# Data Storage on Disk

- Bits stored on disk with some error detection/correction bits
  - Correct random bit flips
  - Detect corruption of data
- Disk controller or OS can handle some errors(e.g., blacklisting certain sectors)
- If errors cannot be masked, user perceives hard disk failures
- Technologies such as RAID (Redundant Array of Inexpensive Disks) provide high reliability and performance by replicating across multiple disks.

# March 31, RAID

- Why more than one disk?
- What are the different RAID levels? (striping, mirroring, parity)
- Which RAID levels are best for reliability?  for capacity?
- Which are best for performance?  (sequential vs. random reads and writes)

# Only one disk?

Sometimes we want many disks —why?

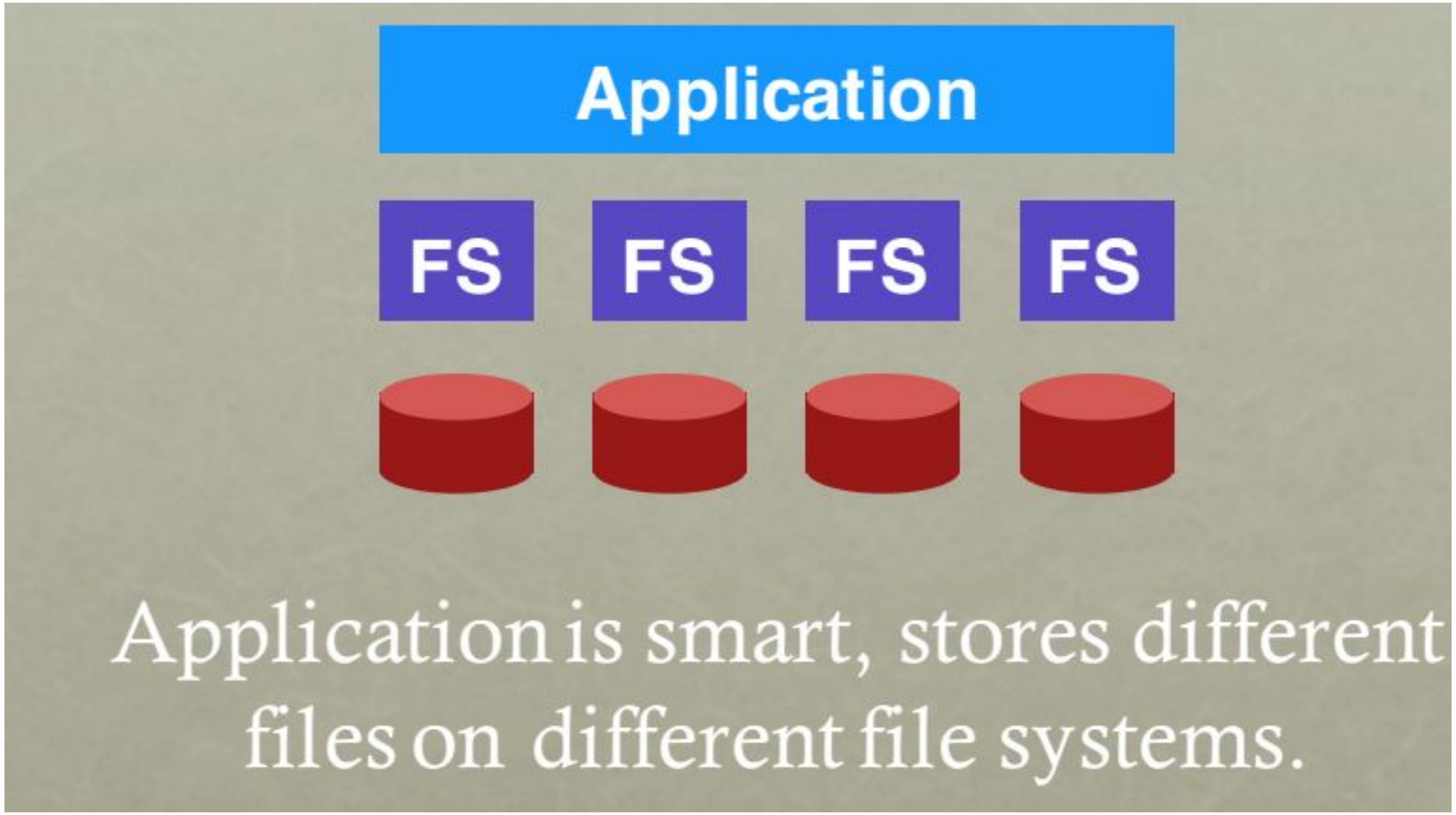
-capacity

-reliability

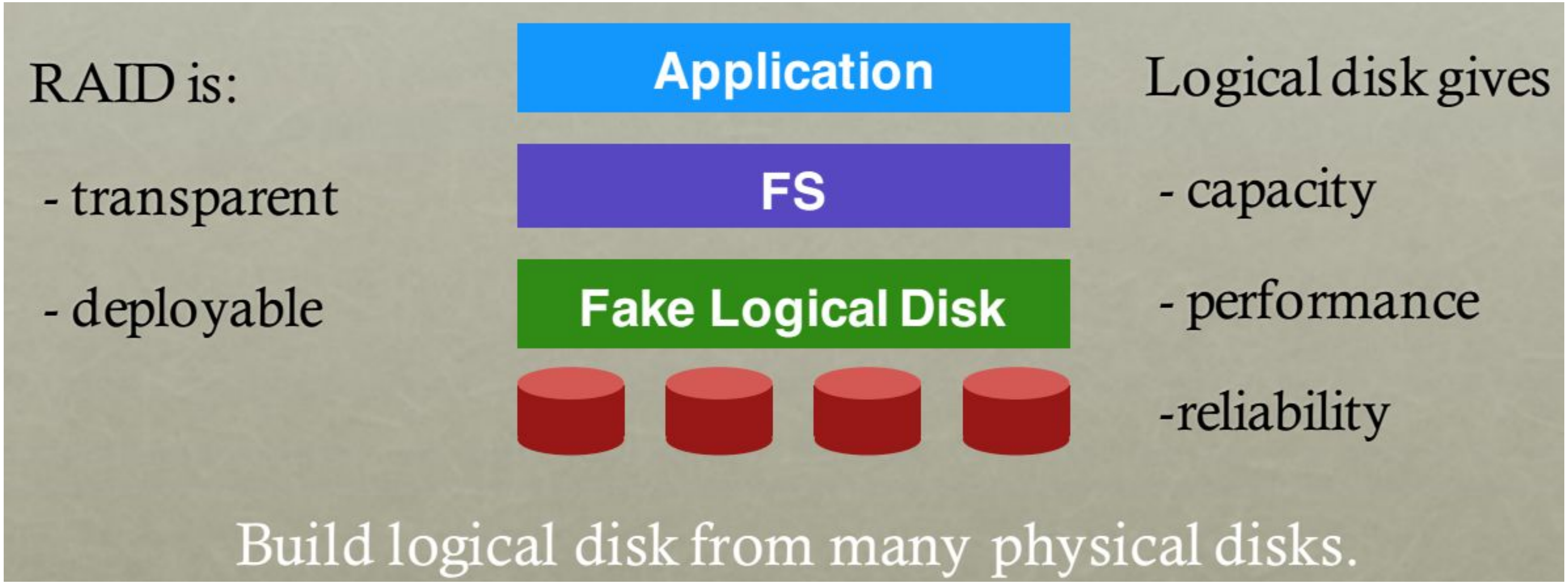
-Performance

Challenge: most file systems work on only one disk

# Sol 1: JBOD


Application
FS
FS
FS
FS
Application is smart, stores different files on different file systems.

