



Computing Infrastructure

 POLITECNICO DI MILANO



RAID disks



Redundant Arrays of Independent (Inexpensive) Disks



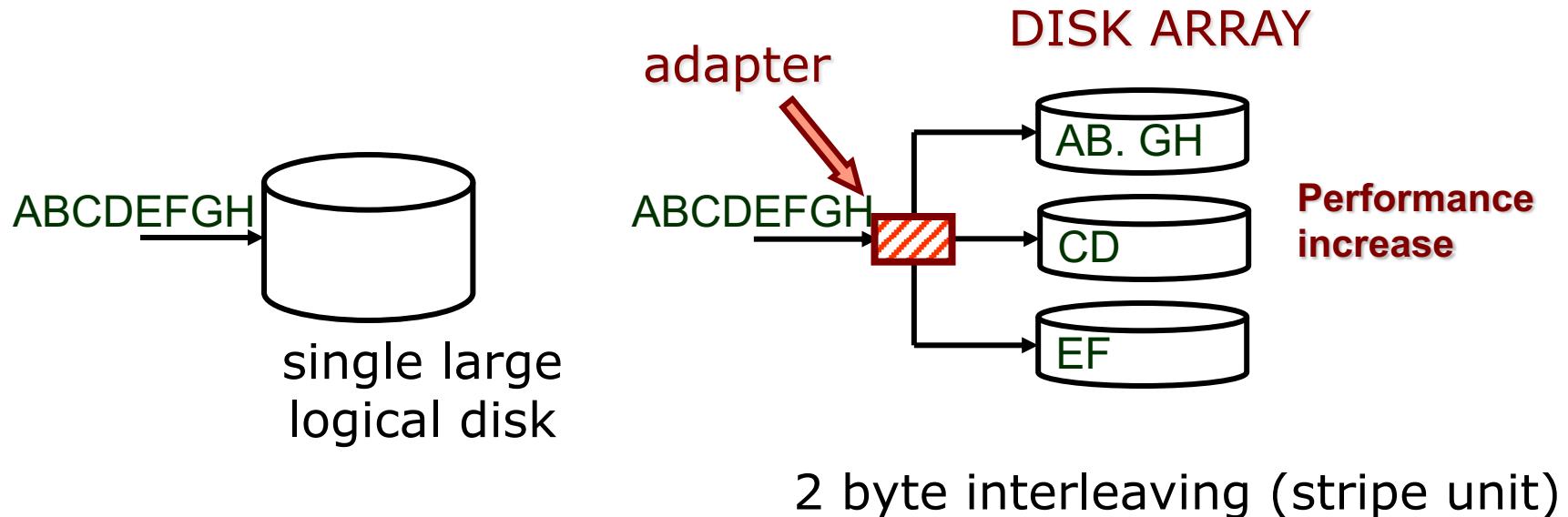
- Disk Arrays: proposed in the 1980's [PATTERSON]
- Need to increase the performance, the size and the reliability of storage systems
- Several *independent* disks that are considered as a single, large, high-performance logical disk
 - In contrast with the JBOD (Just a Bunch of Disks) method where each disk is a separate device with a different mount point
- the data are *striped* across several disks accessed in parallel:
 - **high data transfer rate:** large data accesses (heavy I/O op.)
 - **high I/O rate:** small but frequent data accesses (light I/O op.)
 - **load balancing** across the disks

Two orthogonal techniques:

- **data striping:** to improve **performance**
- **redundancy:** to improve **reliability**



parallelism in a I/O operation (a simple example)



- *I/O virtualization*

- data are distributed transparently over the disks (no action is required to the users)



- **striping**: data are written sequentially (a vector, a file, a table, ...) in units (stripe unit: bit, byte, blocks) on multiple disks according to a cyclic algorithm (*round robin*)
- **stripe unit**: dimension of the unit of data that are written on a single disk
- **stripe width**: number of disks considered by the striping algorithm
 1. **multiple independent I/O requests** will be executed in parallel by several disks decreasing the **queue length** (and **time**) of the disks
 2. **single multiple-block I/O requests** will be executed by multiple disks in parallel **increasing** of the **transfer rate** of a single request



the more physical disks in the array

the **larger** the **size** and **performance** gains
but

the **larger** the **probability of failure** of a disk



this is the main motivation for the introduction of
redundancy



- The probability of a failure (assuming independent failures) in an array of 100 disks is 100 higher the probability of a failure of a single disk
 - if a disk has an Mean Time To Failure (MTTF) of 200,000 hours (~23 years) an array of 100 disks will show a MTTF of 2000 hours (~ 3 months)
- **Redundancy:** error correcting codes (stored on disks different from the ones with the data) are computed to tolerate loss due to disk failures
- Since write operations must update also the redundant information, their performance is **worse** than the one of the traditional writes



example (elementary) of a data reconstruction

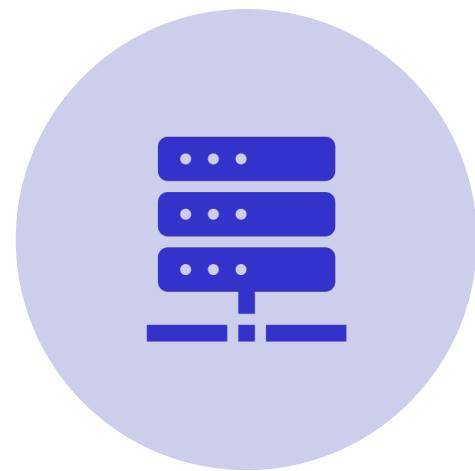


Disk 0	Disk 1	Disk 2	redundant data
10	8	2	20

Values

failed = 20 - (10 + 2) = 8

Sum of the data



RAID



Beyond Single Disks



Hard drives are great devices

- Relatively fast, persistent storage

Shortcomings:

- How to cope with disk failure?
 - Mechanical parts break over time
 - Sectors may become silently corrupted
- Capacity is limited
 - Managing files across multiple physical devices is cumbersome
 - Can we make 10x 1 TB drives look like a 10 TB drive?

Redundant Array of Inexpensive Disks



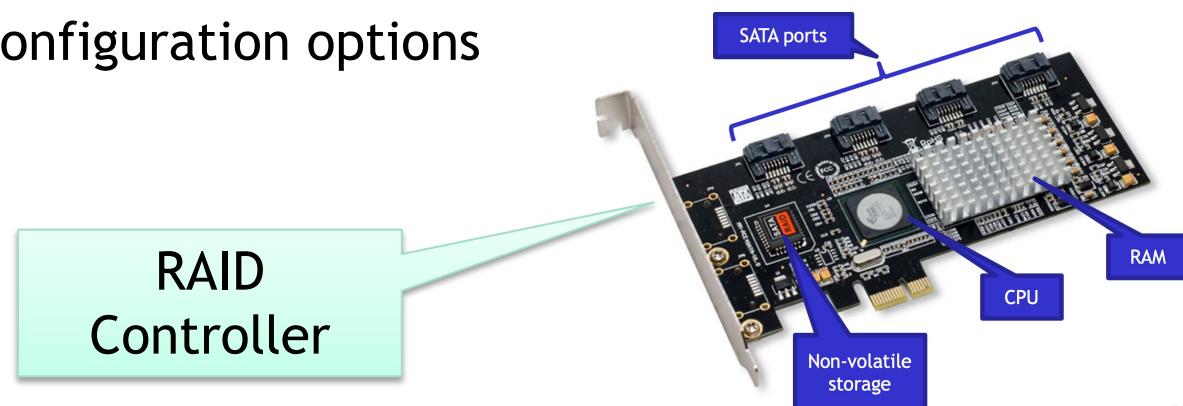
RAID: use multiple disks to create the illusion of a large, faster, more reliable disk

Externally, RAID looks like a single disk

- i.e. RAID is **transparent**
- Data blocks are read/written as usual
- No need for software to explicitly manage multiple disks or perform error checking/recovery

Internally, RAID is a complex computer system

- Disks managed by a dedicated CPU + software
- RAM and non-volatile memory
- Many different configuration options
 - **RAID levels**





- **RAID 0** striping only
- **RAID 1** mirroring only
 - **RAID 0+1** (nested levels)
 - **RAID 1+0** (nested levels)
- RAID 2 bit interleaving (not used)
- RAID 3 byte interleaving - redundancy (parity disk)
- **RAID 4** block interleaving - redundancy (parity disk)
- **RAID 5** block interleaving - redundancy (parity block distributed)
- **RAID 6** greater redundancy (2 failed disks are tolerated)

Only «**RED ARCHITECTURES**» are part of the course

RAID level 0: *striping, no redundancy*

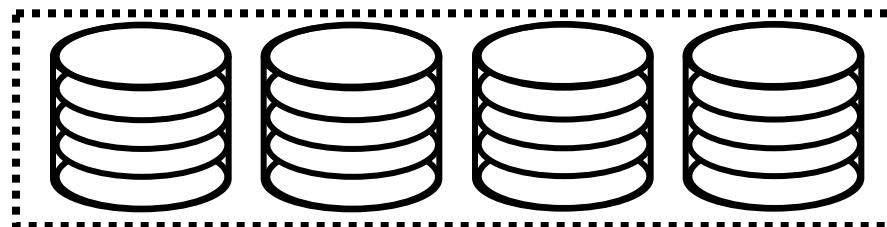


Data are written on a single logical disk and splitted in several blocks distributed across the disks according to a striping algorithm

