15-410

"...What goes around comes around..."

Storage
Oct. 14, 2016

Dave Eckhardt

Dave O'Hallaron

Contributions from

- **Garth Gibson**
- **Brian Railing & Steve Muckle**
- **Eno Thereska, Rahul Iyer**
- **15-213**
- **"How Stuff Works" web site**

Overview

What is “storage”?

Anatomy of a Hard Drive

SSD's

What you should know

Storage

Where is your stuff when you turn your machine off?

Storage

Where is your stuff when you turn your machine off?

- In “the cloud”!

Storage

Where is your stuff when you turn your machine off?

- **In "the cloud"!**

Where does *the cloud* store your stuff?

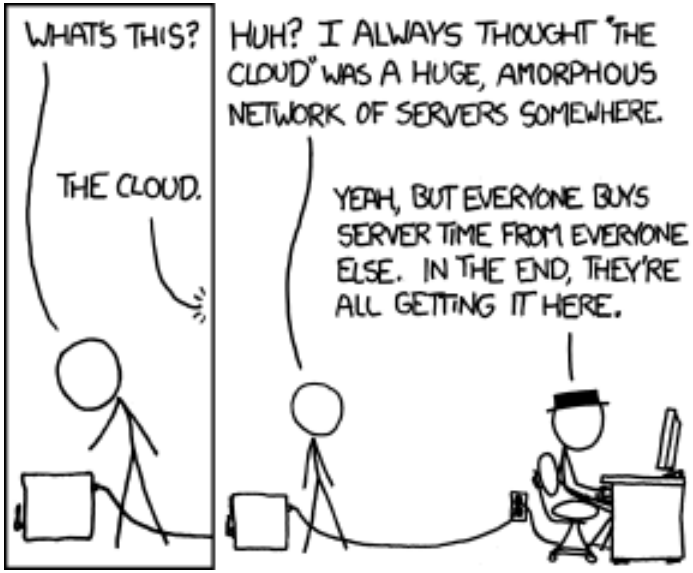
Storage

Where is your stuff when you turn your machine off?

- **In "the cloud"!**

Where does *the cloud* store your stuff?

http://xkcd.com/908/



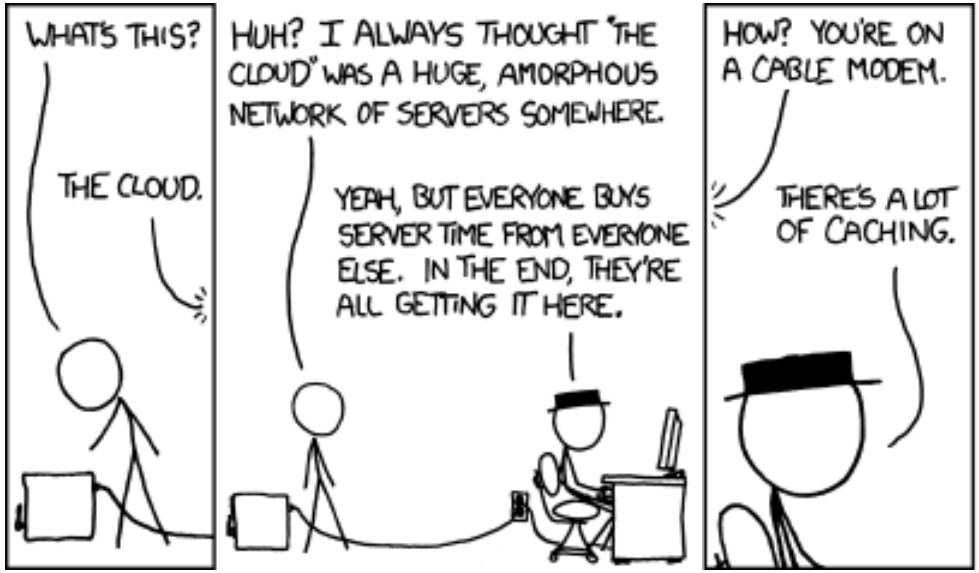
Storage

Where is your stuff when you turn your machine off?

- **In "the cloud"!**

Where does *the cloud* store your stuff?

http://xkcd.com/908/



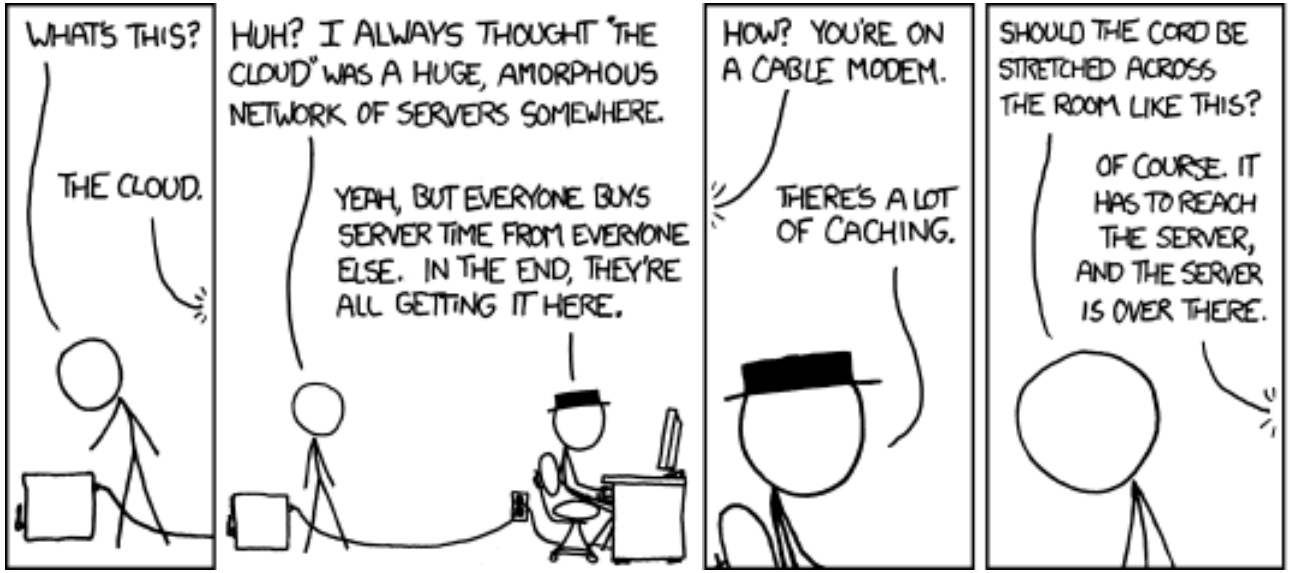
Storage

Where is your stuff when you turn your machine off?

- **In "the cloud"!**

Where does *the cloud* store your stuff?

