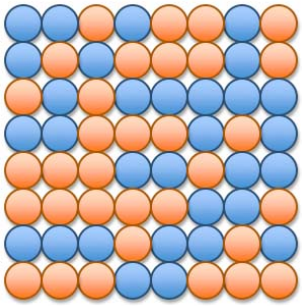


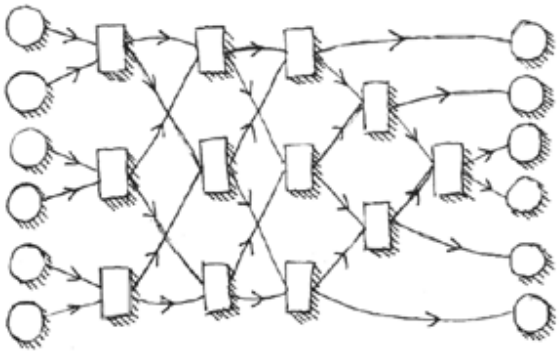
UNIT 10C

Concurrency: Pipelining & Distributed Processing

Review

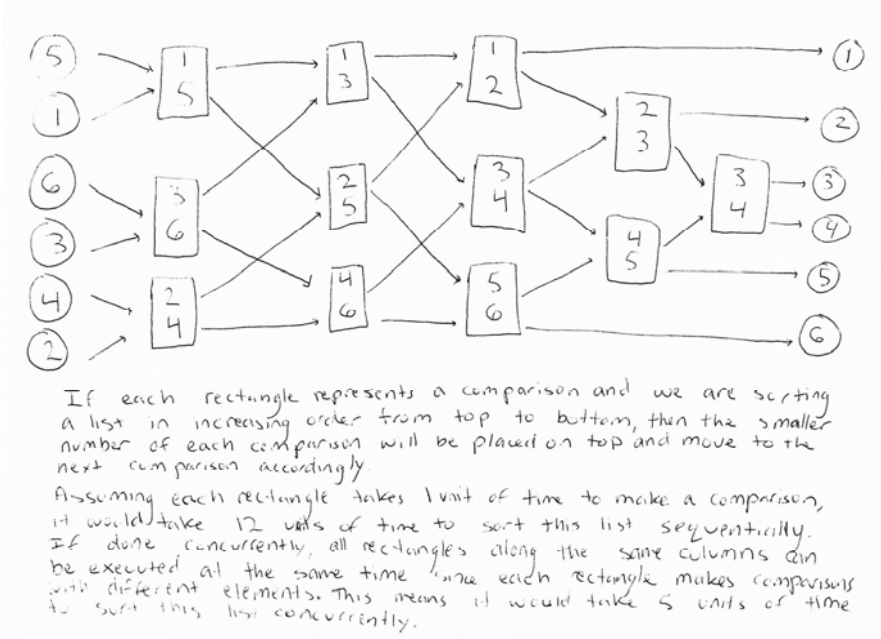
- **Multitasking** - The coordination of several computational processes on one processor or several cores.
- A **critical section** is a section of computer code that must only be executed by one process or thread at a time.
- **Deadlock** is the condition when two or more processes are all waiting for some shared resource that other processes of the group hold, causing all processes to wait forever without proceeding.

Activity: Sorting Network Simulation



Input: [5, 1, 6, 3, 4, 2]

How many steps does this take . . . sequentially? concurrently?



Pipelining

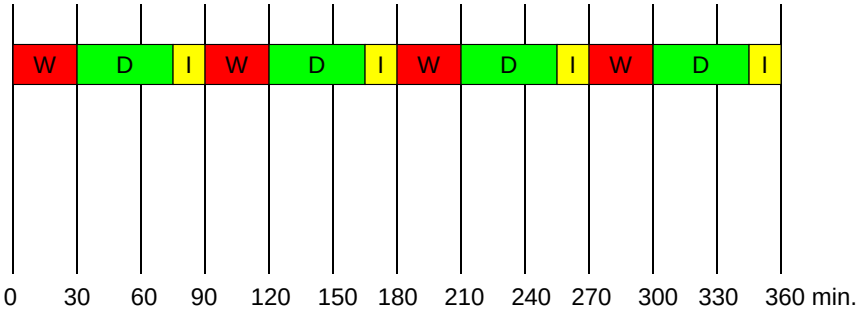
- Pipelining is similar to an assembly line.
  - Instead of completing one computation before starting another, each computation is split into simpler sub-steps, and computations are started as others are in progress.

