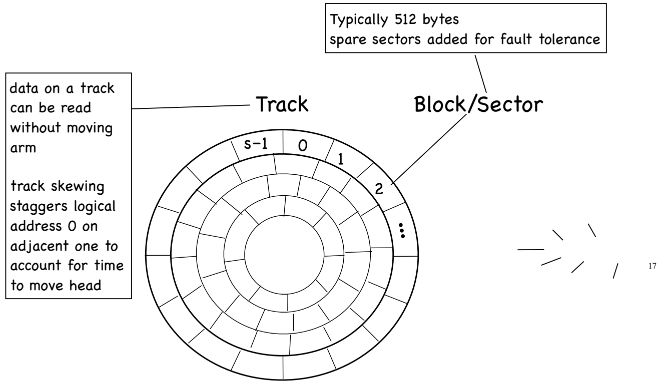
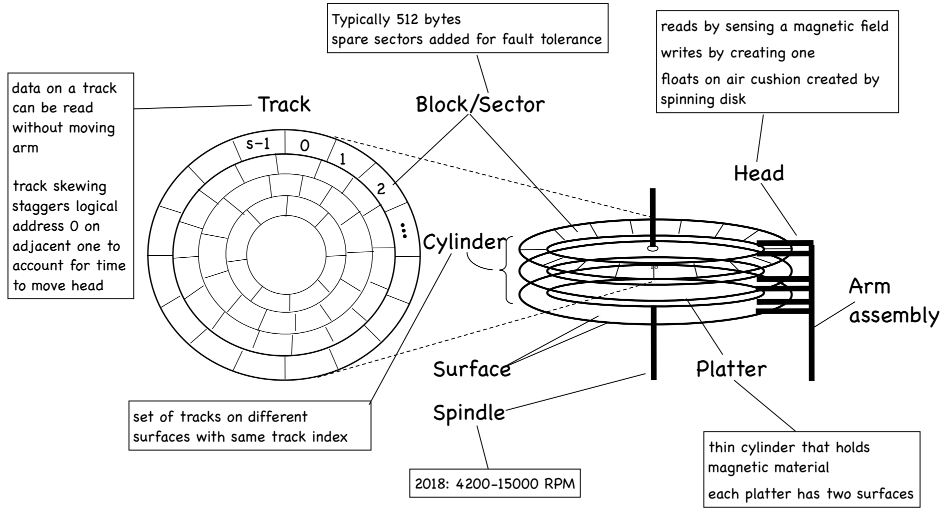
# Disk Drive Schematic

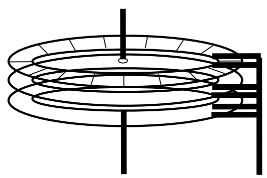



# Disk Drive Schematic




# Disk Read/Write

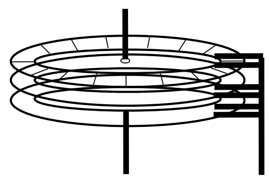
- Present disk with a sector address
  - Old: CHS = (cylinder, head, sector)
  - New abstraction: Logical Block Address (LBA)
    - linear addressing 0…N-1
- Heads move to appropriate track
  - seek
  - settle
- Appropriate head is enabled
- Wait for sector to appear under head
  - rotational latency
- Read/Write sector
  - transfer time



Disk access time:

# Disk Read/Write

- Present disk with a sector address
  - Old: CHS = (cylinder, head, sector)
  - New abstraction: Logical Block Address (LBA)
    - linear addressing 0…N-1
- Heads move to appropriate track
  - seek (and though shalt approximately find)
  - settle (fine adustments)
- Appropriate head is enabled
- Wait for sector to appear under head
  - rotational latency
- Read/Write sector
  - transfer time

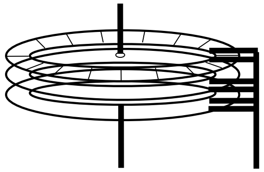


Disk access time:

seek time +

# Disk Read/Write

- Present disk with a sector address
  - Old: CHS = (cylinder, head, sector)
  - New abstraction: Logical Block Address (LBA)
    - linear addressing 0…N-1
- Heads move to appropriate track
  - seek (and though shalt approximately find)
  - settle (fine adustments)
- Appropriate head is enabled
- Wait for sector to appear under head
  - rotational latency
- Read/Write sector
  - transfer time



