1×(-6)+2×8=10
C10=3×(-5)+4×7=13
C11=3×(-6)+4×8=14



(a)初始加载 (b)A,B沿k维复制 (c)A沿j维复制
(d)B沿i维复制 (e)点积 (f)沿k维求和

图9. 12

DNS乘法

//令$r^{(m)}$表示r的第m位取反;

//{p, r_m=d}表示r(0≤r≤p-1)的集合, 这里r的二

//进制第m位为d;

//输入: $A_{n\times n}$, $B_{n\times n}$;　输出: $C_{n\times n}$

Begin //以n=2, p=8=2^3举例, q=1, r=$(r_2r_1r_0)_2$

(1)for m=3q-1 to 2q do //按k维复制A,B, m=2

for all r in {p, r_m=0} par-do //r_2=0的r

(1.1) $A_{r(m)} \leftarrow A_r$ //A(100)←A(000)等

(1.2) $B_{r(m)} \leftarrow B_r$ //B(100)←B(000)等

endfor

endfor

(2)for m=q-1 to 0 do //按j维复制A, m=0

for all r in {p, r_m= r_{2q+m}} par-do //r_0=r_2的r

$A_{r(m)} \leftarrow A_r$ //A(001)←A(000),A(100)←A(101)

endfor //A(011)←A(010),A(110)←A(111)

endfor

(3)for m=2q-1 to q do //按i维复制B,m=1

for all r in {p, r_m= r_{q+m}} par-do//r_1=r_2的r

$B_{r(m)} \leftarrow B_r$ //B(010)←B(000),B(100)←B(110)

endfor //B(011)←B(001),B(101)←B(111)

endfor

(4)for r=0 to p-1 par-do //相乘, all P_r

$C_r=A_r\times B_r$

endfor

(5)for m=2q to 3q-1 do //求和,m=2

for r=0 to p-1 par-do

$C_r=C_r+C_{r(m)}$

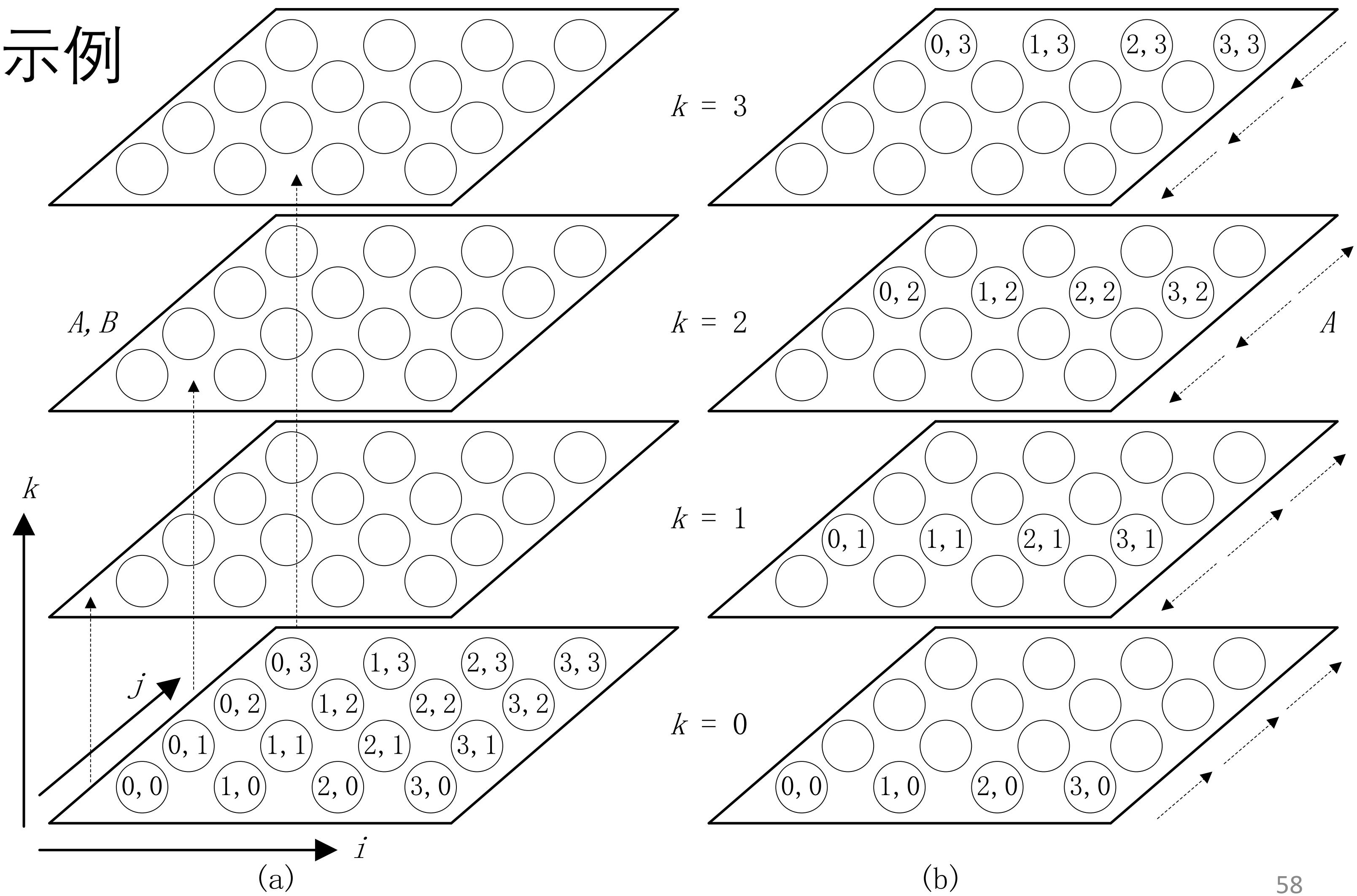
endfor

endfor

End

DNS乘法

- 示例



DNS乘法

- 示例