RAID: (Redundant?) Arrays of Inexpensive Disks

CSCI 333

Why RAID?

1. The ideas used by RAID are “big ideas” in systems

- Striping
- Replication
- Parity

2. Fault tolerance is important in practice

- We must first define a model for how things can fail
- Then we can design systems to overcome those failures

How to Read

Do not spend your time memorizing RAID levels

- Instead, think about the “big ideas” and their tradeoffs
  - When would you stripe writes? When is striping not worth the work?
  - Should you use replication or parity? How many replicates do you need?

Think about each idea in terms of:

- Performance
- Capacity
- Fault tolerance

Think about how to apply the ideas elsewhere:

- Modern systems comprise many abstract layers. Where can you apply these ideas, and where does abstraction get in the way?

Other Related Topics

Crash recovery/consistency

Erasure coding (e.g., Reed-Solomon codes)

“Byzantine” fault tolerance

Deduplication (perhaps the inverse of replication…)

RAID: (Redundant?) Arrays of Inexpensive Disks

CSCI 333

Lecture Overview

Redundant Arrays of Inexpensive Disks

- Three “techniques”
  - Striping
  - Mirroring
  - Parity

- Three evaluation criteria
  - Performance
  - Reliability
  - Capacity

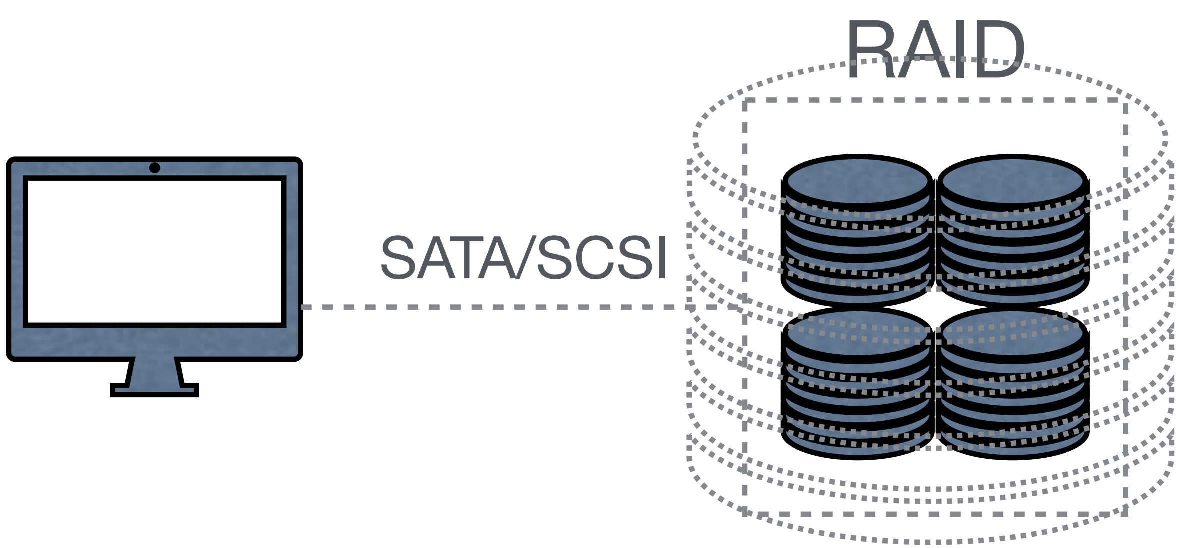
- Failure Model
  - Fail-stop

- RAID Levels

RAID from the user's view

Hardware RAID is *transparent* to the user

- An array of disks are connected to the computer, and all the computer sees is a single logical disk
- Nothing about the RAID setup is externally visible: it's just a single LBA space that appears and acts as one device



Why?

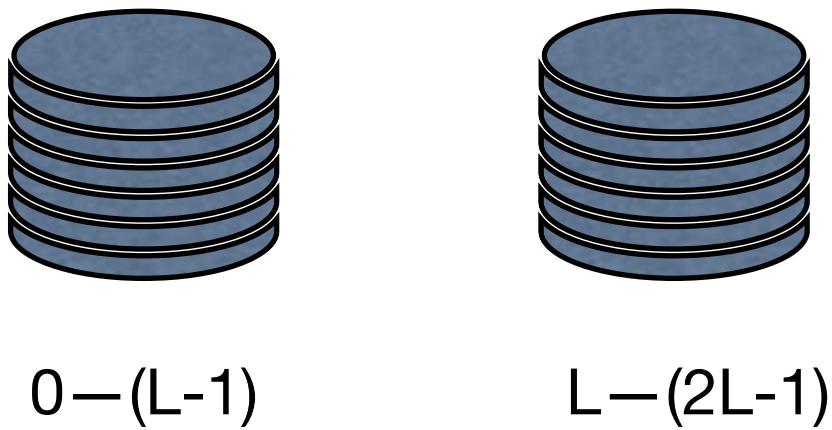
Why masquerade a set of N>1 disks as a single volume of storage? What types of things might we want to improve?

- Capacity
  - we may just want to store more data than fits on a single disk, but not change our software to manage multiple physical devices
- Performance (parallelism and/or choice)
  - a single disk has one disk arm, so it can read from one location at a time. N disks have N disk arms. We can parallelize some operations
- Recovery
  - if all of our data is on a single disk, we are extremely vulnerable to any disk failures
  - if our data is on N disks, we may not lose everything if we lose 1 disk

Capacity

Suppose we have an array of 2 disks, each capable of storing L logical blocks.

