6.1800 Spring 2024

Lecture #15: Reliability

building reliable systems from unreliable components

Katrina LaCurts | lacurts@mit.edu | 6.1800 2024

you have an exam on Thursday

there are a lot of things you can use to study, all on the website

- lecture outlines, slides
- recitation notes
- practice exams

the exam is open book but not open Internet. you will turn your network devices off during the exam. download everything you might need ahead of time.

6.1800 in the news

we will spend the next few weeks improving reliability by adding redundancy to systems, but reliability is also a human problem

Some of the crucial layers of redundancies that are supposed to ensure that Boeing's planes are safe appear to be strained, the people said. The experience level of Boeing's work force has dropped since the start of the pandemic. The inspection process intended to provide a vital check on work done by its mechanics has been weakened over the years. And some suppliers have struggled to adhere to quality standards while producing parts at the pace Boeing wanted them.

we will also see that reliability often comes at the cost of performance

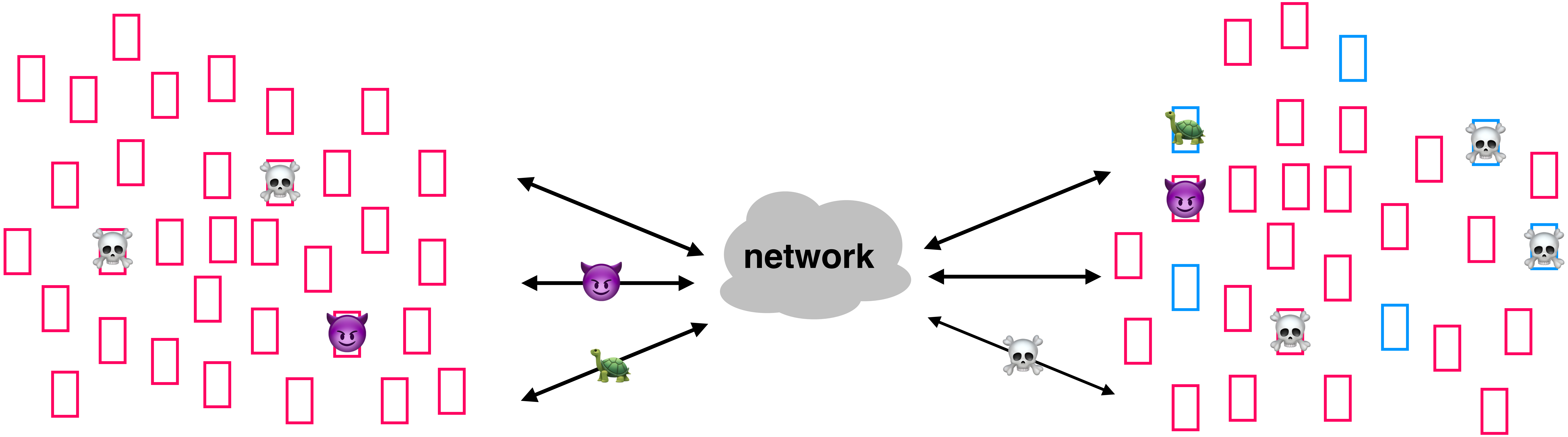
'Shortcuts Everywhere': How Boeing Favored Speed Over Quality

Problems have plagued the manufacturer even after two fatal crashes, and many current and former employees blame its focus on making planes more quickly.

https://www.nytimes.com/2024/03/28/business/boeing-quality-problems-speed.html

scalability: how does our system behave as we increase the number of machines, users, requests, data, etc.?

fault-tolerance/reliability: how does our system deal with failures (☠️)? machines crashing, network links breaking, etc.



security: how does our system cope in the face of targeted attacks (😈)?

performance: how do we define our performance requirements, and know if our system is meeting them? what do we do if performance is subpar (🐢)?

who is impacted by our design and implementation choices?
who makes those choices?

how to build reliable systems

1. **identify** **all possible faults; decide which ones we're going to handle**

2. **detect/contain** **faults**

3. **handle** **faults ("recover")**

how to measure success

there are many things we could measure, but we will typically focus on availability: what fraction of time is the system up and available to use

today we'll focus on handling disk failure

we want to make the disk as reliable as possible so that we don't lose data; we especially don't want to lose data when the machine fails (which is what will start happening in the next lecture)

2.12.1 Annualized Failure Rate (AFR) and Mean Time Between Failures (MTBF)

The product shall achieve an Annualized Failure Rate - AFR - of 0.73% (Mean Time Between Failures - MTBF - of 1.2 Million hrs) when operated in an environment that ensures the HDA case temperatures do not exceed 40°C. Operation at case temperatures outside the specifications in Section 2.9 may increase the product Annualized Failure Rate (decrease MTBF). AFR and MTBF are population statistics that are not relevant to individual units.

