I/O and Storage:
RAID

CS 571: Operating Systems (Spring 2020)

Lecture 10b

Yue Cheng

Some material taken/derived from:

- Wisconsin CS-537 materials created by Remzi Arpaci-Dusseau.

Properties of A Single Disk

- A single disk is slow
    - Kind of Okay sequential I/O performance
    - Really bad for random I/O

Properties of A Single Disk

- A single disk is slow
  - Kind of Okay sequential I/O performance
  - Really bad for random I/O

- The storage capacity of a single disk is limited

Properties of A Single Disk

- A single disk is slow
  - Kind of Okay sequential I/O performance
  - Really bad for random I/O

- The storage capacity of a single disk is limited

- A single disk is not reliable

RAID: Redundant Array of Inexpensive Disks

Wish List for a Disk

- Wish it to be faster
  - I/O is always the performance bottleneck

Wish List for a Disk

- Wish it to be faster
    - I/O is always the performance bottleneck
- Wish it to be larger
    - More and more data needs to be stored

Wish List for a Disk

- Wish it to be faster
  - I/O is always the performance bottleneck
- Wish it to be larger
  - More and more data needs to be stored
- Wish it to be more reliable
  - We don’t want our valuable data to be gone

Only One Disk?

- Sometimes we want many disks
  - For higher performance
  - For larger capacity
  - For better reliability

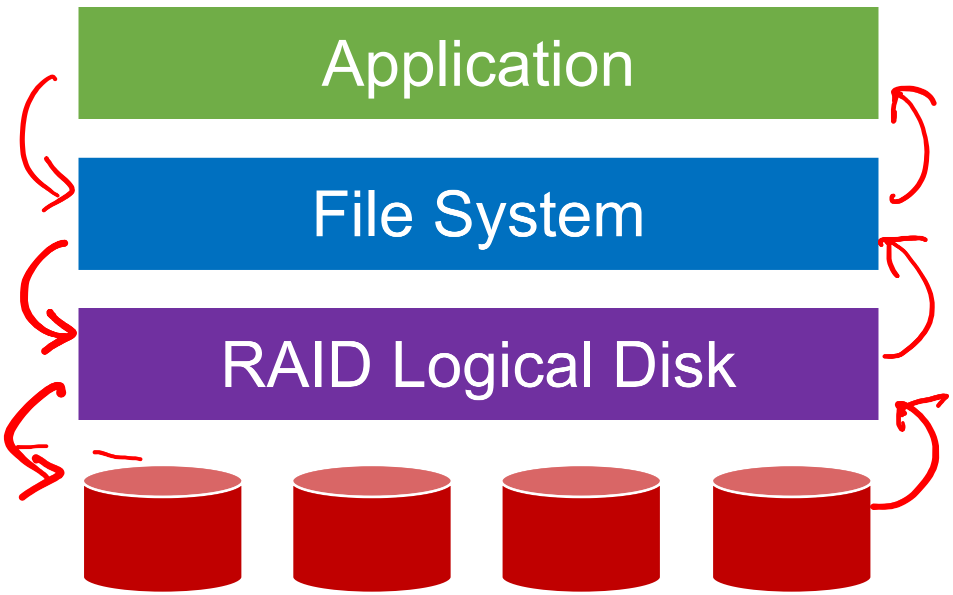
- Challenge: Most file systems work on only one disk

Solution: RAID

Focus of class. → S/W RAID

H/W RAID

RAID: Redundant Array of Inexpensive Disks



S/W.

(H/W) HDDs

Build a logical disk from many physical disks

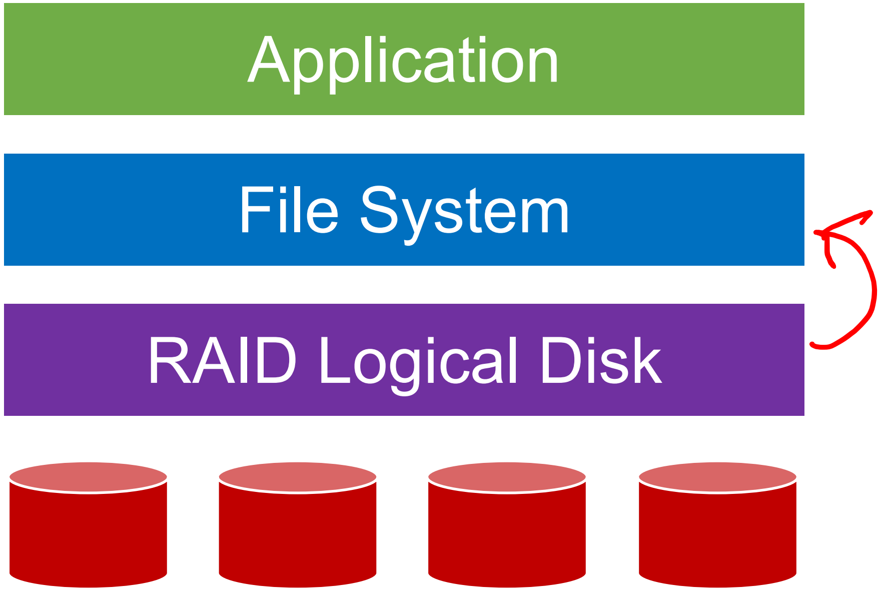
Solution: RAID

RAID: Redundant Array of Inexpensive Disks

RAID is

- Transparent
- Deployable

One more layer.

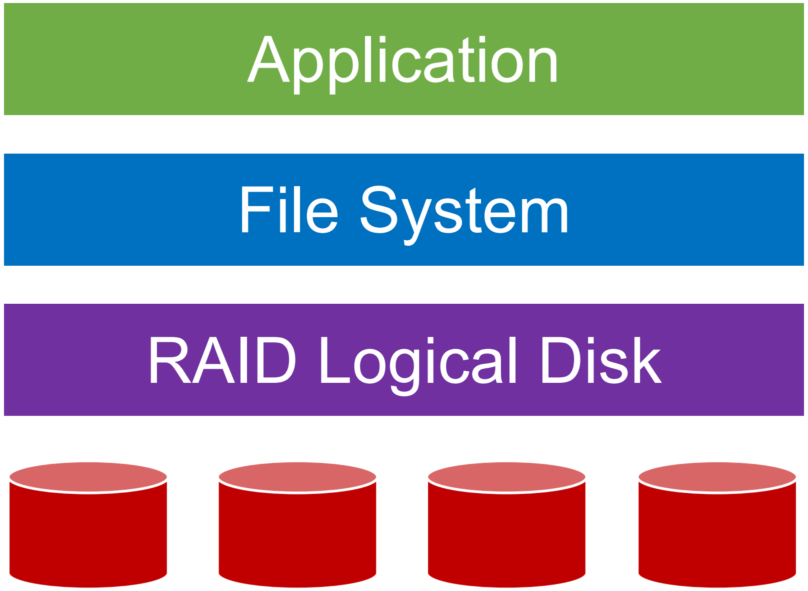


Build a logical disk from many physical disks

Solution: RAID

RAID: Redundant Array of Inexpensive Disks

RAID is
- Transparent
- Deployable



Application

File System

RAID Logical Disk

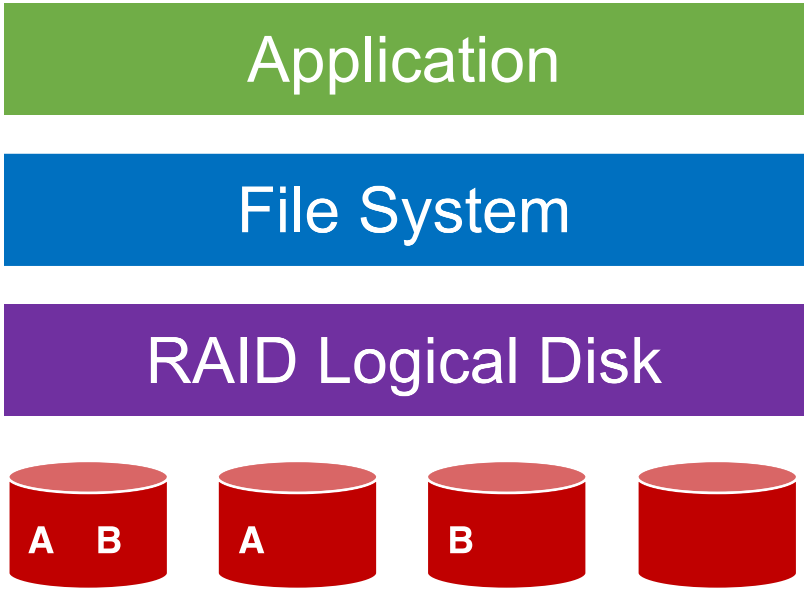
Logical disks gives
- Performance
- Capacity
- Reliability

Build a logical disk from many physical disks

Solution: RAID

RAID: Redundant Array of Inexpensive Disks

RAID is
- Transparent
- Deployable



Logical disks gives
- Performance
- Capacity
- Reliability

redundancy!

replication!

Build a logical disk from many physical disks

Why Inexpensive Disks?

wins

- Economies of scale! Cheap disks are popular.
- You can often get many commodity hardware components for the same price as a few expensive components

Why Inexpensive Disks?

- Economies of scale! Cheap disks are popular.

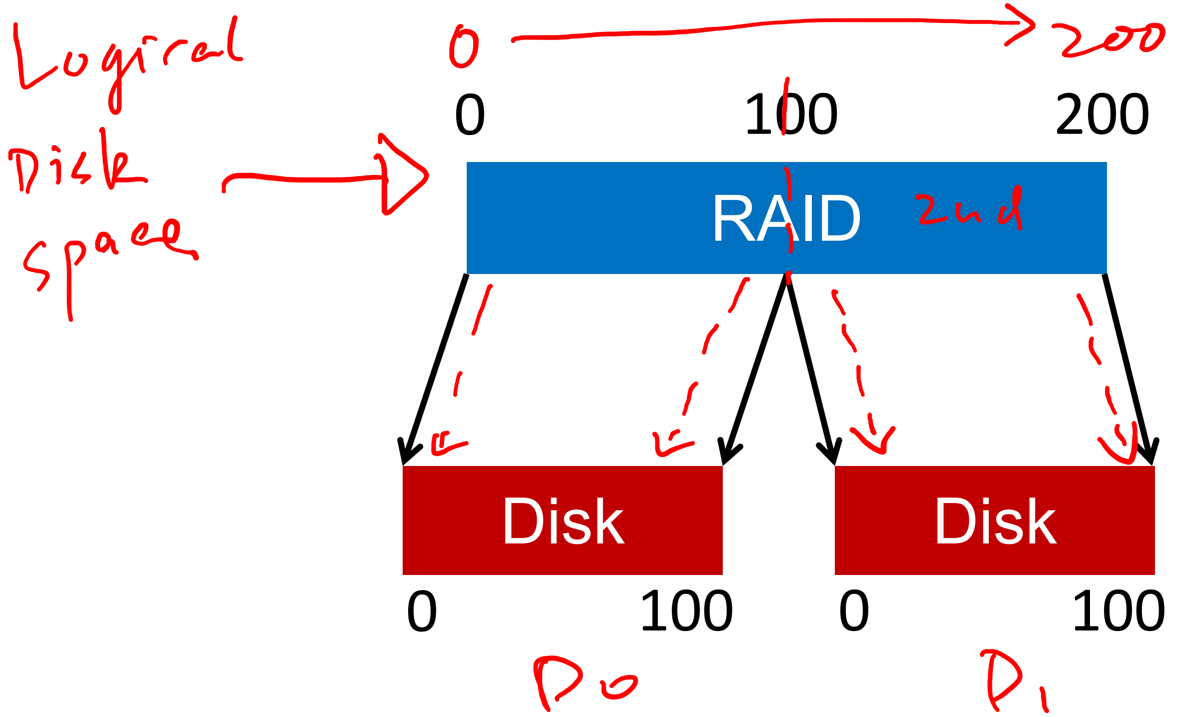
- You can often get many commodity hardware components for the same price as a few expensive components

- Strategy: Write software to build high-quality logical devices from many cheap devices
  - Tradeoff: To compensate poor properties of cheap devices

General Strategy

JBOD.