- used where **performance** and **capacity**, rather than **reliability**, are the primary concerns, minimum two drives required
- + **lowest cost** because it does not employ redundancy (no error-correcting codes are computed and stored)
- + **best write performance** (it does not need to update redundant data and it is parallelized)
- **single** disk failure will result in data loss

disk array

data



user capacity of 4 physical disks configured as a single logical disk RAID 0

used by the servers with read-only data

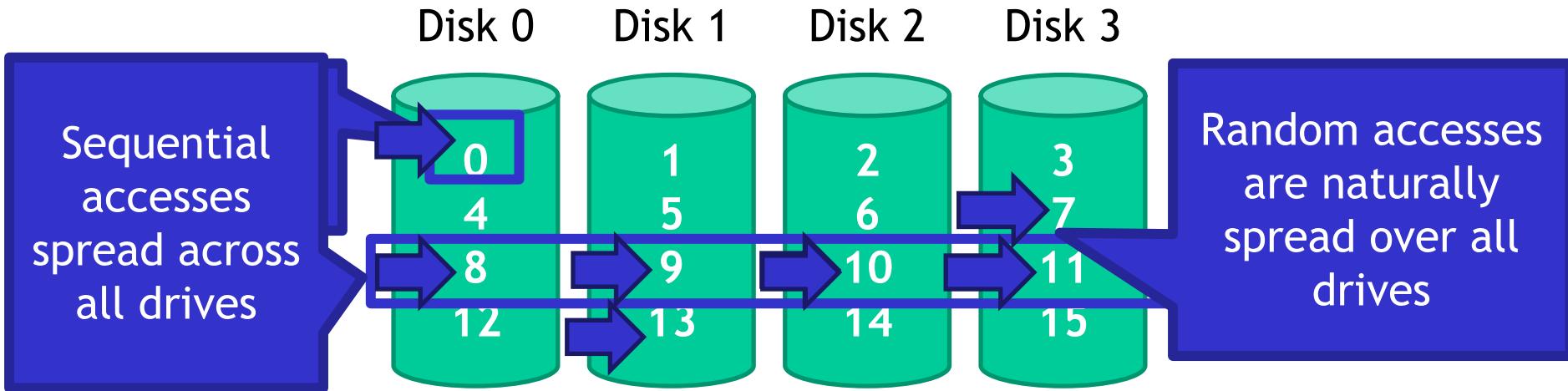
	Disk 0	Disk 1	Disk 2	Disk 3	
Stripe 1	<i>Block 1</i>	<i>Block 2</i>	<i>Block 3</i>	<i>Block 4</i>	
Stripe 2	<i>Block 5</i>	<i>Block 6</i>	<i>Block 7</i>	<i>Block 8</i>	



RAID 0: Striping



- Key idea: present an **array** of disks as a single large disk
- Maximize parallelism by **striping** data cross all N disks





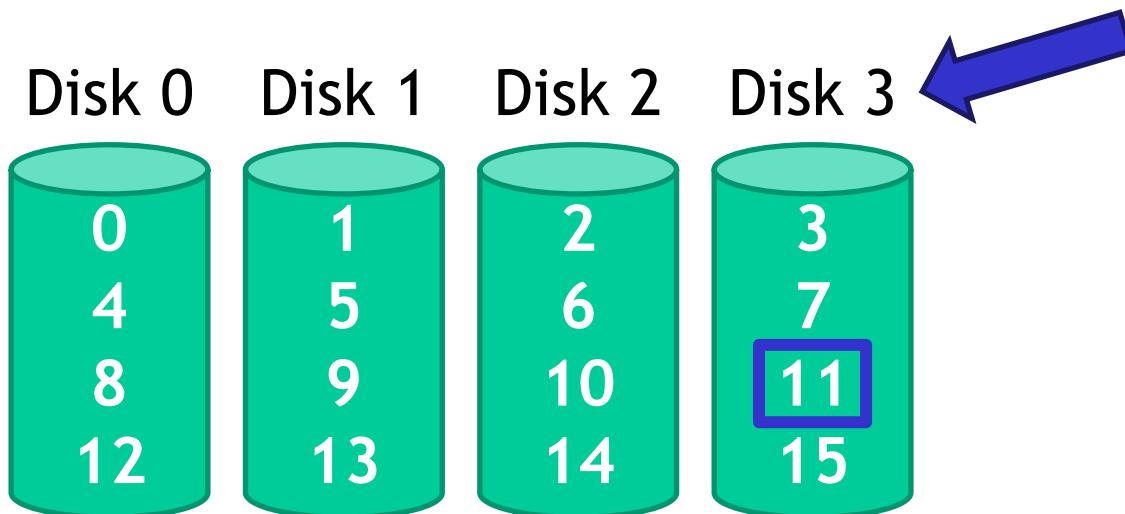
Addressing Blocks

How do you access specific data blocks?

- Disk = $\text{logical_block_number \% number_of_disks}$
- Offset = $\text{logical_block_number / number_of_disks}$

Example: read block 11

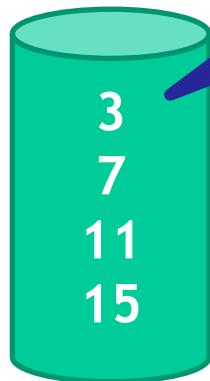
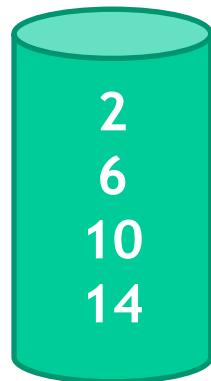
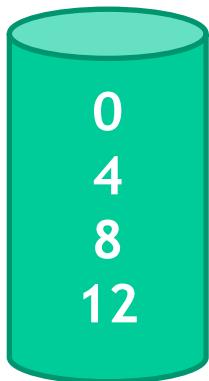
- $11 \% 4 = \text{Disk 3}$
- $11 / 4 = \text{Physical Block 2 (starting from 0)}$





Chunk Sizing

Disk 0 Disk 1 Disk 2 Disk 3

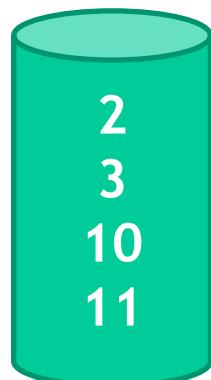
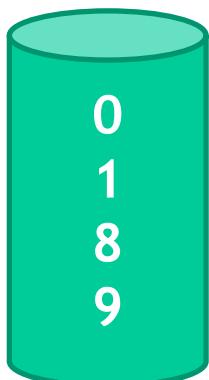


Chunk size = 1 block

Chunk size impacts array performance

- Smaller chunks → greater parallelism
- Big chunks → reduced seek times

Disk 0 Disk 1 Disk 2 Disk 3



Typical arrays use 64KB chunks

Chunk size = 2 block

Measuring RAID Performance (1) - Seq. Transfer Rate



As usual, we focus on **sequential** and **random** workloads

Assume disks in the array have **sequential transfer rate S**

E.g. Single large transfer

- 10 MB transfer
- $S = \text{transfer_size} / \text{time_to_access}$
- $10 \text{ MB} / (7 \text{ ms} + 3 \text{ ms} + 10 \text{ MB} / 50 \text{ MB/s}) = 47.62 \text{ MB/s}$



Average seek time	7 ms
Average rotational delay	3 ms
Transfer rate	50 MB/s

Measuring RAID Performance (2) - Rnd. Transfer Rate



As usual, we focus on **sequential** and **random** workloads

Assume disks in the array have **random transfer rate R**

E.g. Set of small files

- 10 KB transfer
- $R = \text{transfer_size} / \text{time_to_access}$
- $10 \text{ KB} / (7 \text{ ms} + 3 \text{ ms} + 10 \text{ KB} / 50 \text{ MB/s}) = 0.98 \text{ MB/s}$



Average seek time	7 ms
Average rotational delay	3 ms
Transfer rate	50 MB/s



Analysis of RAID 0



Capacity: N

- All space on all drives can be filled with data

Reliability: 0

- If any drive fails, data is permanently lost
- MTTDL = MTTF

Sequential read and write: $N * S$

- Full parallelization across drives

Random read and write: $N * R$

- Full parallelization across all drives

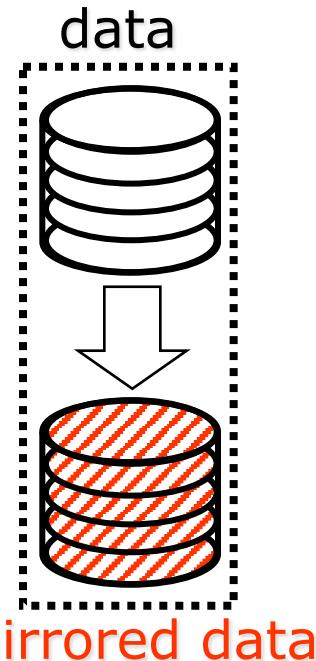


RAID level 1: *mirroring*



whenever data is written to a disk it is also duplicated (**mirrored**) to a second disk (there are always **two copies** of the data), minimum 2 disk drives

- + **high reliability**: when a disk fails the second copy is used
- + **read** of a data: it can be retrieved from the disk with the **shorter** queueing, seek, and latency delays
- + **fast writes** (no error correcting code should be computed) – but still slower than standard disks (due to duplication)
- **high costs** (50% of the capacity is used)



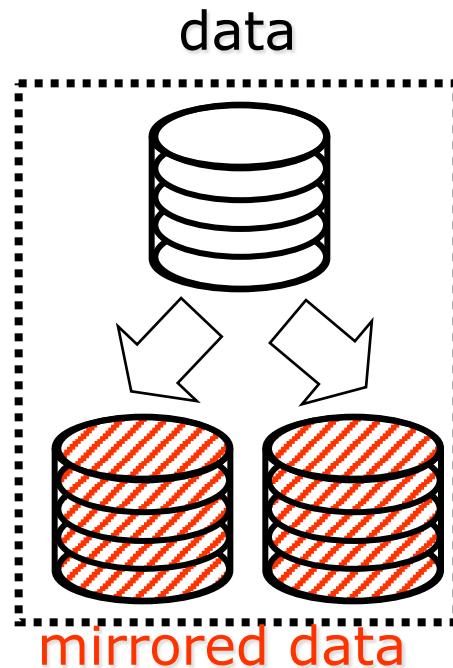
Disk 0	Disk 1
Block 1	Block 1
Block 2	Block 2
Block 3	Block 3



RAID level 1: *mirroring (Theoretical)*



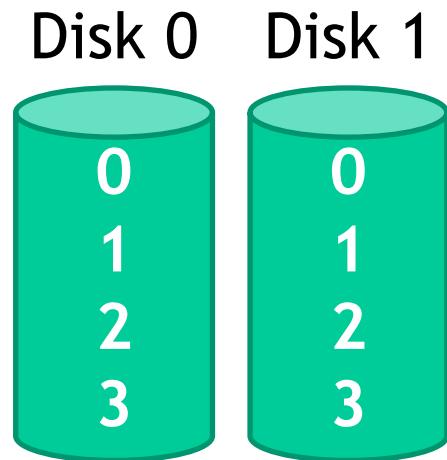
- In principle, a RAID 1 can mirror the content over more than one disk.
 - This give resiliency to errors even if more than one disk breaks
 - It allows with a *voting mechanism* to identify errors not reported by the disk controller.
- In practice this is never used, because the overhead and costs are too high.