- Let's partition the LBAs as follows:

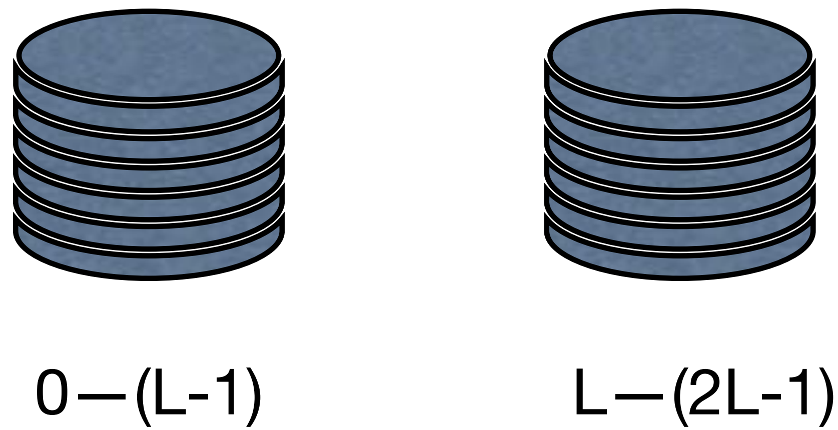


- What is the capacity?
- What is the performance?
- How many disk failures can we survive?

Capacity

Suppose we have an array of 2 disks, each capable of storing L logical blocks.

- Let's partition the LBAs as follows:

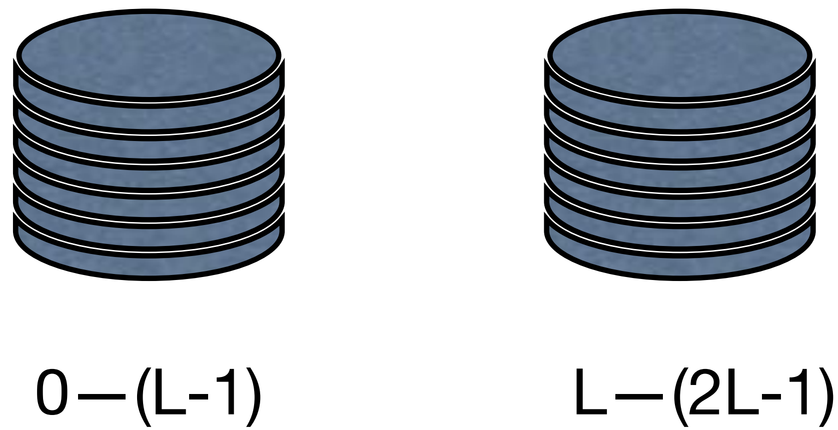


- What is the capacity?
  - Capacity is equal to the sum of the capacity of both disks
    - No capacity is sacrificed

Capacity

Suppose we have an array of 2 disks, each capable of storing L logical blocks.

- Let's partition the LBAs as follows:

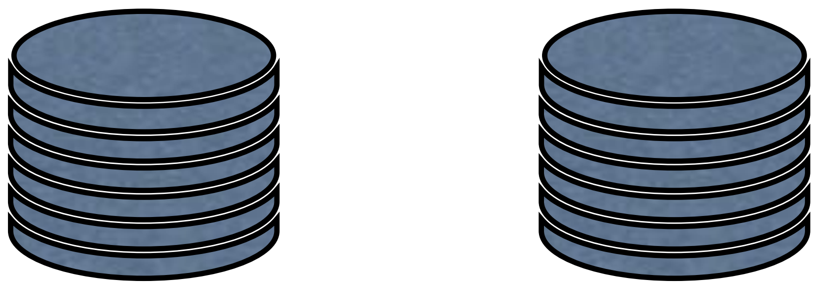


- What is the performance?
  - Sequential read/write: No gain.
    - Sequential writes go to one disk or the other (in expectation)
  - Random read/write: Potential for parallelism.
    - Can utilize both disks simultaneously (in expectation).

Capacity

Suppose we have an array of 2 disks, each capable of storing L logical blocks.

- Let's partition the LBAs as follows:



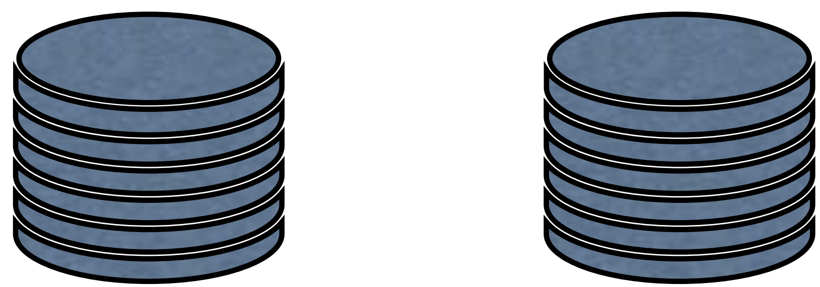
0—(L-1) L—(2L-1)

- How many disk failures can we survive?
  - 0 failures: If any disk fails, we lose all data on that disk

Striping

Suppose we have an array of 2 disks, each capable of storing L logical blocks.

- Let's partition the LBAs as follows:



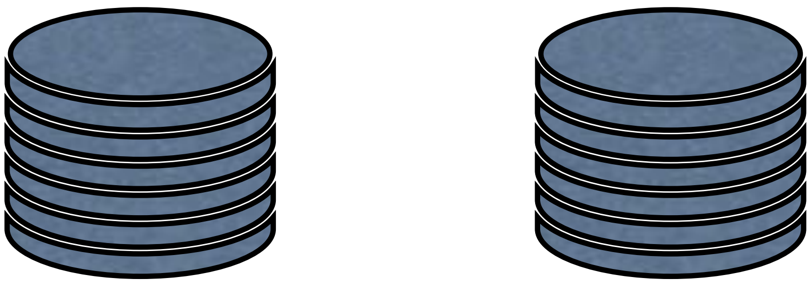
0,2,4,…,(2L-2) 1,3,5,…,(2L-1)

The difference in this scenario is that we assign LBAs to disks in a way that **stripes** our data across the disks (we can extend this idea to N>2 disks).