AFR and MTBF specifications are based on the following assumptions for business critical storage system environments:

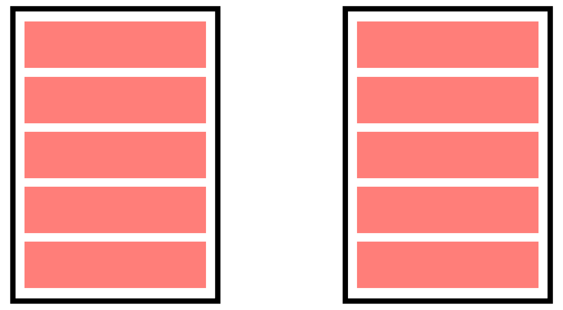
- 8,760 power-on-hours per year.
- 250 average motor start/stop cycles per year.
- Operations at nominal voltages.
- Systems will provide adequate cooling to ensure the case temperatures do not exceed 40°C. Temperatures outside the specifications in Section 2.9 will increase the product AFR and decrease MTBF.

Nonrecoverable read errors	1 per 10^{15} bits read, max
Annualized Failure Rate (AFR)	0.73% (nominal power, 40°C case temperature)

https://www.seagate.com/staticfiles/support/disc/manuals/NL35%20Series%20&%20BC%20ES%20Series/Barracuda%20ES.2%20Series/100468393f.pdf