RAID 1: Mirroring - Key Idea



- RAID 0 offers high performance, but zero error recovery
- Key idea: make two copies of all data



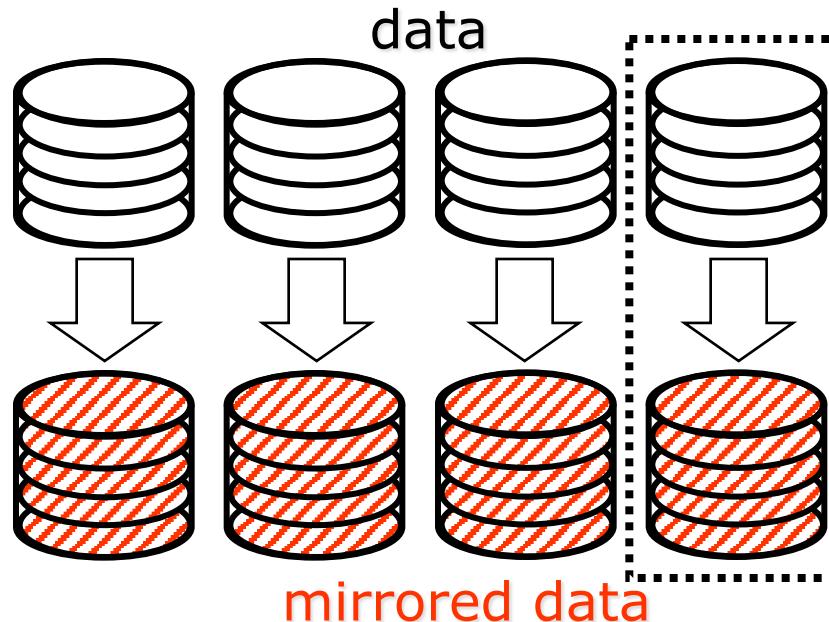


RAID level 1: *mirroring – using more disks*



- However if several disks are available (always in an even number), disks could be coupled.
- The total capacity is halved.
- Each disk has a mirror.
- How to organize this? RAID levels can be combined
 - Raid 0 + 1
 - Raid 1 + 0

**user capacity
of 4 disks
(for data)**





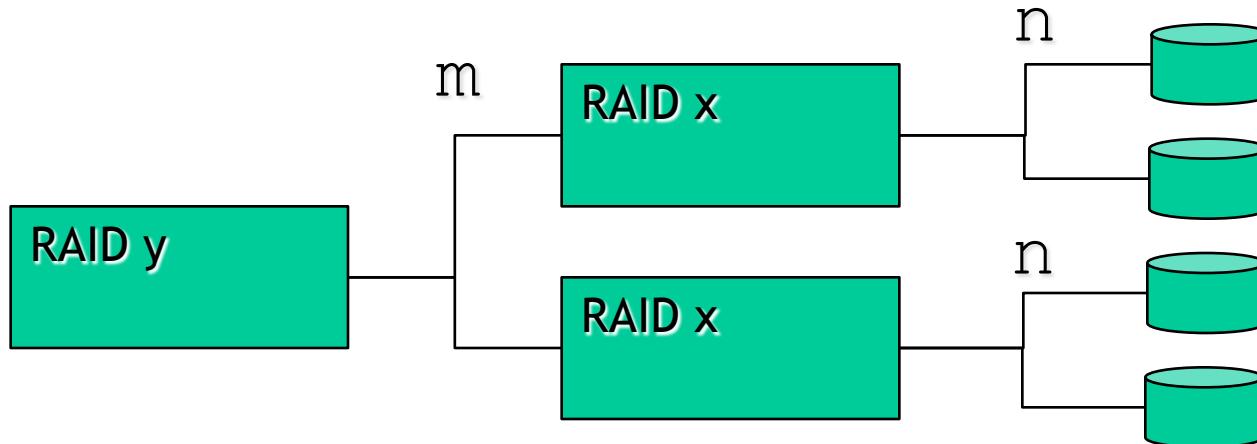
RAID levels (combined)



RAID levels can be combined

RAID $x + y$ (or RAID xy) =>

- $n \times m$ disks in total
- Consider m groups of n disks
- Apply RAID x to each group of n disks
- Apply RAID y considering the m groups as single disks





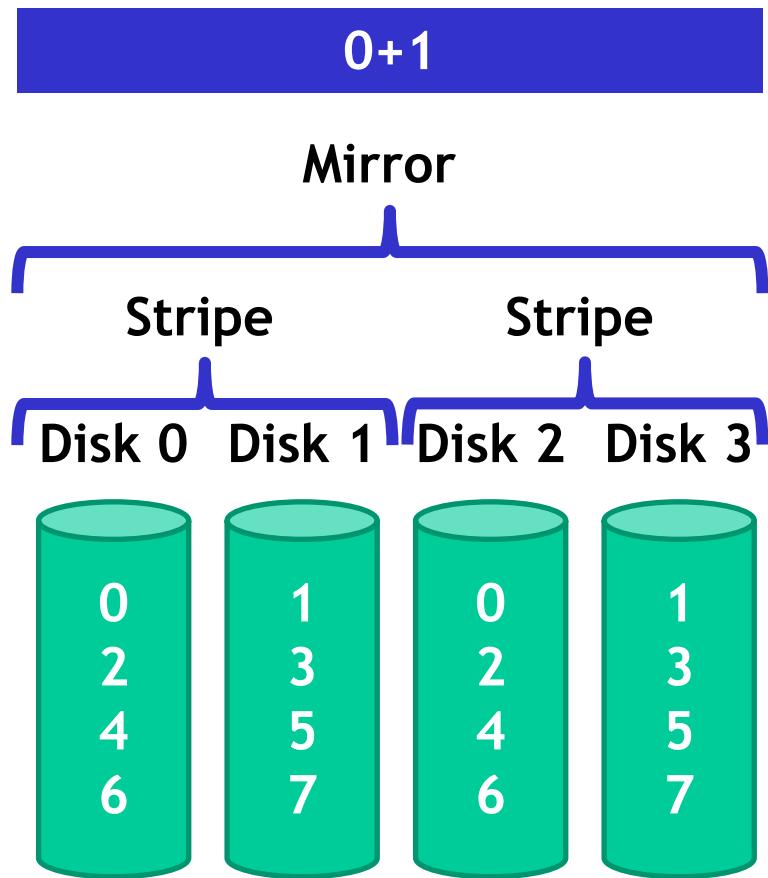
RAID level 0 + 1



"Group of striped disks (RAID 0) that are then mirrored (RAID 1)"

striping first (RAID 0)
then mirroring (RAID 1)

- minimum 4 drives
- after the first failure the model becomes as a RAID 0





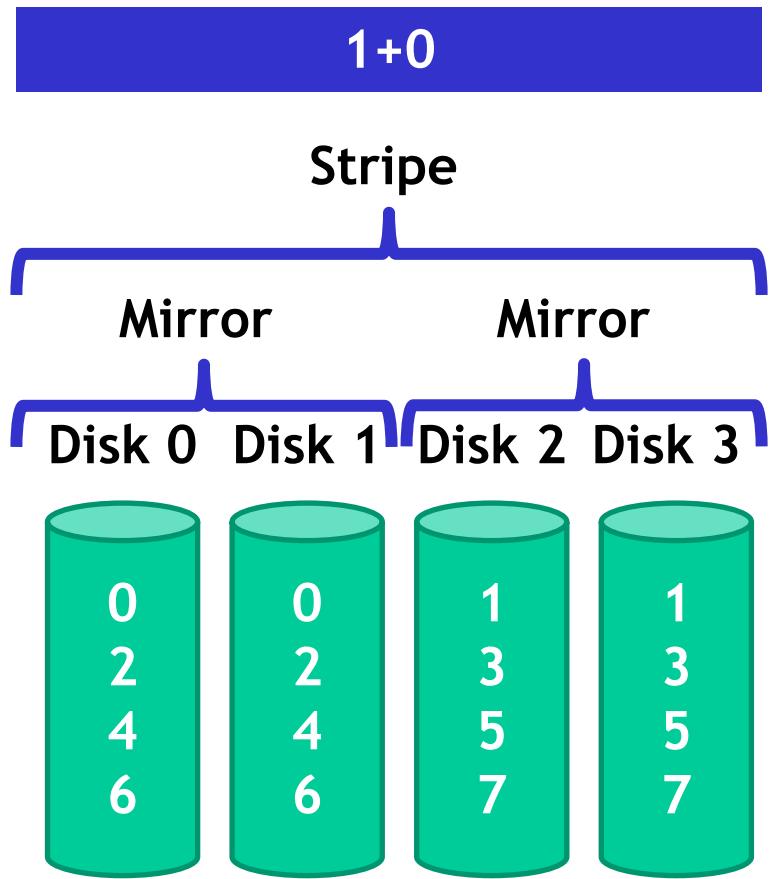
RAID level 1 + 0



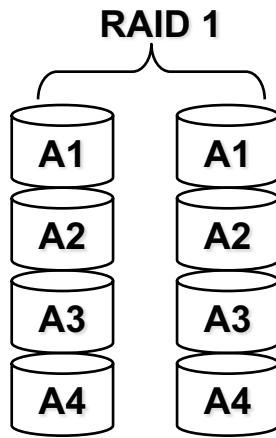
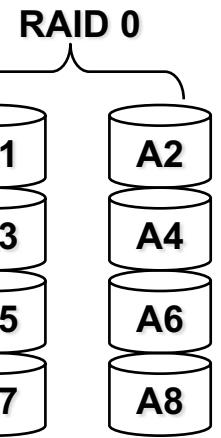
“Group of mirrored disks (RAID 1) that are then striped (RAID 0)”

mirroring first (RAID 1)
then striping (RAID 0)