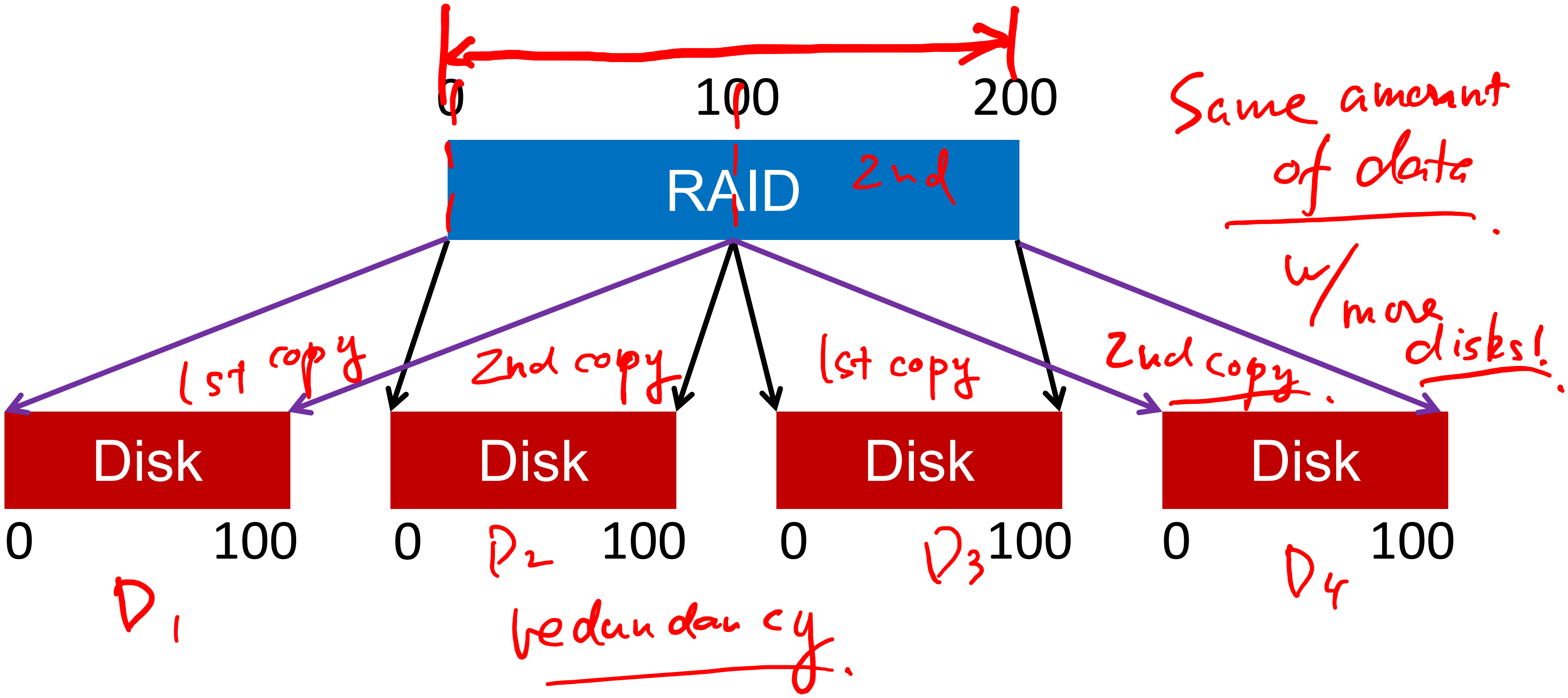
Build fast and large disks from smaller ones



General Strategy

Build fast and large disks from smaller ones

Add more disks for reliability++!



RAID Metrics

- Performance → throughput (large sequential IOs)
  → latency (small, random IOs)
  - How long does each workload take?
- Capacity
  - How much space can apps use?
- Reliability
  - How many disks can we safely lose?

RAID Metrics

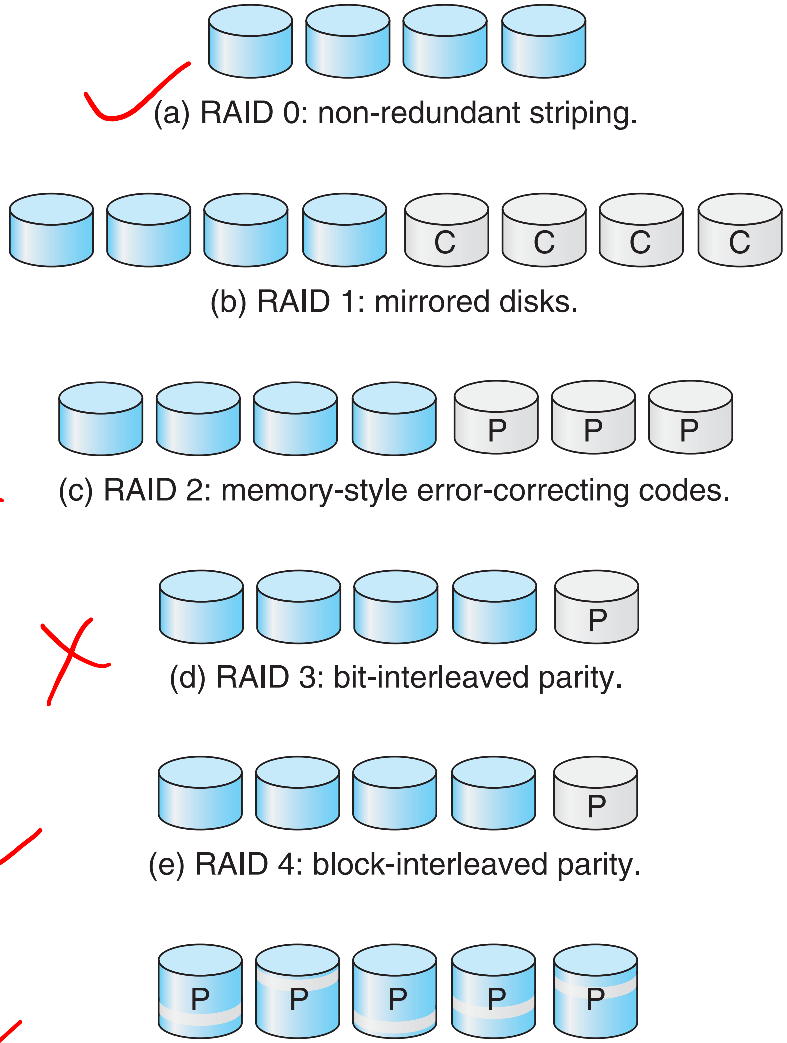
- Performance
  - How long does each workload take?
- Capacity
  - How much space can apps use?
- Reliability
  - How many disks can we safely lose?
  - Assume fail-stop model!

binary.

{ works.
fails.

RAID Levels

Configs.



✓ (a) RAID 0: non-redundant striping.

✓ (b) RAID 1: mirrored disks.

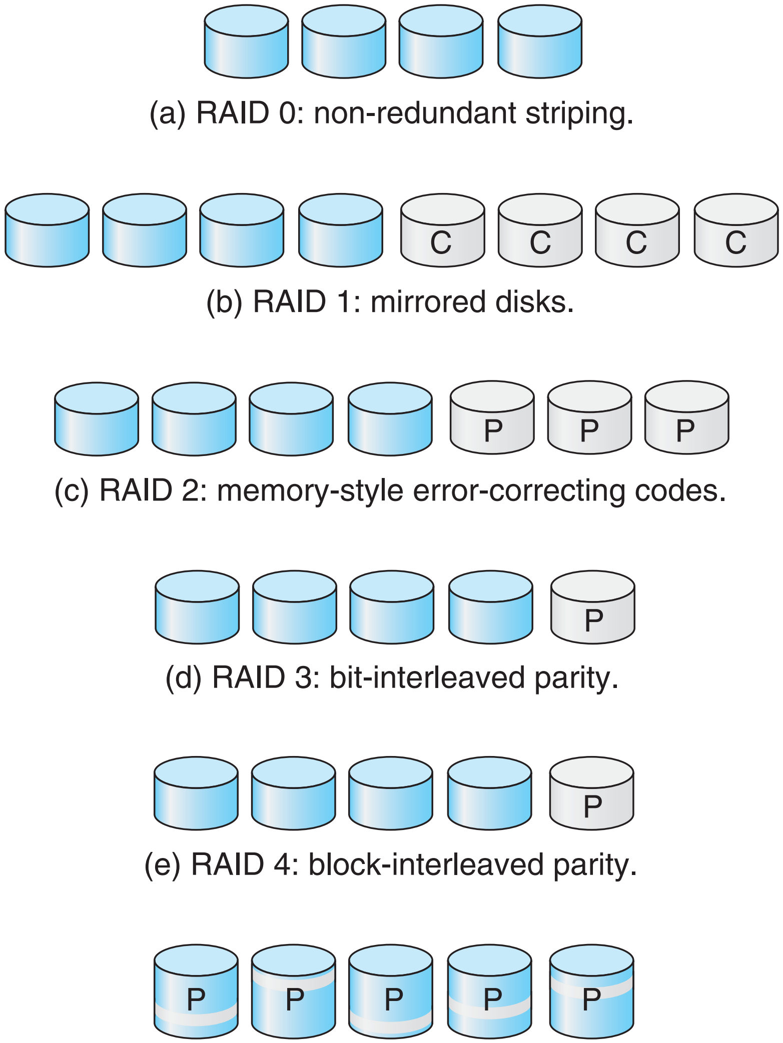
✗ (c) RAID 2: memory-style error-correcting codes.

✗ (d) RAID 3: bit-interleaved parity.

✓ (e) RAID 4: block-interleaved parity.

✓ (f) RAID 5: block-interleaved distributed parity.

RAID Level 0



(a) RAID 0: non-redundant striping.

(b) RAID 1: mirrored disks.

(c) RAID 2: memory-style error-correcting codes.

(d) RAID 3: bit-interleaved parity.

(e) RAID 4: block-interleaved parity.

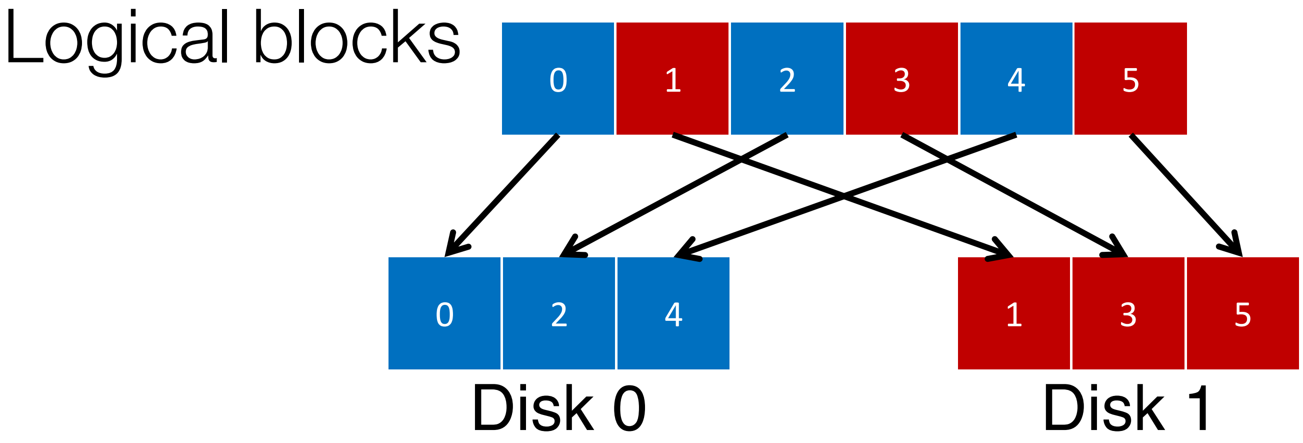
(f) RAID 5: block-interleaved distributed parity.

RAID-0: Striping

- No redundancy

best case

- Serves as upper bound for
  - Performance
  - Capacity



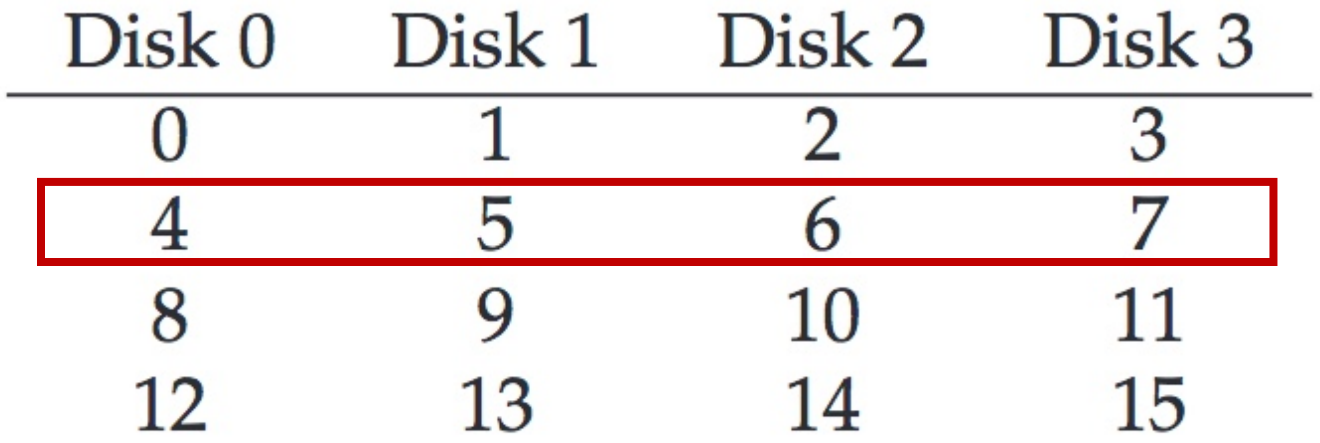
4 Disks

RAID - 0 .

stripe .

