



COSC 6365  
Lecture 20  
2008-03-27

**CS@UH**

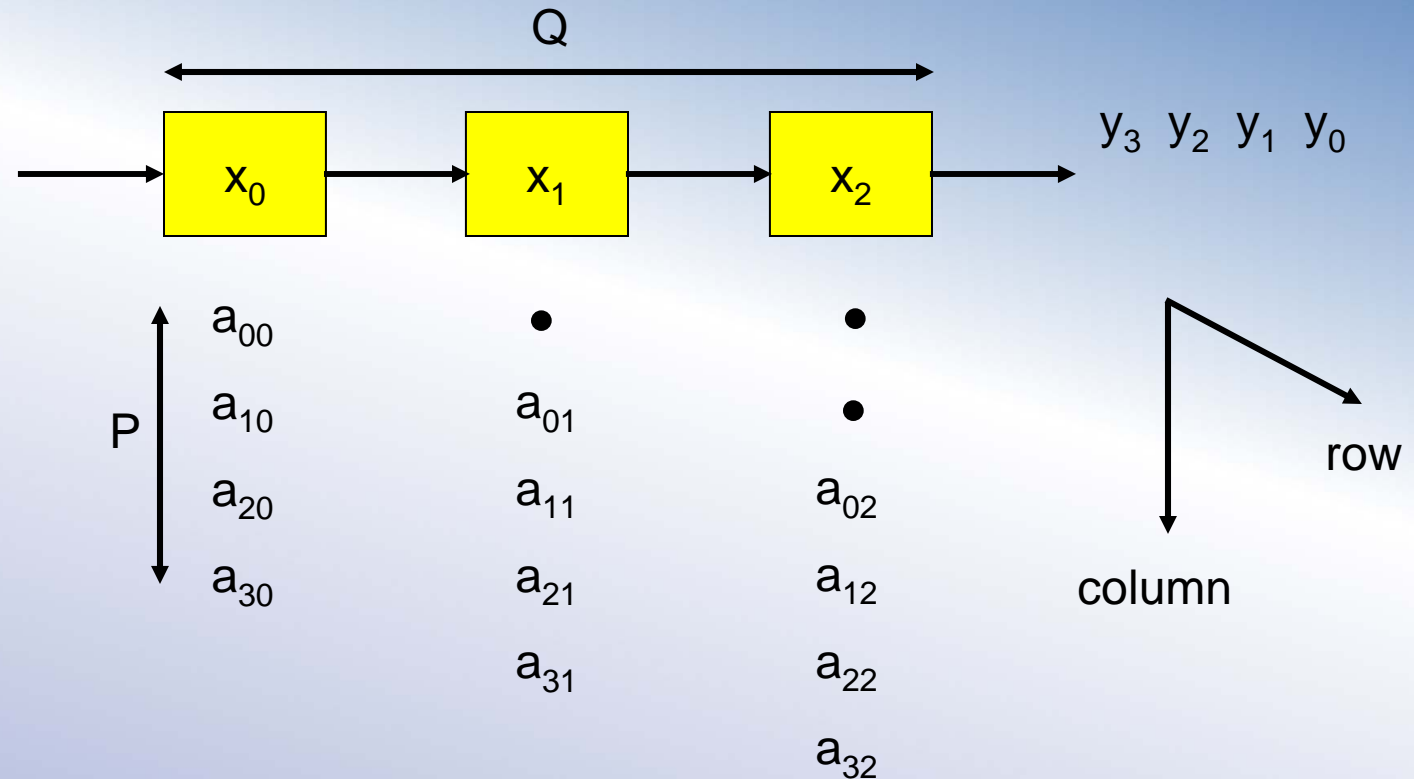
# Introduction to HPC

## Lecture 20

Lennart Johnsson  
Dept of Computer Science  
Director TLC<sup>2</sup>



# Matrix-Vector Multiplication



$$\text{Time: } (1+2(Q-1)+2(P-1))t_a + (P+Q-1)t_c = (2(P+Q)-3)t_a + (P+Q-1)t_c$$

$$\text{Speed-up: } \frac{2PQt_a}{(2(P+Q)-3)t_a + (P+Q-1)t_c}$$

$$\text{Efficiency: } \frac{2Pt_a}{(2(P+Q)-3)t_a + (P+Q-1)t_c}$$



# Matrix-Vector Multiplication

$$\text{Efficiency: } \frac{2P}{(2(P+Q)-3)+(P+Q-1)t_c/t_a}$$

$P \gg Q$ : Efficiency  $O(1)$

$Q \gg P$ : Efficiency  $O(P/Q)$

Why this difference in efficiency?