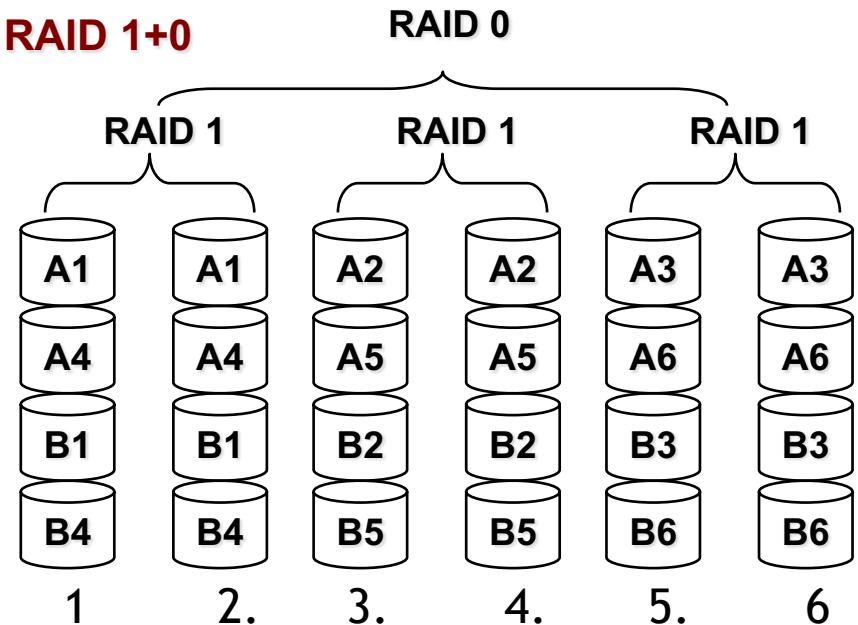
- minimum 4 drives are required
- used in databases with very high workload (fast writes)



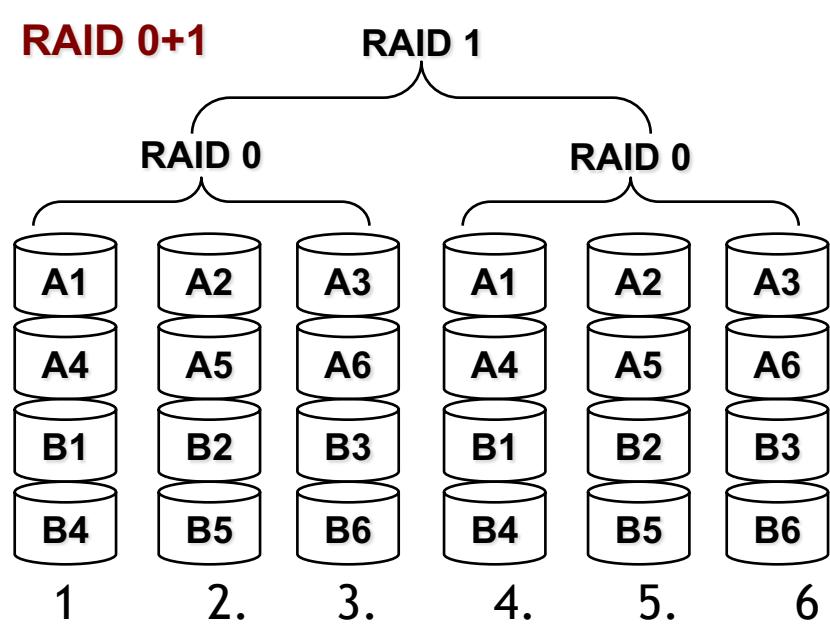
RAID 0, 1, 0+1 and 1+0 organizations



RAID 1+0



RAID 0+1

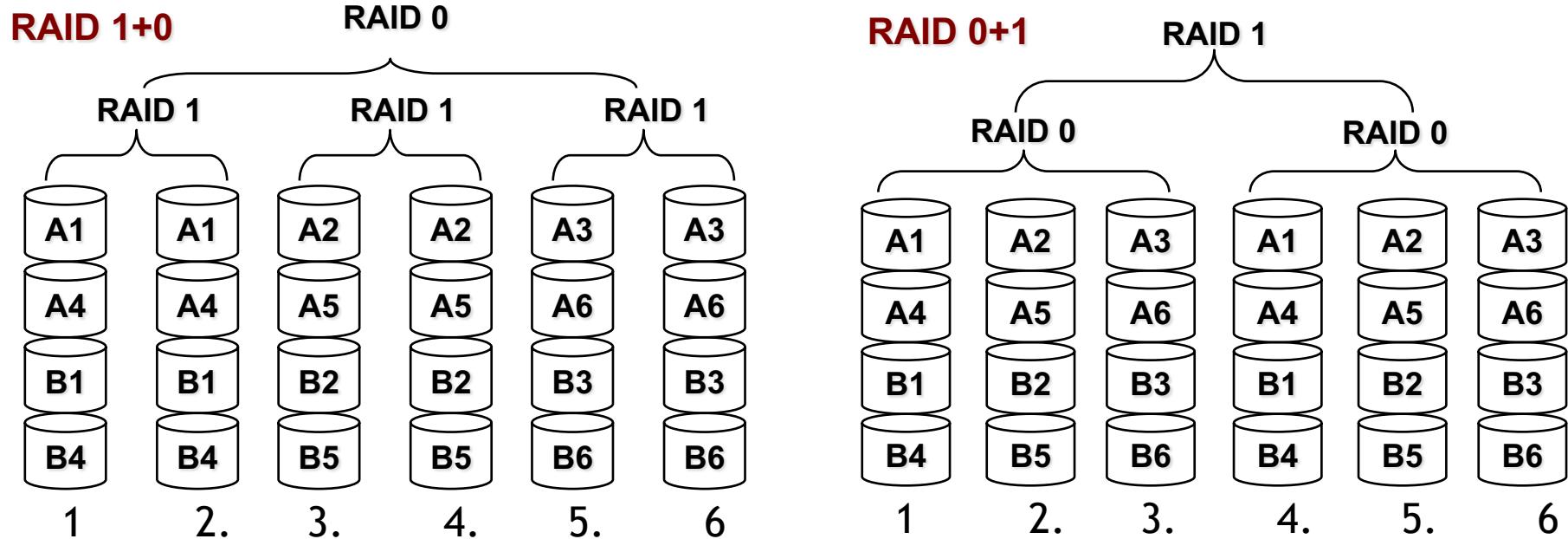




RAID 0, 1, 0+1 and 1+0 organizations



- The blocks are the same but are allocated in a different order
- Performance on both RAID 10 and RAID 01 are the same.
- The storage capacity of RAID 10 and RAID 01 is the same.



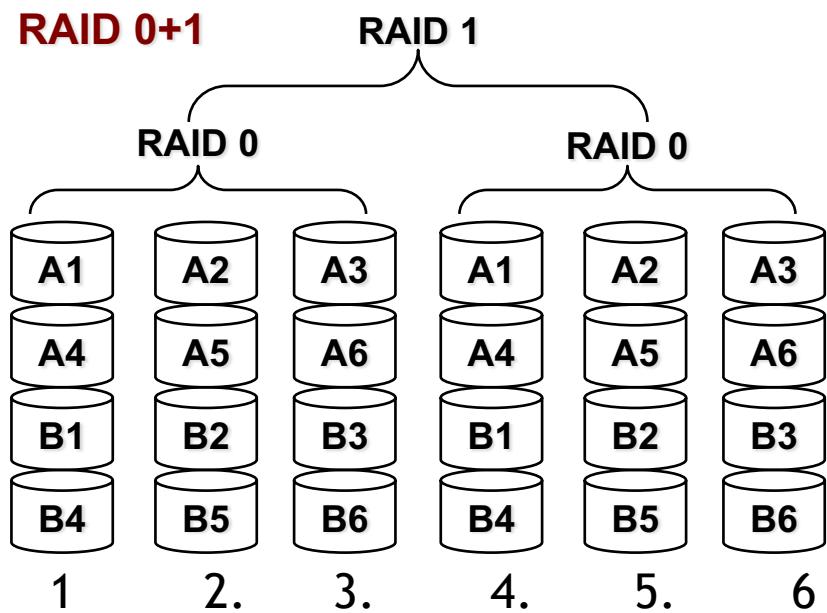
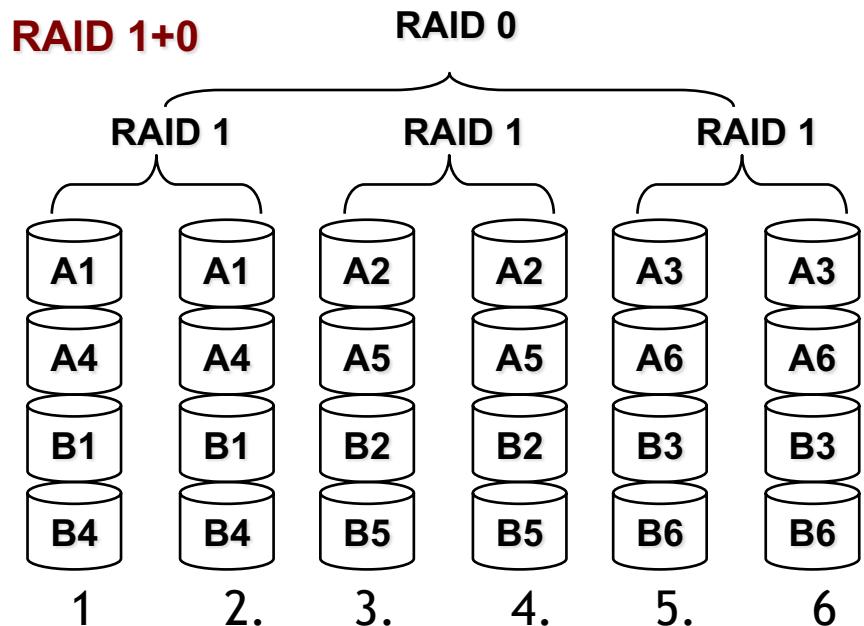


RAID 0, 1, 0+1 and 1+0 organizations



- The main difference is the **fault tolerance level**:

- ✓ On most implementations of RAID controllers,
 - ✓ RAID 0+1 fault tolerance is less.
 - ✓ RAID 1+0 fault tolerance is larger.