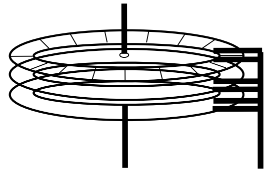
Disk access time:

seek time +

rotation time +

# Disk Read/Write

- Present disk with a sector address
  - Old: CHS = (cylinder, head, sector)
  - New abstraction: Logical Block Address (LBA)
    - linear addressing 0…N-1
- Heads move to appropriate track
  - seek (and though shalt approximately find)
  - settle (fine adustments)
- Appropriate head is enabled
- Wait for sector to appear under head
  - rotational latency
- Read/Write sector
  - transfer time



Disk access time:

seek time +

rotation time +

transfer time

# A closer look: seek time

- Minimum: time to go from one track to the next
  - 0.3-1.5 ms
- Maximum: time to go from innermost to outermost track
  - more than 10ms; up to over 20ms
- Average: average across seeks between each possible pair of tracks
  - approximately time to seek 1/3 of the way across disk

# How did we get that?

- To compute average seek time, add distance between every possible pair of tracks and divide by total number of pairs
  - assuming tracks, pairs, and sum of distances is

    which we compute as

# How did we get that?

- To compute average seek time, add distance between every possible pair of tracks and divide by total number of pairs
  - assuming tracks, pairs, and sum of distances is
  - which we compute as
  - The inner integral expands to
  - which evaluates to

# How did we get that?

- To compute average seek time, add distance between every possible pair of tracks and divide by total number of pairs
  - assuming tracks, pairs, and sum of distances is
  - which we compute as
  - The inner integral expands to
  - which evaluates to
  - The outer integral becomes
  - which we divide by the number of pairs to obtain $N/3$

# A closer look: seek time

- Minimum: time to go from one track to the next
  - 0.3-1.5 ms
- Maximum: time to go from innermost to outermost track
  - more than 10ms; up to over 20ms
- Average: average across seeks between each possible pair of tracks
  - approximately time to seek 1/3 of the way across disk
- Head switch time: time to move from track on one surface to the same track on a different surface
  - range similar to minimum seek time

# A closer look: rotation time

- Today most disk rotate at 4200 to 15,000 RPM
  - ≈15ms to 4ms per rotation
  - good estimate for rotational latency is half that amount
- Head starts reading as soon as it settles on a track
  - track buffering to avoid "shoulda coulda" if any of the sectors flying under the head turn out to be needed

# A closer look: transfer time

- Surface transfer time
  - Time to transfer one or more sequential sectors to/from surface after head reads/writes first sector
  - Much smaller that seek time or rotational latency
    - 512 bytes at 100MB/s ≈ 5$\mu$s (0.005 ms)
  - Lower for outer tracks than inner ones
    - same RPM, but more sectors/track: higher bandwidth!
- Host transfer time
  - time to transfer data between host memory and disk buffer
    - 60MB/s (USB 2.0) to 2.5GB/s (Fibre Channel 20GFC)

# Buffer Memory

- Small cache (8 to 16 MB) that holds data
  - read from disk
  - about to be written to disk
- On write
  - write back (return from write as soon as data is cached)
  - write through (return once it is on disk)

# Computing I/O time

- The rate of I/O is computed as

# Example: Toshiba MK3254GSY (2008)

| Size | |
|---|---|
| Platters/Heads | 2/4 |
| Capacity | 320GB |
| Performance | |
| Spindle speed | 7200 RPM |
| Avg. seek time R/W | 10.5/12.0 ms |
| Max. seek time R/W | 19 ms |
| Track-to-track | 1 ms |
| Surface transfer time | 54-128 MB/s |
| Host transfer time | 375 MB/s |
| Buffer memory | 16MB |
| Power | |
| Typical | 16.35 W |
| Idle | 11.68 W |

# 500 Random Reads

| Size | |
|---|---|
| Platters/Heads | 2/4 |
| Capacity | 320GB |
| Performance | |
| Spindle speed | 7200 RPM |
| Avg. seek time R/W | 10.5/12.0 ms |
| Max. seek time R/W | 19 ms |
| Track-to-track | 1 ms |
| Surface transfer time | 54-128 MB/s |
| Host transfer time | 375 MB/s |
| Buffer memory | 16MB |
| Power | |
| Typical | 16.35 W |
| Idle | 11.68 W |