$P \gg Q$ : Time is  $O(P)$ , Speed-up is  $O(Q)$

Small degree of parallelism compared to workload

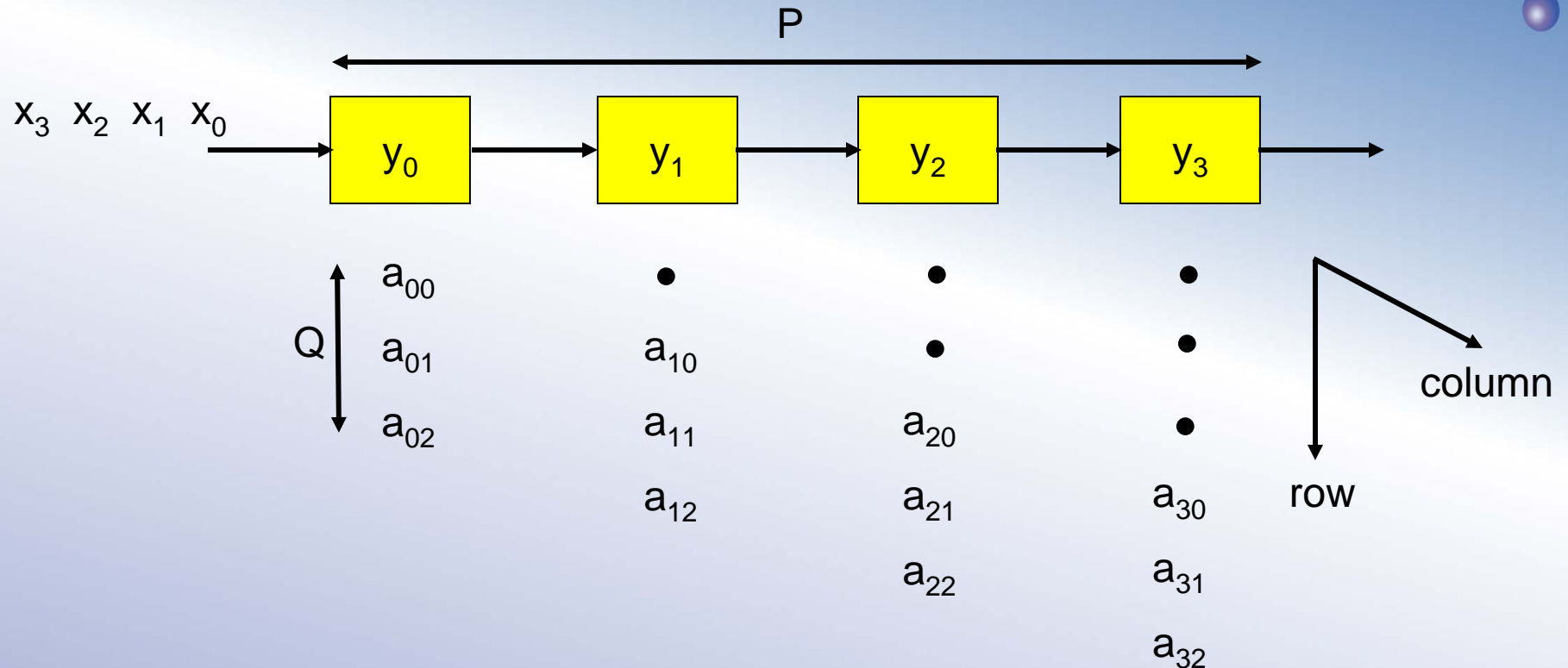
$Q \gg P$ : Time is  $O(Q)$ , Speed-up is  $O(P)$

Small degree of parallelism compared to workload. Time dominated by pipeline delay.

How can the efficiency be increased for  $Q \gg P$ ?