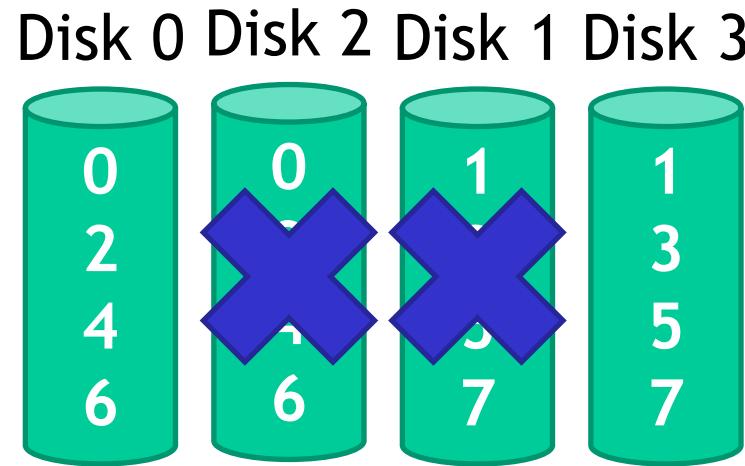
Analysis of RAID 1 (1)

Capacity: $N / 2$

- Two copies of all data, thus half capacity

Reliability: 1 drive can fail, sometime more

- If you are lucky, $N / 2$ drives can fail without data loss





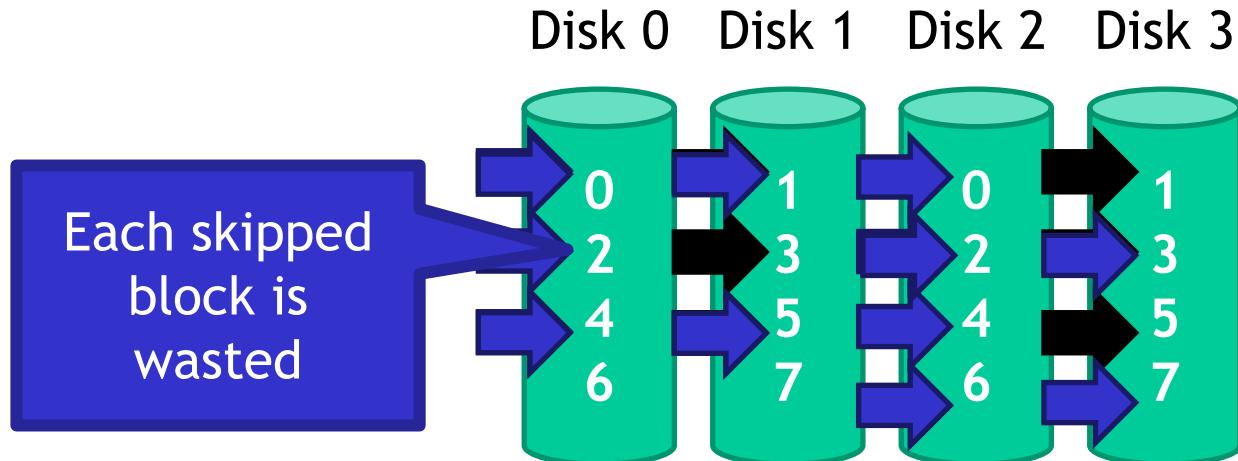
Analysis of RAID 1 (2)

Sequential write: $(N / 2) * S$

- Two copies of all data, thus half throughput

Sequential read: $(N / 2) * S$

- Half of the read blocks are wasted, thus halving throughput





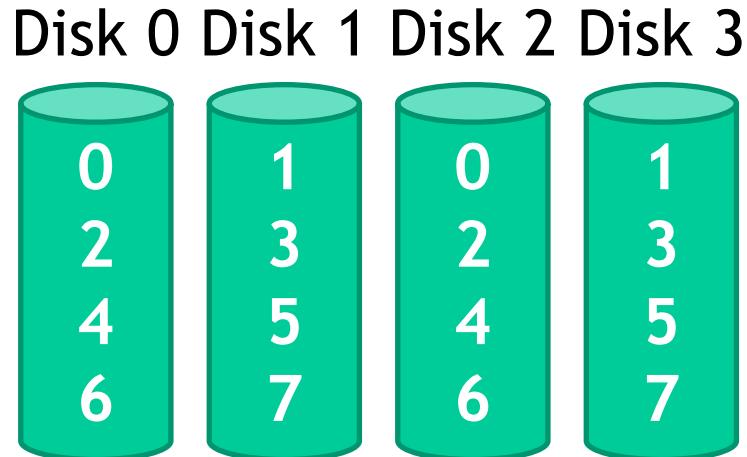
Analysis of RAID 1 (3)

Random read: $N * R$

- Best case scenario for RAID 1
- Reads can parallelize across all disks

Random write: $(N / 2) * R$

- Two copies of all data, thus half throughput

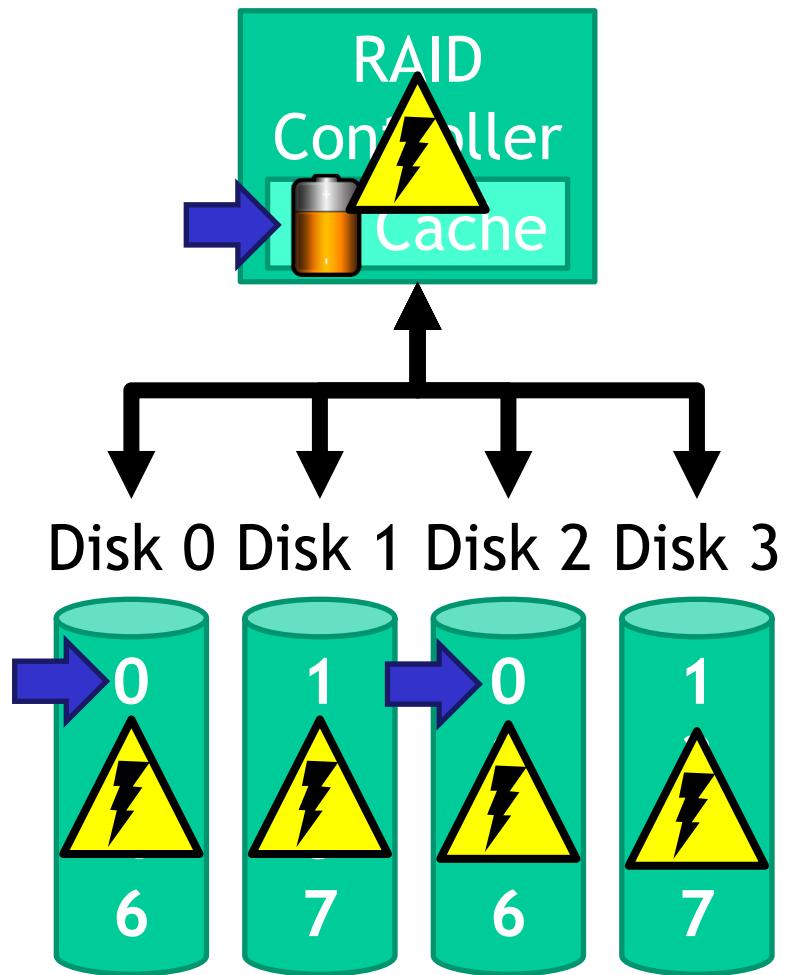




The Consistent Update Problem



- Mirrored writes should be **atomic**
 - All copies are written, or none are written
- However, this is difficult to guarantee
 - Example: power failure
- Many RAID controllers include a **write-ahead log**
 - Battery backed, non-volatile storage of pending writes
 - A recovery procedure ensures to recover the out-of-sync mirrored copies



Decreasing the Cost of Reliability



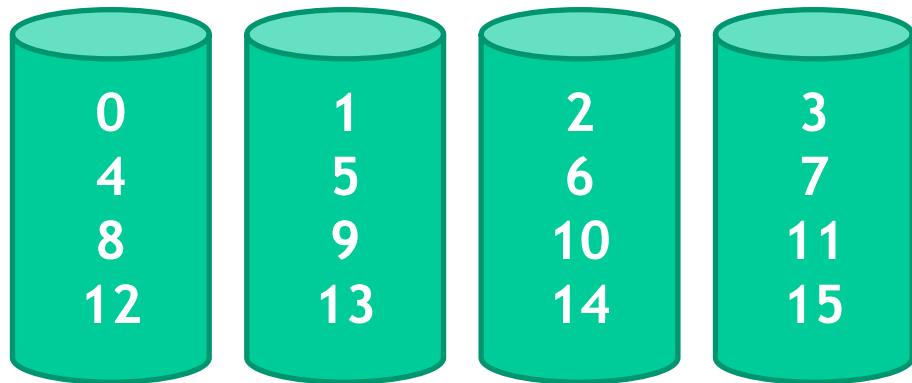
- RAID 1 offers highly reliable data storage
- But, it uses $N / 2$ of the array capacity
- Can we achieve the same level of reliability without wasting so much capacity?
 - Yes!
 - Use information coding techniques to build light-weight error recovery mechanisms



RAID 4: Parity Drive



Disk 0 Disk 1 Disk 2 Disk 3 Disk 4



Disk N only stores parity information for the other $N-1$ disks

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	0	1	1	$0 \wedge 0 \wedge 1 \wedge 1 = 0$
0	1	0	0	$0 \wedge 1 \wedge 0 \wedge 0 = 1$
1	1	1	1	$1 \wedge 1 \wedge 1 \wedge 1 = 0$
0	1	1	1	$0 \wedge 1 \wedge 1 \wedge 1 = 1$