http://xkcd.com/908/



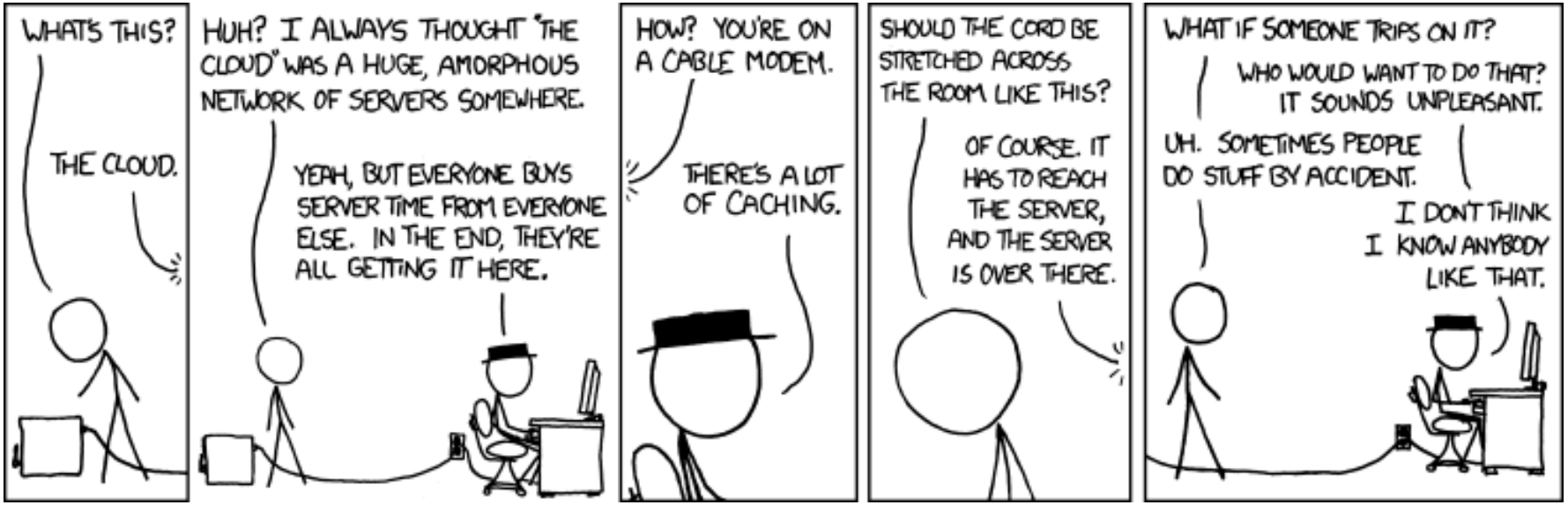
Storage

Where is your stuff when you turn your machine off?

- **In "the cloud"!**

Where does *the cloud* store your stuff?

http://xkcd.com/908/



Storage

Various devices

- "Drum"
- Magnetic tape
- "Hard disk"
- Floppy disc
- CD-ROM
- Magneto-Optical (MO) disk
- Flash memory
- ?MRAM?

What do they have in common?

How do they differ?

Storage Features

"Non-volatile"

- **Remembers without electricity**

Slow **(compared to RAM)**

- **Milliseconds or seconds instead of nanoseconds**
- **Can't execute programs from it (must fetch first)**

"Block oriented"

- **Fetch and store large clumps of data**
  - **Floppy: 128/256/512 bytes**
  - **Spinning disk: 512/4096 bytes**
  - **CD-ROM: 2048 bytes**
  - **Flash: "hard to say"**
- **Time to fetch 1 byte == time to fetch 1 block**

Storage Model

"Non-volatile"

- Write, power-off, read: should return same value
  - *Years* later!

Address space

- Blocks have numbers
  - Ancient times: (C,H,S) tuple
    - C, H, S were geometric features of old disks
  - Modern: (LBA)
    - "Logical Block Address" runs from 0..N

Storage Model

Reading and writing

- Read-block(N) ⇒ [huge delay] ⇒ block else failure
  - Sometimes a re-try helps (usually not)

Storage Model

Reading and writing

- Read-block(N) ⇒ [huge delay] ⇒ block else failure
  - Sometimes a re-try helps (usually not)
- Write-block(N) ⇒ [huge delay] ⇒ “ok” else failure
  - Failures usually indicate “obvious” bad things
    - The disk motor stopped
  - “Successful” write doesn't *guarantee* a later read
  - Devices usually contain a power buffer
    - A write operation either completes or has no effect

Storage Model

Reading and writing

- Read-block(N) ⇒ [huge delay] ⇒ block else failure
  - Sometimes a re-try helps (usually not)
- Write-block(N) ⇒ [huge delay] ⇒ “ok” else failure
  - Failures usually indicate “obvious” bad things
    - The disk motor stopped
  - “Successful” write doesn't *guarantee* a later read
  - Devices usually contain a power buffer
    - A write operation either completes or has no effect
- Modern devices support “tagged command queueing”
  - OS can issue multiple requests, each has a “tag”
  - Device can return results in any order, with the OS's tag

Command Queueing In Action

Disks serve read requests out of order

- OS queues: “read 37”, “read 83”, “read 2”
  - Disk returns 37, 2, 83
    - Great! That's why we buy smart disks and queue multiple requests
- OS queues: “read 37”, “read 38”, “read 39”
  - Disk does one seek, reads 37-40, plus also 40-72 while it's in the neighborhood
  - Sends sectors to OS as they become available

Command Queueing In Action

Disks serve read requests out of order

- OS queues: “read 37”, “read 83”, “read 2”
  - Disk returns 37, 2, 83
    - Great! That's why we buy smart disks and queue multiple requests
- OS queues: “read 37”, “read 38”, “read 39”
  - Disk does one seek, reads 37-40, plus also 40-72 while it's in the neighborhood
  - Sends sectors to OS as they become available

Disks serve ***write*** requests out of order, too

- OS queues “write 23”, “write 24”, “write 1000”, “read 4-8”, ...
  - Disk writes 24, 23 (!!), gives you 4, 5, 6, 7, 8, writes 1000
    - What if power fails before last write?
    - What if power fails between first two writes?

Command Queueing In Action

How can OS ensure data-structure integrity?

- Special commands
  - "Flush all pending writes"
    - Think "my disk is 'modern'", think "disk barrier"
    - Can even queue a flush to apply to all before now
    - Can apply these "barrier" flushes to subsets of requests
    - Rarely used by operating system
  - "Disable write cache"
    - Think "please don't be quite so modern"

Examples

"Hard drive"

- Parts
- Execution model

Flash memory

- Challenges
  - Write amplification
  - Wear leveling

Anatomy of a Hard Drive

On the outside, a hard drive looks like this

Taken from "How Hard Disks Work"
http://computer.howstuffworks.com/hard-disk2.htm

Anatomy of a Hard Drive

If we take the cover off, we see that there actually is a "hard disk" inside

Taken from "How Hard Disks Work"
http://computer.howstuffworks.com/hard-disk2.htm

Anatomy of a Hard Drive

A hard drive usually contains multiple disks, called *platters*

These spin at thousands of RPM (5400, 7200, 10000, …)

Slower disks use less power

Taken from "How Hard Disks Work"
http://computer.howstuffworks.com/hard-disk2.htm

Anatomy of a Hard Drive

Information is written to and read from the platters by the *read/write heads* on the end of the *disk arm*

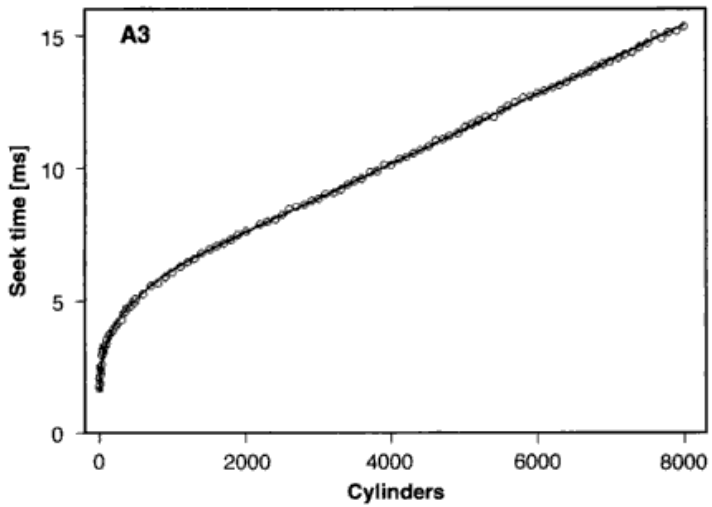
Taken from "How Hard Disks Work"
http://computer.howstuffworks.com/hard-disk2.htm