- Workload
  - 500 read requests, randomly chosen sector
  - served in FIFO order
- How long to service them?
  - 500 times (seek + rotation + transfer)
  - seek time: 10.5 ms (avg)
  - rotation time:
    - 7200 RPM = 120 RPS
    - rotation time 8.3 ms
    - on average, half of that: 4.15 ms
  - transfer time
    - at least 54 MB/s
    - 512 bytes transferred in (.5/54,000) seconds = 9.26μs
  - Total time:
    - 500 x (10.5 + 4.15 + 0.009) ≈ 7.33 sec

$$R_{I/O} = \frac{500 \times .5 \times 10^{-3} MB}{7.33\, s} = 0.034\, MB/s$$

# 500 Sequential Reads


| Size | |
|---|---|
| Platters/Heads | 2/4 |
| Capacity | 320GB |
| Performance | |
| Spindle speed | 7200 RPM |
| Avg. seek time R/W | 10.5/12.0 ms |
| Max. seek time R/W | 19 ms |
| Track-to-track | 1 ms |
| Surface transfer time | 54-128 MB/s |
| Host transfer time | 375 MB/s |
| Buffer memory | 16MB |
| Power | |
| Typical | 16.35 W |
| Idle | 11.68 W |


- Workload
  - 500 read requests for sequential sectors on the same track
  - served in FIFO order
- How long to service them?
  - seek + rotation + 500 times transfer
  - seek time: 10.5 ms (avg)
  - rotation time:
    - 4.15 ms, as before
  - transfer time
    - outer track: 500 x (.5/128000) ≈ 2ms
    - inner track: 500 x (.5/54000) seconds ≈ 4.6ms
  - Total time is between:
    - outer track: (2 + 4.15 + 10.5) ms ≈ 16.65 ms

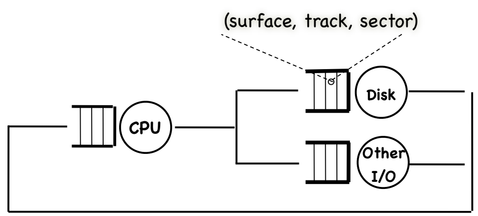
$$R_{I/O} = \frac{500 \times .5 \times 10^{-3} MB}{16.65\, ms} = 15.02\, MB/s$$

    - inner track: (4.6 + 4.15 + 10.5) ms ≈ 19.25 ms

$$R_{I/O} = \frac{500 \times .5 \times 10^{-3} MB}{19.25\, ms} = 12.99\, MB/s$$

# Disk Head Scheduling

- In a multiprogramming/time sharing environment, a queue of disk I/Os can form

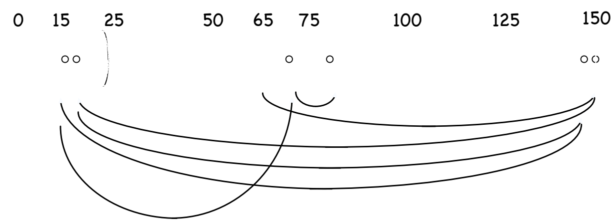



- OS maximizes disk I/O throughput by minimizing head movement through disk head scheduling
  - and this time we have a good sense of the length of the task!

# FCFS

- Assume a queue of request exists to read/write tracks

···· 83 | 72 | 14 | 147 | 16 | 150 and the head is on track 65

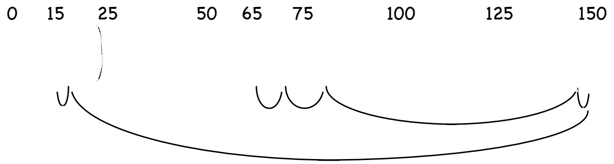



FCFS scheduling results in disk head moving 550 tracks

and makes no use of what we know about the length of the tasks!