# Sol2: RAID




**Redundant Array of Inexpensive Disks**

- **Use multiple disks** in concert to build a **faster**, **bigger**, and more **reliable** disk system.
    - RAID just looks like a big disk to the host system.

- Advantage
    - **Performance** & **Capacity**: Using multiple disks in parallel
    - **Reliability**: RAID can tolerate the loss of a disk.

**RAIDs provide these advantages transparently to systems that use them.**

# Why inexpensive Disks?

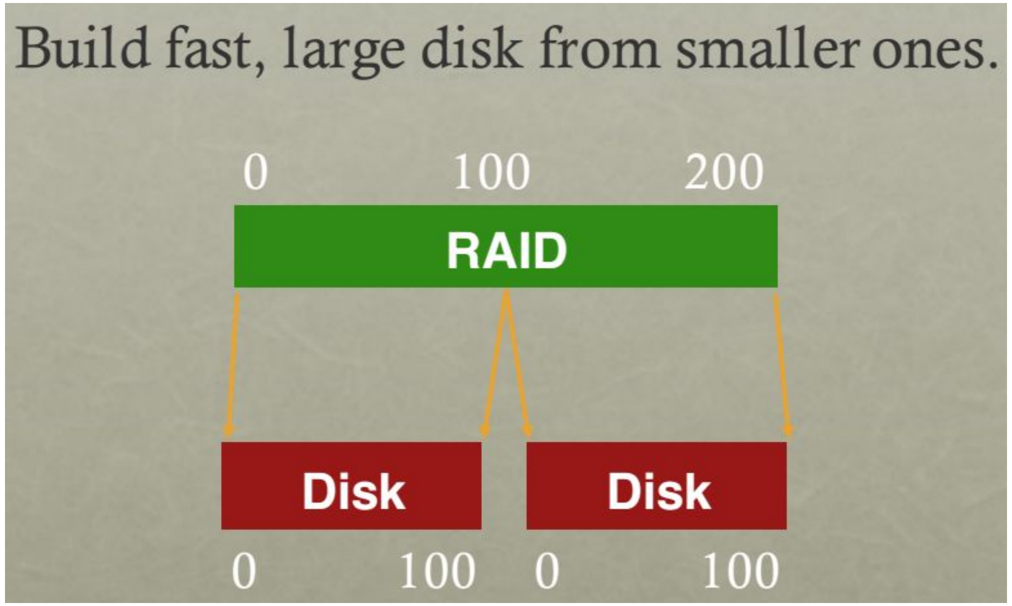
Economies of scale!  Commodity disks cost less

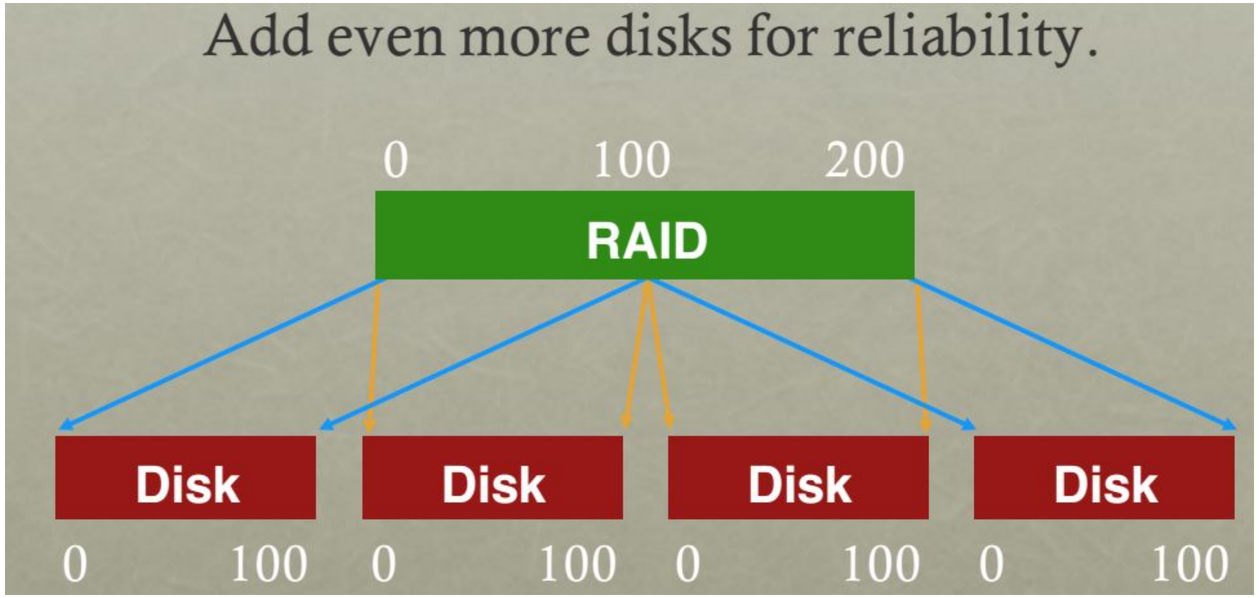
Can buy many commodity H/W components for the same price as few high-end components