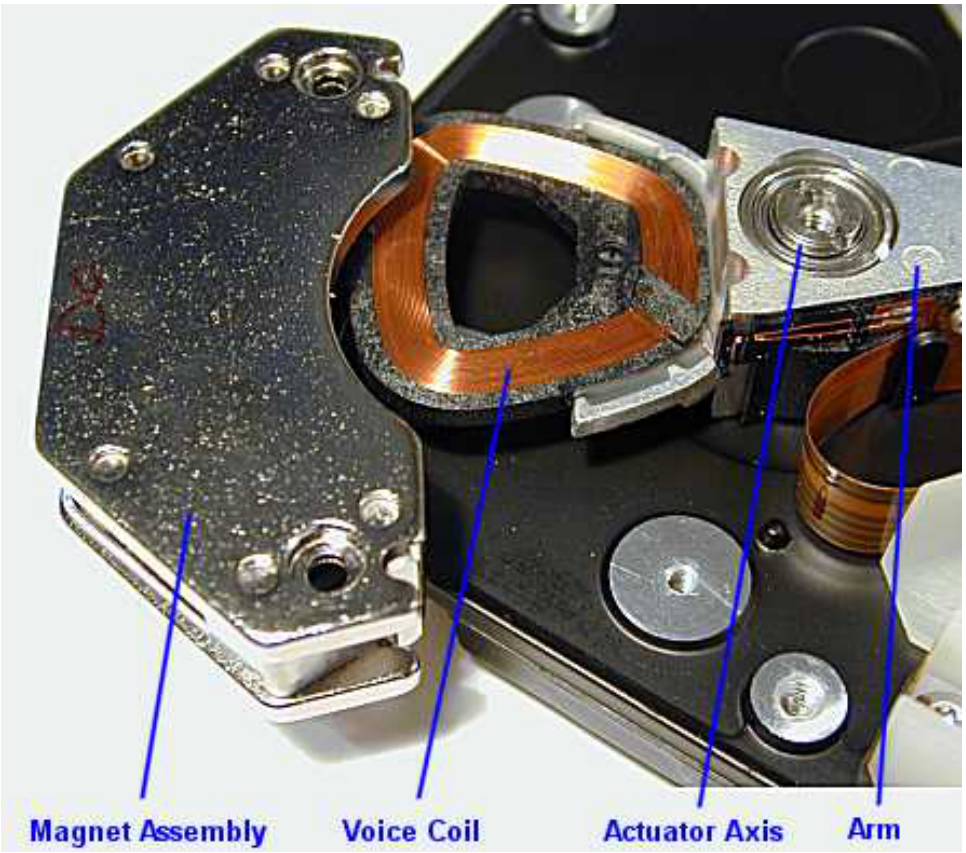
Anatomy of a Hard Drive

The arm is moved by a voice coil actuator

Slow, as computers go

- **Acceleration time**
- **Travel time**



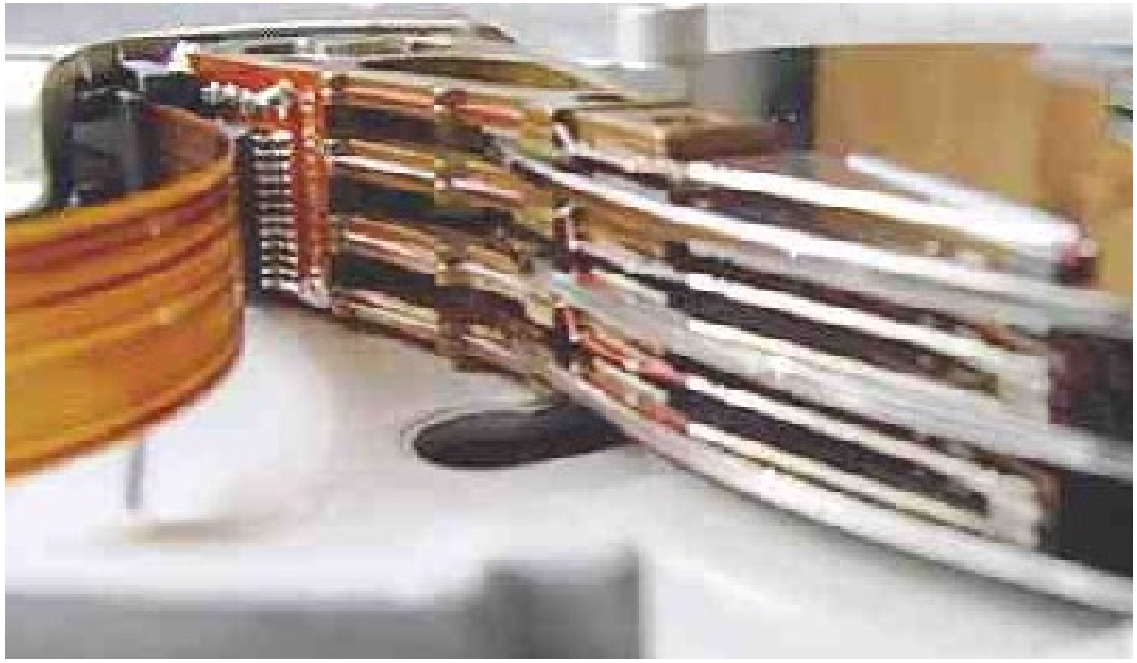


Taken from "Hard Disk Drives"
http://www.pcguide.com/ref/hdd

Oklobdzija, Comp. Eng. Handbook, 2002

Anatomy of a Hard Drive

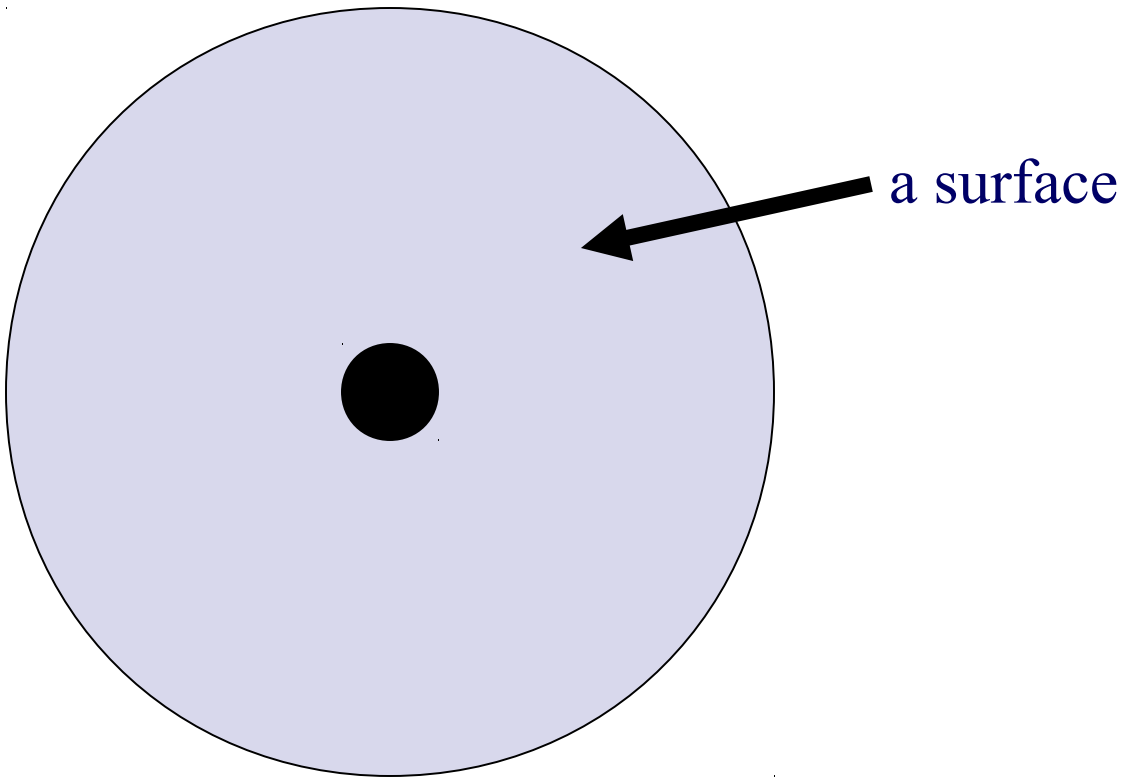
- Both sides of each platter store information
- Each side of a platter is called a *surface*
- Each surface has its own read/write head