Laundry Without Pipelining

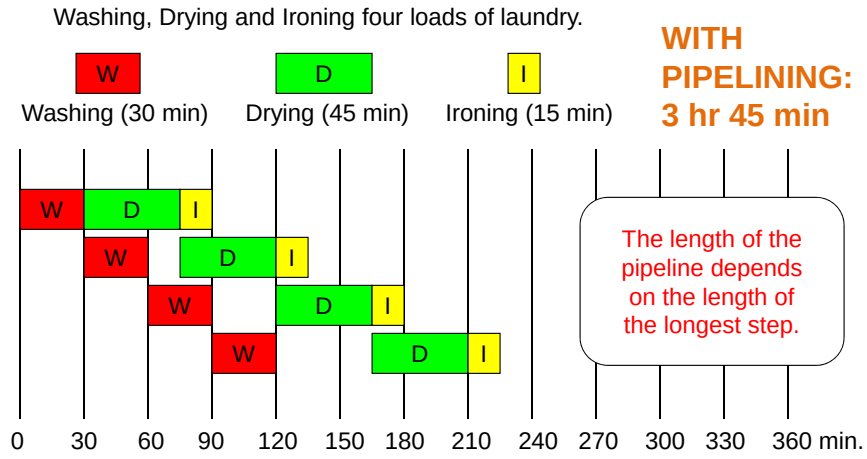
Washing, Drying and Ironing four loads of laundry.

W Washing (30 min) D Drying (45 min) I Ironing (15 min)

WITHOUT PIPELINING: 6 hr

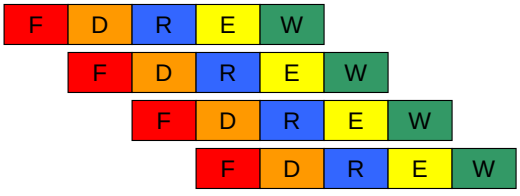


Laundry With Pipelining



Pipelining in Computing

- Fetch instruction from memory
- Decode the instruction
- Read data from registers
- Execute the instruction
- Write the result into a register



Dealing with Dependencies

```
ADD R3, R1
ADD R5, R3
ADD R8, R7
ADD R11, R10
```

"Add the contents of R1 and R3 and store the results in R3."

This instruction depends on the result of the previous instruction. (This will hold up the pipeline.)

```
ADD R3, R1
ADD R8, R7
ADD R11, R10
ADD R5, R3
```

Reorder the instructions to minimize the delay on the pipeline due to the dependency, if possible.

Dealing with Dependencies

```
A: ADD R3, R1
   SUB R6, R4
   JMN A, R6
   MOV R3, R1
```

"Jump to label A if R6 != 0."

The JMN instruction will stall in the pipeline since the final values of R6 is not known yet.

Possible solutions:

1. Assume the jump occurs. If we find later that R6 is equal to 0, clear the pipeline and begin computing with the MOV instruction.
2. Start decoding the ADD and MOV instructions. When we know if R6 is equal to 0 or not, send the appropriate instructions into the pipeline for completion.

Matrix Multiplication

	hw	paper	exam1	exam2	exam3	final
student1	95	90	93	91	85	92
student2	73	80	75	63	79	75
student3	85	73	80	85	88	91
student4	50	65	50	60	56	47
student5	100	95	98	96	96	90
student6	75	75	75	75	75	75
student7	90	80	80	90	100	100
student8	88	80	80	70	60	55

	weight
hw	0.15
paper	0.1
exam1	0.15
exam2	0.15
exam3	0.15
final	0.3

	average
student1	
student2	
student3	
student4	
student5	
student6	
student7	
student8	

Matrix Multiplication

0 + 95*0.15 + 90*0.1 + 93*0.15 + 91*0.15 + 85*0.15 + 92*0.3 = 91.2

	hw	paper	exam1	exam2	exam3	final
student1	95	90	93	91	85	92
student2	73	80	75	63	79	75
student3	85	73	80	85	88	91
student4	50	65	50	60	56	47
student5	100	95	98	96	96	90
student6	75	75	75	75	75	75
student7	90	80	80	90	100	100
student8	88	80	80	70	60	55

	weight
hw	0.15
paper	0.1
exam1	0.15
exam2	0.15
exam3	0.15
final	0.3

	average
student1	91.2
student2	
student3	
student4	
student5	
student6	
student7	
student8	