Disk 0	Disk 1	Disk 2	Disk 3
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

4 Disks



Disk 0	Disk 1	Disk 2	Disk 3
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

stripe:

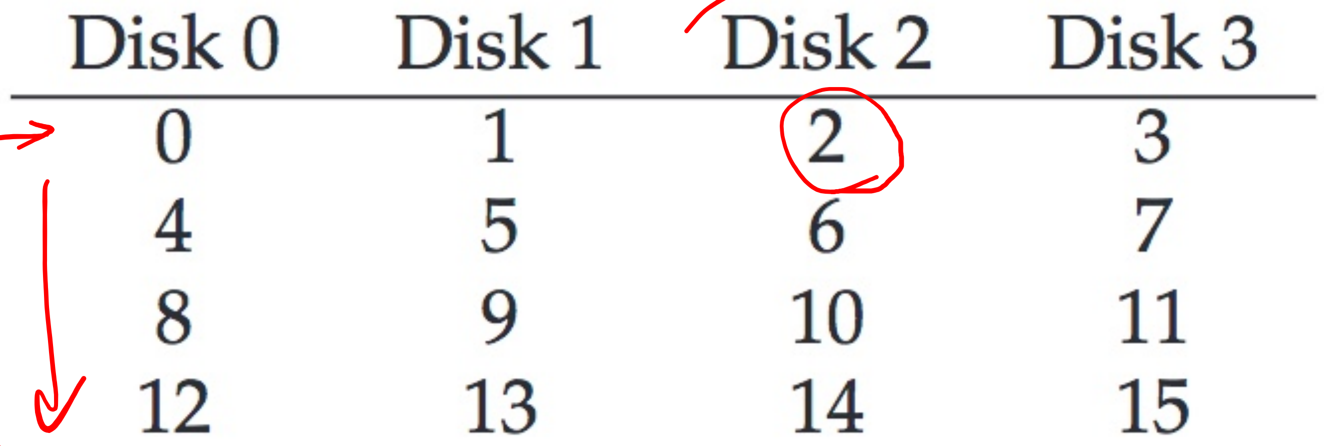
How to Map?

- Given logical address A:
  - physical: Disk = … A % #Disks.
  - Offset = … A / #Disks.

stripe ID →

2 % 4 = 2

2 / 4 = 0



Disk 0	Disk 1	Disk 2	Disk 3
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

stripe 0 →

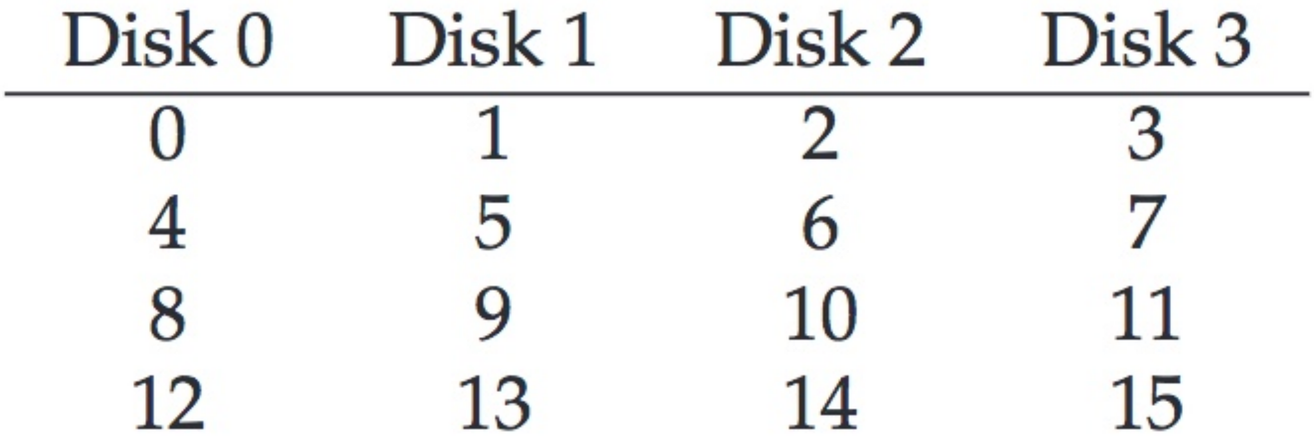
stripe 1

2

stripe 3

How to Map?

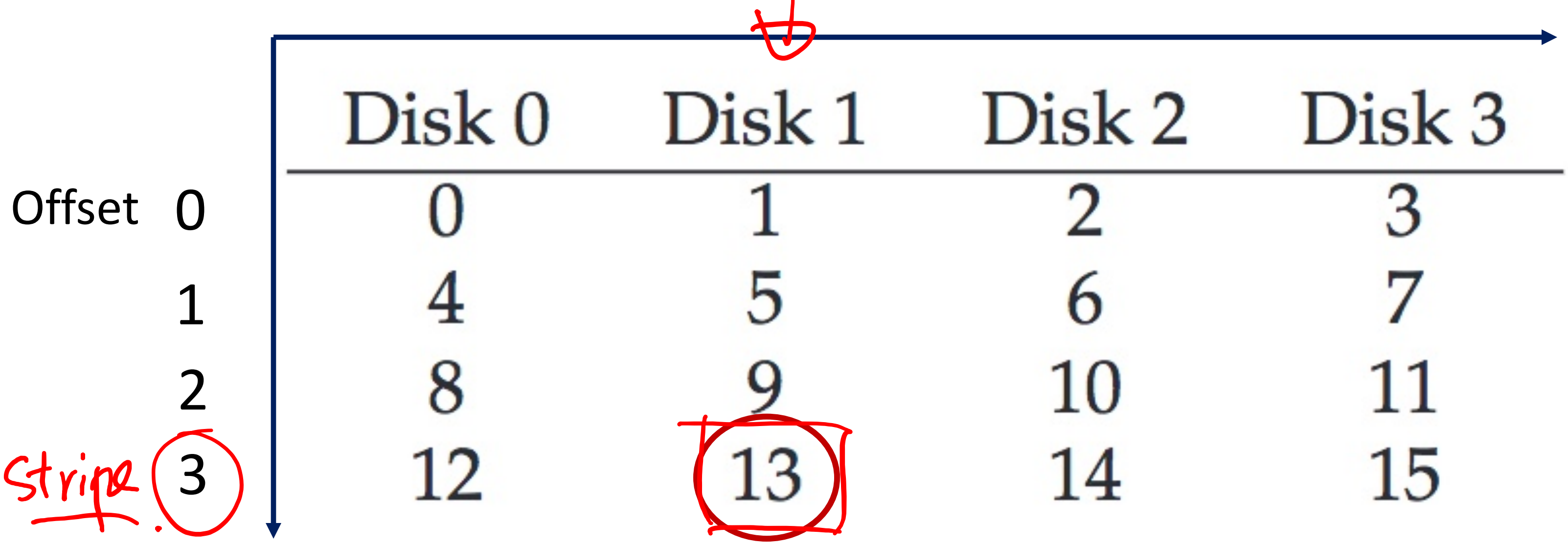
- Given logical address A:
  - Disk = `A % disk_count`
  - Offset = `A / disk_count`



Disk 0	Disk 1	Disk 2	Disk 3
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

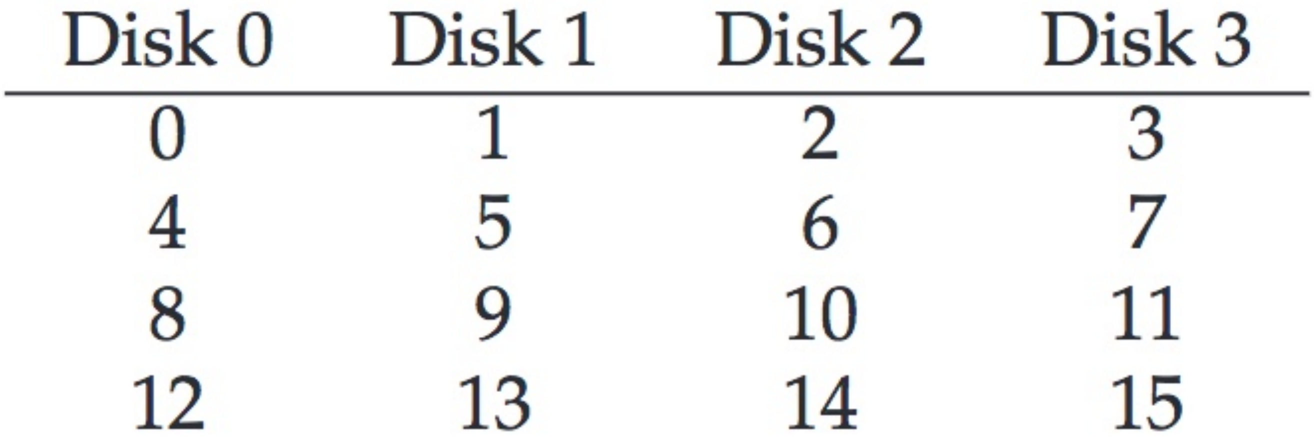
Mapping Example: Find Block 13

- Given logical address 13:
  - Disk = `13 % 4 = 1`
  - Offset = `13 / 4 = 3`



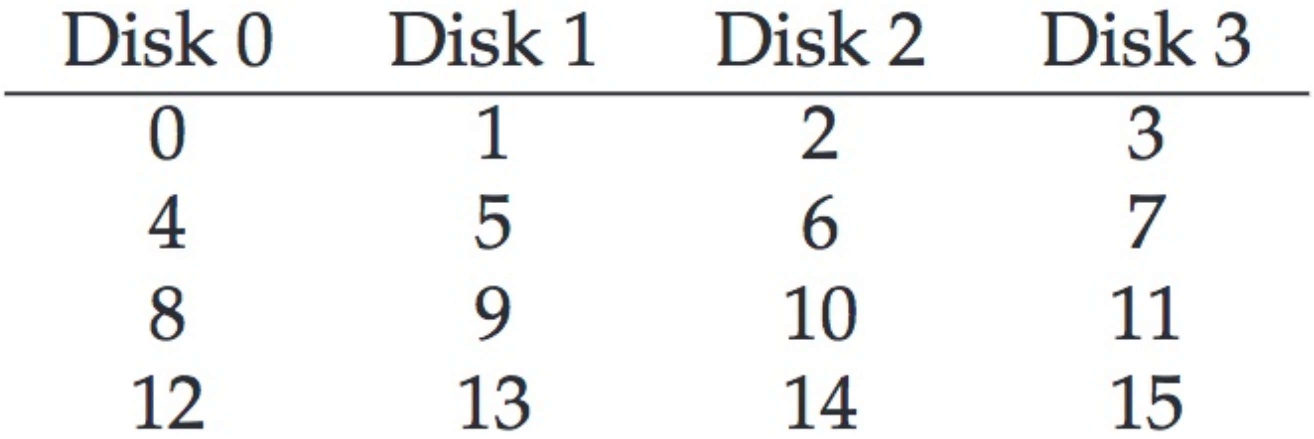
		Disk 0	Disk 1	Disk 2	Disk 3
Offset	0	0	1	2	3
	1	4	5	6	7
	2	8	9	10	11
Stripe	3	12	13	14	15

Chunk Size = 1



Disk 0	Disk 1	Disk 2	Disk 3
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

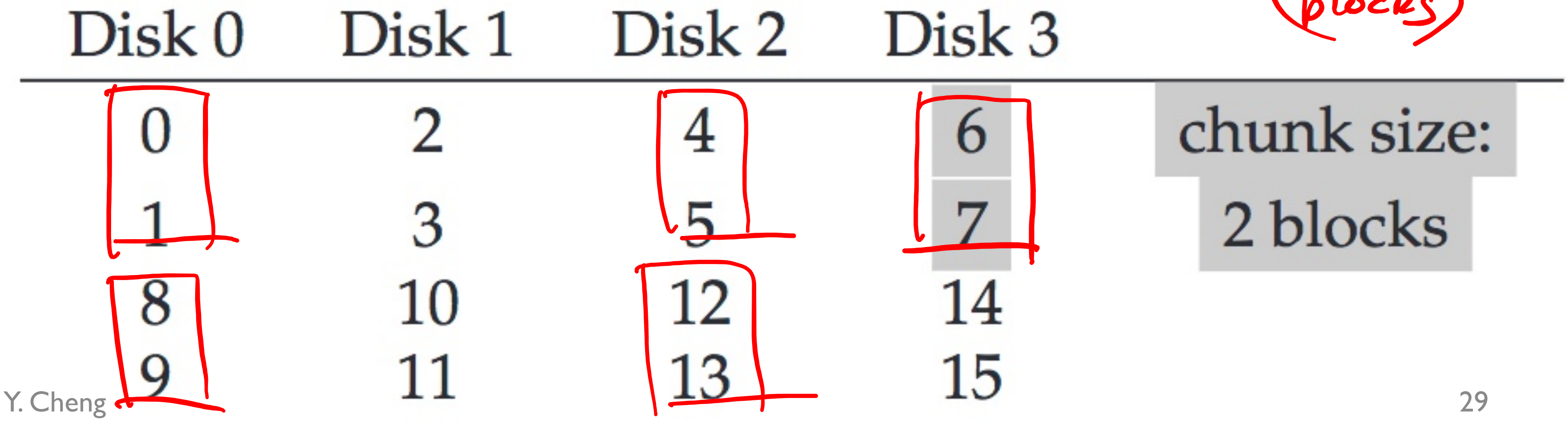
Chunk Size = 1



Disk 0	Disk 1	Disk 2	Disk 3
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