Strategy: write S/W to build high-quality logical devices from many cheap devices

Alternative to RAID: buy an expensive, high-end disk

# General Strategy



# Mapping

How should we map logical block addresses to physical block addresses?

-Some similarity to virtual memory

1) Dynamicmapping: use data structure (hash table, tree)
-page tables

2) Staticmapping: use simple math - RAID

# Redundancy

Trade-offs to amount of redundancy

Increase number of copies:

-improves reliability (and maybe performance)

Decrease number of copies (deduplication)

-improves space efficiency

# Reasoning about RAID

**RAID:** system for mapping logical to physical blocks

**Workload:** types of reads/writes issued by applications (sequential vs. random)

**Metric:** capacity, reliability, performance

# RAID Decisions

Which logical blocks map to which physical blocks?

How do we use extra physical blocks (if any)?

Different RAID levels make different trade-offs

# Metrics

Capacity: how much space can apps use?

Reliability: how many disks can we safely lose? (assume fail stop!)

Performance: how long does each workload take?

Normalize each to characteristics of one disk

N := number of disks
C := capacity of 1 disk
S := sequential throughput of 1 disk
R := random throughput of 1 disk
D := latency of one small I/O operation

# Interface

- When a RAID receives I/O request,
  1. The RAID **calculates** which disk to access.
  2. The RAID **issue** one or more **physical I/Os** to do so.

- RAID example: A mirrored RAID system
  - Keep <u>two copies</u> of each block (each one on a separate disk)
  - Perform <u>two physical I/Os</u> for every one logical I/O it is issued.

# Internals

- A microcontroller
  - Run firmware to direct the operation of the RAID
- Volatile memory (such as DRAM)
  - Buffer data blocks
- Non-volatile memory
  - Buffer writes safely
- Specialized logic to perform parity calculation

# Fault Model

- RAIDs are designed to **detect** and **recover** from certain kinds of disk faults.

- Fail-stop fault model
  - A disk can be in one of two states: *Working* or *Failed*.
    - Working: all blocks can be read or written.
    - Failed: the disk is permanently lost.
  - RAID controller can immediately observe when a disk has failed.