# SSTF: Shortest Seek Time First

- Greedy scheduling
  - Rearrange queue from: .... 83 | 72 | 14 | 147 | 16 | 150
  - to: .... 14 | 16 | 150 | 147 | 83 | 72




Head moves 221 tracks BUT

mismatch with array-of-blocks interface

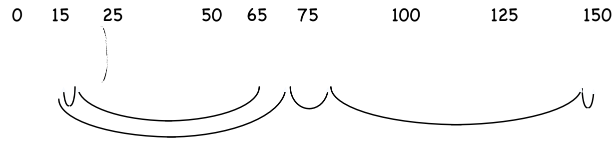
starvation

# SCAN Scheduling "Elevator"

- Move the head in one direction until all requests have been serviced, and then reverse
  - sweeps disk back and forth

Rearrange queue from: .... 83 | 72 | 14 | 147 | 16 | 150

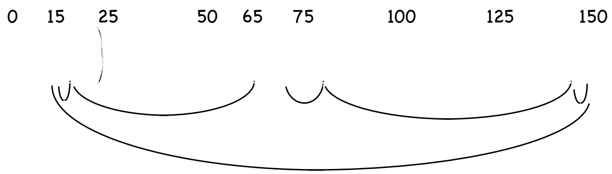
to: .... 150 | 147 | 83 | 72 | 14 | 16




Head moves 187 tracks.

# C-SCAN scheduling

- Circular SCAN
  - sweeps disk in one direction (from outer to inner track), then resets to outer track and repeats e




- More uniform wait time than SCAN
  - moves head to serve requests that are likely to have waited longer

# Outsourcing Scheduling Decisions

- Selecting which track to serve next should include rotation time (not just seek time!)
  - SPTF: Shortest Positioning Time First
- Hard for the OS to estimate rotation time accurately
  - Hierarchical decision process
    - OS sends disk controller a batch of "reasonable" requests
    - disk controller makes final scheduling decisions

# Error detection and correction

- A layered approach
  - At the hardware level, checksums and device-level checks
    - remedy through error correcting codes
  - At the system level, redundancy, as in RAID
  - End-to-end checks at the file system level

# Storage device failures and mitigation - I

- Sector/page failure (i.e., Partial failure)
  - Data lost, rest of device operates correctly
    - Permanent (e.g. due to scratches) or transient (e.g., due to "high fly writes" producing weak magnetic fields, or write/read disturb errors)
    - Non recoverable read errors: in 2011, one bad sector/page per $10^{14}$ to $10^{18}$ bits read
  - Mitigations
    - data encoded with additional redundancy (error correcting codes + error notification)
    - for non recoverable read errors, remapping (device includes spare sectors/pages)
  - Pitfalls
    - non-recoverable error rates are negligible - 10% when reading a 2TB disk with a bad sector/$10^{14}$ bits
    - non-recoverable error rates are constant - they depend on load, age, workload
    - failures are independent - errors often correlated in time or space
    - error rates are uniform - different causes can contribute differently to nonrecoverable read errors

# Example: unrecoverable read errors

- Your 500GB laptop disk just crashed BUT you have just made a full backup on a 500GB disk
  - non recoverable read error rate: 1 sector/$10^{14}$ bits read
- What is the probability of reading successfully the entire disk during restore?

Expected number of failures while reading the data:

$$500 \text{ GB} \times \frac{8 \times 10^9 \text{ bits}}{\text{GB}} \times \frac{1 \text{ error}}{10^{14} \text{ bits}} = 0.04$$

Alternatively...

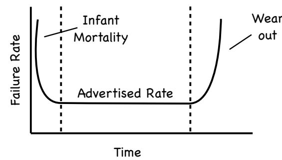
Assume each bit has a $10^{-14}$ chance of being wrong and that failures are independent

Probability to read all bits successfully:

$$(1 - 10^{-14})^{(500 \times 8 \times 10^9)} = 0.9608$$

# Storage device failures and mitigations - II

- Device failures
  - Device stops to be able to serve reads and writes to all sectors/pages (e.g. due to capacitor failure, damaged disk head, wear-out)
  - Annual failure rate
    - fraction of disks expected to fail/year
      - 2011: 0.5% to 0.9%
  - Mean Time To Failure (MTTF)
    - inverse of annual failure rate
      - 2011: $10^6$ hours (0.9%) to $1.7 \times 10^6$ hours (0.5%)
  - Pitfalls
    - MTTF measures a device's useful life (MTTF applies to device's intended service life)
    - advertised failure rates are trustworthy
    - failures are independent
    - failure rates are constant
    - devices behave identically
    - ignore warning signs (SMART technology)
      - Self Monitornig, Analysis, ReportTing



# Example: disk failures in a large system

- File server with 100 disks
- MTTF for each disk: $1.5 \times 10^6$ hours
- What is the expected time before one disk fails?