Parity calculated using XOR

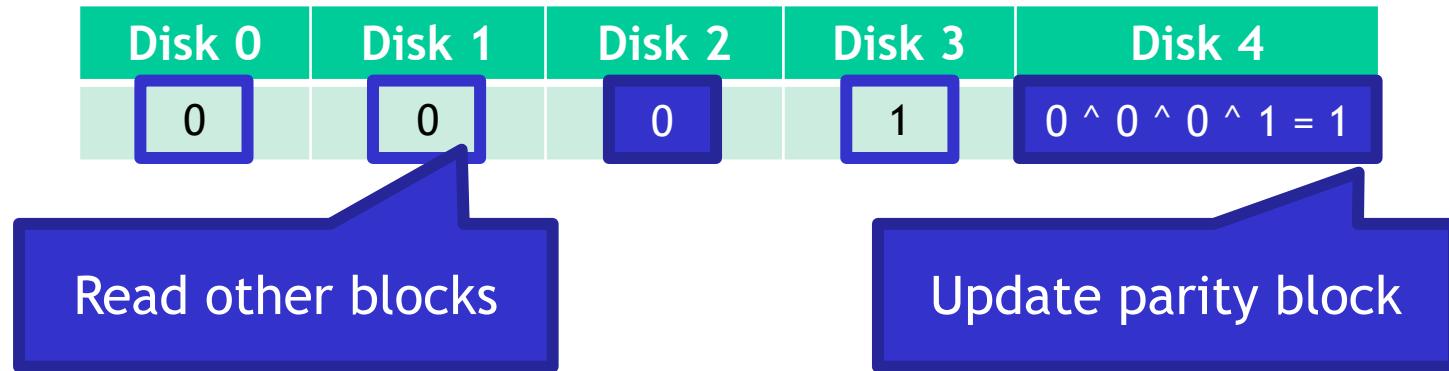


Updating Parity on Write

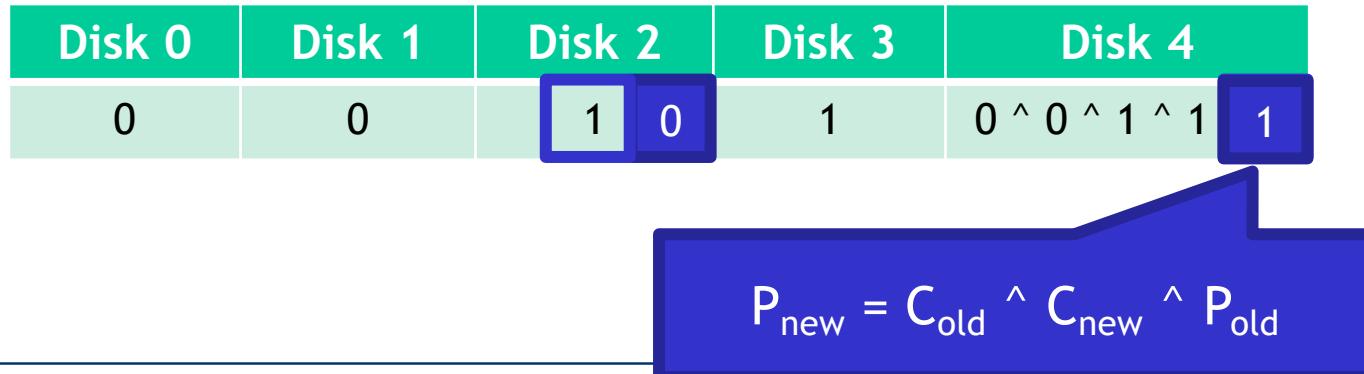


How is parity updated when blocks are written?

1. Additive parity



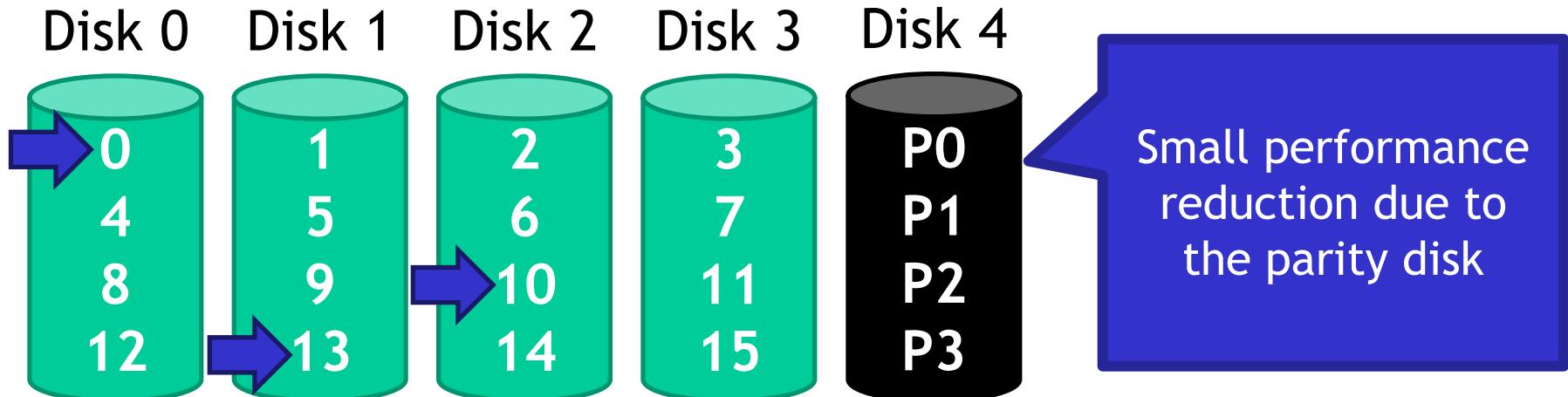
2. Subtractive parity





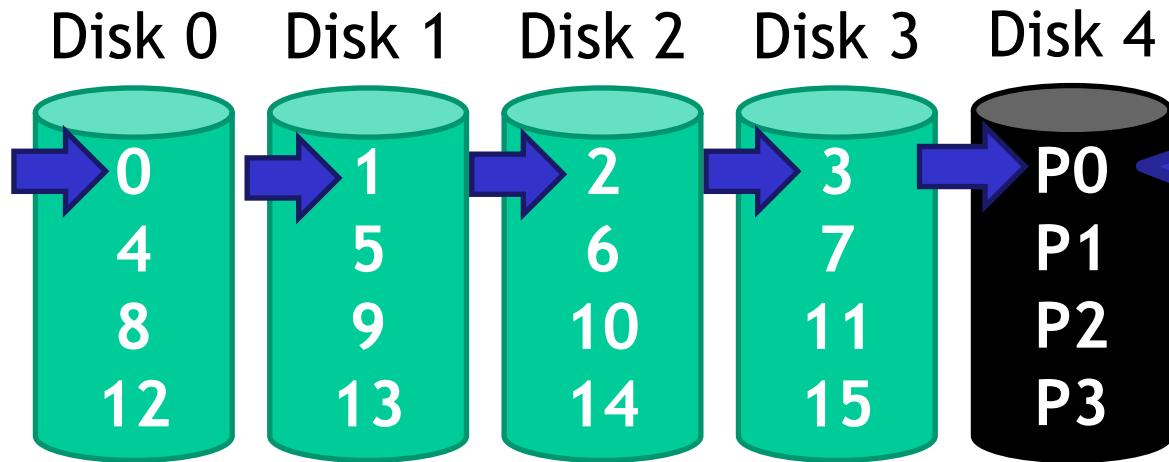
Reads and on RAID 4

- Reads (Serial or Random) are not a problem in RAID 4
- Parallelization across all non-parity blocks in the stripe





Serial Writes and RAID 4



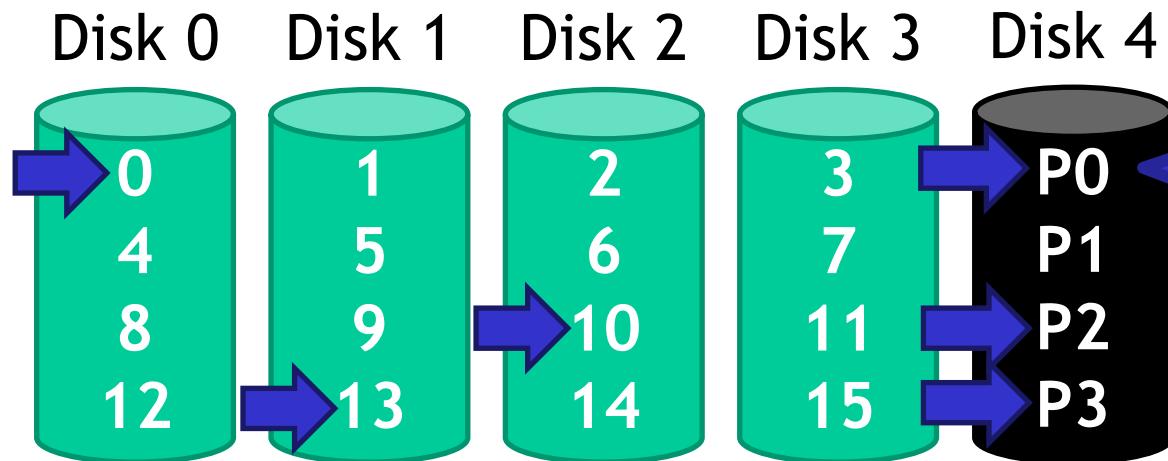
All writes on the same stripe update the parity drive once

RAID 4 has the same Read and Write write performance

- Parallelization across all non-parity blocks in the stripe



Random Writes and RAID 4



All writes must update the parity drive, causing serialization :(

Random writes in RAID 4

1. Read the target block and the parity block
2. Use subtraction to calculate the new parity block
3. Write the target block and the parity block

RAID 4 has terrible write performance

- Bottlenecked by the parity drive



Analysis of RAID 4



Capacity:

- $N - 1$
- Space on the parity drive is lost

Reliability:

- 1 drive can fail
- Massive performance degradation during partial outage

Sequential Read and write: $(N - 1) * S$

- Parallelization across all non-parity blocks in the stripe

Random Read: $(N - 1) * R$

- Reads parallelize over all but the parity drive

Random Write: $R / 2$

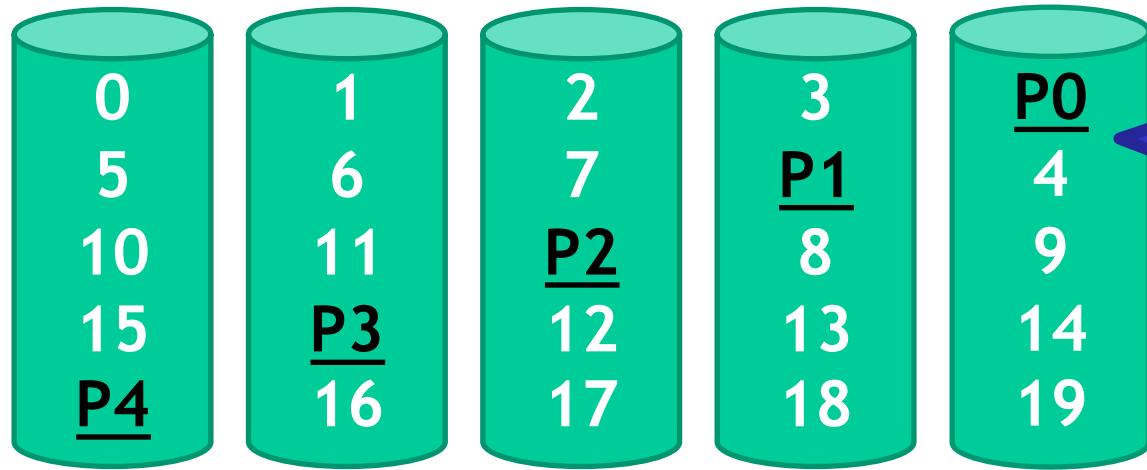
- Writes serialize due to the parity drive
- Each write requires 1 read and 1 write of the parity drive, thus $R / 2$