Matrix Multiplication

0 + 73*0.15 + 80*0.1 + 75*0.15 + 63*0.15 + 79*0.15 + 75*0.3 = 74.0

	hw	paper	exam1	exam2	exam3	final
student1	95	90	93	91	85	92
student2	73	80	75	63	79	75
student3	85	73	80	85	88	91
student4	50	65	50	60	56	47
student5	100	95	98	96	96	90
student6	75	75	75	75	75	75
student7	90	80	80	90	100	100
student8	88	80	80	70	60	55

	weight
hw	0.15
paper	0.1
exam1	0.15
exam2	0.15
exam3	0.15
final	0.3

	average
student1	91.2
student2	74.0
student3	
student4	
student5	
student6	
student7	
student8	

Matrix Multiplication

0 + 85*0.15 + 73*0.1 + 80*0.15 + 85*0.15 + 88*0.15 + 91*0.3 = 85.3

	hw	paper	exam1	exam2	exam3	final
student1	95	90	93	91	85	92
student2	73	80	75	63	79	75
student3	85	73	80	85	88	91
student4	50	65	50	60	56	47
student5	100	95	98	96	96	90
student6	75	75	75	75	75	75
student7	90	80	80	90	100	100
student8	88	80	80	70	60	55

	weight
hw	0.15
paper	0.1
exam1	0.15
exam2	0.15
exam3	0.15
final	0.3

	average
student1	91.2
student2	74.0
student3	85.3
student4	
student5	
student6	
student7	
student8	

....and so on...

Matrix Multiplication

If each multiply/add takes 1 time unit,
this non-pipelined matrix multiplication takes 48 time units.

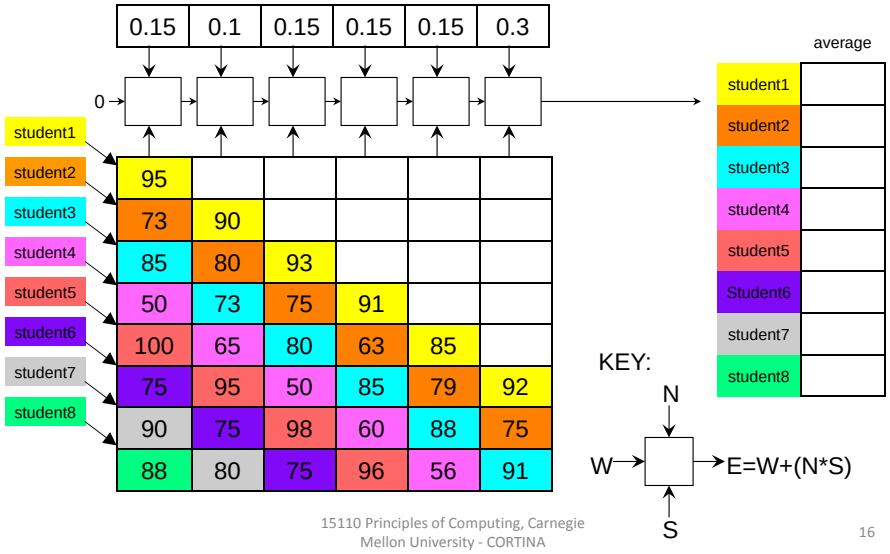
	hw	paper	exam1	exam2	exam3	final
student1	95	90	93	91	85	92
student2	73	80	75	63	79	75
student3	85	73	80	85	88	91
student4	50	65	50	60	56	47
student5	100	95	98	96	96	90
student6	75	75	75	75	75	75
student7	90	80	80	90	100	100
student8	88	80	80	70	60	55

	weight
hw	0.15
paper	0.1
exam1	0.15
exam2	0.15
exam3	0.15
final	0.3

	average
student1	91.2
student2	74.0
student3	85.3
student4	53.0
student5	95.0
student6	75.0
student7	92.0
student8	69.2