Taken from "How Hard Disks Work"
http://computer.howstuffworks.com/hard-disk2.htm

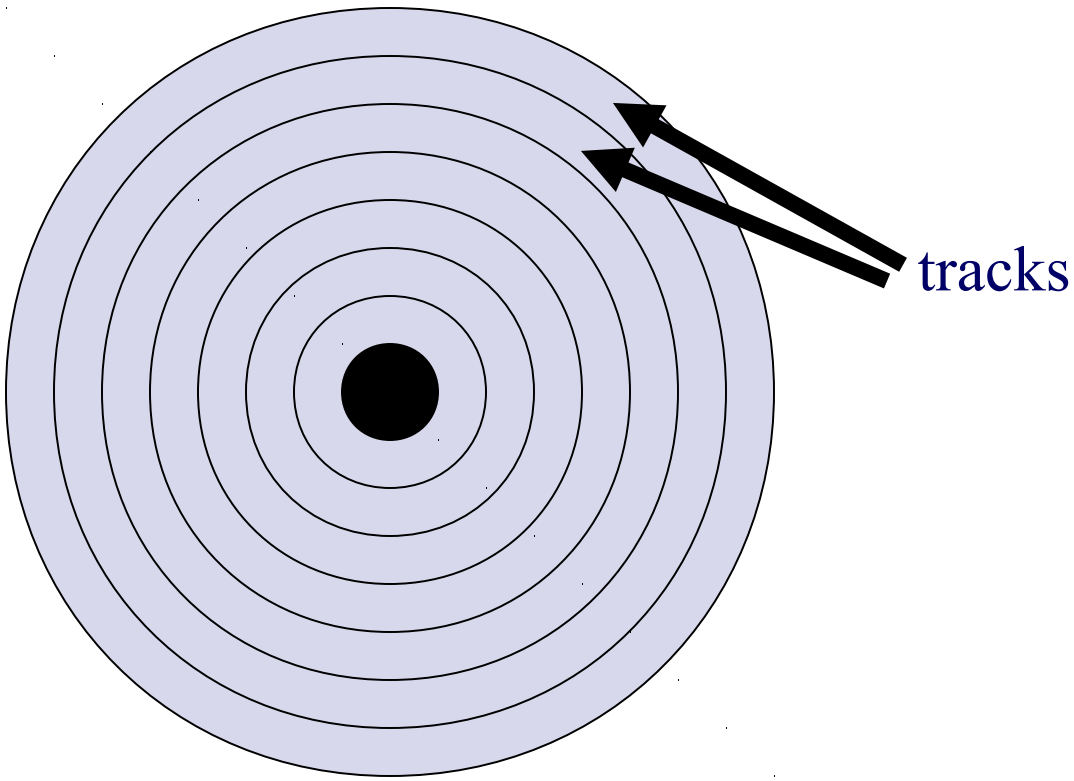
Anatomy of a Hard Drive

How are the surfaces organized?



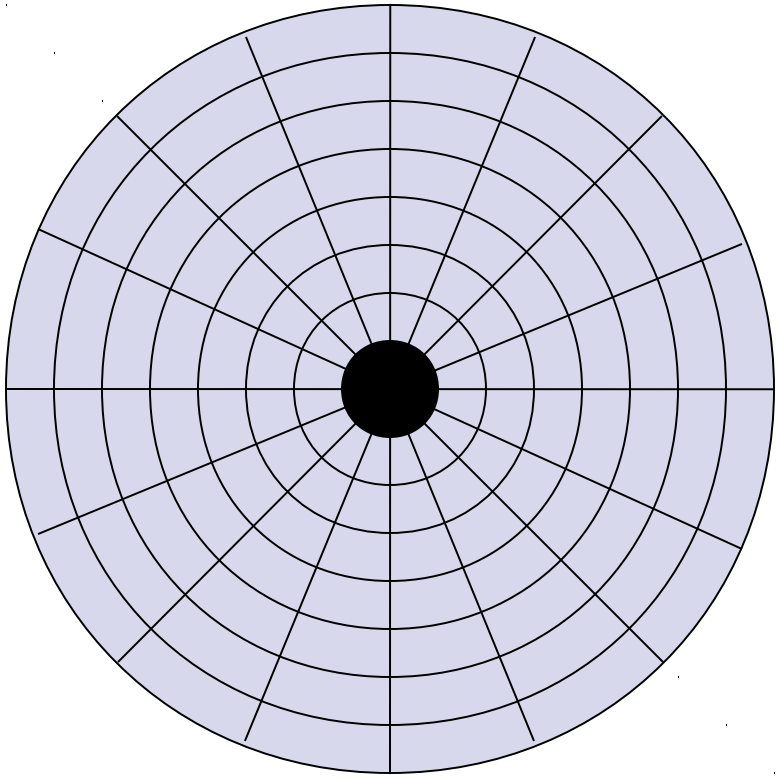
Anatomy of a Hard Drive

Each surface is divided by concentric circles, creating *tracks*



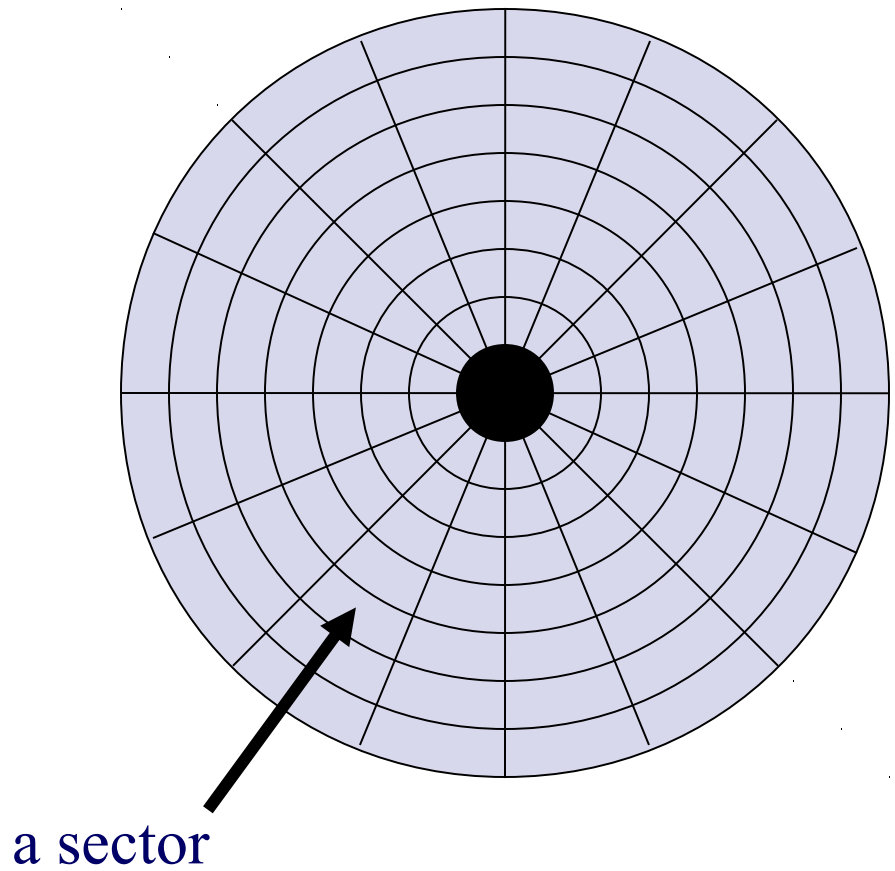
Anatomy of a Hard Drive

These tracks are further divided into *sectors*



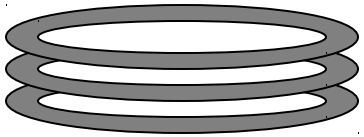
Anatomy of a Hard Drive

- **These tracks are further divided into *sectors***
- **A sector is the smallest unit of data transfer to or from the disk**
  - **512 bytes – traditional disks**
  - **2048 bytes – CD-ROMs**
  - **4096 bytes – 2010 disks**
    - **(pretend to be 512!)**