RAID 5: Rotating Parity



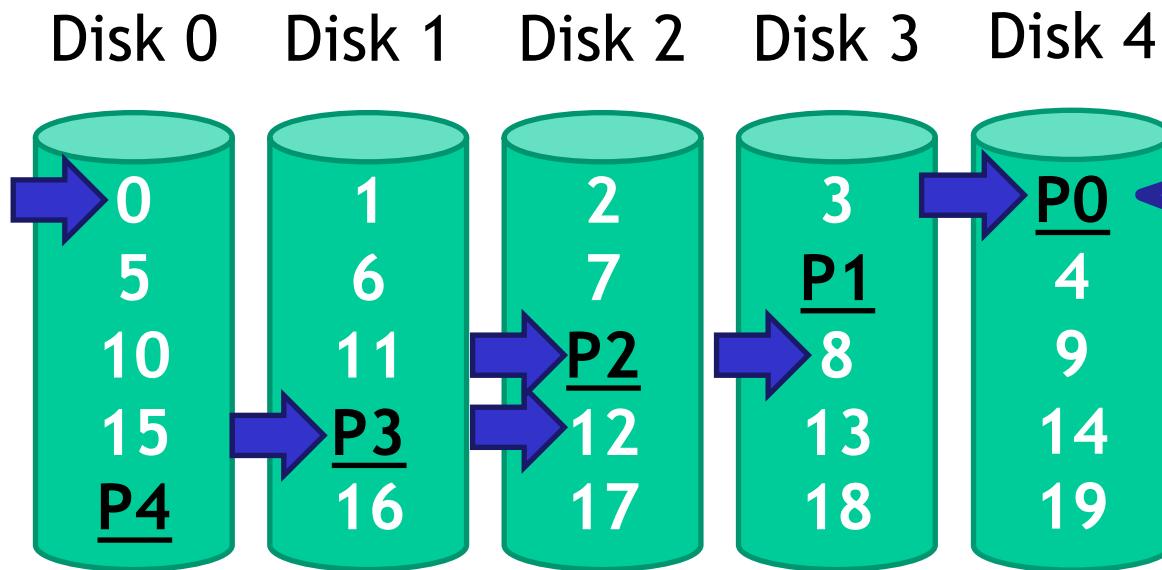
Disk 0 Disk 1 Disk 2 Disk 3 Disk 4



Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	0	1	1	$0 \wedge 0 \wedge 1 \wedge 1 = 0$
1	0	0	$0 \wedge 1 \wedge 0 \wedge 0 = 1$	0
1	1	$1 \wedge 1 \wedge 1 \wedge 1 = 0$	1	1
1	$0 \wedge 1 \wedge 1 \wedge 1 = 1$	0	1	1



Random Writes and RAID 5



Random writes in RAID 5

1. Read the target block and the parity block
2. Use subtraction to calculate the new parity block
3. Write the target block and the parity block

Thus, 4 total operations (2 reads, 2 writes)

- Distributed across all drives



Analysis of Raid 5



Capacity: [same as RAID 4]

- $N - 1$

Reliability: [same as RAID 4]

- 1 drive can fail
- Massive performance degradation during partial outage

Sequential Read and write:

- $(N - 1) * S$ [same]
- Parallelization across all non-parity blocks

Random Read:

- $N * R$ [vs. $(N - 1) * R$]
- Unlike RAID 4, reads parallelize over all drives

Random Write:

- $N / 4 * R$ [vs. $R / 2$ for RAID 4]
- Unlike RAID 4, writes parallelize over all drives
- Each write requires 2 reads and 2 write, hence $N / 4$



Comparison of RAID Levels



- N - number of drives
- S - sequential access speed
- R - random access speed
- D - latency to access a single disk

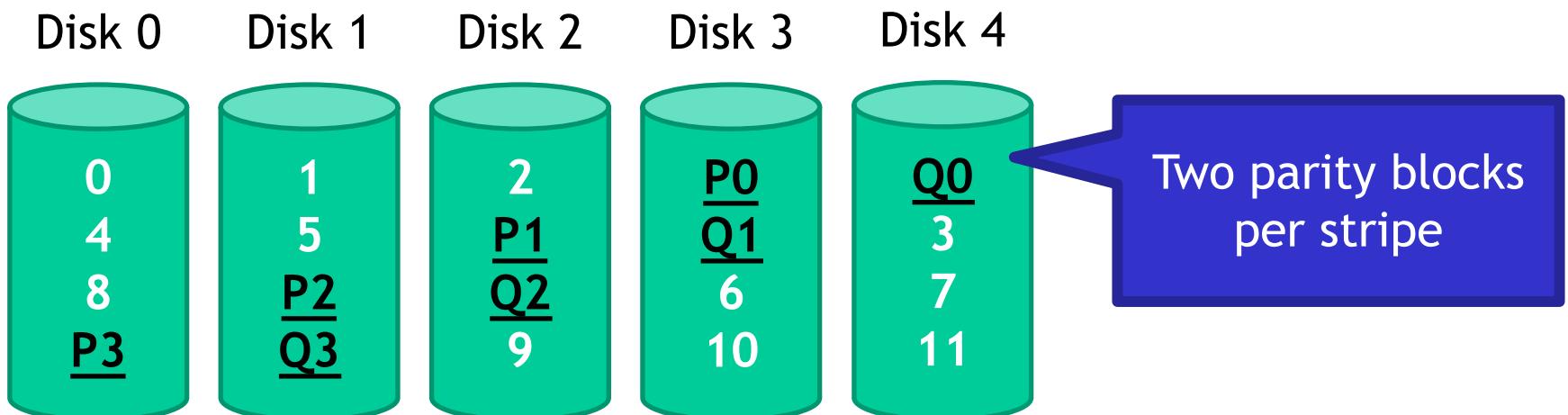
		RAID 0	RAID 1	RAID 4	RAID 5
Throughput	Capacity	N	$N / 2$	$N - 1$	$N - 1$
	Reliability	0	1 (maybe $N / 2$)	1	1
	Sequential Read	$N * S$	$(N / 2) * S$	$(N - 1) * S$	$(N - 1) * S$
	Sequential Write	$N * S$	$(N / 2) * S$	$(N - 1) * S$	$(N - 1) * S$
	Random Read	$N * R$	$N * R$	$(N - 1) * R$	$N * R$
	Random Write	$N * R$	$(N / 2) * R$	$R / 2$	$(N / 4) * R$
Latency	Read	D	D	D	D
	Write	D	D	$2 * D$	$2 * D$



RAID level 6



- More *fault tolerance* with respect RAID5
 - 2 concurrent failures are tolerated
 - Uses Solomon-Reeds codes with two redundancy schemes
 - (P+Q) distributed and independent
 - N + 2 disks required
 - High overhead for writes (computation of parities)
 - each write require 6 disk accesses due to the need to update both the P and Q parity blocks (slow writes)
 - Minimum set of 4 data disks





Characteristics of RAID levels



RAID level	Capacity	Reliability	R/W performance	Rebuild performance	Suggested applications
0	100%	N/A	Very good	Good	Non critical data
1	50%	Excellent	Very good / good	good	Critical information
5	(n-1)/n	Good	Good/ fair	Poor	Database, transaction based applications
6	(n-2)/n	Excellent	Very good/ poor	Poor	Critical information, w/minimal
1+0	50%	Excellent	Very good/ good	Good	Critical information, w/better performance

Best performance and most capacity? -> RAID 0

Greatest error recovery? -> RAID 1 (1+0 or 0+1) or RAID 6

Balance between space, performance, and recoverability? -> RAID 5



Many RAID systems include a hot spare

- An idle, unused disk installed in the system
- If a drive fails, the array is immediately rebuilt using the hot spare

RAID can be implemented in hardware or software

- Hardware is faster and more reliable...
- But, migrating a hardware RAID array to a different hardware controller almost never works
- Software arrays are simpler to migrate and cheaper, but have worse performance and weaker reliability
 - Due to the consistent update problem



POLITECNICO DI MILANO

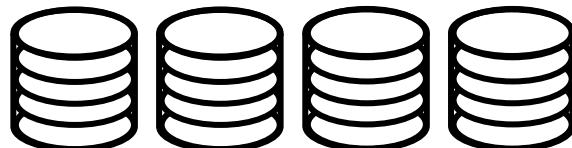


RAID Disks: Reliability Calculation



MTTF for an Array of Disks

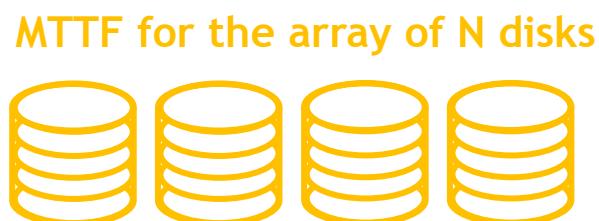
- Let's assume a constant Failure Rate, an exponentially distributed time to failure, and the case of independent failures
 - Conditions normally used to determine the disk MTTF
- $\text{MTTF}_{\text{diskArray}} = \text{MTTF}_{\text{singleDisk}} / \# \text{Disks}$
- Without any fault tolerance approach, large disk arrays are *too instable to be used*
 - Disks do not have very large MTTF since it is highly probable that will be replaced in “short-time”
- RAID0 has no redundancy
 - $\text{MTTF}_{\text{RAID0}} = \text{MTTF}_{\text{diskArray}} = \text{MTTD}_{\text{singleDisk}} / \# \text{Disks}$