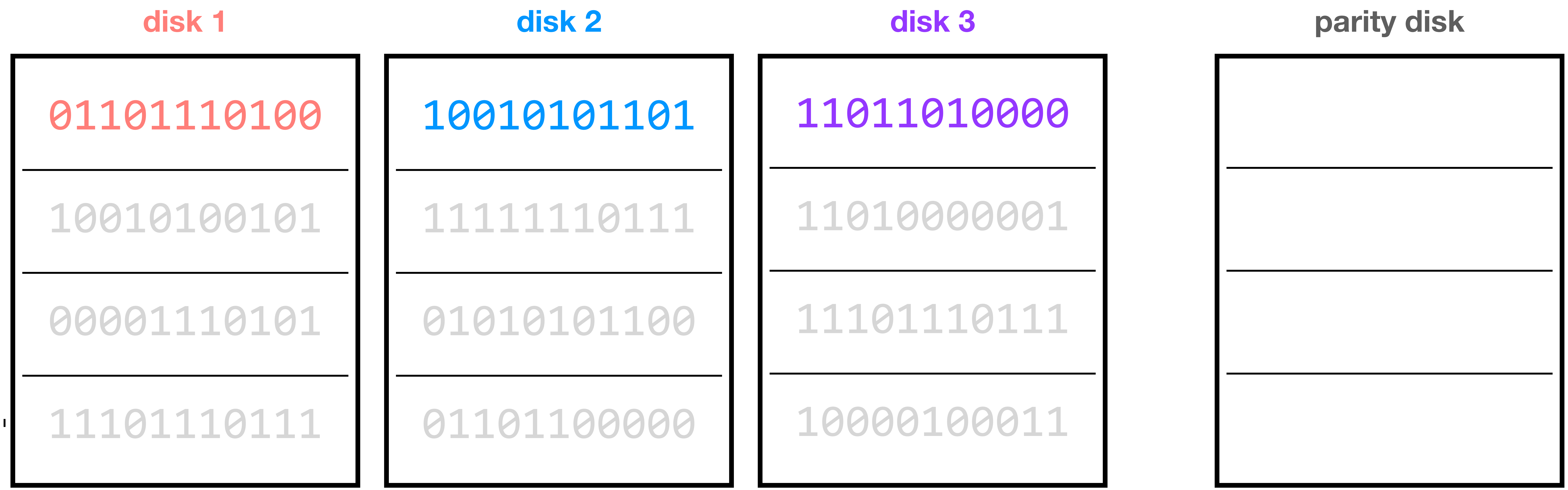
RAID 1

mirroring

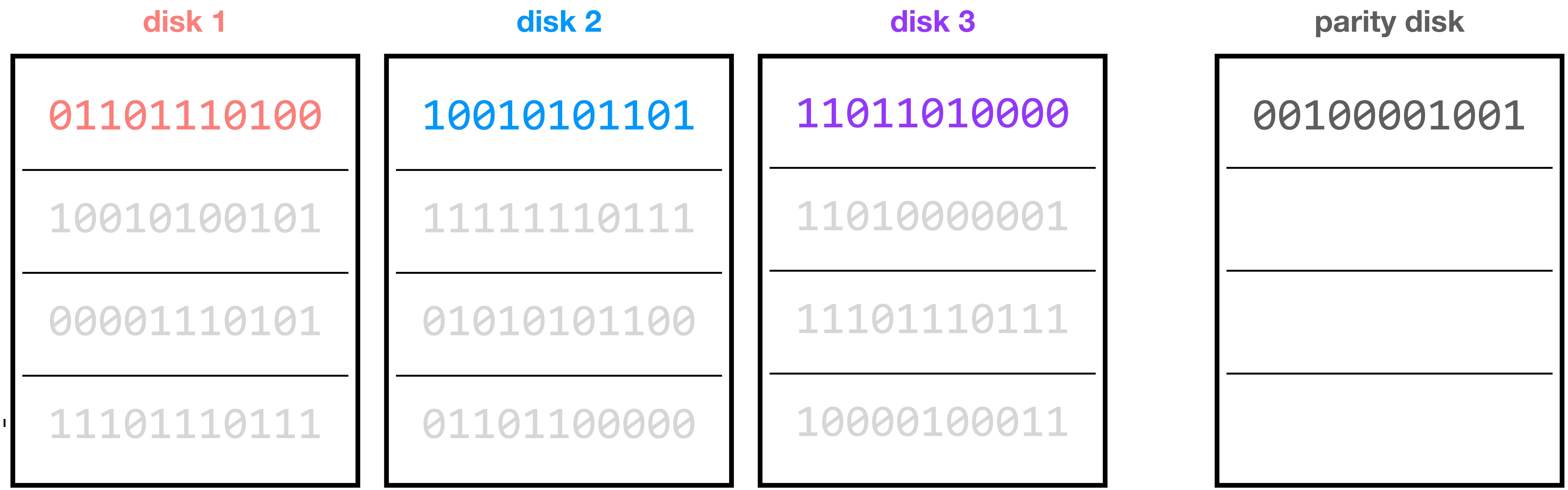


+ can recover from single-disk failure
- requires 2N disks

disk 1	disk 2	disk 3	parity disk
01101110100	10010101101	11011010000	
10010100101	11111110111	11010000001	
00001110101	01010101100	11101110111	
11101110111	01101100000	10000100011	



xor sector 1 from each disk to get sector 1 on the parity disk



xor sector 1 from each disk to get sector 1 on the parity disk

disk 1	disk 2	disk 3	parity disk
01101110100	10010101101	11011010000	00100001001
10010100101	11111110111	11010000001	10111010011
00001110101	01010101100	11101110111	10110101110
11101110111	01101100000	10000100011	00000110100

xor sector *i* from each disk to get sector *i* on the parity disk

Katrina LaCurts | lacurts@mit.edu | 6.1800 2024

suppose disk 2 fails, and after recovery, all data has been lost

disk 1	disk 2	disk 3	parity disk
01101110100		11011010000	00100001001
10010100101		11010000001	10111010011
00001110101		11101110111	10110101110
11101110111		10000100011	00000110100

suppose disk 2 fails, and after recovery, all data has been lost

disk 1	disk 2	disk 3	parity disk
01101110100		11011010000	00100001001
10010100101		11010000001	10111010011
00001110101		11101110111	10110101110
11101110111		10000100011	00000110100

xor sector 1 from the remaining disks to recover sector 1 on the failed disk

suppose disk 2 fails, and after recovery, all data has been lost

disk 1	disk 2	disk 3	parity disk
01101110100	10010101101	11011010000	00100001001
10010100101		11010000001	10111010011
00001110101		11101110111	10110101110
11101110111		10000100011	00000110100