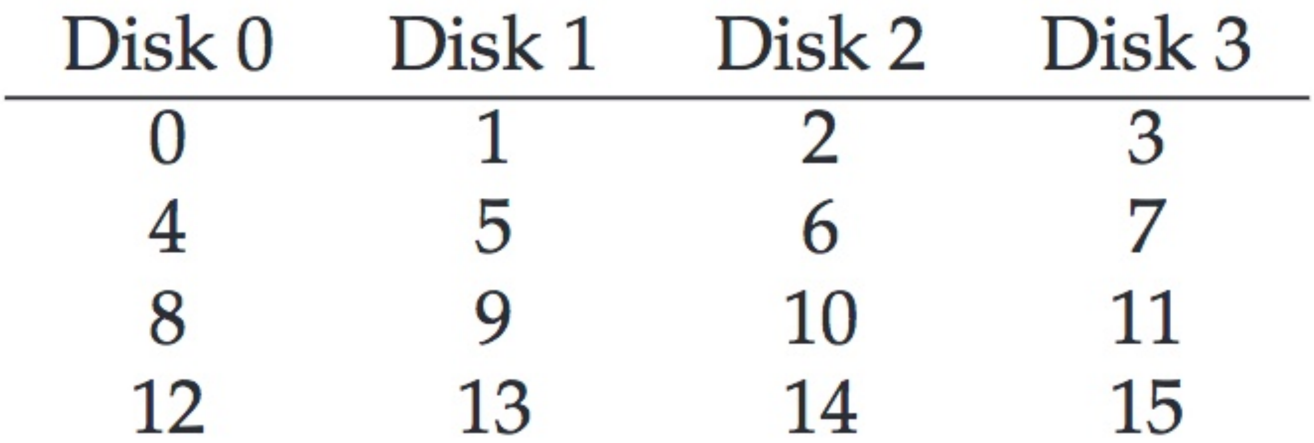
Chunk Size = 2

Disk sectors
(blocks)



Disk 0	Disk 1	Disk 2	Disk 3	
0	2	4	6	chunk size:
1	3	5	7	2 blocks
8	10	12	14	
9	11	13	15	

Chunk Size = 1

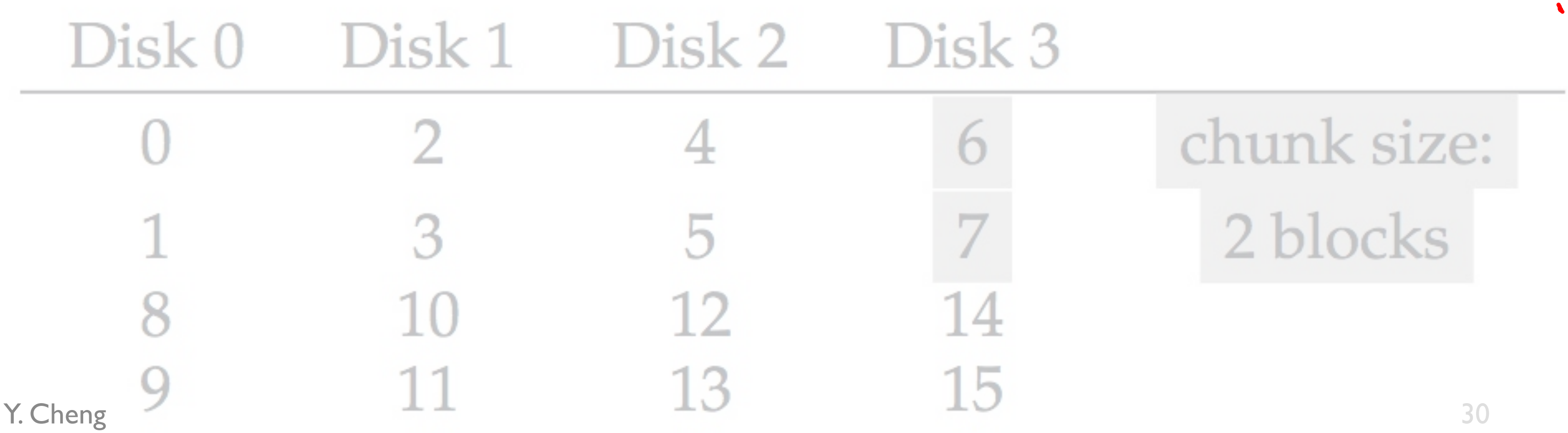


Disk 0	Disk 1	Disk 2	Disk 3
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

In all following examples, we assume chunk size of 1

Disk sector

Chunk Size = 2



Disk 0	Disk 1	Disk 2	Disk 3
0	2	4	6
1	3	5	7
8	10	12	14
9	11	13	15

chunk size:
2 blocks

Theoretical

RAID-0 Analysis

1. What is capacity?

Reliability

2. How many disks can fail?

3. Throughput?

4. Latency?

Performance

RAID-0 Analysis

1. What is capacity? N * C
2. How many disks can fail? 0
3. Throughput? N * S and N * R

   Seq Rand
4. Latency? D

N: # Disks.

C: capacity of one Disk.

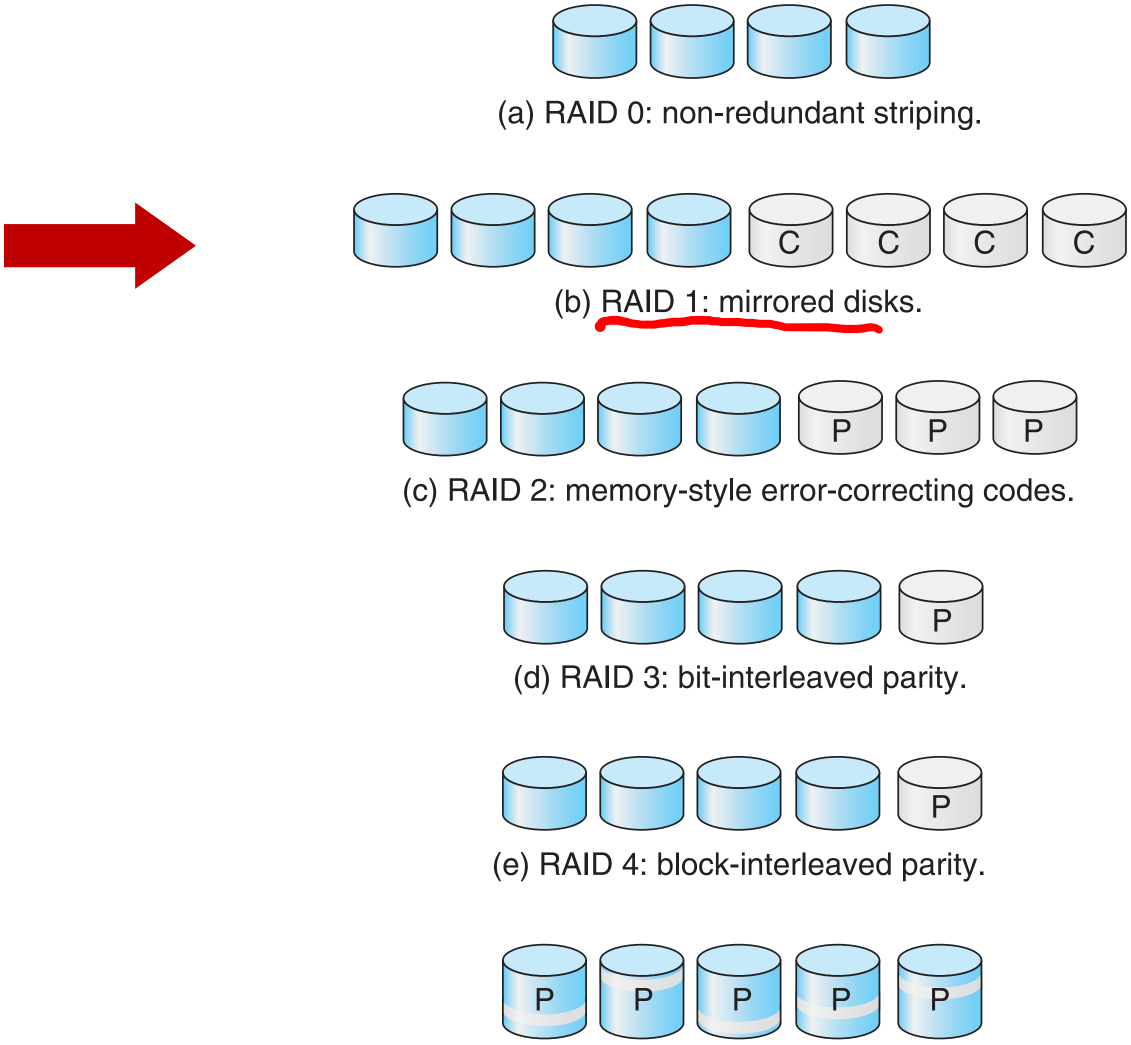
S: Sequential throughput

R: Random I/O operation per sec. (IOPS)

D: Latency for any single random I/O op

both Reads Writes.

RAID Level 1



(a) RAID 0: non-redundant striping.

(b) RAID 1: mirrored disks.

(c) RAID 2: memory-style error-correcting codes.

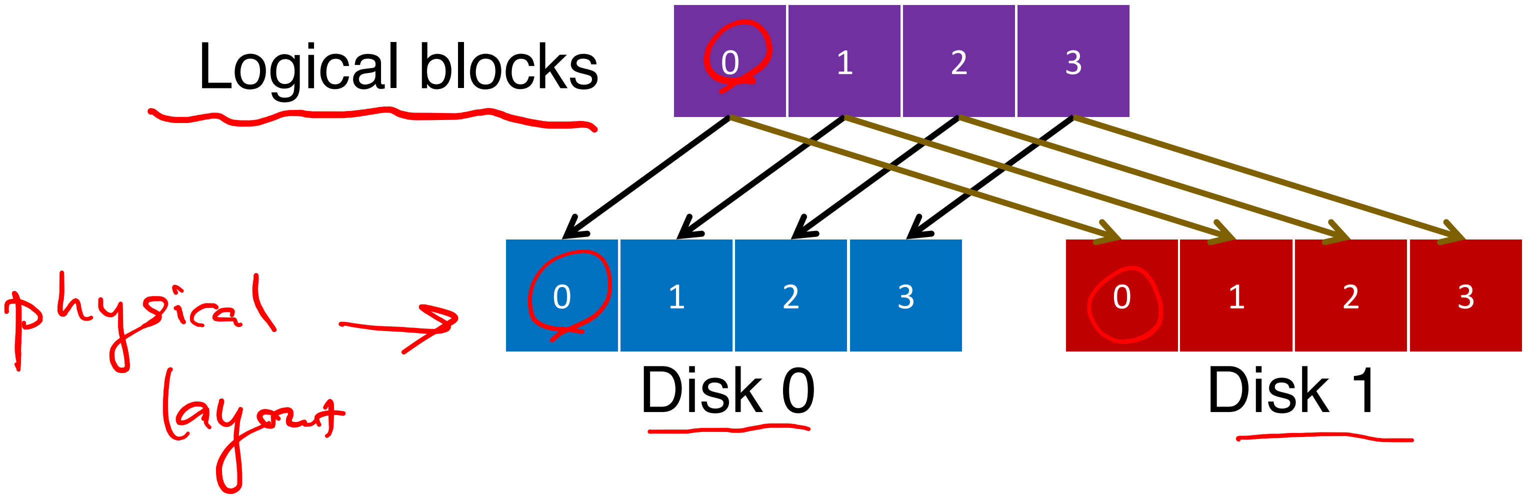
(d) RAID 3: bit-interleaved parity.

(e) RAID 4: block-interleaved parity.

(f) RAID 5: block-interleaved distributed parity.

RAID-1: Mirroring

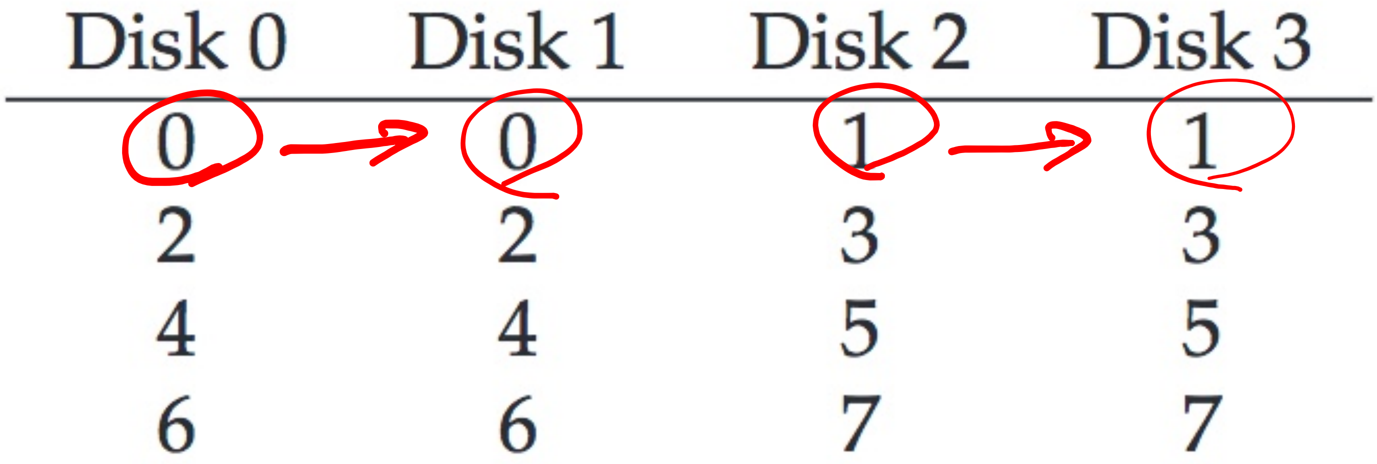
- RAID-1 keeps two copies of each block



Assumption

- Assume disks are fail-stop
  - Two states
    - They work or they don’t
  - We know when they don’t work

4 Disks



Disk 0	Disk 1	Disk 2	Disk 3
0	0	1	1
2	2	3	3
4	4	5	5
6	6	7	7

4 Disks

Best-case scenario.