12V ONLY


146 GB
Fibre Channel
450 GB

# RAID-0

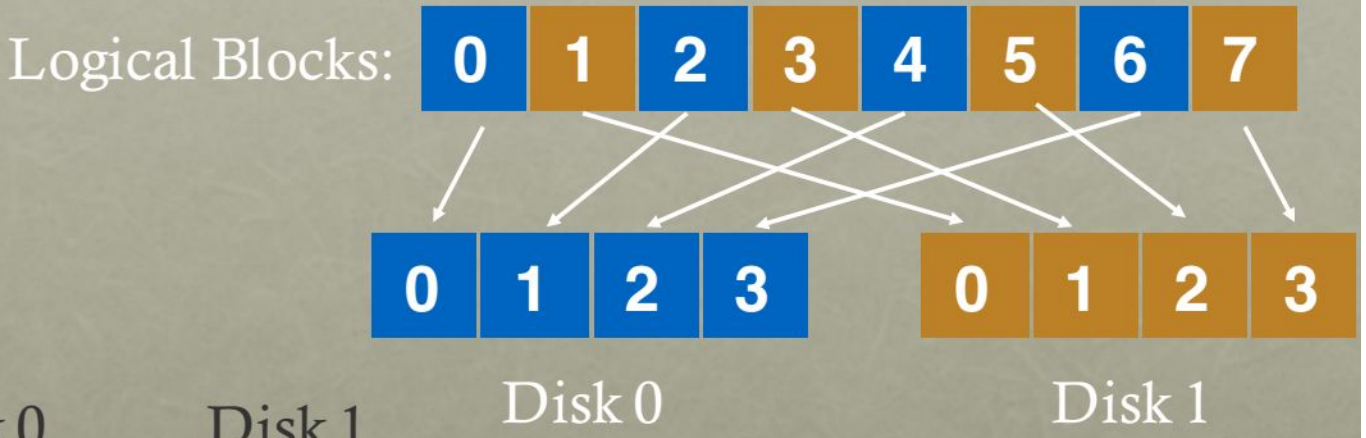
- RAID Level 0 is the simplest form as **striping** blocks.
  - **Spread the blocks** across the disks in a round-robin fashion.
  - No redundancy
  - Excellent performance and capacity

| Disk 0 | Disk 1 | Disk 2 | Disk 3 | |
|---|---|---|---|---|
| 0 | 1 | 2 | 3 | Stripe (The blocks in the same row) |
| 4 | 5 | 6 | 7 | |
| 8 | 9 | 10 | 11 | |
| 12 | 13 | 14 | 15 | |

**RAID-0: Simple Striping**
**(Assume here a 4-disk array)**

# RAID-0 [STRIPING]

Optimize for capacity. No redundancy




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

# RAID-0

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

Given logical address A, find:

Disk = A % disk_count

Offset = A / disk_count

- Example) RAID-0 with a bigger chunk size
  - Chunk size : 2 blocks (8 KB)
  - A Stripe: 4 chunks (32 KB)

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

chunk size: 2blocks

**Striping with a Bigger Chunk Size**

- Chunk size mostly affects performance of the array
  - **Small chunk size**
    - Increasing the parallelism
    - Increasing positioning time to access blocks
  - **Big chunk size**
    - Reducing intra-file parallelism
    - Reducing positioning time

**Determining the "best" chunk size is hard to do.**

**Most arrays use larger chunk sizes (e.g., 64 KB)**

# RAID-0 Analysis

- **Capacity** → RAID-0 is perfect.
  - Striping delivers N disks worth of useful capacity.

- **Performance** of striping → RAID-0 is excellent.
  - All disks are utilized often in parallel.

- **Reliability** → RAID-0 is bad.
  - Any disk failure will lead to data loss.

- Consider two performance metrics
  - Single request latency
  - Steady-state throughput

- Workload
  - **Sequential**: access 1MB of data (block (B) ~ block (B + 1MB))
  - **Random**: access 4KB at random logical address

- A disk can transfer data at
  - `S` MB/s under a sequential workload
  - `R` MB/s under a random workload

- sequential (S) vs random (R)
  - **Sequential** : transfer 10 MB on average as continuous data
  - **Random** : transfer 10 KB on average.
  - Average seek time: 7 ms
  - Average rotational delay: 3 ms
  - Transfer rate of disk: 50 MB/s

- Results:
  - S $= \frac{Amount\ of\ Data}{Time\ to\ access} = \frac{10\ MB}{210\ ms}$ = 47.62 MB /s
  - R $= \frac{Amount\ of\ Data}{Time\ to\ access} = \frac{10\ KB}{10.195\ ms}$ = 0.981 MB /s