Striping

Suppose we have an array of 2 disks, each capable of storing L logical blocks.

- Let's partition the LBAs as follows:



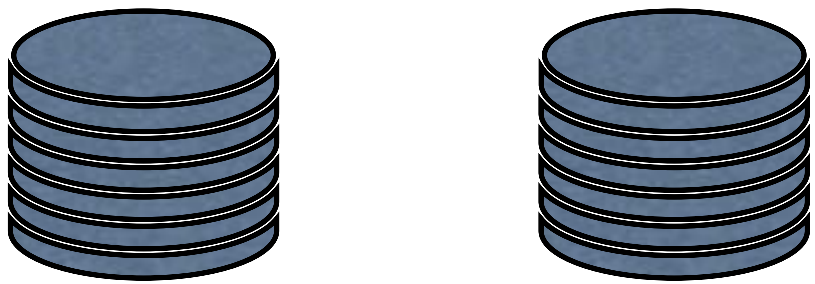
0,2,4,…,(2L-2)　　　1,3,5,…,(2L-1)

- What is the capacity?
  - Same as before. The sum of the capacity of both disks

Striping

Suppose we have an array of 2 disks, each capable of storing L logical blocks.

- Let's partition the LBAs as follows:



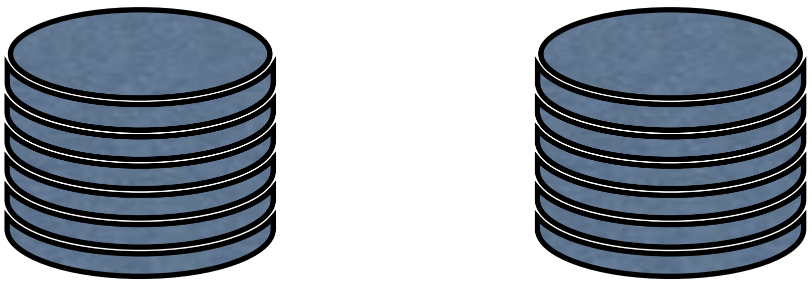
0,2,4,…,(2L-2)　　　1,3,5,…,(2L-1)

- What is the performance?
  - Sequential reads/writes: benefit from parallelism
    - We can utilize both disks at once
  - Random reads/writes: benefit from parallelism
    - We can utilize both disks at once

Striping

Suppose we have an array of 2 disks, each capable of storing L logical blocks.

- Let's partition the LBAs as follows:

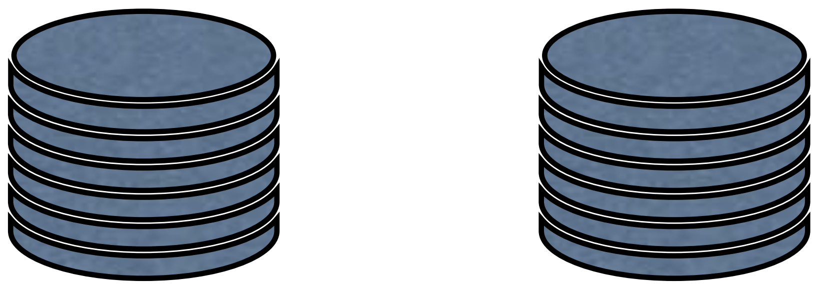


- How many disk failures can we survive?
  - 0 failures: If any disk fails, we lose all data on that disk

Striping

Suppose we have an array of 2 disks, each capable of storing L logical blocks.

- Let's partition the LBAs as follows:



0,2,4,…,(2L-2)　　　1,3,5,…,(2L-1)

Striping adds parallelism to *sequential writes*

Aside: Chunks

How to best "stripe" the data?

- Previous slide has chunk size of 1
- What are the tradeoffs of increasing the chunk size (the number of consecutive LBAs per disk in a stripe)?

Chunk size affects parallelism:

- With a small chunk size, it is more likely that a write will be striped across many disks, increasing parallelism
- With a large chunk size, some writes may directed to fewer disks
  - The system can still get parallelism from making multiple independent requests