Reliability calculations

- RAID levels (>0) uses redundancy to improve reliability
- When a disk fails, it should be replaced and the information reconstructed on the new disk using the redundant information
- MTTR is the time needed for this action
- N is the number of disks (#Disks) in the array
- $\text{MTTF}_{\text{RAID}} = (\text{MTTF}_{\text{singleDisk}} / N) * (1 / \text{Probability}_{\text{additionalCriticalFailuresInMTTR}})$

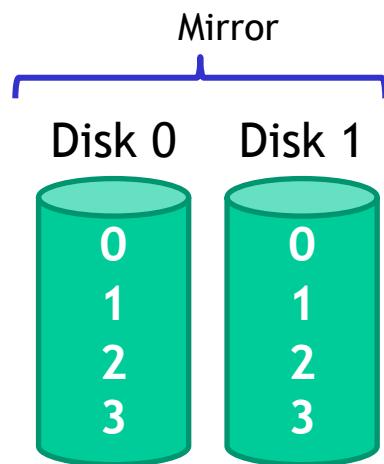


Probability of other critical failures in the array before repairing the failed disk

Determined by the RAID level and type of redundancy



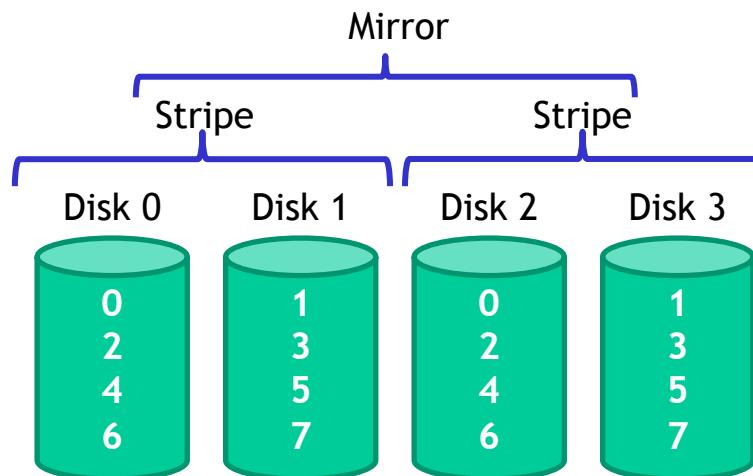
- With a single copy of each disk, 1 drive can fail
 - If you are lucky, $N / 2$ drives can fail without data loss
- $\text{MTTF}_{\text{RAID}} = (\text{MTTF}_{\text{singleDisk}}/N) * (1 / \text{Probability}_{\text{2ndCriticalFailureInMTTR}})$
- $\text{Probability}_{\text{2ndCriticalFailureInMTTR}} = (1 / \text{MTTF}_{\text{singleDisk}}) * \text{MTTR}$
 - $1 / \text{MTTF}_{\text{singleDisk}}$ is the failure rate for the copy of the failed disk
 - MTTR is the period of interest before replacement





RAID0+1 - MTTF

- When 1 disk in a stripe group fails the entire group goes off
- $\text{MTTF}_{\text{RAID}} = (\text{MTTF}_{\text{singleDisk}}/N) * (1 / \text{Probability}_{\text{2ndCriticalFailureInMTTR}})$
- $\text{Probability}_{\text{2ndCriticalFailureInMTTR}} = (G / \text{MTTF}_{\text{singleDisk}}) * \text{MTTR}$
 - G is the number of disks in a stripe group
 - $G/\text{MTTF}_{\text{singleDisk}}$ is the failure rate for one of the disks in the other group (the other one w.r.t. the failed disk)
 - MTTR is the period of interest before replacement

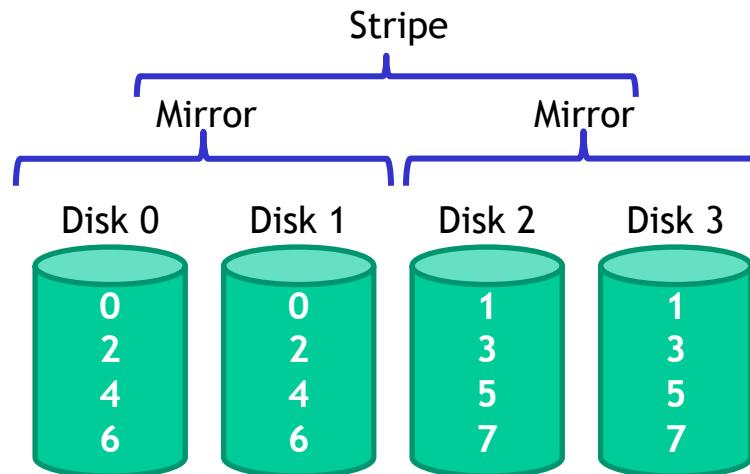




RAID1+0 - MTTF



- To fail, the same copy in both groups has to fail
 - Multiple failure can be tolerated
- $\text{MTTF}_{\text{RAID}} = (\text{MTTF}_{\text{singleDisk}}/N) * (1 / \text{Probability}_{\text{2ndCriticalFailureInMTTR}})$
- $\text{Probability}_{\text{2ndCriticalFailureInMTTR}} = (1 / \text{MTTF}_{\text{singleDisk}}) * \text{MTTR}$
 - $1 / \text{MTTF}_{\text{singleDisk}}$ is the failure rate for the copy of the failed disk
 - MTTR is the period of interest before replacement

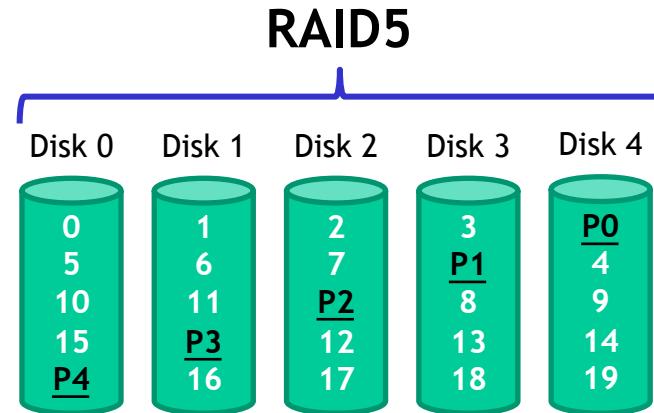
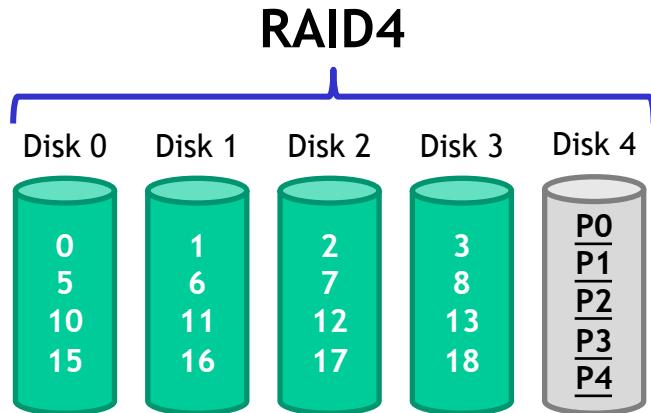




RAID4 and RAID5 - MTTF



- To fail, two disks have to fail before replacement
- $\text{MTTF}_{\text{RAID}} = (\text{MTTF}_{\text{singleDisk}}/N) * (1 / \text{Probability}_{\text{2ndFailureInMTTR}})$
- $\text{Probability}_{\text{2ndFailureInMTTR}} = ((N-1)/\text{MTTF}_{\text{singleDisk}}) * \text{MTTR}$
 - $(N-1)/\text{MTTF}_{\text{singleDisk}}$ is the failure rate for one of the other disks
 - MTTR is the period of interest before replacement

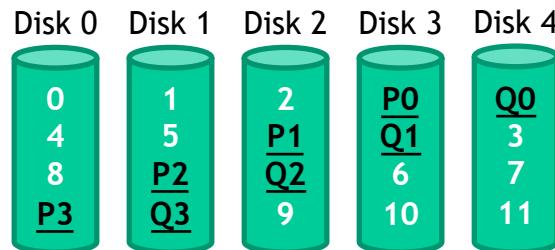




RAID6 - MTTF



- Two disk failures at the same time are tolerated
- $\text{MTTF}_{\text{RAID}} = (\text{MTTF}_{\text{singleDisk}}/N) * (1 / \text{Prob}_{\text{2ndAnd3rdFailureInMTTR}})$
- $\text{Probability}_{\text{2ndAnd3rdFailureInMTTR}} = \text{Probability}_{\text{2ndFailure}} * \text{Probability}_{\text{3rdFailure}}$
- $\text{Probability}_{\text{2ndFailure}} = ((N-1)/\text{MTTF}_{\text{singleDisk}}) * \text{MTTR}$
 - $(N-1)/\text{MTTF}_{\text{singleDisk}}$ is the failure rate for one of the other disks
 - MTTR is the period of interest before the replacement
- $\text{Probability}_{\text{3ndFailure}} = ((N-2)/\text{MTTF}_{\text{singleDisk}}) * \text{MTTR}/2$
 - $(N-2)/\text{MTTF}_{\text{singleDisk}}$ is the failure rate for one of the remaining disks
 - $\text{MTTR}/2$ is the average overlapping period between 1st and 2nd disk replacement (both disk not yet replaced)





Summary



- RAID0
 - $MTTF_{RAID0} = (MTTF_{singleDisk}/N)$
- RAID1+0
 - $MTTF_{RAID1+0} = (MTTF_{singleDisk})^2 / (N * MTTR)$
- RAID0+1
 - $MTTF_{RAID0+1} = (MTTF_{singleDisk})^2 / (N * G * MTTR)$
- RAID4
 - $MTTF_{RAID4} = (MTTF_{singleDisk})^2 / (N * N - 1 * MTTR)$
- RAID5
 - $MTTF_{RAID5} = (MTTF_{singleDisk})^2 / (N * N - 1 * MTTR)$
- RAID6
 - $MTTF_{RAID6} = 2 * (MTTF_{singleDisk})^3 / (N * N - 1 * N - 2 * MTTR^2)$