Assuming independent failures and constant failure rates:

MTTF for some disk = MTTF for single disk / 100 = $1.5 \times 10^4$ hours

Probability that some disk will fail in a year:

$$(365 \times 24) \text{ hours } \times \frac{1}{1.5 \times 10^4} \frac{\text{errors}}{\text{hours}} = 58.5\%$$

Pitfalls:

actual failure rate may be higher than advertised
failure rate may not be constant

# RAID

Redundant Array of Inexpensive* Disks

* In industry, "inexpensive" has been replaced by "independent" :-)

# E Pluribus Unum

- Implement the abstraction of a faster, bigger and more reliable disk using a collection of slower, smaller, and more likely to fail disks
  - different configurations offer different tradeoffs
- Key feature: transparency
  - to the OS looks like a single, large, highly performant and highly reliable single disk
    - a linear array of blocks
    - mapping needed to get to actual disk
    - cost: one logical I/O may translate into multiple physical I/Os
- In the box:
  - microcontroller, DRAM (to buffer blocks) [sometimes non-volatile memory, parity logic]

# Failure Model

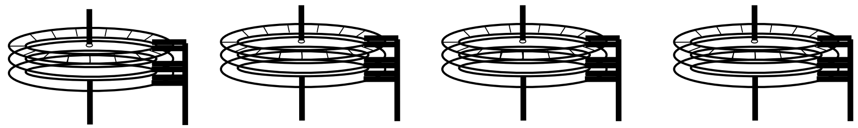
- RAIDs can detect and recover from certain kinds of failures
- Adopt the strong, somewhat unrealistic Fail-Stop failure model
  - component works correctly until it crashes, permanently
    - disk is either working: all sectors can be read and written
    - or has failed: it is permamently lost
  - failure of the component is immediately detected
    - RAID controller can immediately observe when a disk has failed

# How to Evaluate a RAID

- Capacity
  - what fraction of the sum of the storage of its constituent disks does the RAID make available?
- Reliability
  - How many disk fault can a specific RAID configuration tolerate?
- Performance
  - Workload dependent

# RAID-0: Striping

Spread blocks across disks using round robin

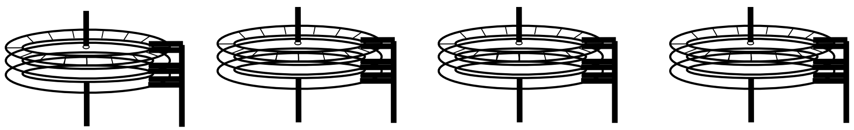


| | | | | |
|---|---|---|---|---|
| Stripe | 0 | 1 | 2 | 3 |
| Stripe | 4 | 5 | 6 | 7 |
| Stripe | 8 | 9 | 10 | 11 |
| Stripe | 12 | 13 | 14 | 15 |

+ Excellent parallelism — high positioning time

# RAID-0: Striping

Spread blocks across disks using round robin



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

+ lower positioning time — lower parallelism

# RAID-0: Evaluation

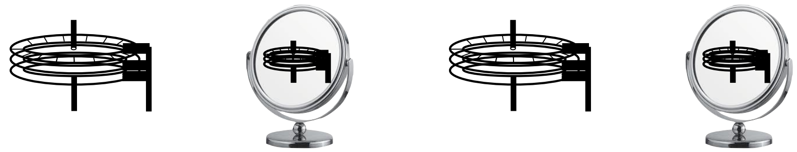
- Capacity
  - Excellent: N disks of B blocks: RAID-0 exports NxB blocks
- Reliability
  - Poor: Any disk failure causes data loss
- Performance
  - Workload dependent, of course
  - We'll consider two
    - Sequential: single disk transfers S MB/s
    - Random: single disk transfer R MB/s
    - S >> R (50 times higher in your textbook example!)