Reliability

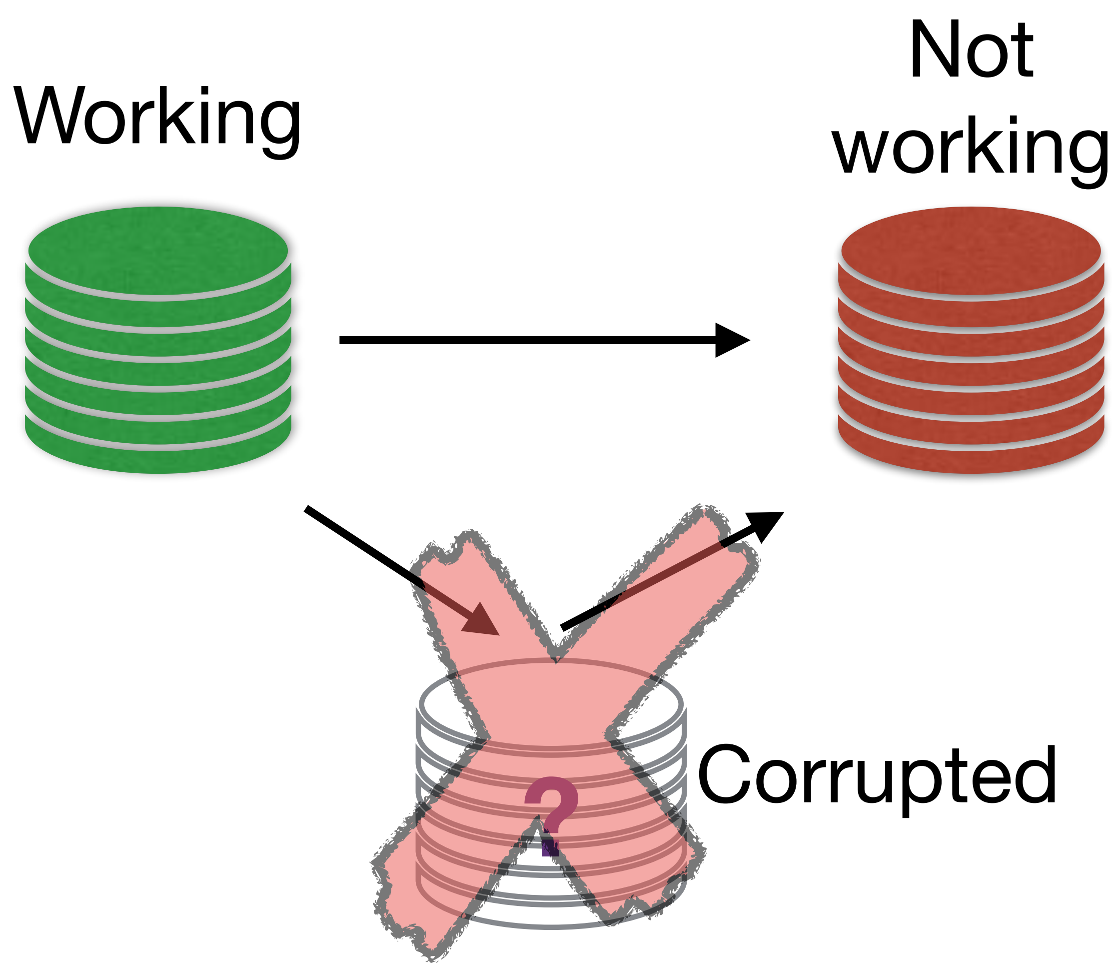
In addition to performance, we may use extra disks to increase the reliability of our storage

- Disks fail for a variety of reasons
- We want to be able to undergo one (or more) disk failures *without losing data*
- If possible, we also want to preserve/improve performance

How Can Disks Fail?

RAID assumes disks are fail-stop

- If there is an error, we can detect the error immediately
- Assume a simple state machine: either the entire disk works, or the entire disk has failed



Reality

What other classes of errors could possibly exist?

- Failures can be transient
  - e.g., a temporary error that fixes itself
- Failures can be unreliable
  - e.g., sometimes an error is returned, sometimes the correct answer
- Failures can be partial
  - e.g., a single sector or range of sectors become unusable

Disk losses may be correlated

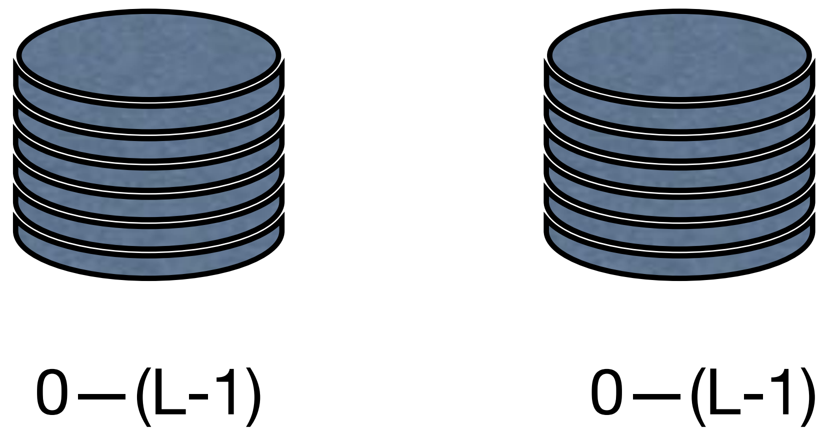
- If your power supply goes, it may take all disks with it
- Flood/fire?
- Theft?

RAID doesn't attempt to handle these errors

Redundancy: Mirroring

Suppose we have an array of 2 disks, each capable of storing L logical blocks.

- Let's partition the LBAs as follows:

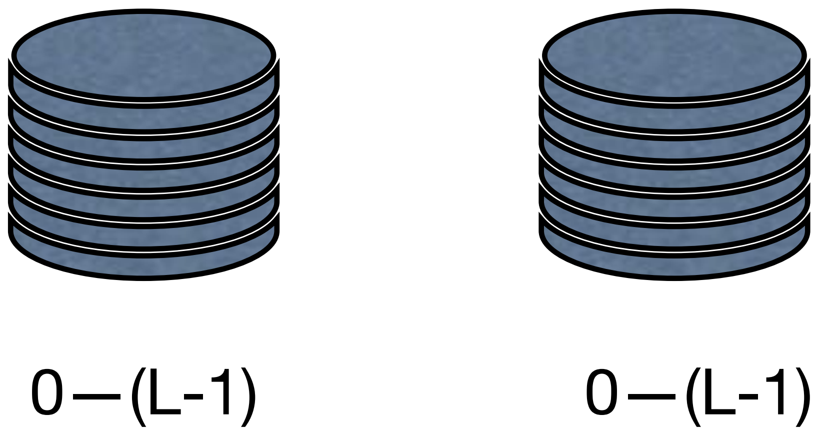


- What is the capacity?
- What is the performance?
- How many disk errors can we survive?

Redundancy: Mirroring

Suppose we have an array of 2 disks, each capable of storing L logical blocks.