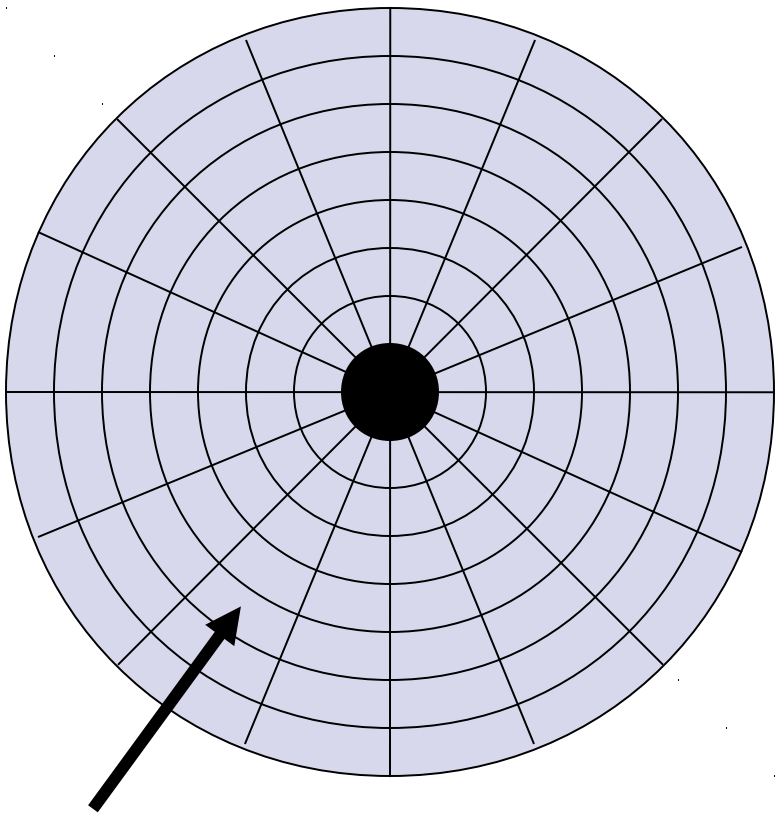
Disk Cylinder

Matching tracks across surfaces are collectively called a *cylinder*



Anatomy of a Hard Drive

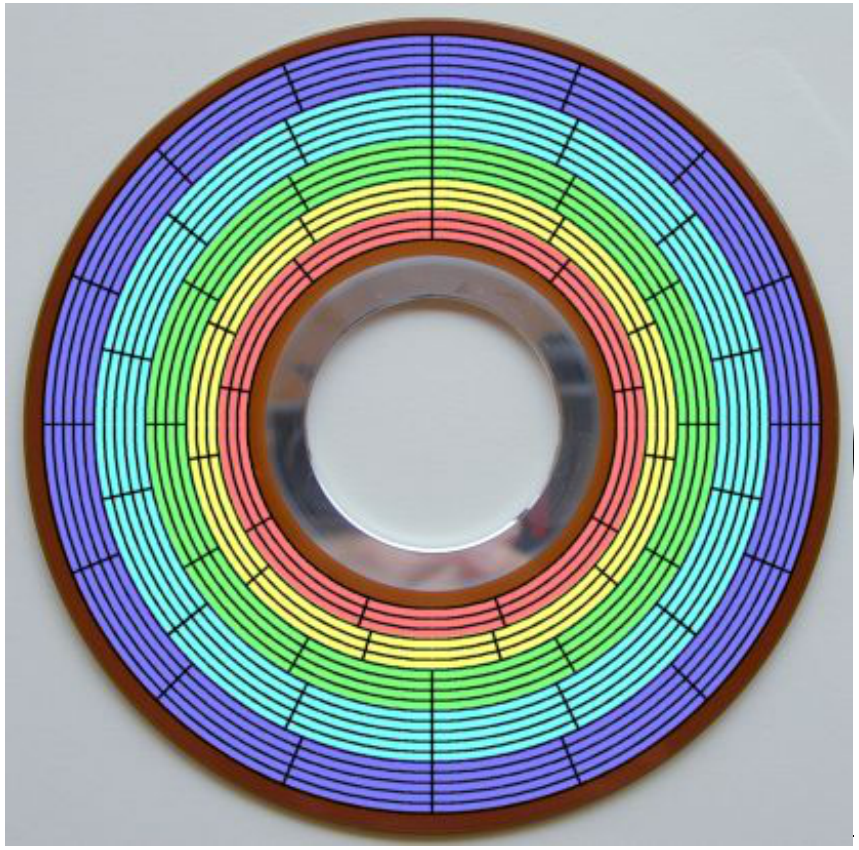
- **These tracks are further divided into *sectors***
- **A sector is the smallest unit of data transfer to or from the disk**
  - 512 bytes – traditional disks
  - 2048 bytes – CD-ROMs
  - 4096 bytes – 2010 disks
    - (pretend to be 512!)
- **"Sector address"**
  - "C/H/S"
- **Gee, those outer sectors look bigger...?**



Anatomy of a Hard Drive, Really

Modern hard drives use *zoned bit recording*

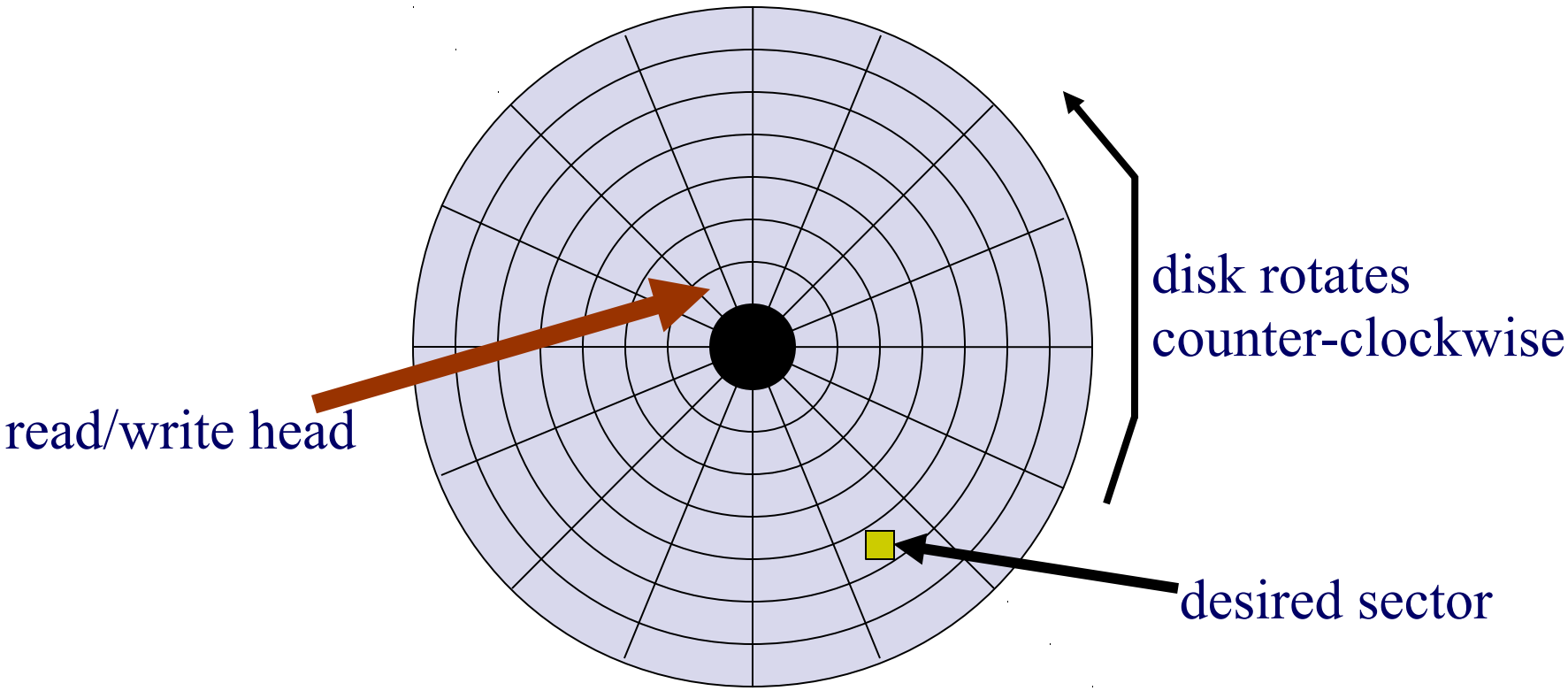
- Disk has tables to map track# to #sectors
- Sectors are all roughly the same linear length
- LBA "sector address" names a sector, like "page number" names a frame