xor sector 1 from the remaining disks to recover sector 1 on the failed disk

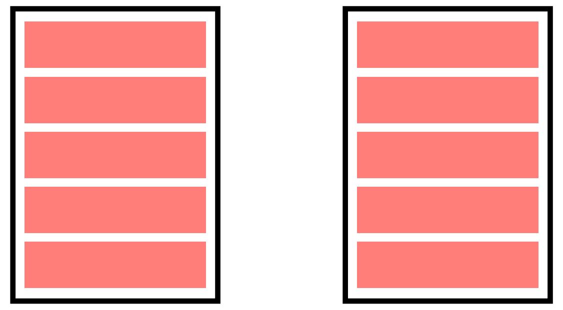
suppose disk 2 fails, and after recovery, all data has been lost

disk 1	disk 2	disk 3	parity disk
01101110100	10010101101	11011010000	00100001001
10010100101	11111110111	11010000001	10111010011
00001110101	01010101100	11101110111	10110101110
11101110111	01101100000	10000100011	00000110100

xor sector *i* from the remaining disks to recover sector *i* on the failed disk

RAID 1

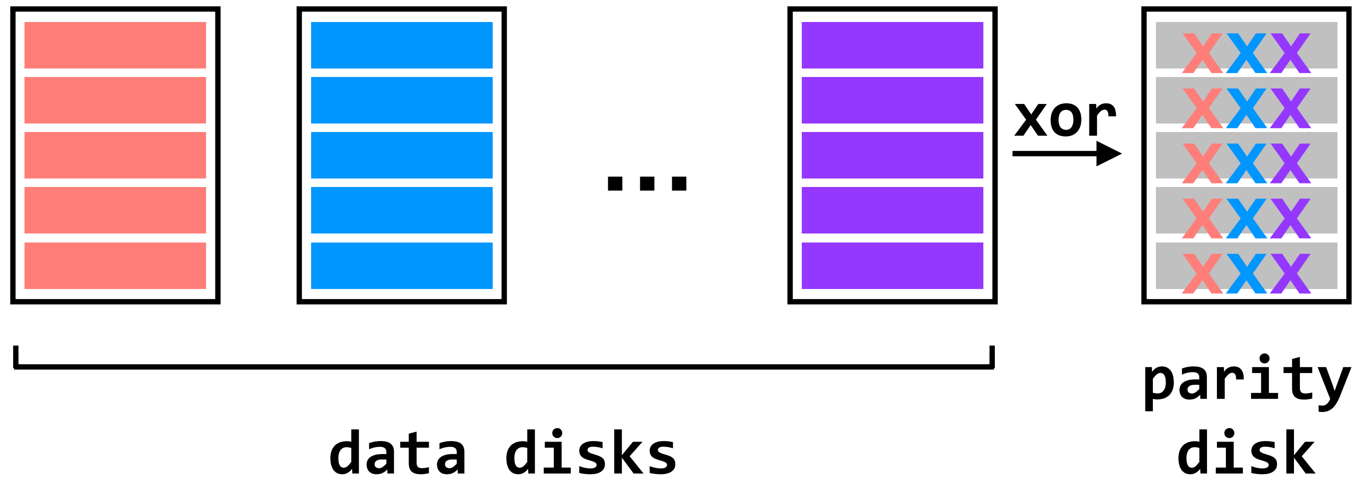
mirroring



+ can recover from single-disk failure
- requires 2N disks

RAID 4

dedicated parity disk



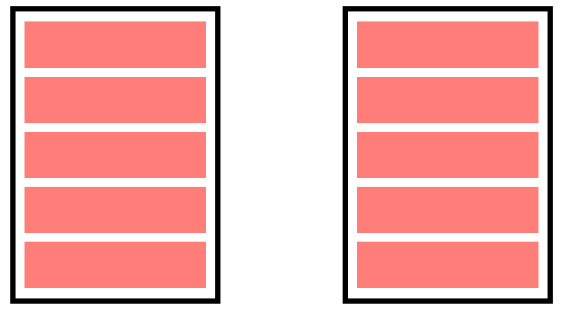
sector i of the parity disk is the xor of sector i from all data disks

+ can recover from single-disk failure
+ requires N+1 disks
+ performance benefits if you stripe a single file across multiple data disks
- all writes go to the parity disk