# Matrix-Vector Multiplication



$$\text{Time: } (1+2(Q-1)+2(P-1))t_a + (P+Q-1)t_c = (2(P+Q)-3)t_a + (P+Q-1)t_c$$

$$\text{Speed-up: } \frac{2PQt_a}{(2(P+Q)-3)t_a + (P+Q-1)t_c}$$

$$\text{Efficiency: } \frac{2Qt_a}{(2(P+Q)-3)t_a + (P+Q-1)t_c}$$



# Matrix-Vector Multiplication

$$\text{Efficiency: } \frac{2Q}{(2(P+Q)-3)+(P+Q-1)t_c/t_a}$$

$P \gg Q$ : Efficiency  $O(Q/P)$

$Q \gg P$ : Efficiency  $O(1)$

Why this difference in efficiency?

$P \gg Q$ : Time is  $O(P)$ , Speed-up is  $O(Q)$

Small degree of parallelism compared to workload. Time dominated by pipeline delay.

$Q \gg P$ : Time is  $O(Q)$ , Speed-up is  $O(P)$

Small degree of parallelism compared to workload.



COSC 6365  
Lecture 20  
2008-03-27

CS@UH

# Matrix-Vector Multiplication

What did we learn?

Small degree of parallelism compared to workload:  
good efficiency

Large degree of parallelism compared to workload:  
poor efficiency

Choose algorithm and data distribution  
depending on the problem at hand



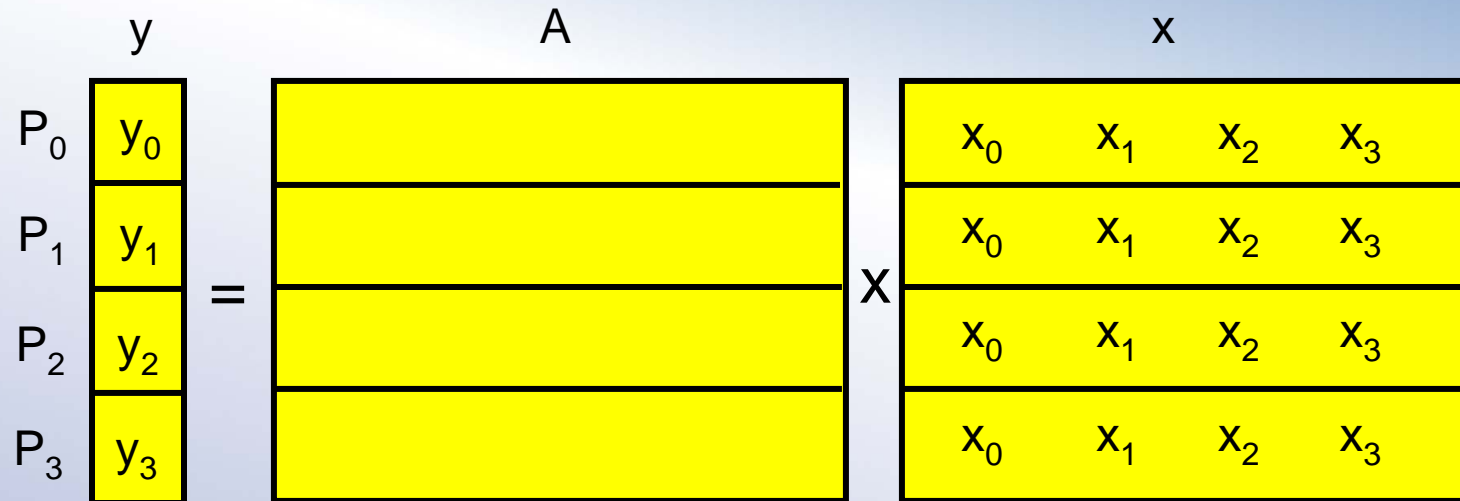
# Matrix-Vector Multiplication

$$\begin{array}{c} y \\ P_0 \\ P_1 \\ P_2 \\ P_3 \end{array} \begin{array}{|c|} \hline y_0 \\ \hline y_1 \\ \hline y_2 \\ \hline y_3 \\ \hline \end{array} = \begin{array}{c} A \\ \hline \hline \hline \hline \end{array} \begin{array}{c} x \\ X_0 \\ X_1 \\ X_2 \\ X_3 \end{array}$$



