Scheduling in Distributed Systems

1. Local Scheduling:

- In a distributed system, each processor typically schedules its own processes without considering what other processors are doing.
- This works fine in most cases, but when processes on different processors need to communicate frequently, independent scheduling can cause delays.

2. Problem with Independent Scheduling:

- For example, if Process A (on one processor) frequently communicates with Process D (on another processor), there can be delays if their schedules don't align.
- A delay occurs when A sends a message to D, but D is not running yet. By the time D processes the message and replies, A may also not be running. This leads to long waiting times between message exchanges (e.g., 200 ms per exchange).

Processor

Tims slot	0	
0	A	C
	B	D
2	A	C
3	B	D
4	A	C
5	B	D

(a)

Processor

Time slot	0		2	3	4	5	6	7
0	X				X			
			X			X		
2		X			X		X	
3	X					X		
4		X		X				X
5			X		X			

(b)

Fig. 4-20. (a) Two jobs running out of phase with each other. (b) Scheduling matrix for eight processors, each with six time slots. The Xs indicated allocated slots.

3. Co-Scheduling Solution:

- To solve this, Ousterhout proposed **co-scheduling**—a technique where related processes that need to communicate frequently are scheduled to run at the same time.
- **Process Grouping:** Processes that communicate a lot are placed in the same "slot" on different processors. This ensures that when one process is running, all the others are too, maximizing communication efficiency.

4. Co-Scheduling Example (Matrix-Based Scheduling):

- Imagine a table (or matrix) where each column represents a processor, and each row represents processes in the same time slot on different processors.
- For example, Process A is placed in slot 1 on Processor 1, Process B in slot 1 on Processor 2, and so on.
- All processors run the processes in slot 1 at the same time, then move to slot 2, and so on, ensuring synchronized scheduling.