01101110100	10010101101	11011010000	00100001001
10010100101	11111110111	11010000001	10111010011
00001110101	01010101100	11101110111	10110101110
11101110111	01101100000	10000100011	00000110100

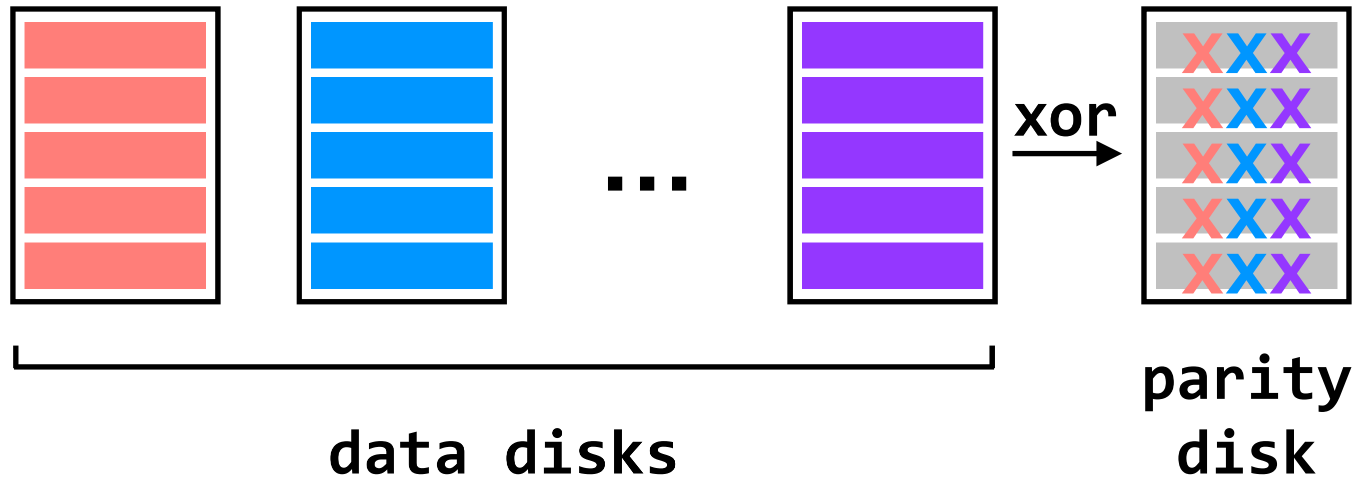
00100001001	01101110100	10010101101	11011010000
10010100101	10111010011	11111110111	11010000001
00001110101	01010101100	10110101110	11101110111
11101110111	01101100000	10000100011	00000110100

RAID 1
mirroring



+ can recover from single-disk failure
- requires 2N disks

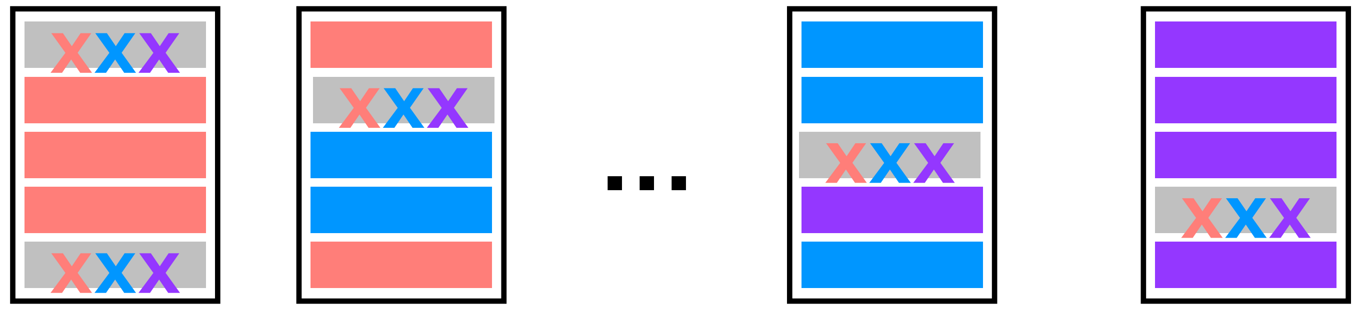
RAID 4
dedicated parity disk



sector i of the parity disk is the xor of sector i from all data disks

+ can recover from single-disk failure
+ requires N+1 disks
+ performance benefits if you stripe a single file across multiple data disks
- all writes go to the parity disk