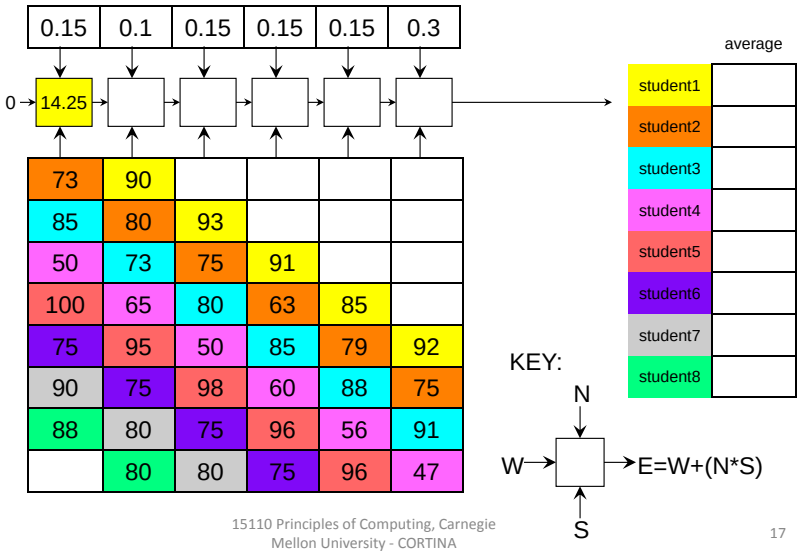
Faster Matrix Multiplication

using Pipelining



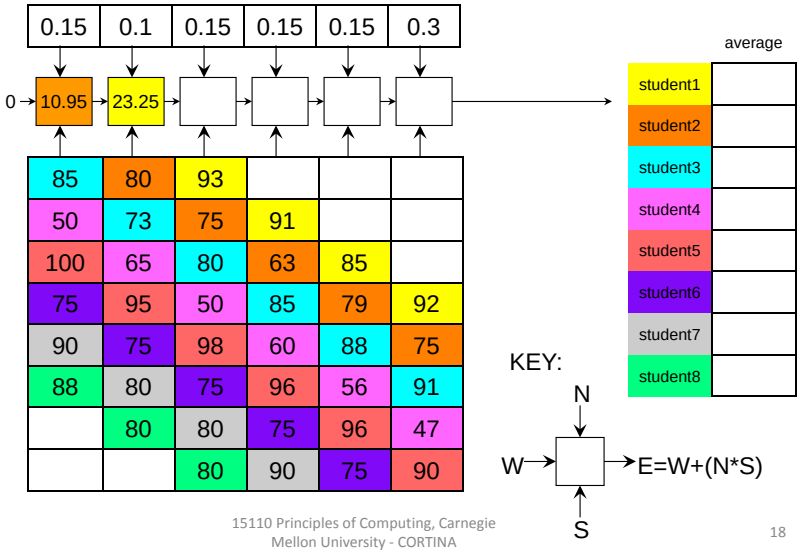
Faster Matrix Multiplication

using Pipelining



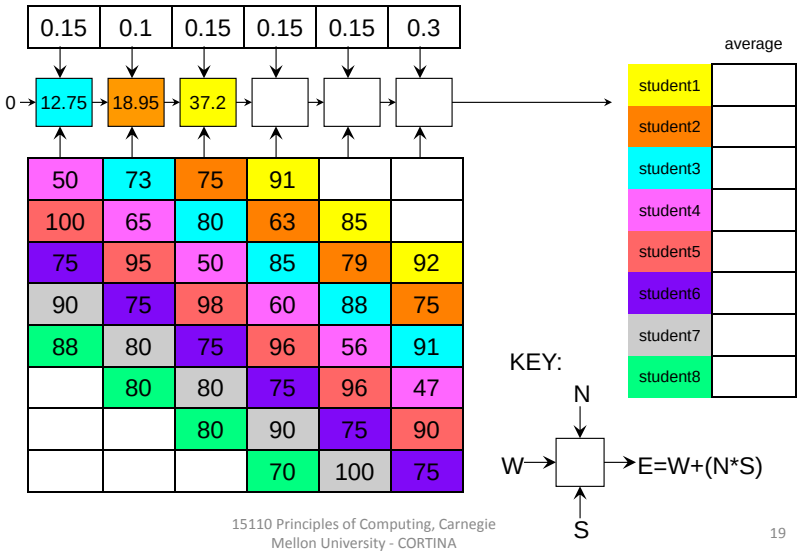
Faster Matrix Multiplication

using Pipelining



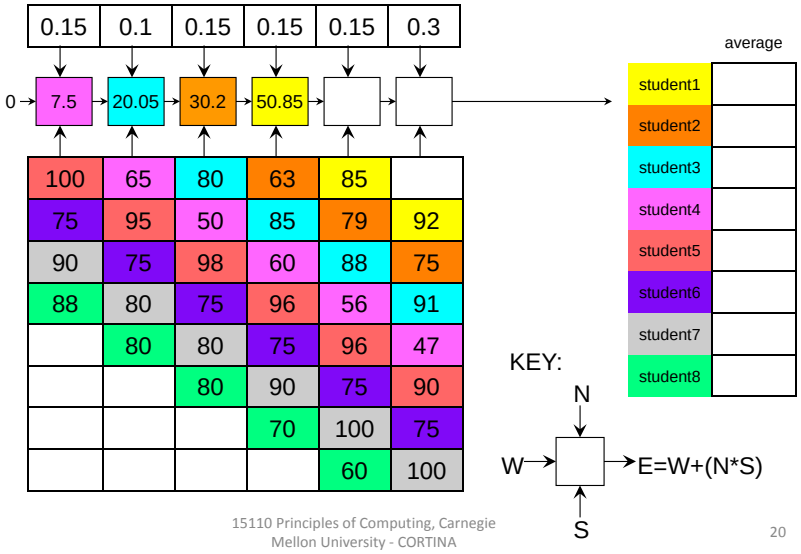
Faster Matrix Multiplication

using Pipelining



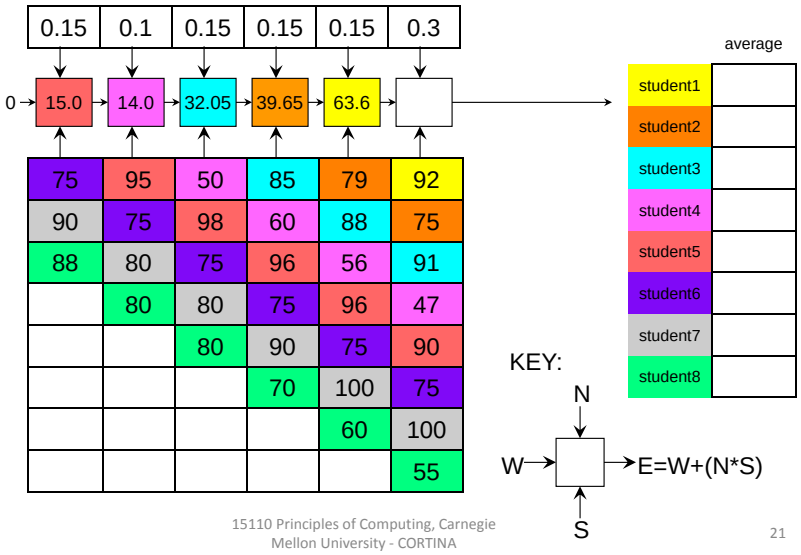
Faster Matrix Multiplication

using Pipelining



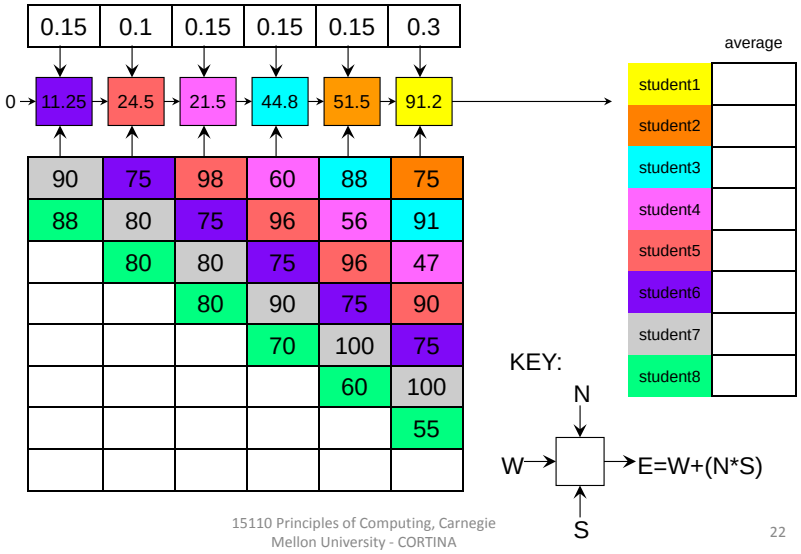
Faster Matrix Multiplication

using Pipelining



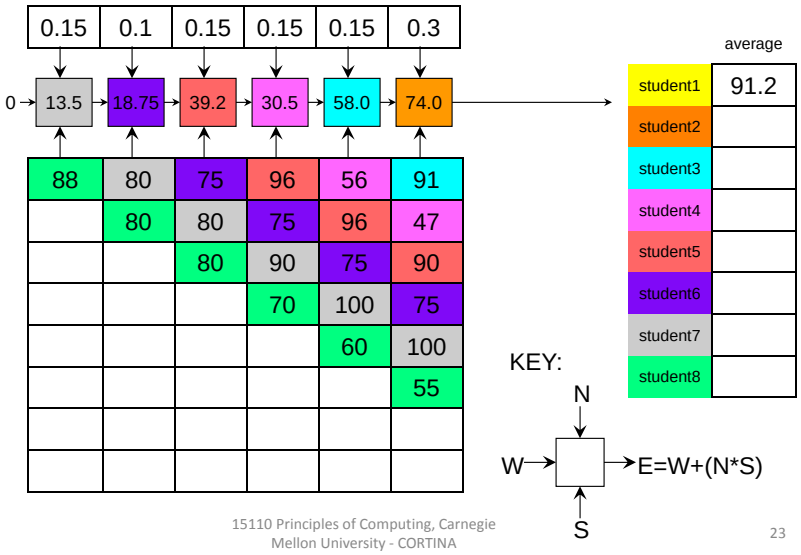
Faster Matrix Multiplication

using Pipelining



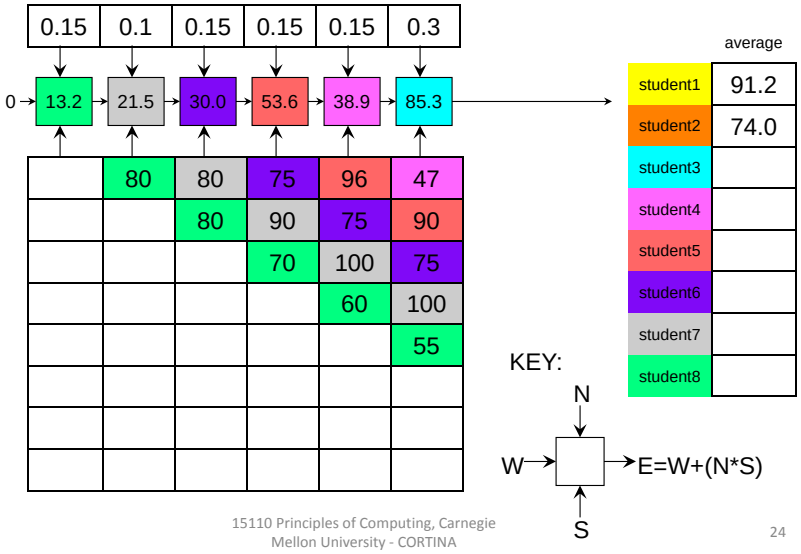
Faster Matrix Multiplication

using Pipelining



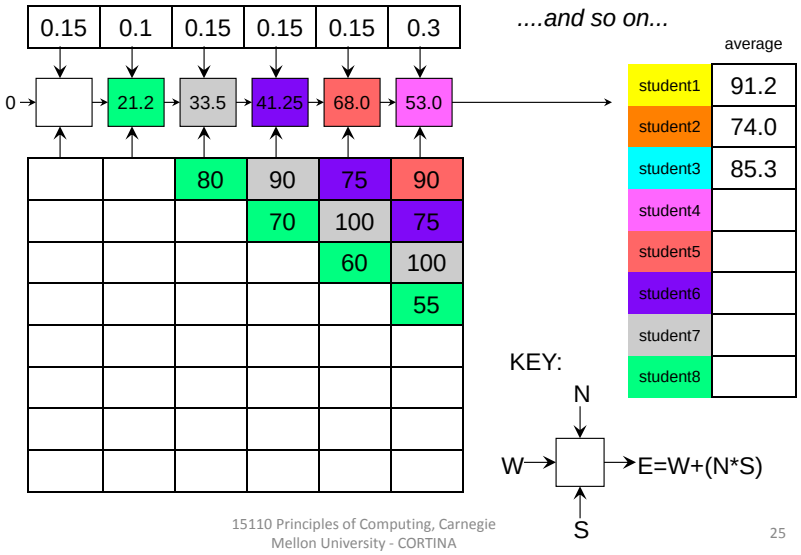
Faster Matrix Multiplication

using Pipelining



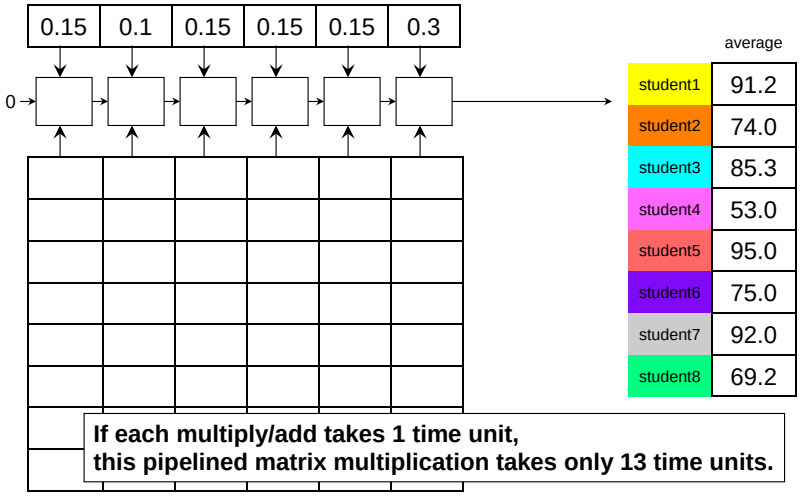
Faster Matrix Multiplication

using Pipelining