# Matrix-Vector Multiplication

All-to-All Broadcast of  $x$



Local Matrix-Vector Multiplication

Time:  $P/N(2Q-1)t_a + Q/N(N-1)t_c$

Speed-up:  $\frac{2PQt_a}{P/N(2Q-1)t_a + Q/N(N-1)t_c}$

Efficiency:  $\frac{2PQt_a}{P(2Q-1)t_a + Q(N-1)t_c}$



# Matrix-Vector Multiplication

$$\text{Efficiency: } \frac{2PQt_a}{P(2Q-1)t_a + Q(N-1)t_c}$$

$P \gg Nt_c/t_a$ : Efficiency  $O(1)$

$P=N$ : Efficiency  $\sim 2t_a/(2t_a+t_c)$

Example:

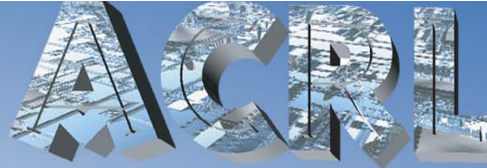
10GF core  $t_a = 10^{-10}$  sec

1 GigE comm  $t_c = 0.7 \times 10^{-7}$  sec for 64-bit data

Efficiency:  $\sim 2t_a/t_c = \sim 3 \times 10^{-3}$

Efficiency is  $O(1)$  when parallelism is small compared to the workload, especially if  $t_a \gg t_c$