2 failures.

Worst-case. Cannot tolerate

can tolerate 1 single failure.

Disk 0	Disk 1	Disk 2	Disk 3
0	0	1	1
2	2	3	3
4	4	5	5
6	6	7	7

Q: **How many disks can fail?**

RAID-1 Analysis

1. What is capacity? N/2 * C

safe side

2. How many disks can fail? 1 or maybe ~~N / 2~~

4/2 = 2 optimistic.

best case.

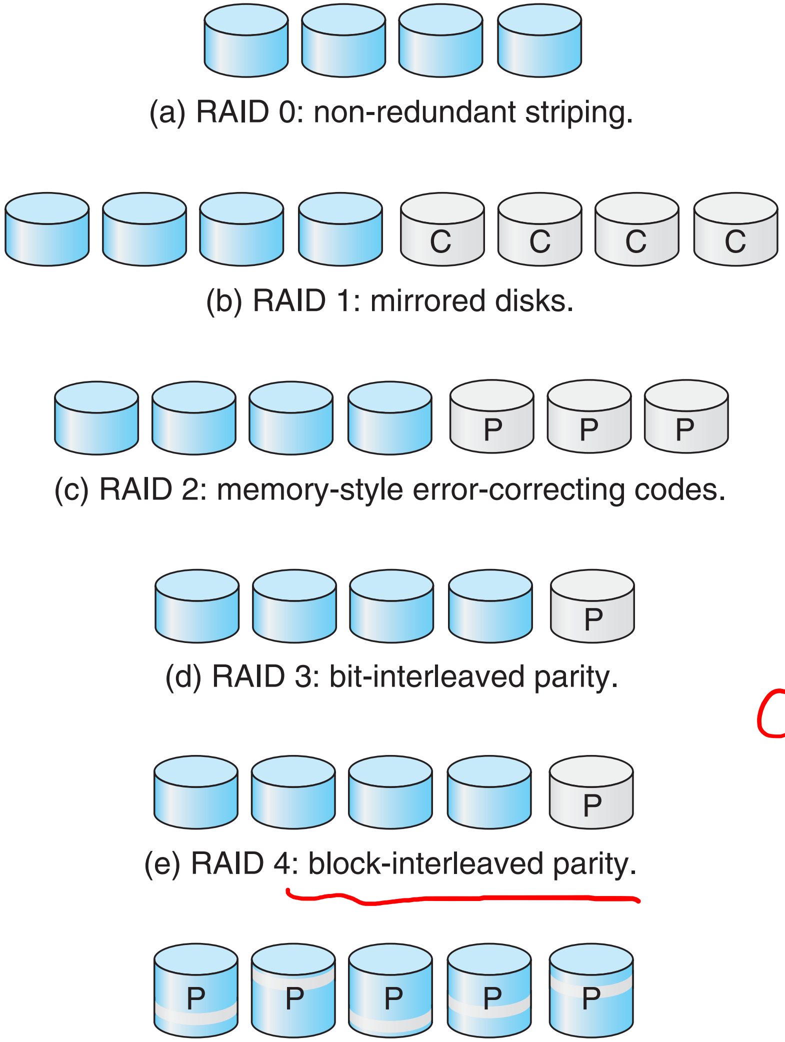
3. Throughput?
   - Seq read: N/2 * S
   - Seq write: N/2 * S
   - Rand read: N * R
   - Rand write: N/2 * R

4. Latency? D

D_1	D_2	D_3	D_4
0	0	1	1
2	2	3	3
4	4	5	5
6✓	6✓	7	7

logical write → two physical writes

RAID Level 4



(a) RAID 0: non-redundant striping.

(b) RAID 1: mirrored disks.

(c) RAID 2: memory-style error-correcting codes.

(d) RAID 3: bit-interleaved parity.

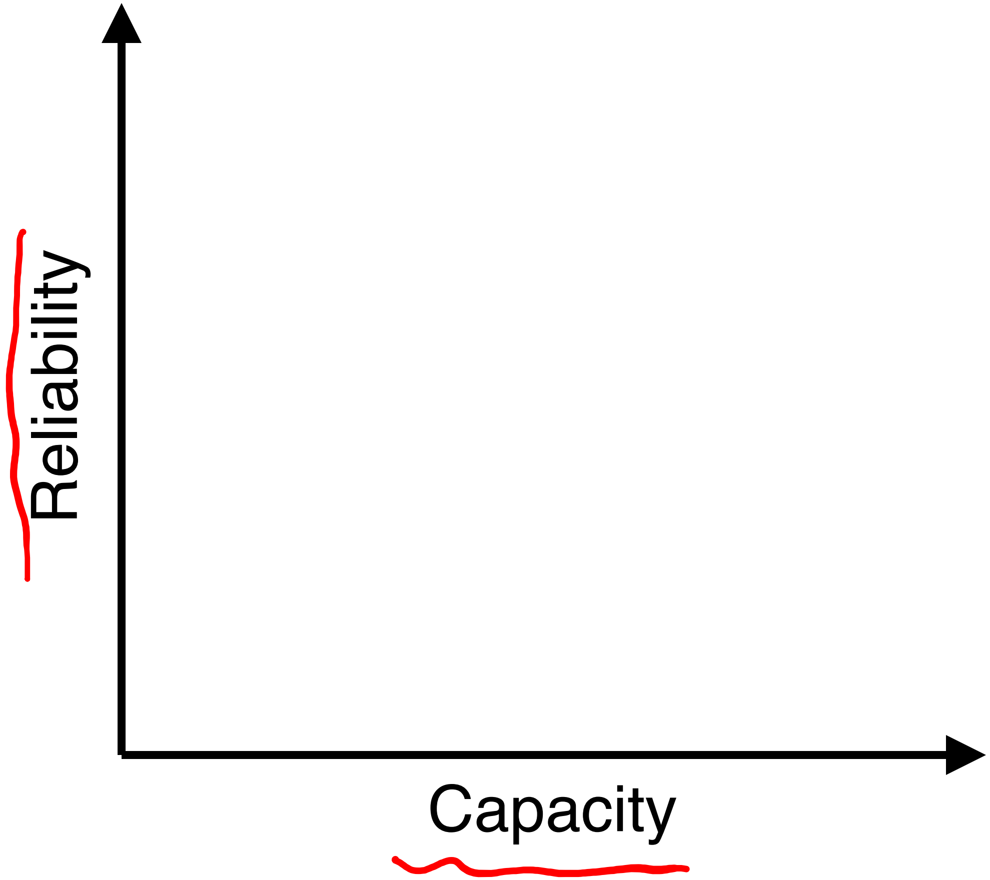
(e) RAID 4: block-interleaved parity.

(f) RAID 5: block-interleaved distributed parity.

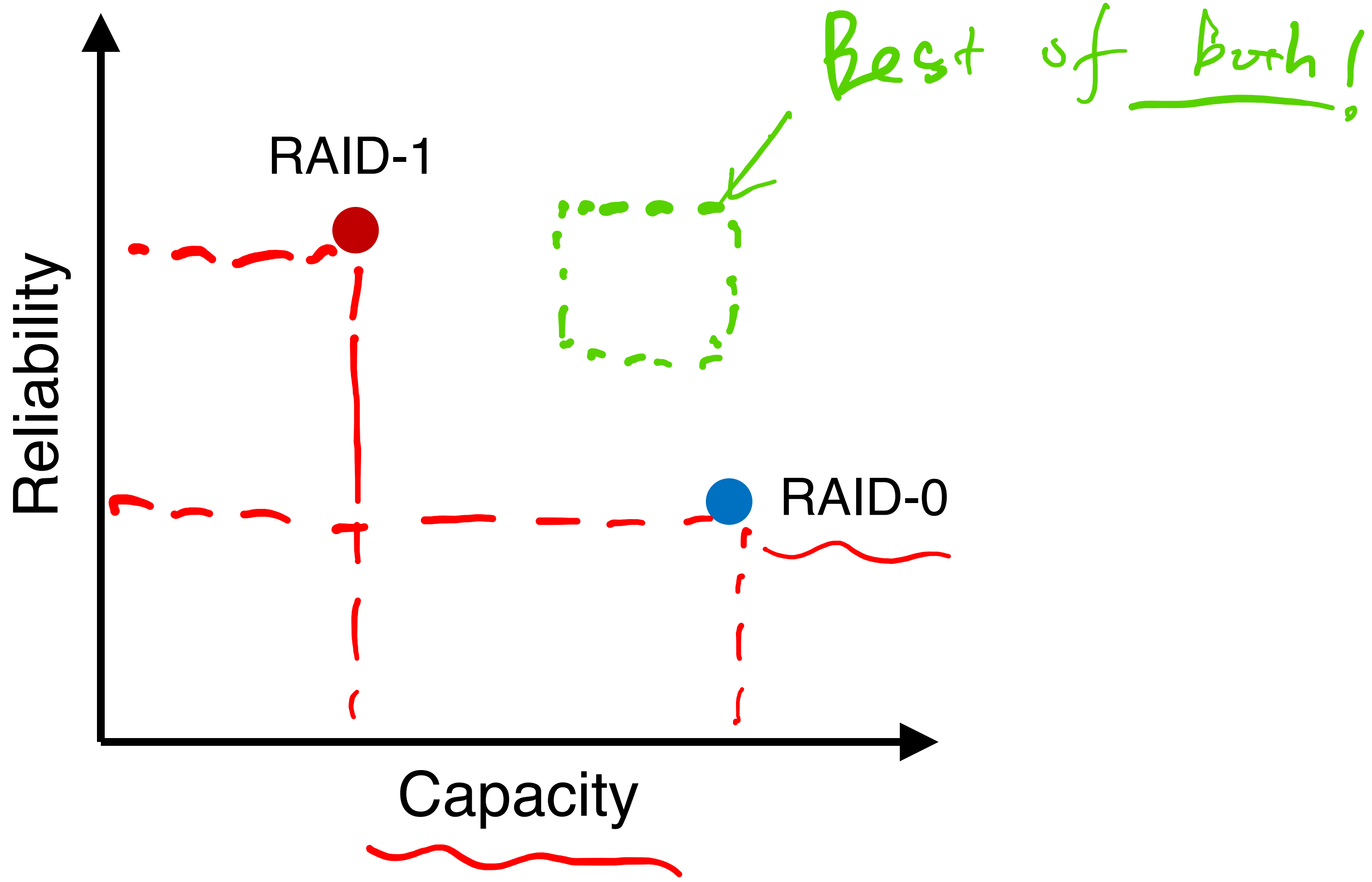
capacity-efficient redundancy.

RAID-4

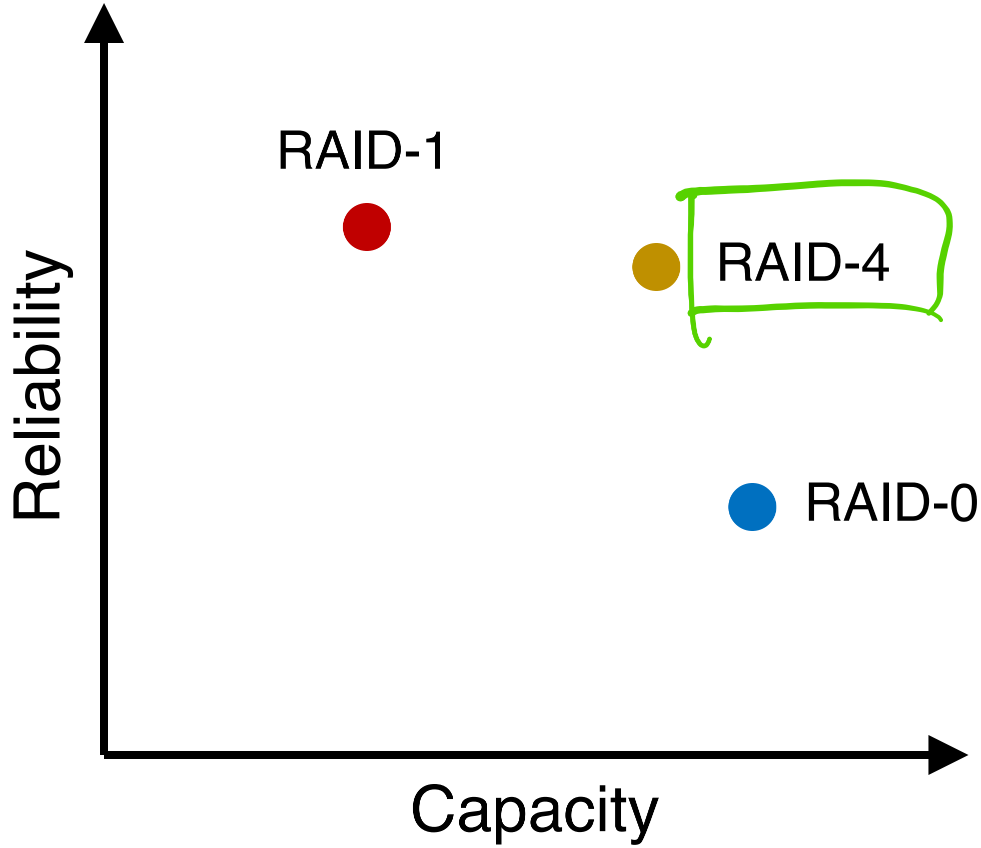
Higher the better!