Faster Matrix Multiplication

using Pipelining



If each multiply/add takes 1 time unit,
this pipelined matrix multiplication takes only 13 time units.

Distributed Systems

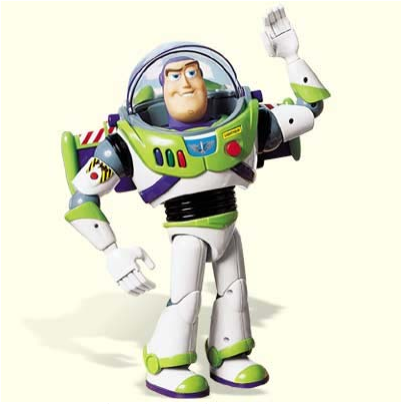
- A distributed system is an application that consists of processes that
  - execute on multiple computers connected through a network, and
  - cooperate to accomplish a task.
- Advantages
  - Reconfigurable: add or rearrange new parts
  - Geographically distributed: Low communication delays for remote users
  - Scalable: can add more processors as demand increases

Render Farms

- A **render farm** is high performance computer system, e.g. a computer cluster, built to render computer-generated imagery (CGI), typically for film and television visual effects.

Render Farms

- Rendering
  - flatten a 3-d space to a 2-d
  - lighting (raytracing)
  - potential concurrency
    - frames
    - pixels within a frame