Taken from "Reference Guide – Hard Disk Drives"
http://www.storagereview.com/map/lm.cgi/zone

Anatomy of a Hard Drive

Let's read in a sector from the disk



Anatomy of a Hard Drive

We need to do two things to transfer a sector

1. **Move the read/write head to the appropriate track ("seek time")**

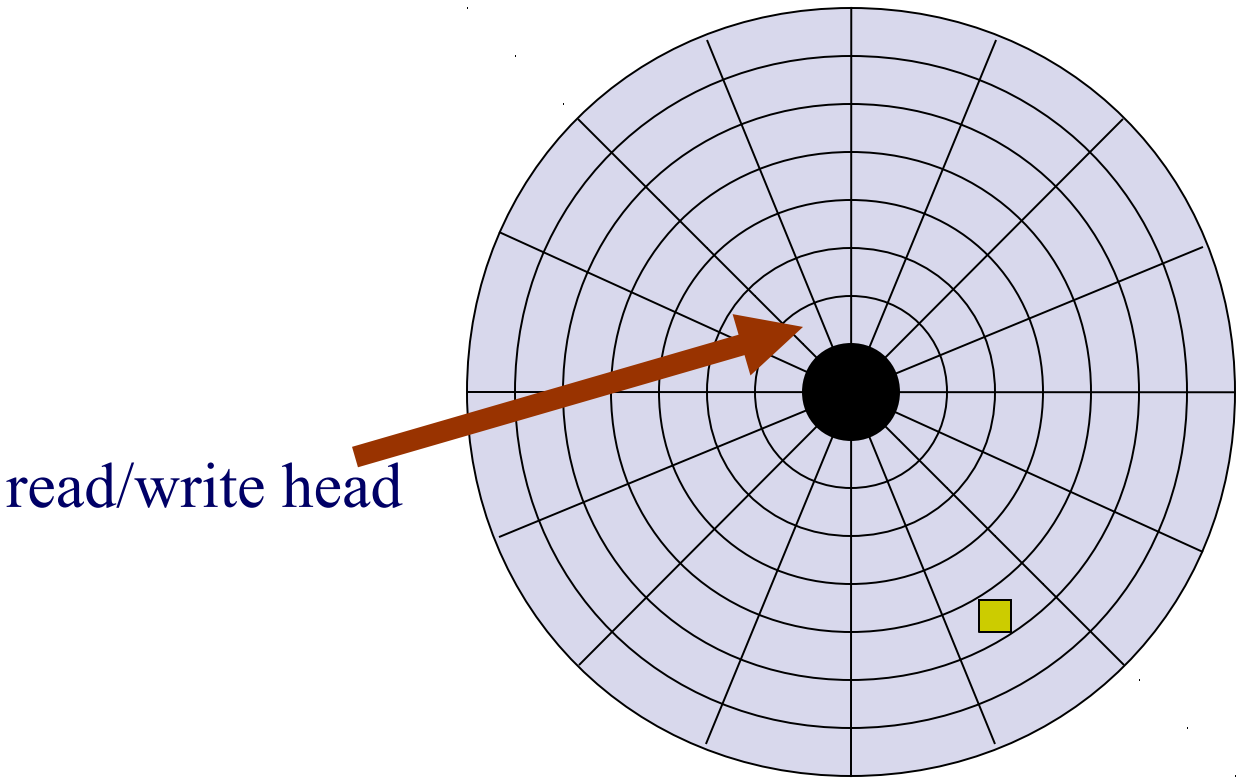
2. **Wait until the desired sector spins around ("rotational delay"/"rotational latency")**

Observe

- **Average seeks are 2 – 10 msec**
- **Rotation of 5400/7200/10K/15K rpm means rotational delay of 11/8/6/4 msec**
- **Rotation dominates short seeks, matches average seeks**