RAID-4



RAID-4



Parity

redundancy

RAID-4: Strategy

- Use parity disk

linear

- In algebra, if an equation has N variables, and N-1 are known, you can also solve for the unknown

parity calculation { addition / XOR

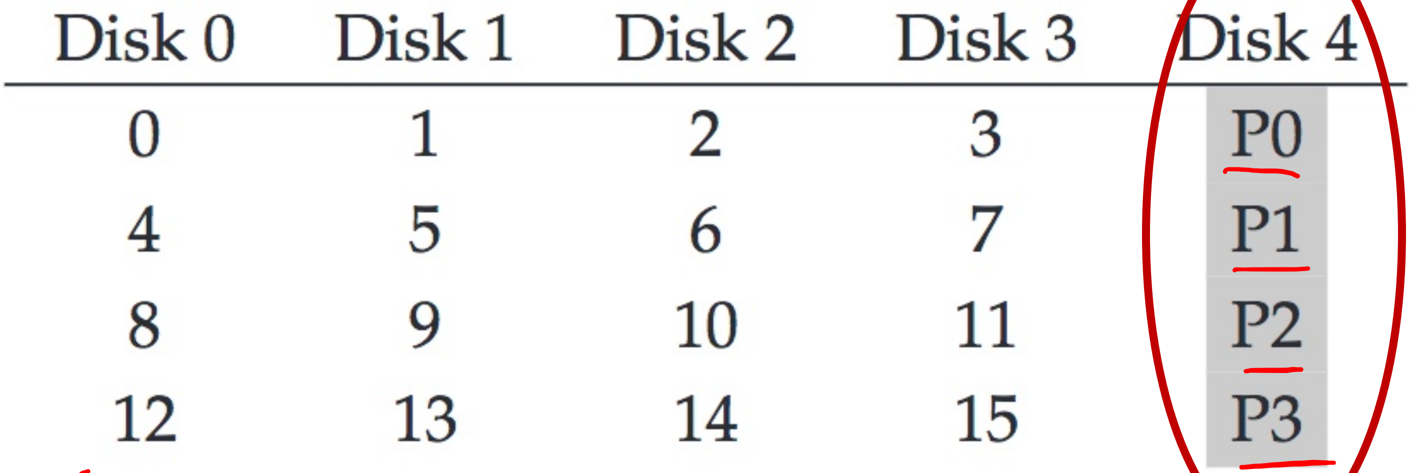
- Treat the sectors/blocks across disks in a stripe as an equation

RAID-4: Strategy

- Use parity disk
- In algebra, if an equation has N variables, and N-1 are known, you can also solve for the unknown
- Treat the sectors/blocks across disks in a stripe as an equation
- A failed disk is like an unknown in that equation

5 Disks

Parity Disk

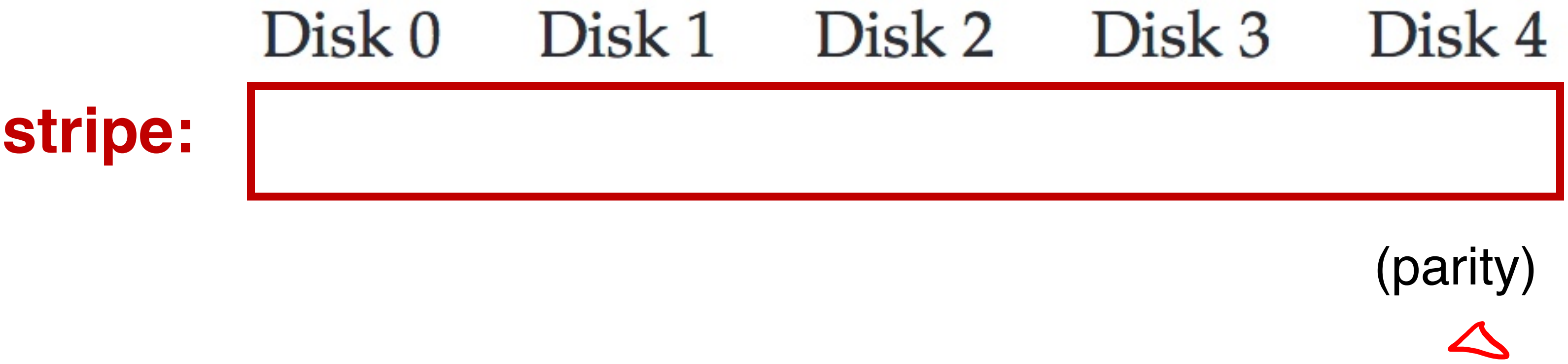


Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
4	5	6	7	P1
8	9	10	11	P2
12	13	14	15	P3

Regular Data Chunks.

Parity chunks.

Example



Example

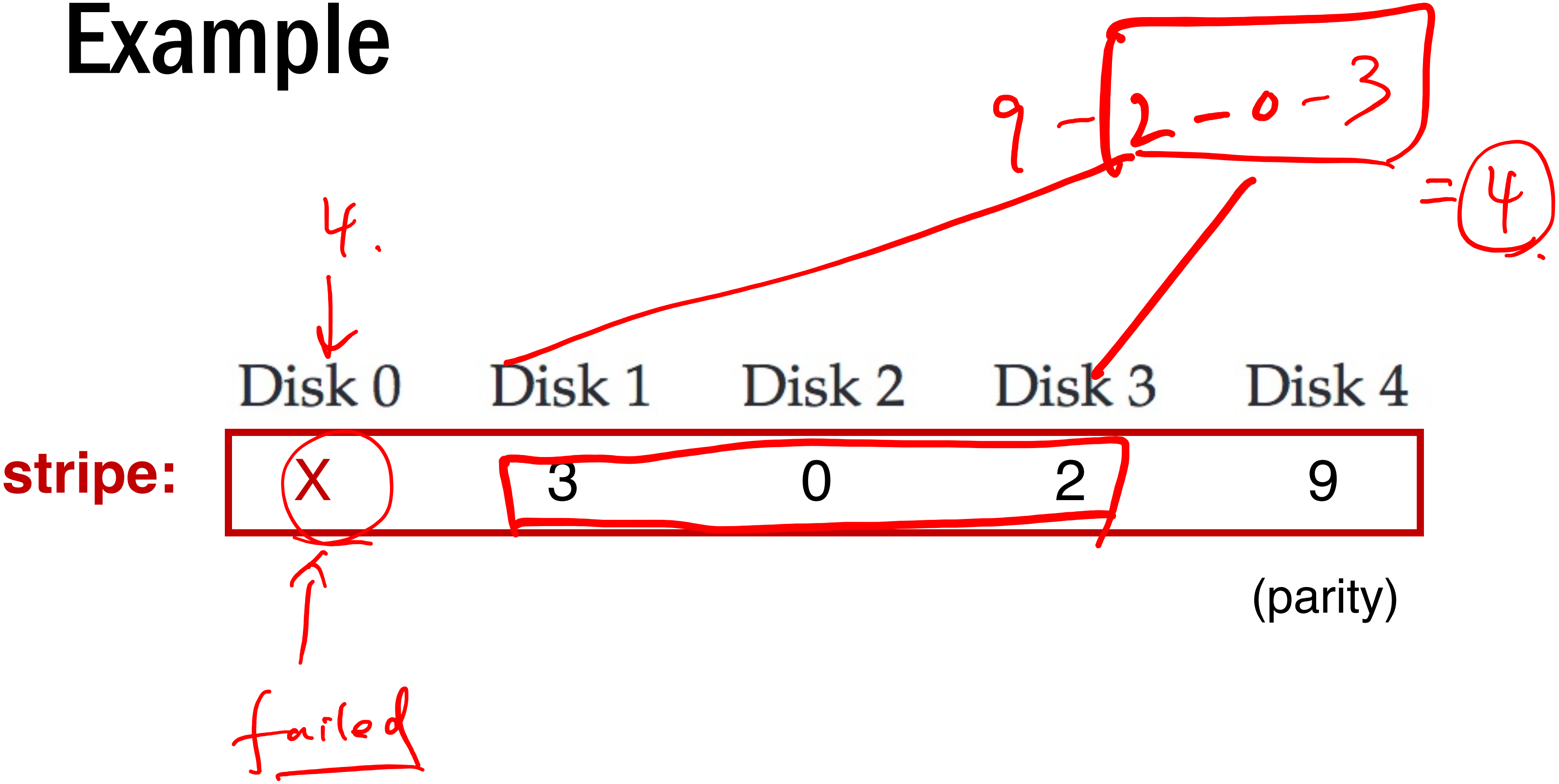
Add.

	Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
stripe:	4 +	3 +	0 +	2 =	$\boxed{9}$
					(parity)

Example

	Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
stripe:	4	3	0	2	9
					(parity)

Example



	Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
stripe:	X	3	0	2	9
					(parity)

$9 - 2 - 0 - 3 = 4$.

4.

failed

Example

RAID4: Trade off

computation overhead ✓

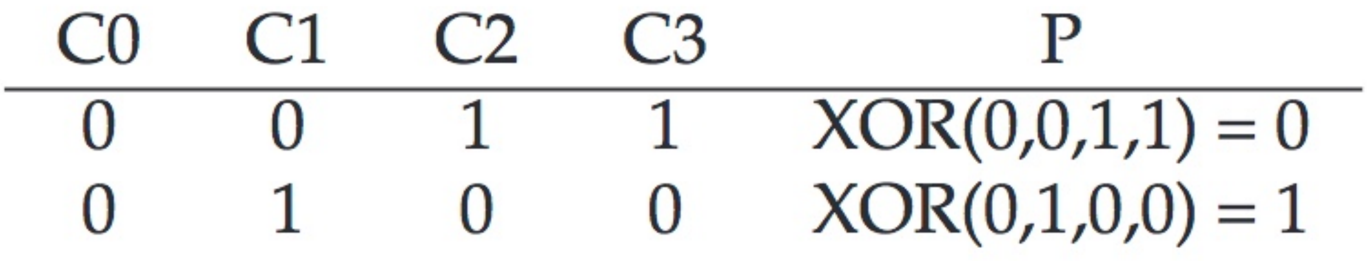
↓

capacity efficiency ✓

	Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
stripe:	**4**	3	0	2	9

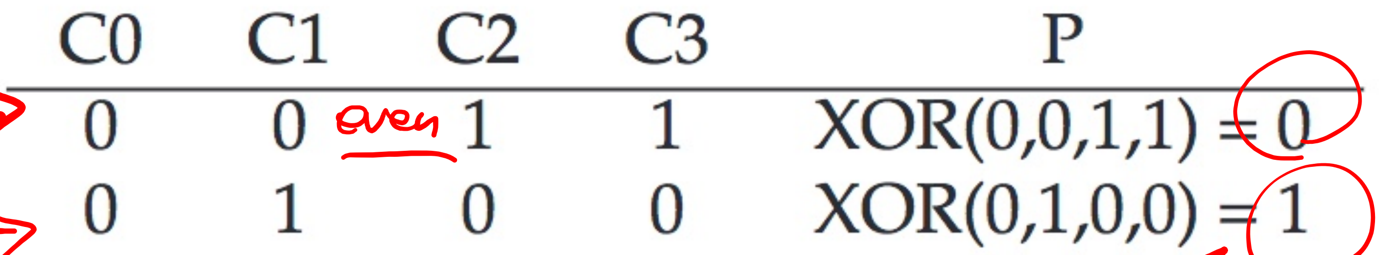
(parity)

Parity Function: XOR Example



C0	C1	C2	C3	P
0	0	1	1	XOR(0,0,1,1) = 0
0	1	0	0	XOR(0,1,0,0) = 1

Parity Function: XOR Example



C0	C1	C2	C3	P
0	0	1	1	XOR(0,0,1,1) = 0
0	1	0	0	XOR(0,1,0,0) = 1

even

odd

XOR function:

- P = 0: The number of 1 in a stripe must be an even number
- P = 1: The number of 1 in a stripe must be an odd number

Parity Function: XOR Example

XOR(00, 10, 11, 10) = 11

	Block0	Block1	Block2	Block3	Parity
stripe:	00	10	11	10	11
	10	01	00	01	10

XOR function:

- P = 0: The number of 1 in a stripe must be an even number
- P = 1: The number of 1 in a stripe must be an odd number

Parity Function: XOR Example

Disk0

	Block0	Block1	Block2	Block3	Parity
stripe:	X	10	11	10	11
	10	01	00	01	10

XOR(10, 11, 10, 11) = 00

4 1's → 0

2 1's → 0

XOR function:

- P = 0: The number of 1 in a stripe must be an even number
- P = 1: The number of 1 in a stripe must be an odd number

Parity Function: XOR Example

00

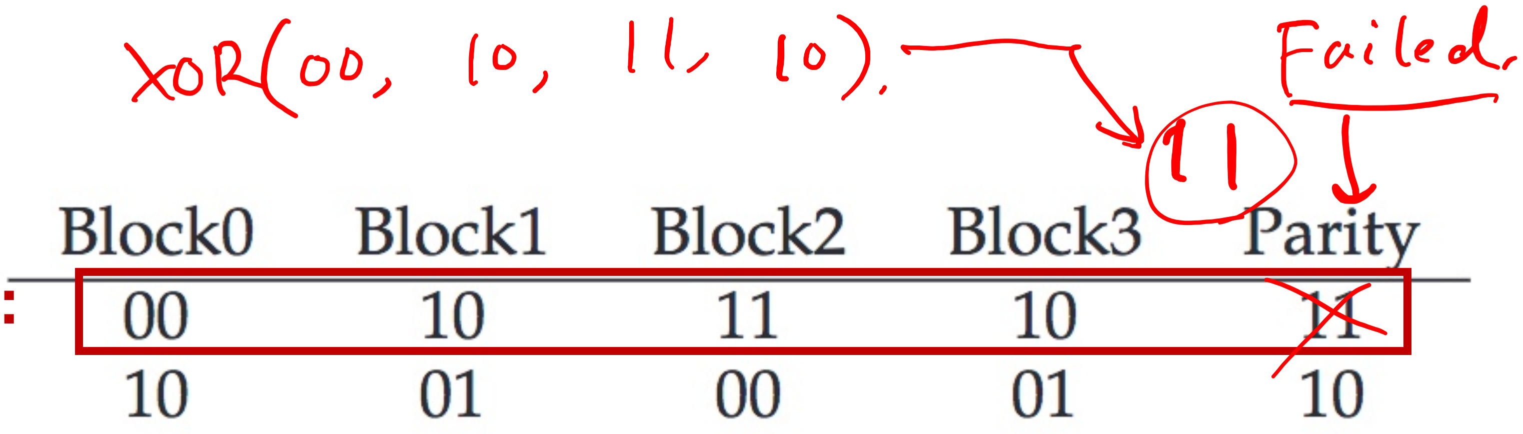
	Block0	Block1	Block2	Block3	Parity
stripe:	X	10	11	10	11
	10	01	00	01	10

Block0 = XOR(10,11,10,11) = 00

XOR function:

- P = 0: The number of 1 in a stripe must be an even number
- P = 1: The number of 1 in a stripe must be an odd number

Parity Function: XOR Example



XOR(00, 10, 11, 10). → 11

Failed

stripe:

Block0	Block1	Block2	Block3	Parity
00	10	11	10	~~11~~
10	01	00	01	10

Block0 = XOR(10,11,10,11) = **00**

XOR function:

- P = 0: The number of 1 in a stripe must be an even number
- P = 1: The number of 1 in a stripe must be an odd number

RAID-4 Analysis

effective

per-disk capacity

1. What is capacity? (N-1) * C

2. How many disks can fail? 1

3. Throughput?
   - Seq read: (N-1) * S
   - Seq write: (N-1) * S
   - Rand read: (N-1) * R
   - Rand write: R/2

logical W → Pysical writes

R W W.

4. Latency? D, 2D → two physical writes

D_1	D_2	D_3	D_4	P.
1	2	3	4	P_0
5	6	7	8	P_1

W

Data	Parity
(N-1)	1

Linear equation.

Erasure Coding

(D, P)

D=4 P=2

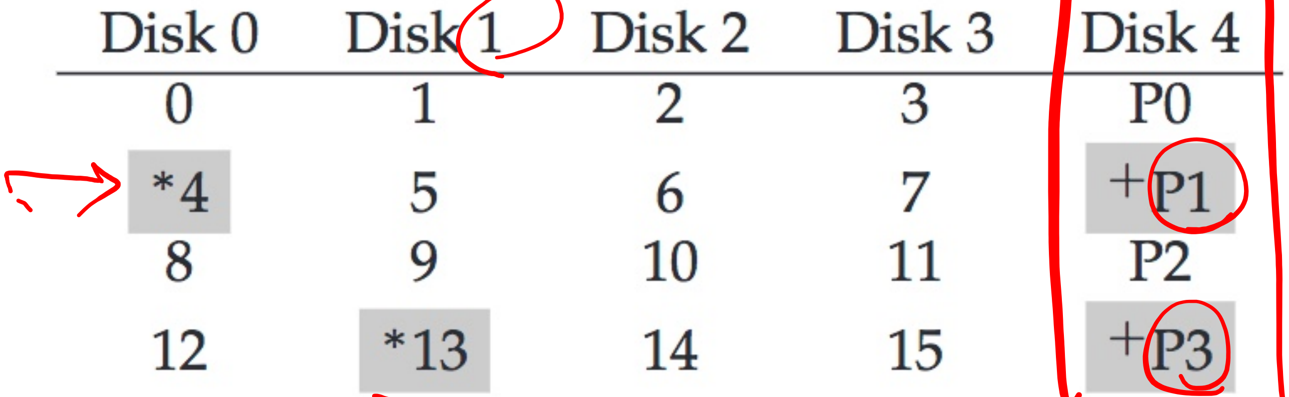
2 failures.

Out of scope

RAID-4 Analysis: Random Write

R/2

Random write to 4, 13, and respective parity blocks



Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
*4	5	6	7	+P1
8	9	10	11	P2
12	*13	14	15	+P3

serialized at bottleneck

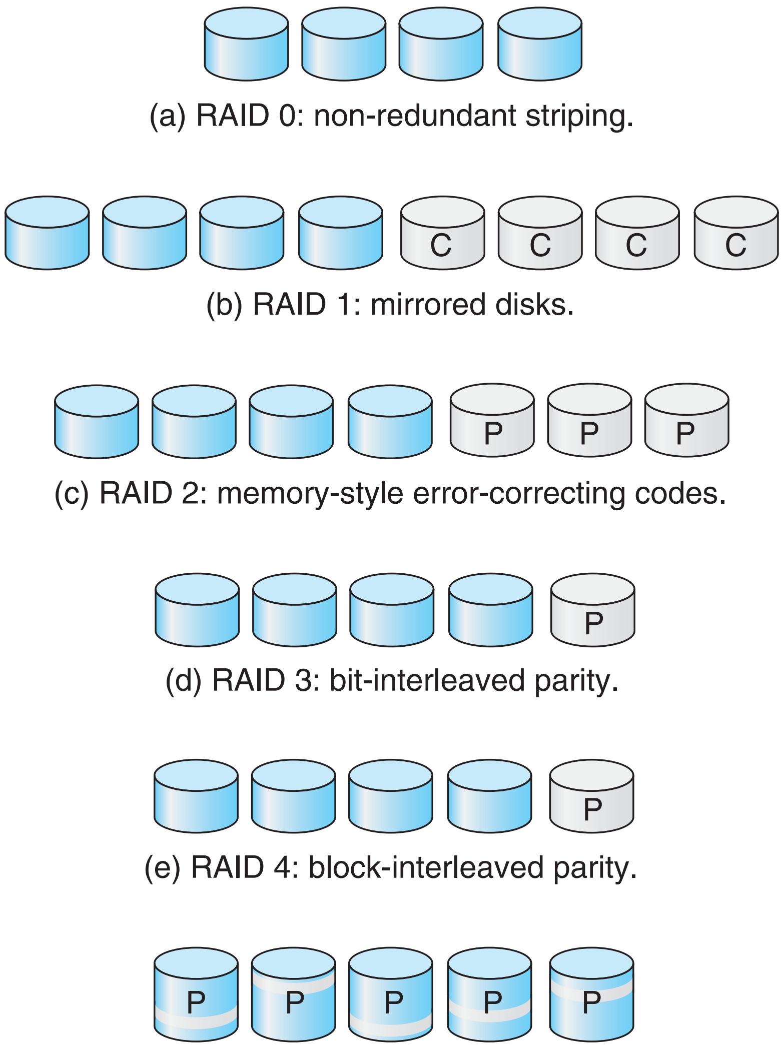
Parity Disk

13

Small write problem (for parity-based RAIDs): Parity disk serializes all random writes; and each **logical** I/O generates two **physical** I/Os (**one read and one write for parity P1**)

R/2

RAID Level 5



(a) RAID 0: non-redundant striping.

(b) RAID 1: mirrored disks.

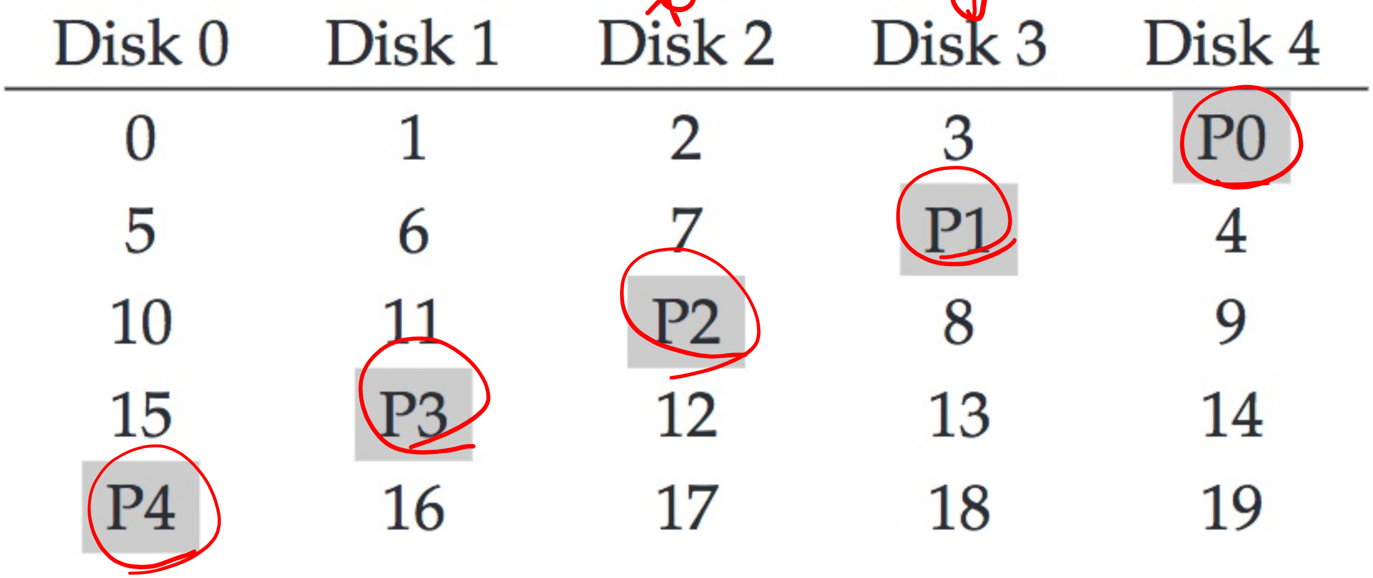
(c) RAID 2: memory-style error-correcting codes.

(d) RAID 3: bit-interleaved parity.

(e) RAID 4: block-interleaved parity.

(f) RAID 5: block-interleaved distributed parity.

RAID-5: Rotating Parity



Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
5	6	7	P1	4
10	11	P2	8	9
15	P3	12	13	14
P4	16	17	18	19

stripes.

offset 0.

offset 1

offset 2.