- Let's partition the LBAs as follows:

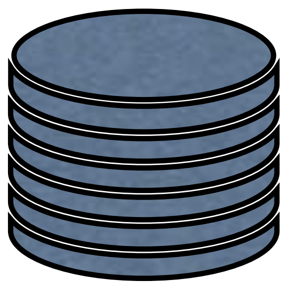


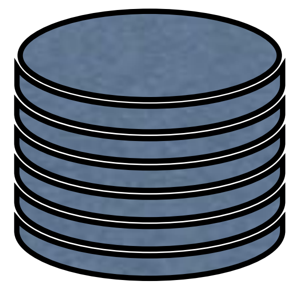
- What is the capacity?
  - In this scenario, we lose half of the capacity of our total disk space

Redundancy: Mirroring

Suppose we have an array of 2 disks, each capable of storing L logical blocks.

- Let's partition the LBAs as follows:





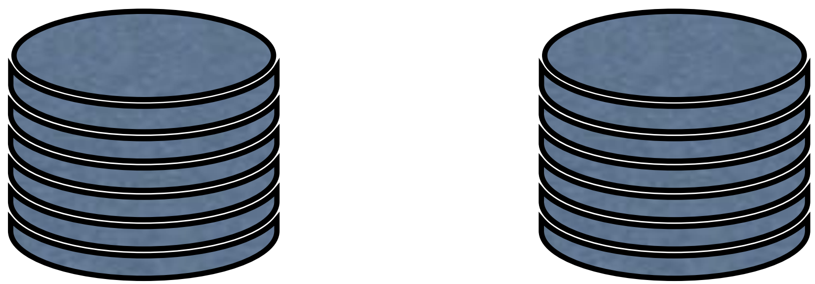
0—(L-1) 0—(L-1)

- What is the performance?
  - Writes: slightly worse
    - We must write to both mirrors, so we get the performance of the slowest disk
  - Reads: potential improvements
    - We can read from either disk, so we get the performance of the fastest disk
    - We can distribute our random reads to either disk, increasing parallelism

Redundancy: Mirroring

Suppose we have an array of 2 disks, each capable of storing L logical blocks.

- Let's partition the LBAs as follows:



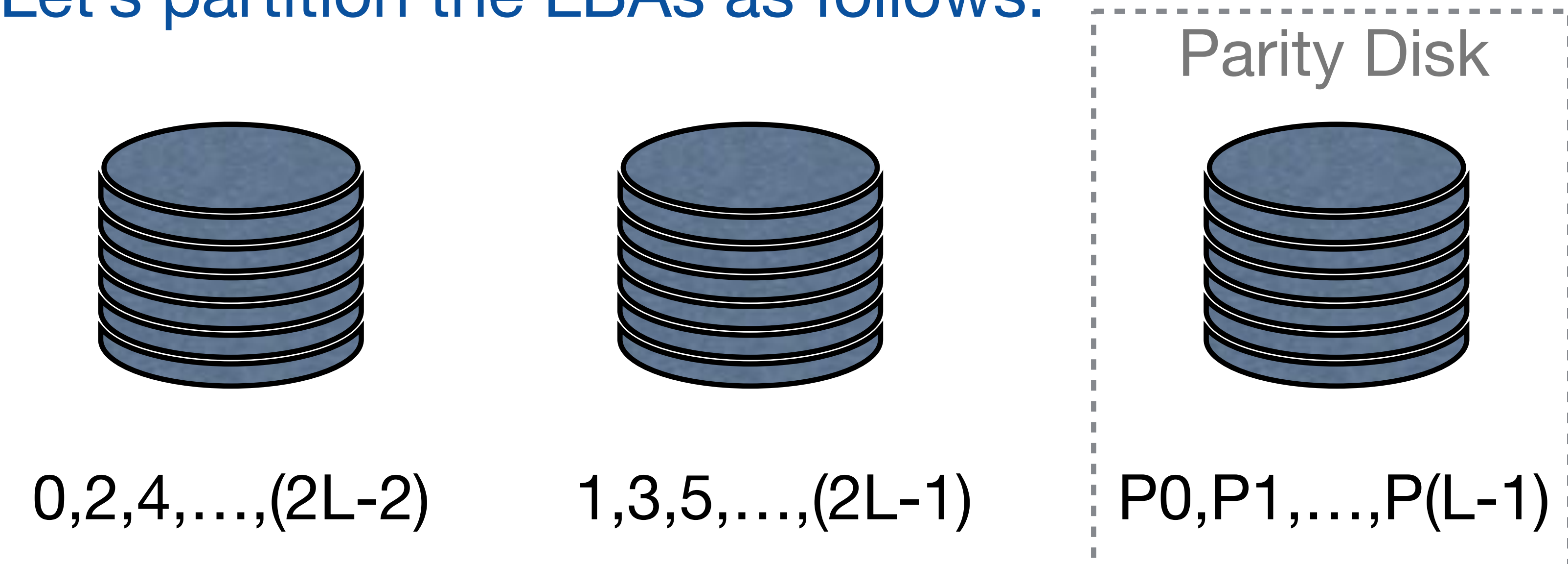
0—(L-1) 0—(L-1)

- How many disk errors can we survive?
  - If we lose one disk, we still have all of our data.
  - The number of failures we can tolerate is equal to the number of replicas.

Redundancy: Parity

Suppose we have an array of 3 disks, each capable of storing L logical blocks.