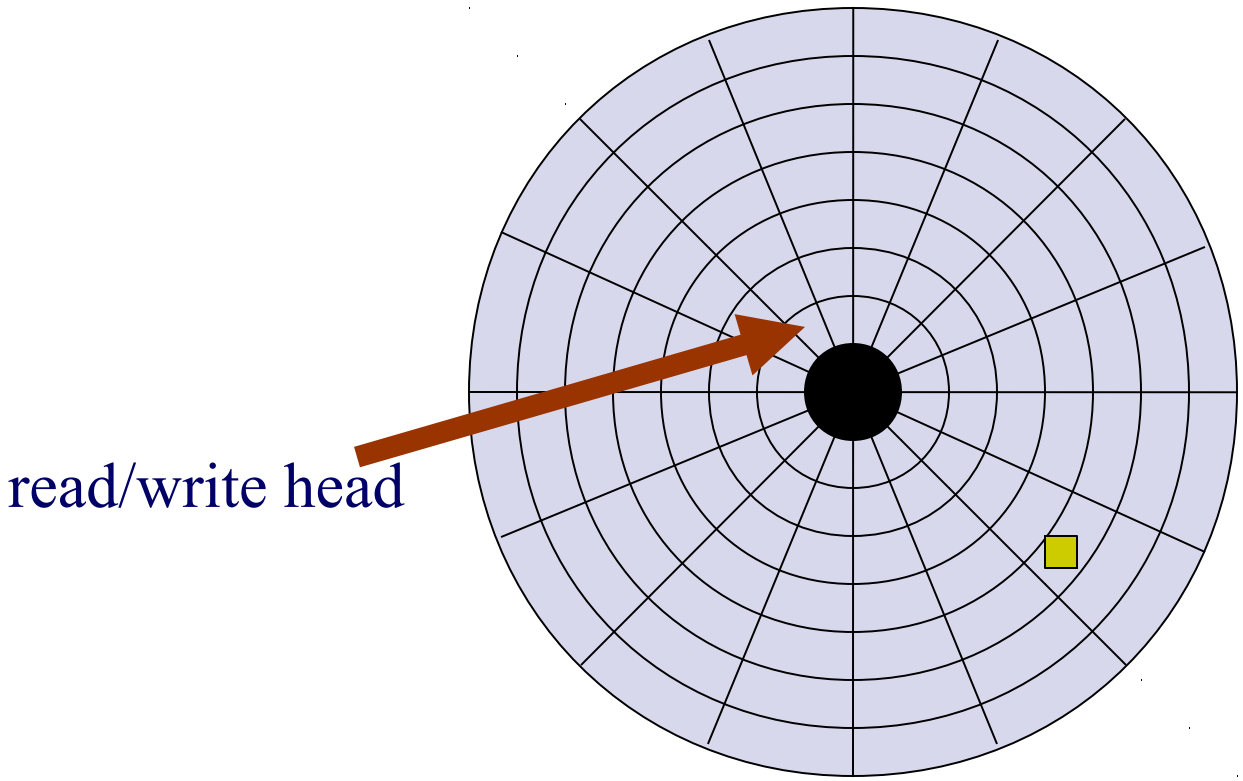
Anatomy of a Hard Drive

Let's read in a sector from the disk



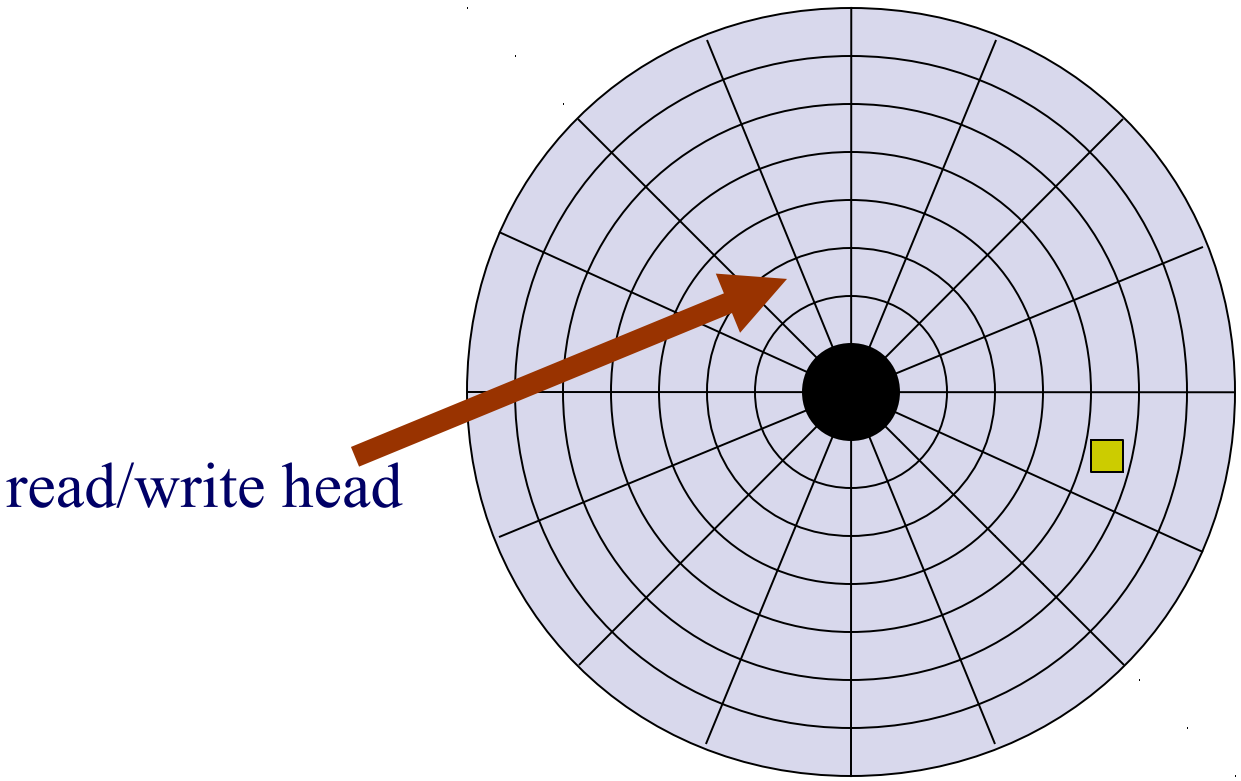
Anatomy of a Hard Drive

Let's read in a sector from the disk



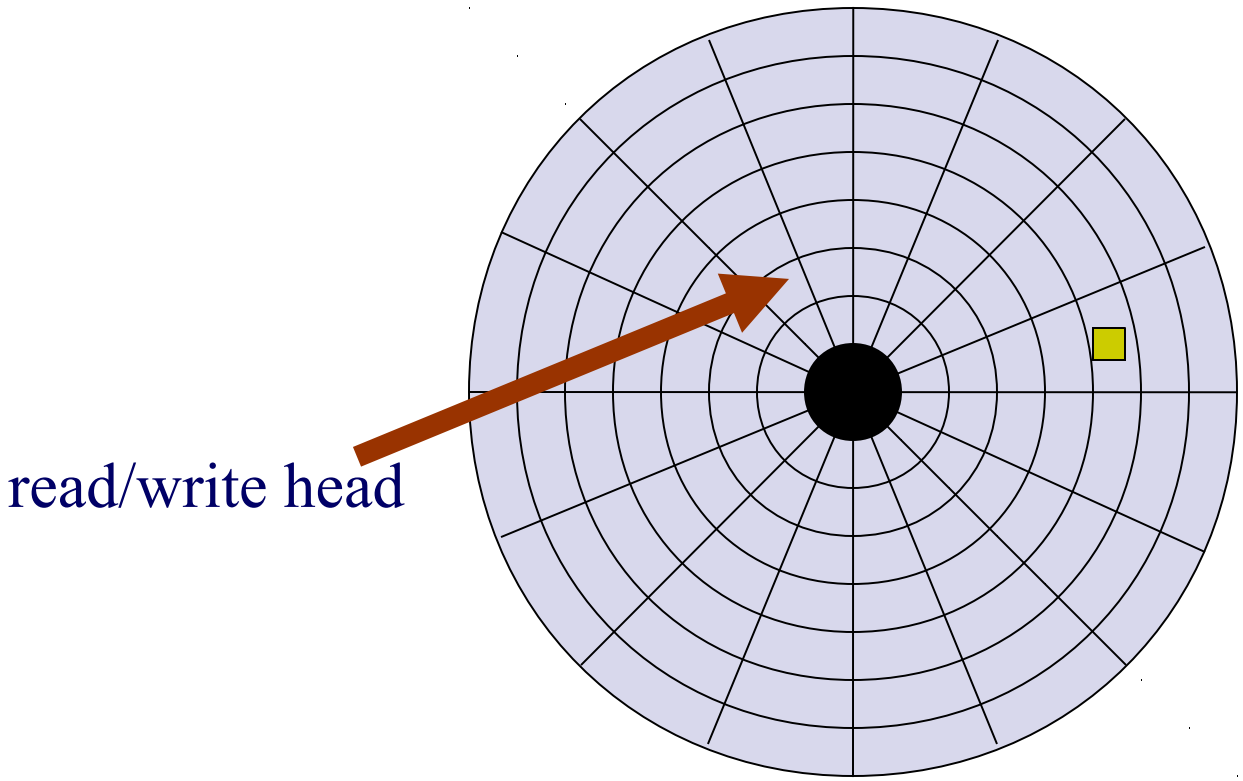
Anatomy of a Hard Drive

Let's read in a sector from the disk