# RAID-0: Performance

- Single-block read/write thoughput
  - about the same as accessing a single disk
- Latency
  - Read: T ms (latency of one I/O op to disk)
  - Write: T ms
- Steady-state read/write throughput
  - Sequential: N x S MB/s
  - Random: N x R MB/s

# RAID-1: Mirroring

Each block is replicated twice



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

Read from any        Write to both

# RAID-1: Evaluation

- Capacity
  - Poor: N disks of B blocks yield (N x B)/2 blocks
- Reliability
  - Good: Can tolerate the failure of any one disk
    - and if you can pick who fails, can tolerate up to N/2 disk failures [NOT ROBUST!]
- Performance
  - Fine for reads: can choose any disk
  - Poor for writes: every logical write requires writing to both disks
    - suffers worst seek+rotational delay of the two writes

# RAID-1: Performance

- Steady-state throughput
  - Sequential Writes: N/2 x S MB/s
    - Each logical W involves two physical W
  - Sequential Reads: N/2 x S MB/s

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

Suppose we want to read
0, 1, 2, 3, 4, 5, 6, 7

# RAID-1: Performance

- Steady-state throughput
  - Sequential Writes: N/2 x S MB/s
    - Each logical W involves two physical Ws
  - Sequential Reads: N/2 x S MB/s

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

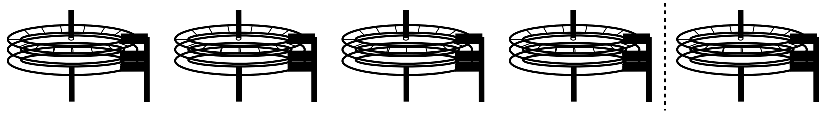
Suppose we want to read
0, 1, 2, 3, 4, 5, 6, 7

Each disk only delivers half of his bandwidth

  - Random Writes: N/2 x R MB/s
    - Each logical W involves two physical Ws
  - Random Reads: N x R MB/s
    - Reads can be distributed across all disks
- Latency for Reads and Writes: T ms

# RAID-4: Block Striped, with Parity

Data disks　　Parity disk



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

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 1 | 1 | 0 |
| 0 | 1 | 1 |

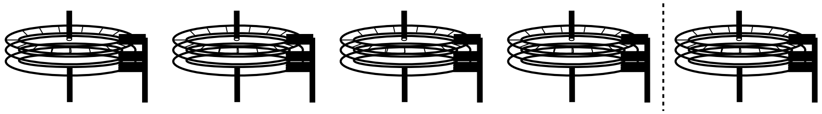
| | | |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 1 | 1 |
| 0 | 0 | 1 |

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 1 |
| 1 | 1 | 0 |

# RAID-4: Block Striped, with Parity

Data disks　　Parity disk



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

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 1 | 1 | 0 |
| 0 | 1 | 1 |

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

| | | |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 1 | 1 |
| 0 | 0 | 1 |

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 1 |
| 1 | 1 | 0 |

Disk controller can identify faulty disk
single parity disk can detect and correct errors

# RAID-4: Evaluation

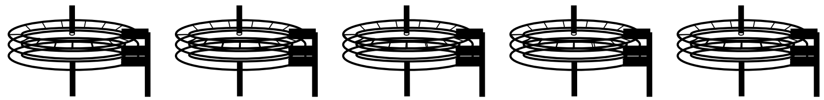
- Capacity
  - Pretty good: N disks of B blocks yield (N-1) x B blocks
- Reliability
  - Pretty Good: Can tolerate the failure of any one disk
- Performance
  - Fine for sequential read/write accesses and random reads
  - Random writes are a problem!

# RAID-4: Performance

- Steady-state throughput
  - Sequential Writes: (N-1) x S MB/s
  - Sequential Reads: (N-1) x S MB/s
  - Random Read: (N-1) x S MB/s
  - Random Writes: R/2 MB/s (Yikes!)
    - need to read block from disk and parity block
    - Compute $P_{new}$ = ($B_{old}$ XOR $B_{new}$) XOR $P_{old}$
    - Write back $B_{new}$ and $P_{new}$
    - Bottleneck accessing P disk eliminates any parallelism for random writes
- Latency
  - Reads: T ms    Writes: 2T m

# RAID-5: Rotating Parity

Parity and Data distributed across all disks



| | | | | |
|---|---|---|---|---|
| 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: Evaluation

- Capacity
  - As in Raid-4
- Reliability
  - As in Raid-4
- Performance
  - Sequential read/write accesses as in RAID-4
  - Random Reads are slightly better
    - N x R MB/s (instead of (N-1) x R MB/s
  - Random Writes are much better than in RAID-4
    - (N/4) x R MBs (each logical read causes 4 I/O ops)