# RAID-0 Analysis

What is capacity? **N * C**

How many disks can fail? **0**

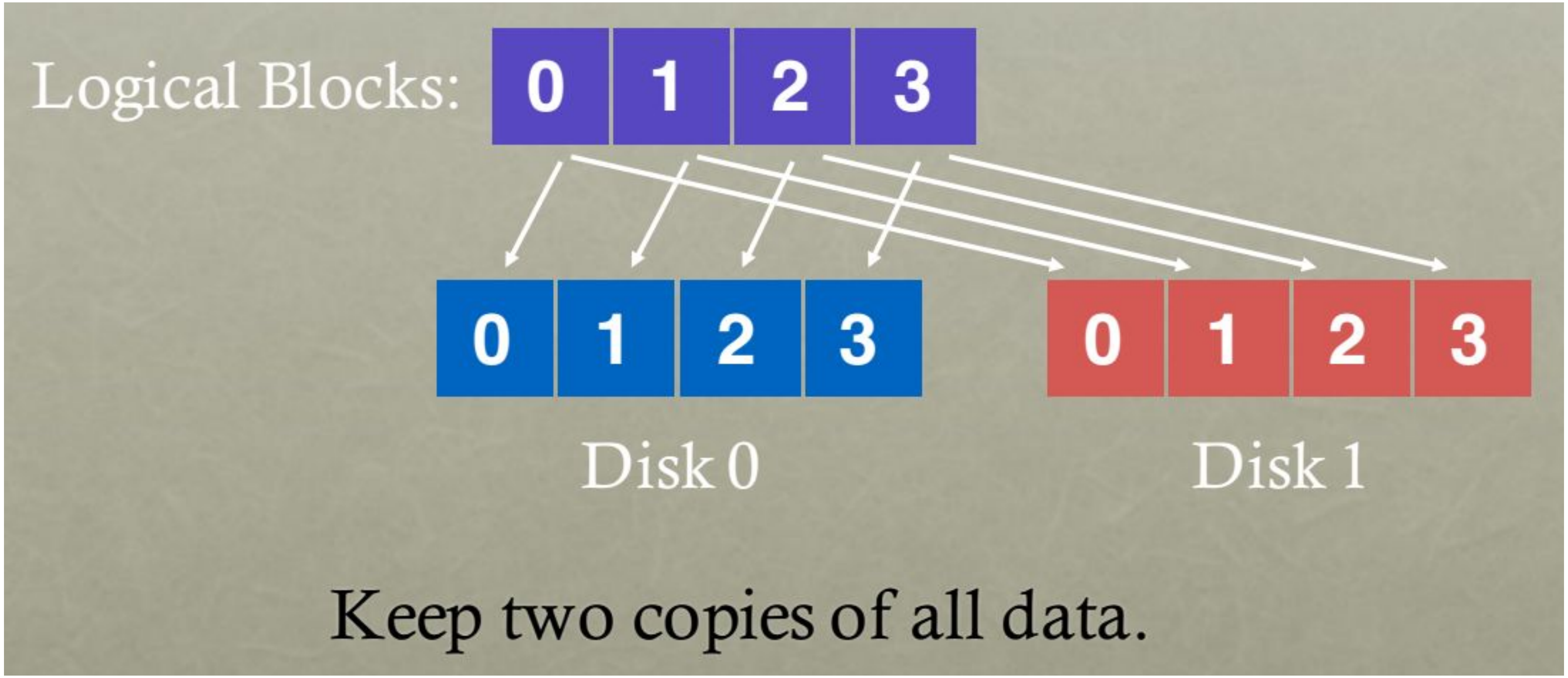
Latency **D**

Throughput (sequential, random)? **N*S , N*R**

Buying more disks improves throughput, but not latency!