- Let's partition the LBAs as follows:



- The parity disk stores redundant information that is calculated based on the contents of our other disks
  - This parity information lets us detect certain types of errors, and possibly recover from them

Aside: Bitwise XOR

Rule: Count the number of 1s, and

- If the number of 1s is odd, the parity bit is 1
- If the number of 1s is even, the parity bit is 0

To extend the idea to disk blocks:

- bitwise XOR the i^{th} bit of each block; result is the i^{th} bit of the parity block

Example:

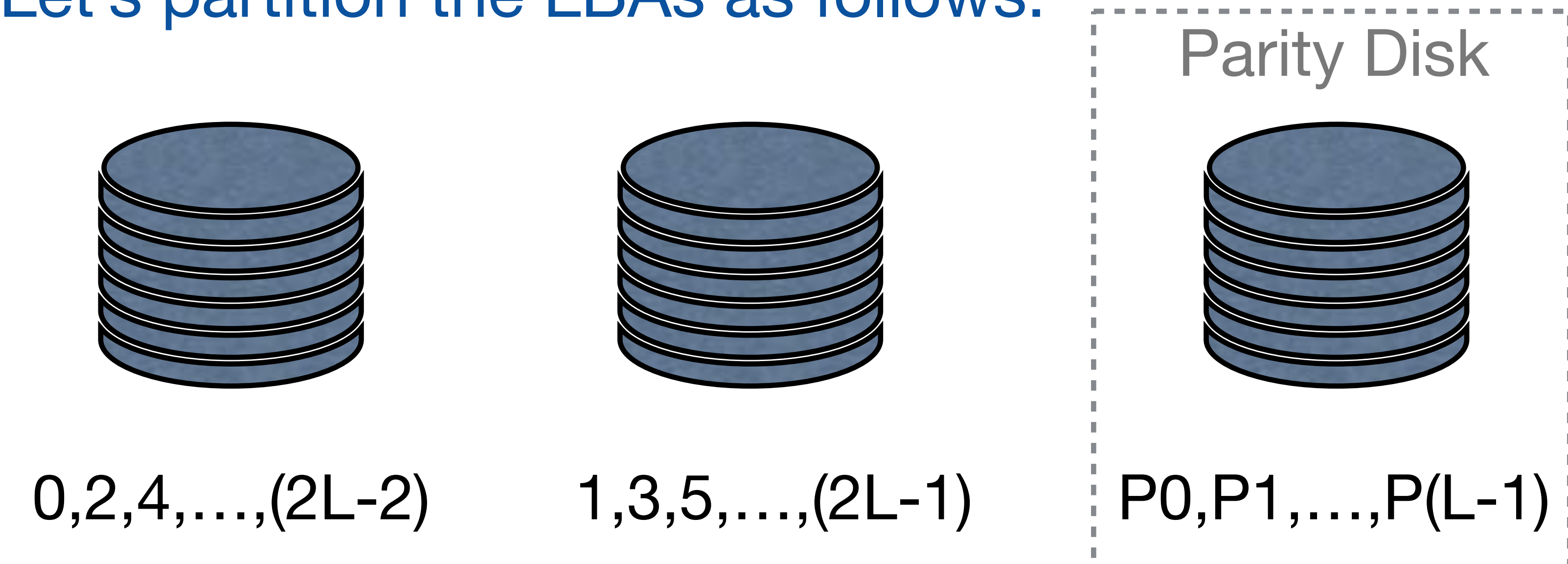
2 Bits:
XOR(1,1) = 0
XOR(0,1) = 1
XOR(1,0) = 1
XOR(0,0) = 0

3 Bits:
XOR(1,1,1) = 1
XOR(1,1,0) = 0
XOR(1,0,1) = 0
XOR(0,1,1) = 0
XOR(1,0,0) = 1
XOR(0,0,1) = 1
XOR(0,1,0) = 1
XOR(0,0,0) = 0

Redundancy: Parity

Suppose we have an array of 3 disks, each capable of storing L logical blocks.

- Let's partition the LBAs as follows:

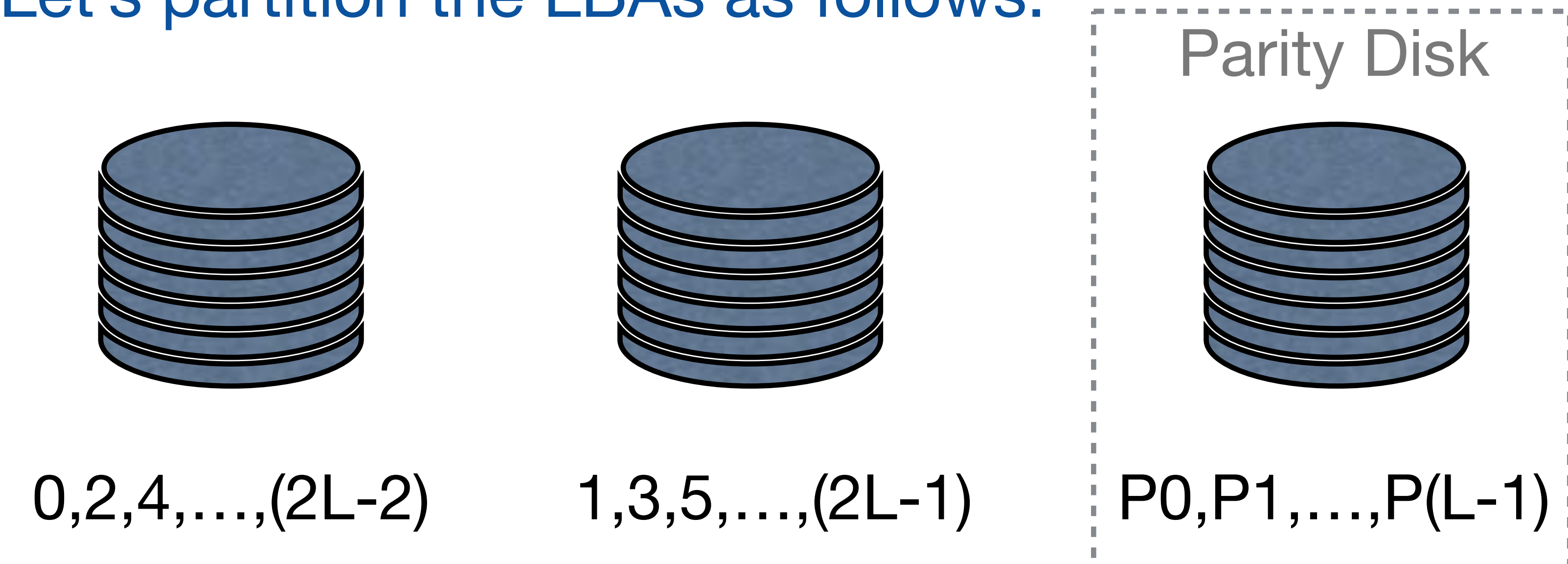


- What is the capacity?
- What is the performance?
- How many disk errors can we survive?