- Many Disney animated movies use this technique

Challenge of Distributed Computing: Reliability in Context of Failure

Failure is the defining difference between distributed and local programming, so you have to design distributed systems with the expectation of failure. Imagine asking people, "If the probability of something happening is one in 10^{13}, how often would it happen?" Common sense would be to answer, "Never." That is an infinitely large number in human terms. But if you ask a physicist, she would say, "All the time. In a cubic foot of air, those things happen all the time." When you design distributed systems, you have to say, "Failure happens all the time." So when you design, you design for failure. It is your number one concern.

— Ken Arnold

Examples of Failures

- permanent network failures
- dropped messages between sender and receiver
- an individual computer breaks
- a process crashes or goes into an infinite loop

Can We Fix These Failures?

- Replication/Redundancy
- **RAID** (**redundant array of independent disks**) is a storage technology that combines multiple disk drive components into a logical unit. RAID is now used as an umbrella term for computer data storage schemes that can divide and replicate data among multiple physical drives.
- A **transaction log** is a history of actions executed by a database management system to guarantee backup over crashes or hardware failures.

Summary

- pipelining
  - assembly line: different sub-steps run concurrently
  - Processor Pipelining:
    - runs a sequence of instructions faster
    - splits each into, e.g., fetch, decode, read, execute, write
    - separate hardware for each pipeline stage
- Distributed Systems
  - multiple processes distributed across multiple machines
  - Examples:
    - render farms
    - Google