$P=N$ : Communication time dominates for typical values of  $t_a$  and  $t_c$



ADVANCED COMPUTING RESEARCH LABORATORY

COSC 6365  
Lecture 20  
2008-03-27

**CS@UH**

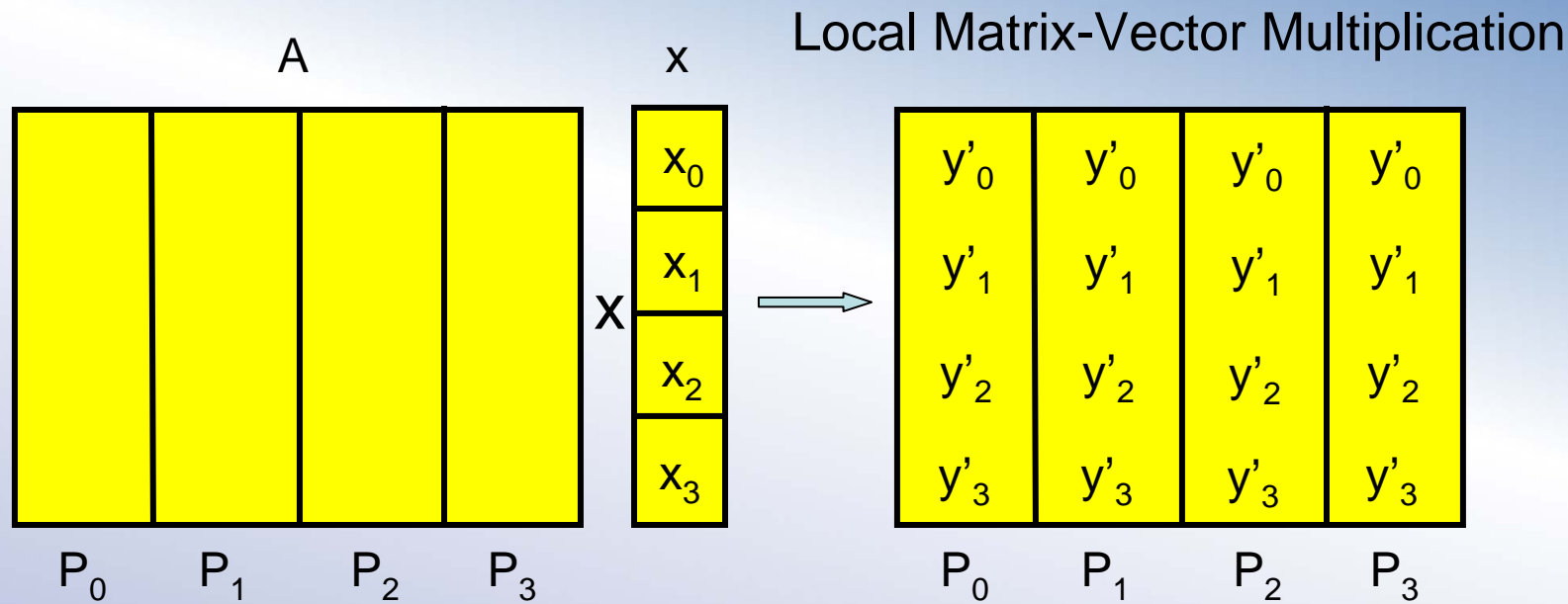
# Matrix-Vector Multiplication

$$\begin{array}{c} y \\ P_0 \\ P_1 \\ P_2 \\ P_3 \end{array} \begin{array}{c} y_0 \\ y_1 \\ y_2 \\ y_3 \end{array} = \begin{array}{c} A \\ P_0 \\ P_1 \\ P_2 \\ P_3 \end{array} \begin{array}{c} x \\ x_0 \\ x_1 \\ x_2 \\ x_3 \end{array}$$

The diagram illustrates the matrix-vector multiplication  $y = Ax$ . On the left, a vertical vector  $y$  is shown with elements  $y_0, y_1, y_2, y_3$  and row indices  $P_0, P_1, P_2, P_3$ . This is equal to a matrix  $A$  (represented by a yellow grid) multiplied by a vertical vector  $x$  with elements  $x_0, x_1, x_2, x_3$ . The matrix  $A$  has columns labeled  $P_0, P_1, P_2, P_3$  at the bottom. The multiplication is indicated by an equals sign and a large 'X' symbol.