N := number of disks
C := capacity of 1 disk
S := sequential throughput of 1 disk
R := random throughput of 1 disk
D := latency of one small I/O operation

# RAID-1 Mirroring



- RAID Level 1 tolerates **disk failures**.
  - **Copy** more than one of **each block** in the system.
  - Copy block places on a separate disk.

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

**Simple RAID-1: Mirroring (Keep two physical copies)**

- RAID-10 (RAID 1+0) : mirrored pairs and then stripe
- RAID-01 (RAID 0+1) : contain two large striping arrays, and then mirrors

# RAID-1 Analysis

$N$ : the number of disks

- **Capacity**: RAID-1 is Expensive
  - The useful capacity of RAID-1 is N/2.

- **Reliability**: RAID-1 does well.
  - It can tolerate the failure of any one disk (up to N/2 failures depending on which disk fail).

# RAID-1 Analysis

| | |
|---|---|
| What is capacity? | N/2 * C |
| How many disks can fail? | 1 (or maybe N / 2) |
| Latency (read, write)? | D |

# RAID-1 Throughput

- Two physical writes to complete
  - It suffers the worst-case seek and rotational delay of the two request.
  - Steady-state throughput
    - **Sequential Write** : $\frac{N}{2} \cdot S$ MB/s
      - Each logical write must result in two physical writes.
    - **Sequential Read** : $\frac{N}{2} \cdot S$ MB/s
      - Each disk will only deliver half its peak bandwidth.
    - **Random Write** : $\frac{N}{2} \cdot R$ MB/s
      - Each logical write must turn into two physical writes.
    - **Random Read** : $N \cdot R$ MB/s
      - Distribute the reads across all the disks.

# RAID-4

Use parity disk

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

Treat sectors across disks in a stripe as an equation.

Data on bad disk is like an unknown in the equation.

- Add **a single parity block**
  - **A Parity block** stores the *redundant information* for that stripe of blocks.

* P: Parity

| Disk 0 | Disk 1 | Disk 2 | Disk 3 | Disk 4 |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | P0 |
| 2 | 2 | 3 | 3 | P1 |
| 4 | 4 | 5 | 5 | P2 |
| 6 | 6 | 7 | 7 | P3 |

- **Compute parity** : the XOR of all of bits

| 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 |

- **Recover from parity**
  - Imagine the bit of the C2 in the first row is lost.
    1. Reading the other values in that row : 0, 0, 1
    2. The parity bit is 0 → <u>even number of 1's</u> in the row
    3. What the missing data must be: a 1.

# RAID-4 Analysis

| | |
|---|---|
| What is capacity? | (N-1) * C |
| How many disks can fail? | 1 |
| Latency (read, write)? | D, 2*D (read and write parity disk) |

# RAID-4 Throughput

- **Performance**
  - Steady-state throughput
    - Sequential read: $(N-1) \cdot S$ MB/s
    - Sequential write: $(N-1) \cdot S$ MB/s

| 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 |

**Full-stripe Writes In RAID-4**

    - Random read: $(N-1) \cdot R$ MB/s

# Random Write in RAID-4

- Overwrite a block + update the parity
- **Method 1**: *additive parity*
  - Read in all of the other data blocks in the stripe
  - XOR those blocks with the new block (1)
  - **Problem**: the performance <u>scales with</u> the number of disks
- **Method 2**: *subtractive parity*

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

  - Update C2(old) → C2(new)
    1. Read in the old data at C2 (C2(old)=1) and the old parity (P(old)=0
    2. Calculate P(new): $P(new) = (C2(old)\ XOR\ C2(new))\ XOR\ \ P(old)$
       - If C2(new)==C2(old) → P(new)==P(old)
       - If C2(new)!=C2(old) → Flip the old parity bit

- The parity disk can be a **bottleneck.**
  - Example: update blocks 4 and 13 (marked with *)

| 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 |

**Writes To 4, 13 And Respective Parity Blocks.**

- Disk 0 and Disk 1 can be accessed in parallel.
- Disk 4 prevents any parallelism.