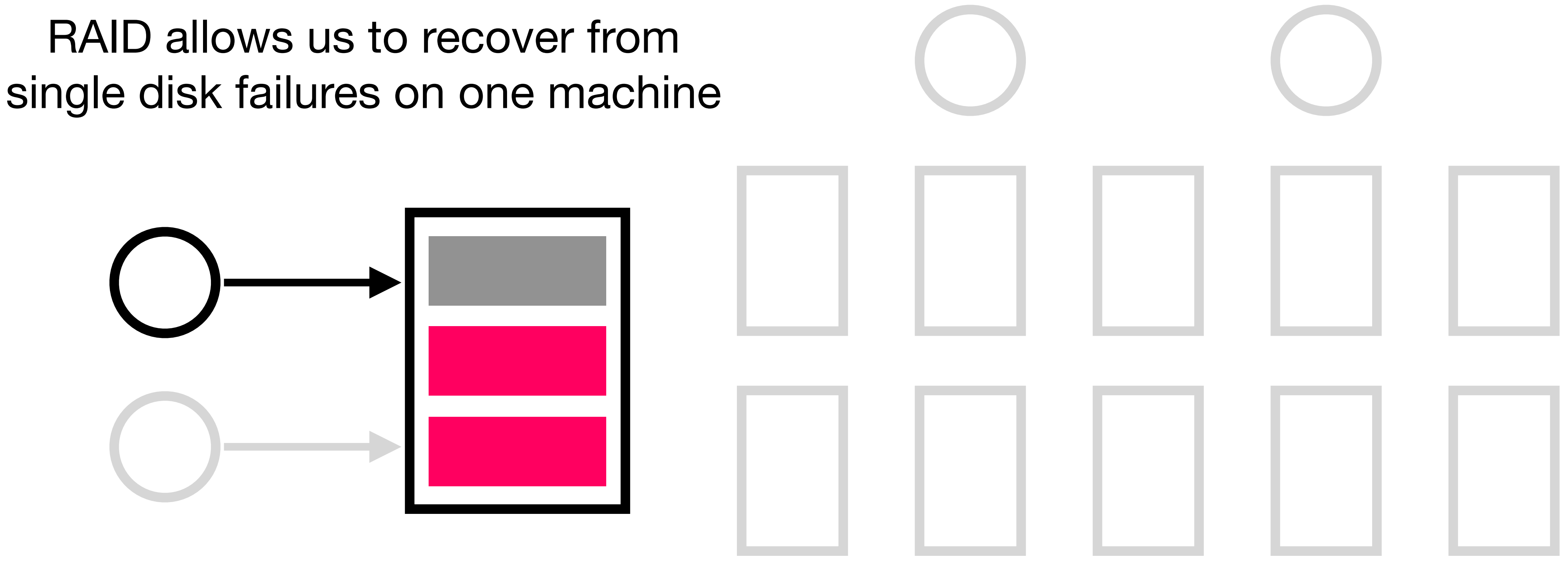
RAID 5
spread out the parity



+ can recover from single-disk failure
+ requires N+1 disks
+ performance benefits if you stripe a single file across multiple data disks
+ writes are spread across disks

Katrina LaCurts | lacurts@mit.edu | 6.1800 2024

our goal is to build **reliable systems from unreliable components**. we want to build systems that serve many clients, store a lot of data, perform well, all while keeping availability high



the high-level process of dealing with failures is to identify the faults, detect/contain the faults, and handle the faults. in lecture, we will build a set of abstractions to make that process more manageable

this slide — which we'll start and end on in each lecture — will get more involved as that goes along systems have faults. we have to take them into account and build reliable, **fault-tolerant systems**. reliability comes at a cost — there are tradeoffs between reliability and simplicity, reliability and performance, etc.

our main tool for improving reliability is **redundancy**. one form of redundancy is **replication**, which can be used to combat many things including disk failures (important, because disk failures mean lost data).

RAID replicates data across disks on a single machine in a smart way. RAID 5 protects against single-disk failures while maintaining good performance

one can extend the ideas in RAID to protect against multiple disk failures. RAID 6 does this, for example