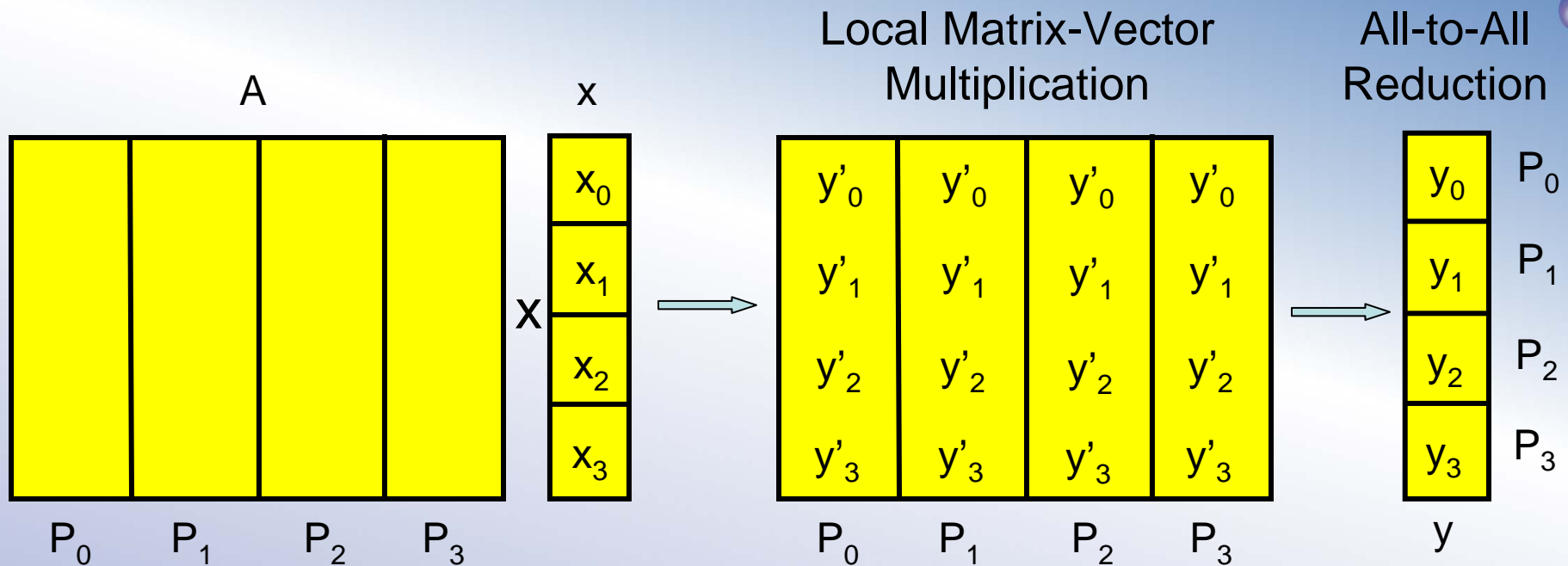


# Matrix-Vector Multiplication





# Matrix-Vector Multiplication



$$\text{Time: } P(2Q/N-1)t_a + P/N(N-1)(t_a + t_c)$$

$$\text{Speed-up: } \frac{2PQt_a}{P(2Q/N-1)t_a + P/N(N-1)(t_a + t_c)}$$

$$\text{Efficiency: } \frac{2PQt_a}{P(2Q-1)t_a + P(N-1)t_c}$$



# Matrix-Vector Multiplication

$$\text{Efficiency: } \frac{2PQt_a}{P(2Q-1)t_a + P(N-1)t_c}$$

$Q \gg Nt_c/t_a$  Efficiency  $O(1)$

$Q=N$ : Efficiency  $\sim 2t_a/(2t_a+t_c)$

Example:

10GF core  $t_a = 10^{-10}$  sec

1 GigE comm  $t_c = 0.7 \times 10^{-7}$  sec for 64-bit data

Efficiency:  $\sim 2t_a/t_c = \sim 3 \times 10^{-3}$

Efficiency is  $O(1)$  when parallelism is small compared to the workload, especially if  $t_a \gg t_c$

$Q=N$ : Communication time dominates for typical values of  $t_a$  and  $t_c$



COSC 6365  
Lecture 20  
2008-03-27

CS@UH

# Matrix-Vector Multiplication

What did we learn?

Small degree of parallelism compared to workload:  
good efficiency

Large degree of parallelism compared to workload:  
poor efficiency

Choose algorithm and data distribution  
depending on the problem at hand



# All-All-Broadcast

$P_0$	$P_1$	$P_2$	$P_3$
$X_0$	$X_1$	$X_2$	$X_3$

Step 1  
Left cyclic shift

$X_0$	$X_1$	$X_2$	$X_3$
$X_1$	$X_2$	$X_3$	$X_0$

Step 2  
Left cyclic shift

$X_0$	$X_1$	$X_2$	$X_3$
$X_1$	$X_2$	$X_3$	$X_0$
$X_2$	$X_3$	$X_0$	$X_1$

Step 3  
Left cyclic shift

$X_0$	$X_1$	$X_2$	$X_3$
$X_1$	$X_2$	$X_3$	$X_0$
$X_2$	$X_3$	$X_0$	$X_1$
$X_3$	$X_0$	$X_1$	$X_2$

No of Processors:  $N$

No of elem./proc.:  $P$

Time:  $(N-1)Pt_c$

Lower bound:  $(N-1)Pt_c$



# All-All-Reduction

$P_0$	$P_1$	$P_2$	$P_3$
$y_0$	$y_0$	$y_0$	$y_0$
$y_1$	$y_1$	$y_1$	$y_1$
$y_2$	$y_2$	$y_2$	$y_2$
$y_3$	$y_3$	$y_3$	$y_3$

$P_0$	$P_1$	$P_2$	$P_3$
$y_0$	$\oplus y_0$	$y_0$	$y_0$
$y_1$	$y_1$	$\oplus y_1$	$y_1$
$y_2$	$y_2$	$y_2$	$\oplus y_2$
$\oplus y_3$	$y_3$	$y_3$	$y_3$

$P_0$	$P_1$	$P_2$	$P_3$
$y_0$		$\oplus y_0$	$y_0$
$y_1$	$y_1$		$\oplus y_1$
$\oplus y_2$	$y_2$	$y_2$	
	$\oplus y_3$	$y_3$	$y_3$

$P_0$	$P_1$	$P_2$	$P_3$
$y_0$			$\oplus y_0$
$\oplus y_1$	$y_1$		
	$\oplus y_2$	$y_2$	
		$\oplus y_3$	$y_3$

Step 1 – right cyclic shift with addition

Step 2 – right cyclic shift with addition

Step 3 – right cyclic shift with addition

Time:  $(N-1)P/Nt_c$

Lower bound:  $(N-1)P/Nt_c$

No of Processors:  $N$

Final no of elem./proc.:  $P/N$