**RAID-4 throughput under random small writes is $(\frac{R}{2})$ MB/s (*terrible*).**

# RAID-5

- RAID-5 **is solution of** small write problem.
  - Rotate the parity blocks across drives.
  - <u>Remove the parity-disk bottleneck</u> for RAID-4

| 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 |

**RAID-5 With Rotated Parity**

# RAID-5 Analysis

| | |
|---|---|
| What is capacity? | **(N-1) * C** |
| How many disks can fail? | **1** |
| Latency (read, write)? | **D, 2*D (read and write parity disk)** |

## Performance

- Sequential read and write
- A single read and write request

Same as RAID-4

- Random read : a little better than RAID-4
  - RAID-5 can utilize all of the disks.

- Random write : $\frac{N}{4} \cdot R$ MB/s
  - The factor of four loss is cost of using parity-based RAID.

Steady-state throughput for RAID-4:

- sequential reads? **(N-1) * S**
- sequential writes? **(N-1) * S**
- random reads? **(N-1) * R**
- random writes? **R/2 (read and write parity disk)**

| Disk0 | Disk1 | Disk2 | Disk3 | Disk4 |
|---|---|---|---|---|
| 3 | 0 | 1 | 2 | 6 |
| | | | | (parity) |

What is steady-state throughput for RAID-5?

- sequential reads? **(N-1) * S**
- sequential writes? **(N-1) * S**
- random reads? **(N) * R**
- random writes? **N * R/4**

| Disk0 | Disk1 | Disk2 | Disk3 | Disk4 |
|---|---|---|---|---|
| - | - | - | - | P |
| - | - | - | P | - |
| - | - | P | - | - |

...

# Comparisions

$N$ : the number of disks
$D$ : the time that a request to a single disk take

| | RAID-0 | RAID-1 | RAID-4 | RAID-5 |
|---|---|---|---|---|
| **Capacity** | N | N/1 | N-1 | N-1 |
| **Reliability** | 0 | 1 (for sure)<br>$\frac{N}{2}$ (if lucky) | 1 | 1 |
| **Throughput** | | | | |
| 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 | $\frac{1}{2}$R | $\frac{N}{4}$ R |
| **Latency** | | | | |
| Read | D | D | D | D |
| Write | D | D | 2D | 2D |

# Summary

- **Performance** and do not care about reliability → RAID-0 (Striping)
- **Random I/O** performance and **Reliability** → RAID-1 (Mirroring)
- **Capacity** and **Reliability** → RAID-5
- **Sequential I/O** and Maximize **Capacity** → RAID-5

# Summary

RAID-0 is always fastest and has best capacity (but at cost of reliability)

RAID-5 better than RAID-1 for sequential workloads

RAID-1 better than RAID-5 for random workloads

For this question, we'll examine how long it takes to perform a small workload consisting of 12 writes to random locations within a RAID. Assume that these random writes are spread "evenly" across the disks of the RAID. To begin with, assume a simple disk model where each read or write takes $D$ time units.

Assume we have a 4-disk RAID-0 (striping). How long does it take to complete the 12 writes?

How long on a 4-disk RAID-1 (mirroring)?

How long on a 4-disk RAID-4 (parity)?

How long on a 4-disk RAID-5 (rotated parity)?

| | | | |
|---|---|---|---|
| 0 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 |

~3D

| | | | |
|---|---|---|---|
| 0 | 0 | 1 | 1 |
| 2 | 2 | 3 | 3 |
| 4 | 4 | 5 | 5 |

~6D

| | | | |
|---|---|---|---|
| 0 | 1 | 2 | P |
| 4 | 5 | 6 | P |
| ~~7~~ | 8 | 9 | P |

~12D ×2 => 24D

| | | | |
|---|---|---|---|
| 0 | 1 | 2 | P |
| 4 | 5 | P | 6 |
| 7 | P | 8 | 9 |
| P | 10 | 11 | 12 |

~6D ×2 => 12D