Redundancy: Parity

Suppose we have an array of 3 disks, each capable of storing L logical blocks.

- Let's partition the LBAs as follows:

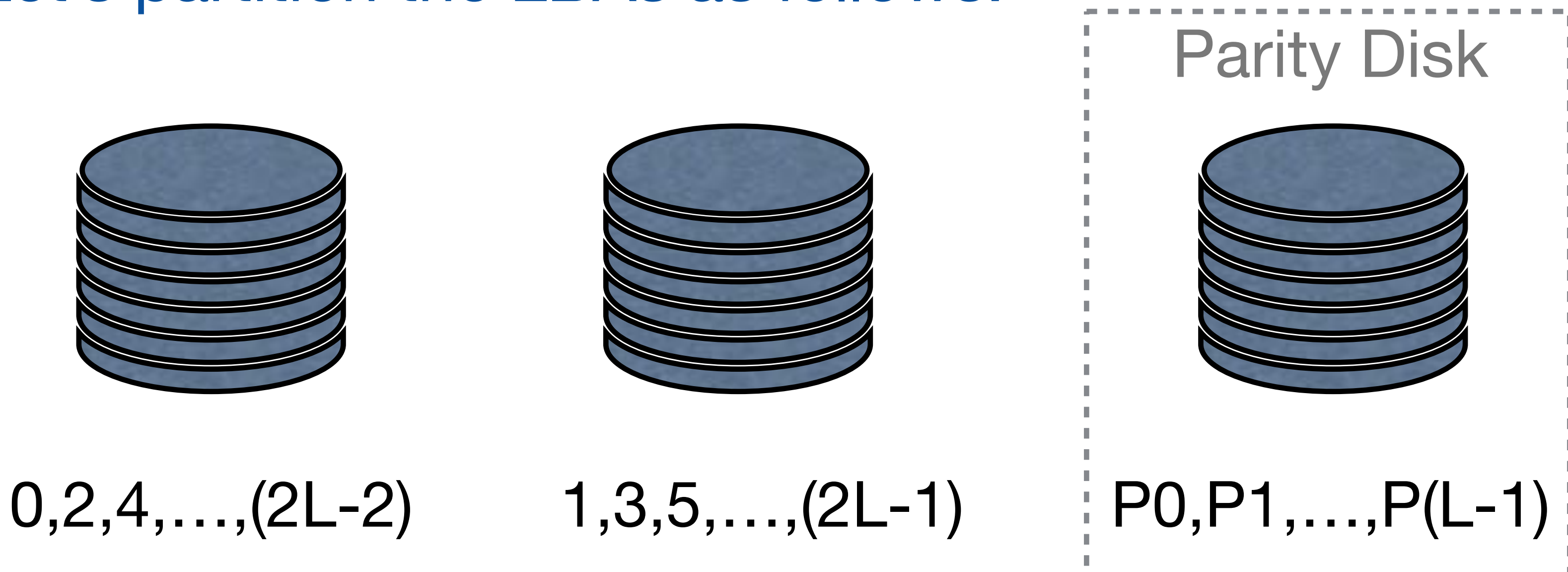


- What is the capacity?
  - We lose one disk to our parity info
    - This is better than mirroring, where we lose half of our capacity

Redundancy: Parity

Suppose we have an array of 3 disks, each capable of storing L logical blocks.

- Let's partition the LBAs as follows:

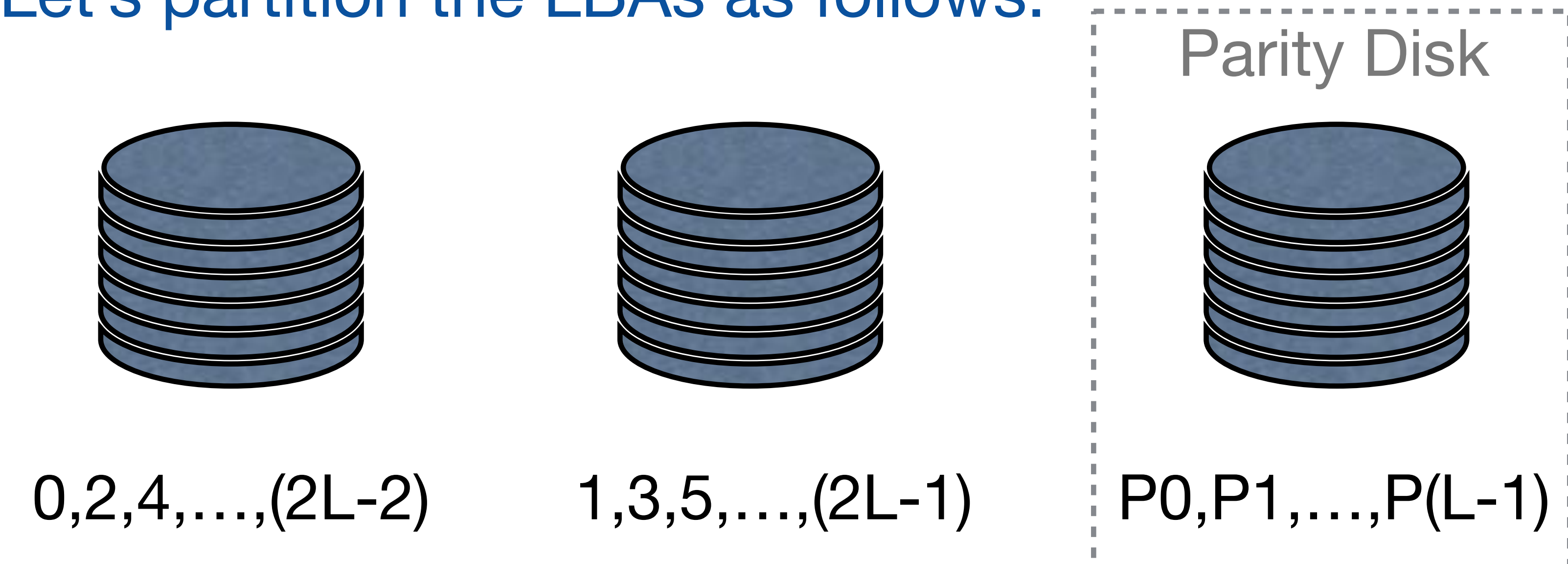


- What is the performance?
  - Sequential read: we can read from our non-parity disks in parallel (striping)
  - Sequential write: we can write a single "stripe" in parallel with its parity
  - Random read: we can read from our non-parity disks in parallel
  - Random write: parity disk becomes a bottleneck
    - Any update to either disk requires an update to the parity disk

Redundancy: Parity

Suppose we have an array of 3 disks, each capable of storing L logical blocks.

- Let's partition the LBAs as follows:

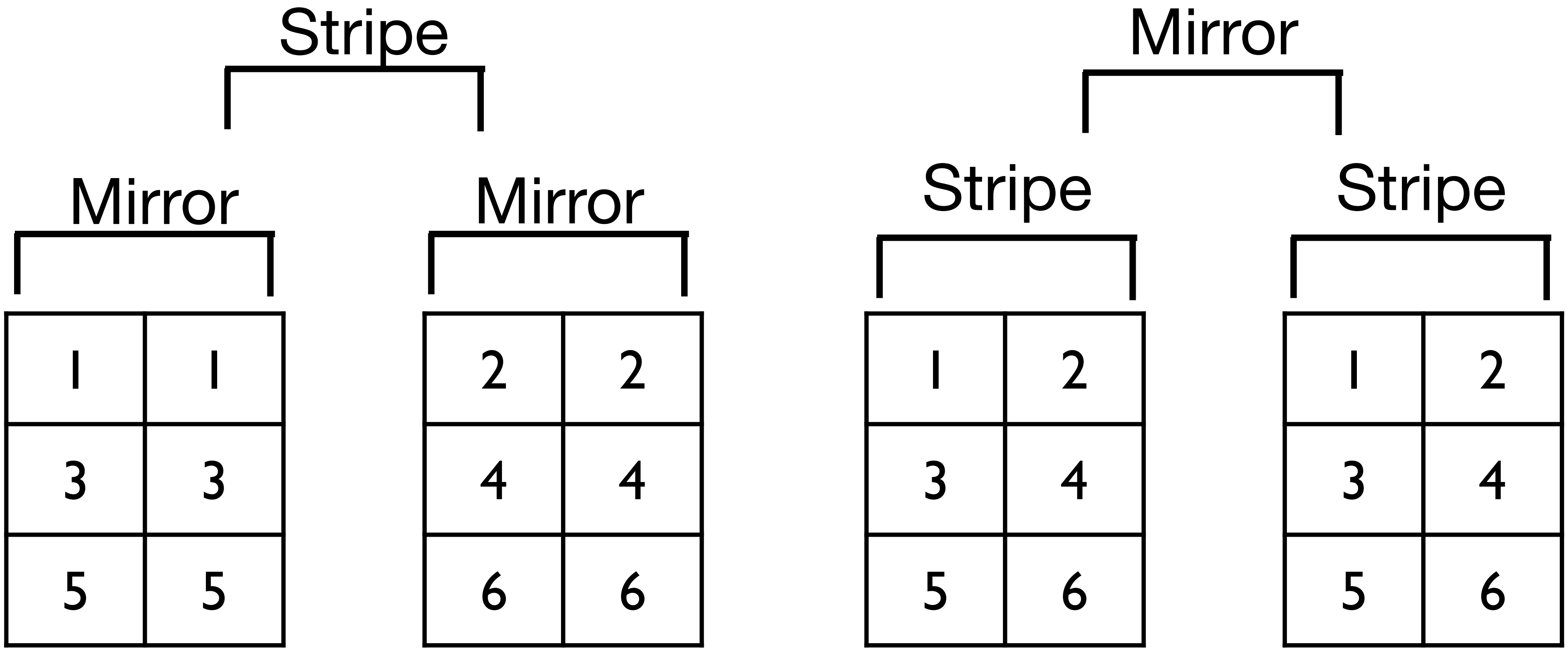


- How many disk errors can we survive?
  - 1: if we lose any disk, we can reconstruct that disk from the remaining disks

Other Considerations

You can combine some RAID levels in fun ways

RAID 10 vs. RAID 01



Other Considerations

You can combine some RAID ideas in fun ways

RAID 4

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

RAID 5

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

Why RAID

It is a relatively straightforward concept, but practical and very useful

The ideas can be applied to distributed storage and other environments (e.g., think of “nodes” as disks)

I’ve seen RAID used as an interview question

- But most of the concepts can be reasoned about on the fly
  - You should remember **mirroring**, **striping**, and **parity**, not Level 0, Level 1, Level 4, Level 5.
  - (Remind your interviewers that you are there to think not to memorize)
- Other levels are less common, but common sense. Explore!