RAID-5 works almost identically to RAID-4, except that it rotates the parity block across drives

RAID-5 Analysis

RAID-4

1. What is capacity? (N-1) * C
2. How many disks can fail? 1
3. Throughput?
   - Seq read: (N-1) * S
   - Seq write: (N-1) * S
   - Rand read: N * R
   - Rand write: ???
4. Latency? D, 2D

RAID-5: Random Write

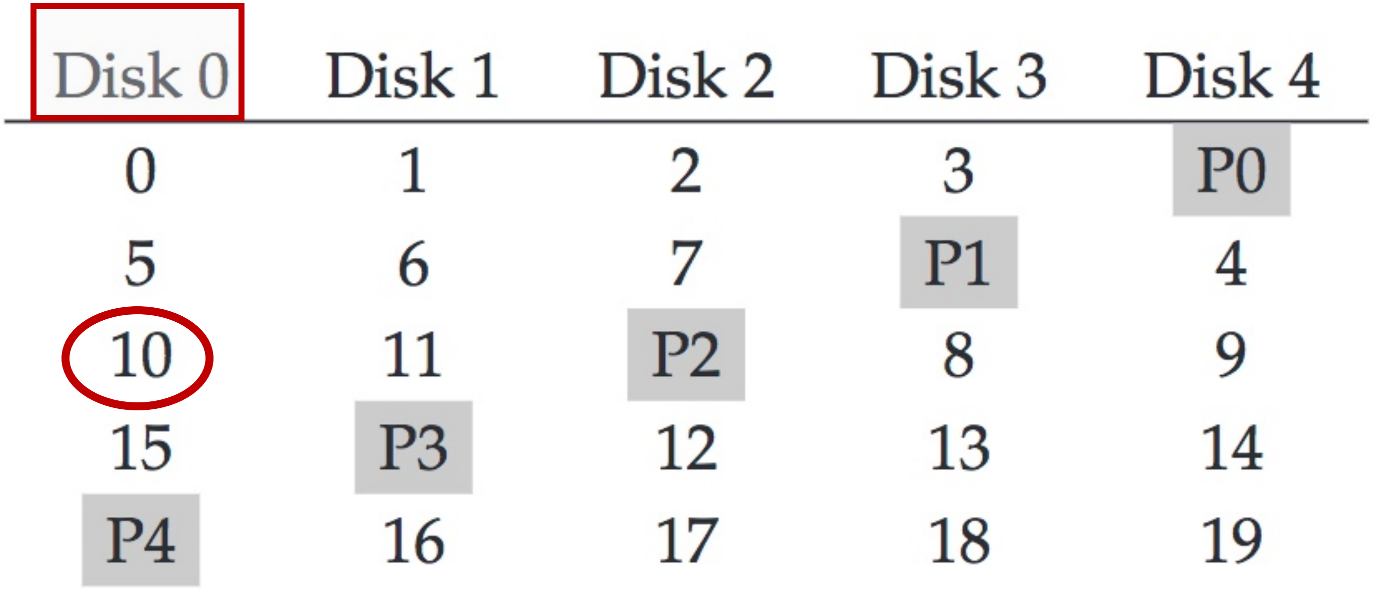
logical Write

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
5	6	7	P1	4
10	11	P2	8	9
15	P3	12	13	14
P4	16	17	18	19

Random write to Block 10 on Disk 0

RAID-5: Random Write

1. Read

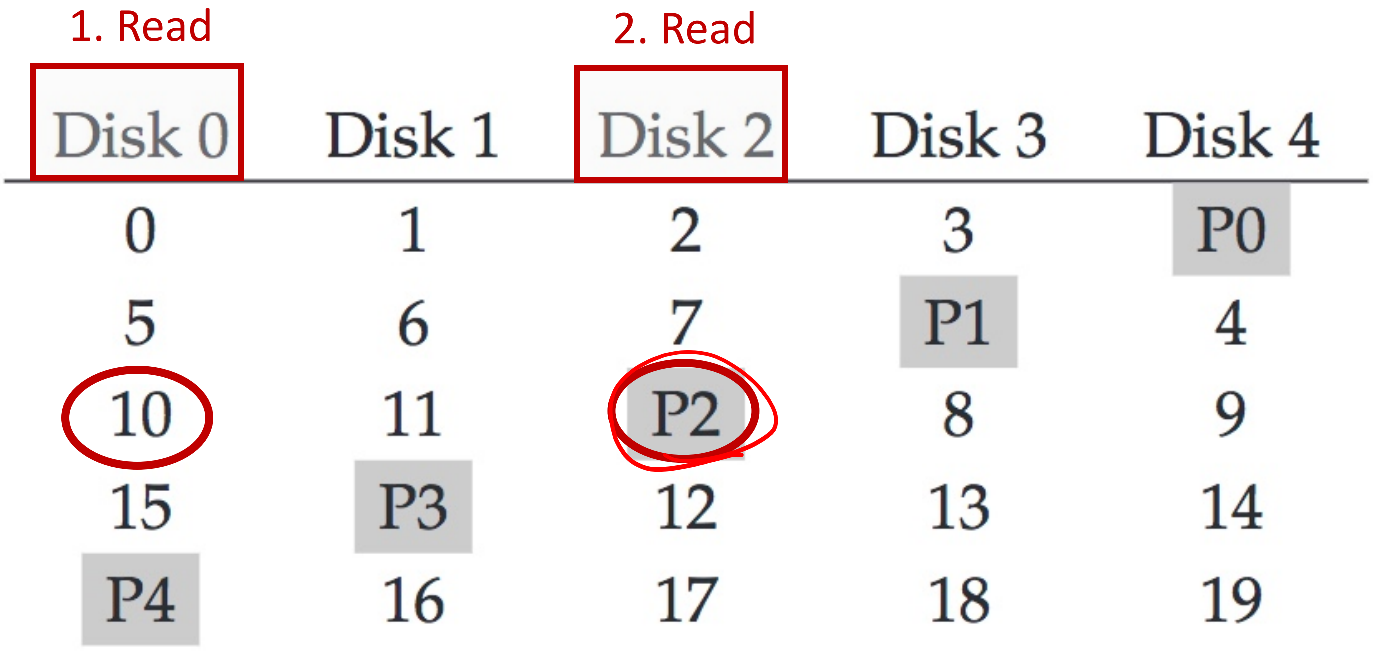


Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
5	6	7	P1	4
10	11	P2	8	9
15	P3	12	13	14
P4	16	17	18	19

Random write to Block 10 on Disk 0

physical op → 1. Read Block 10 → mem.

RAID-5: Random Write



1. Read		2. Read		
Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
5	6	7	P1	4
10	11	P2	8	9
15	P3	12	13	14
P4	16	17	18	19

Random write to Block 10 on Disk 0

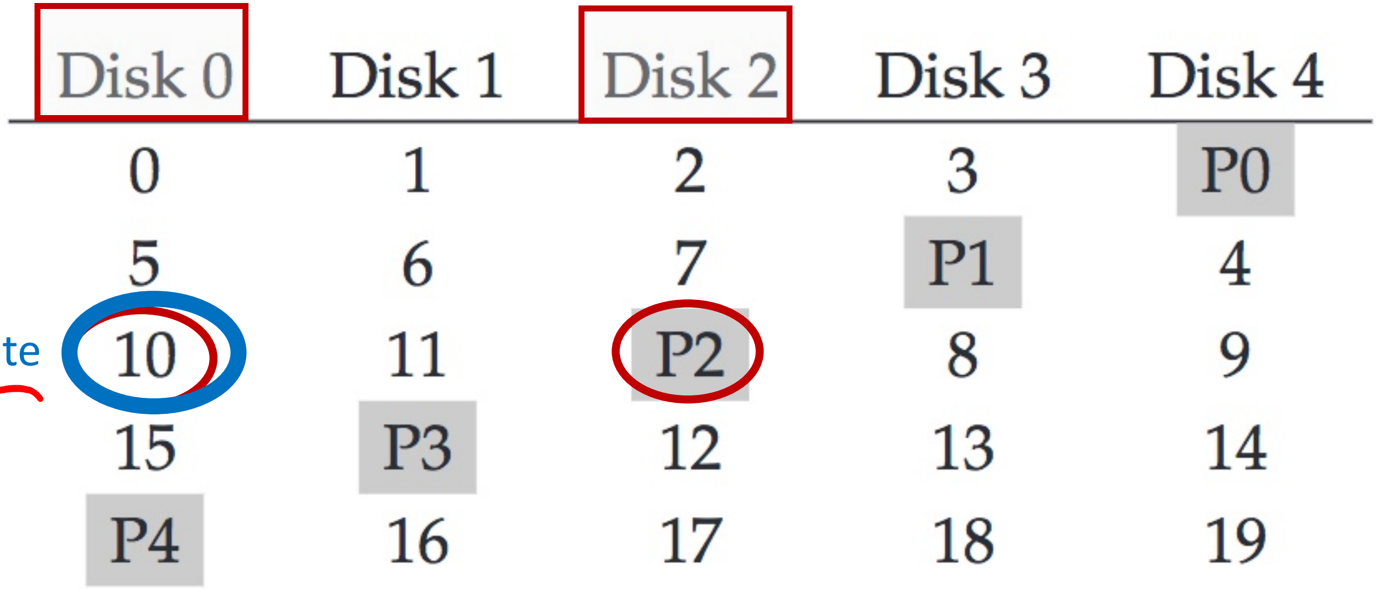
1. Read Block 10

physical op → 2. Read the Parity P2

RAID-5: Random Write

1. Read

2. Read



Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
5	6	7	P1	4
10	11	P2	8	9
15	P3	12	13	14
P4	16	17	18	19

3. Write

Random write to Block 10 on Disk 0

1. Read Block 10
2. Read the Parity P2
3. Write new data in Block 10

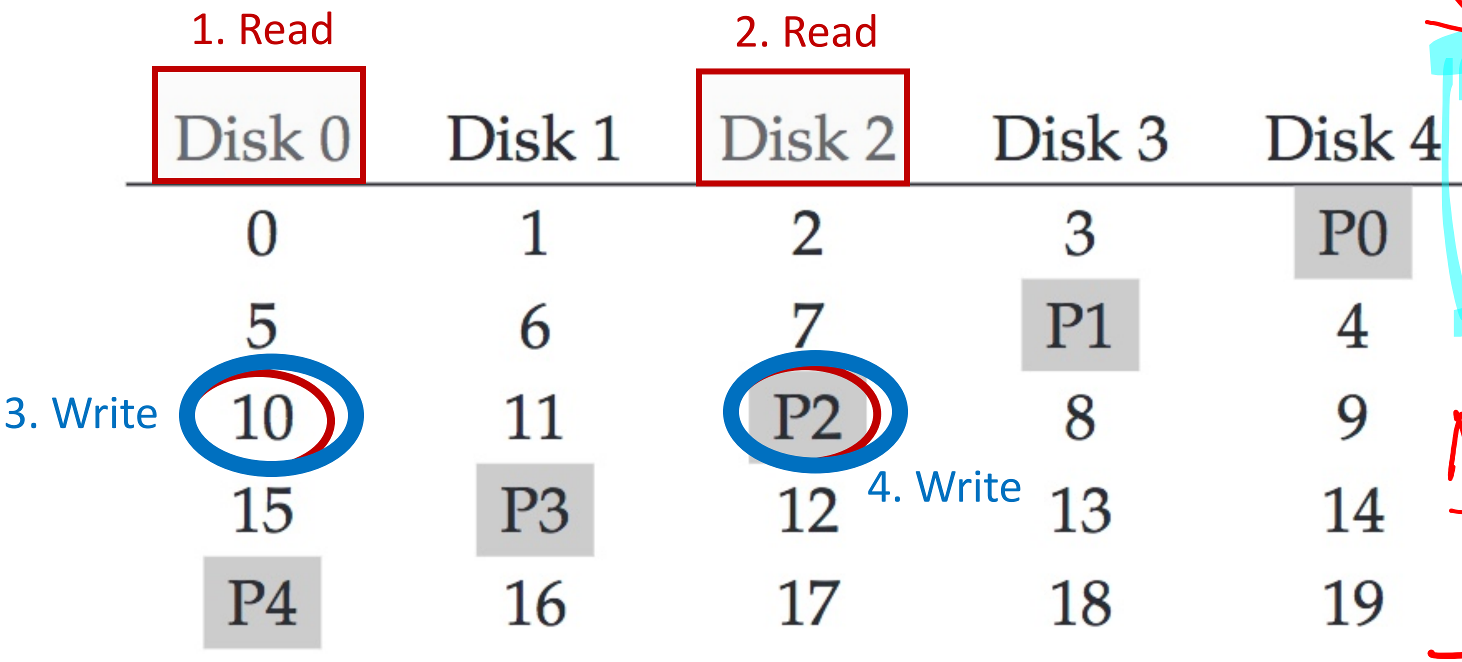
3rd physical op

RAID-5: Random Write

1 single logical Random write op

⇓

4 physical small I/Os



1. Read Disk 0	Disk 1	2. Read Disk 2	Disk 3	Disk 4
0	1	2	3	P0
5	6	7	P1	4
10	11	P2	8	9
15	P3	12	13	14
P4	16	17	18	19

3. Write

4. Write

$N * R$

$$\frac{5 \times R}{4}$$

Random write to Block 10 on Disk 0

1. Read Block 10
2. Read the Parity P2
3. Write new data in Block 10
4. Write new parity P2

4th physical op

RAID-5: Random Write

$\frac{R}{2} \rightarrow \frac{N \times R}{4}$

$\frac{5R}{4}$

$\frac{5}{4} > \frac{1}{2}$

Write 1	Write 2	Write 1 touches P2		Write 2 touches P0
Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
5	6	7	P1	4
10	11	P2	8	9
15	P3	12	13	14
P4	16	17	18	19

Performance reasoning

Generally, for a large number of random read/write requests, RAID-5 will be able to keep all disks busy: thus **N * R**

Rand W

↓

Each random (RAID-5) writes generates 4 physical I/O operations: thus **N * R / 4**

RAID-5 Analysis

1. What is capacity? (N-1) * C

2. How many disks can fail? 1

3. Throughput?
   - Seq read: (N-1) * S
   - Seq write: (N-1) * S
   - Rand read: N * R
   - Rand write: N * R/4

4. Latency? D, 2D

Summary: All RAID’s

	Reliability	Capacity
RAID-0	0	C * N
RAID-1	1 or N/2	C * N/2
RAID-4	1	N-1
RAID-5	1	N-1

Summary: All RAID's

	Seq Read	Seq Write	Rand Read	Rand Write
RAID-0	N * S	N * S	N * R	N * R
RAID-1	N/2 * S	N/2 * S	N * R	N/2 * R
RAID-4	(N-1) * S	(N-1) * S	(N-1) * R	R/2
RAID-5	(N-1) * S	(N-1) * S	N * R	N/4 * R

DO Read the Textbook!

Please do read the textbook chapter “RAID” to gain a deeper understanding of the various analyses covered in lecture.