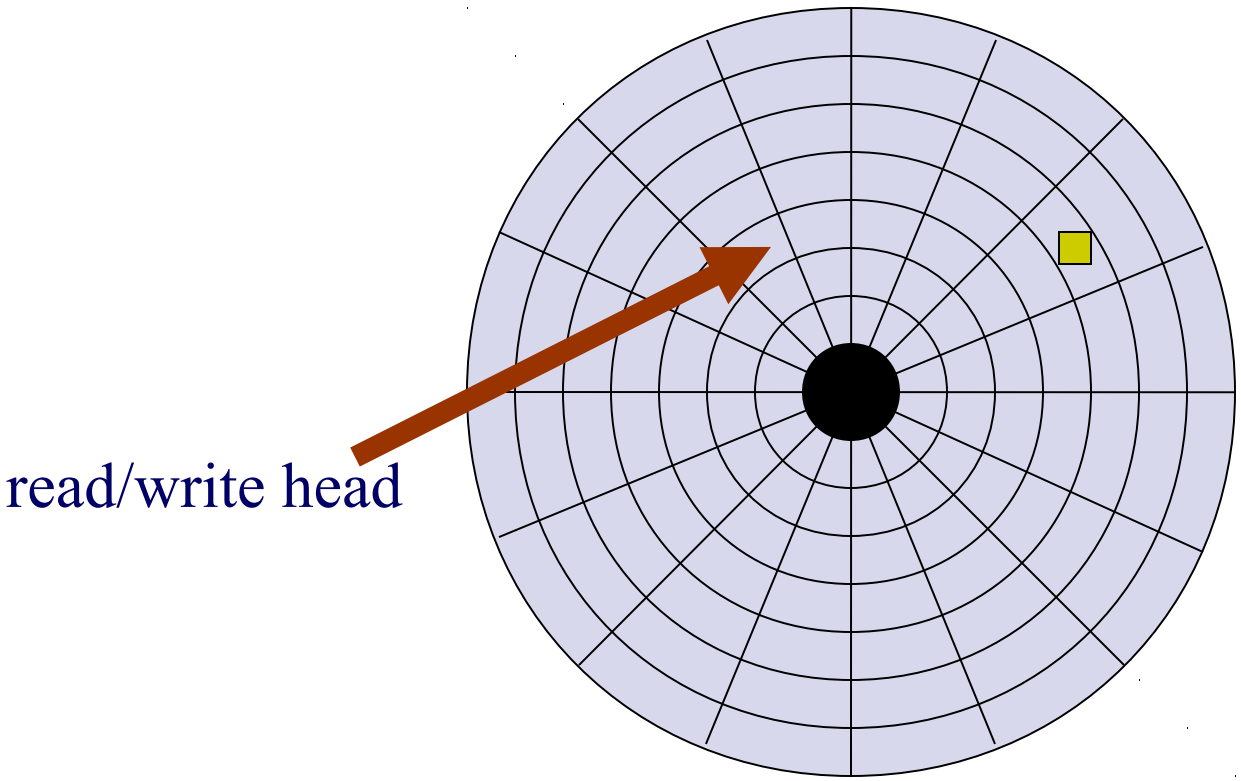
Anatomy of a Hard Drive

Let's read in a sector from the disk



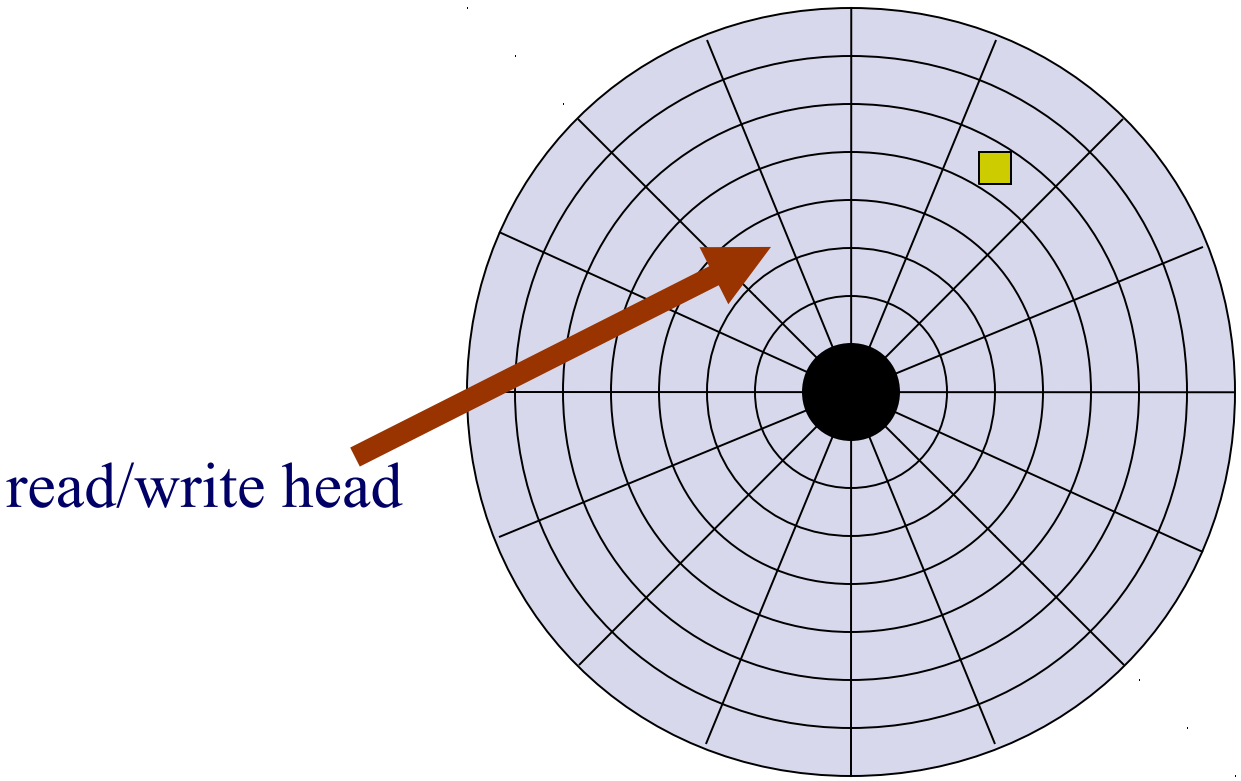
Anatomy of a Hard Drive

Let's read in a sector from the disk



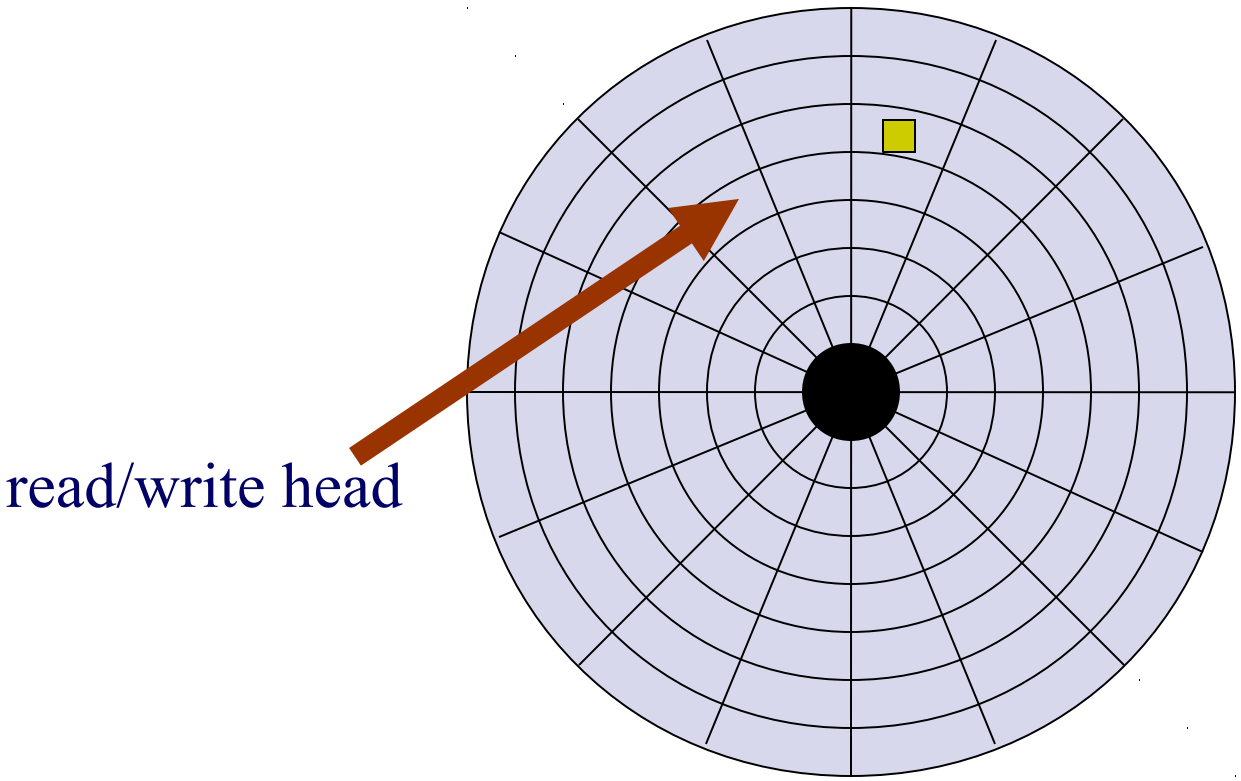
Anatomy of a Hard Drive

Let's read in a sector from the disk



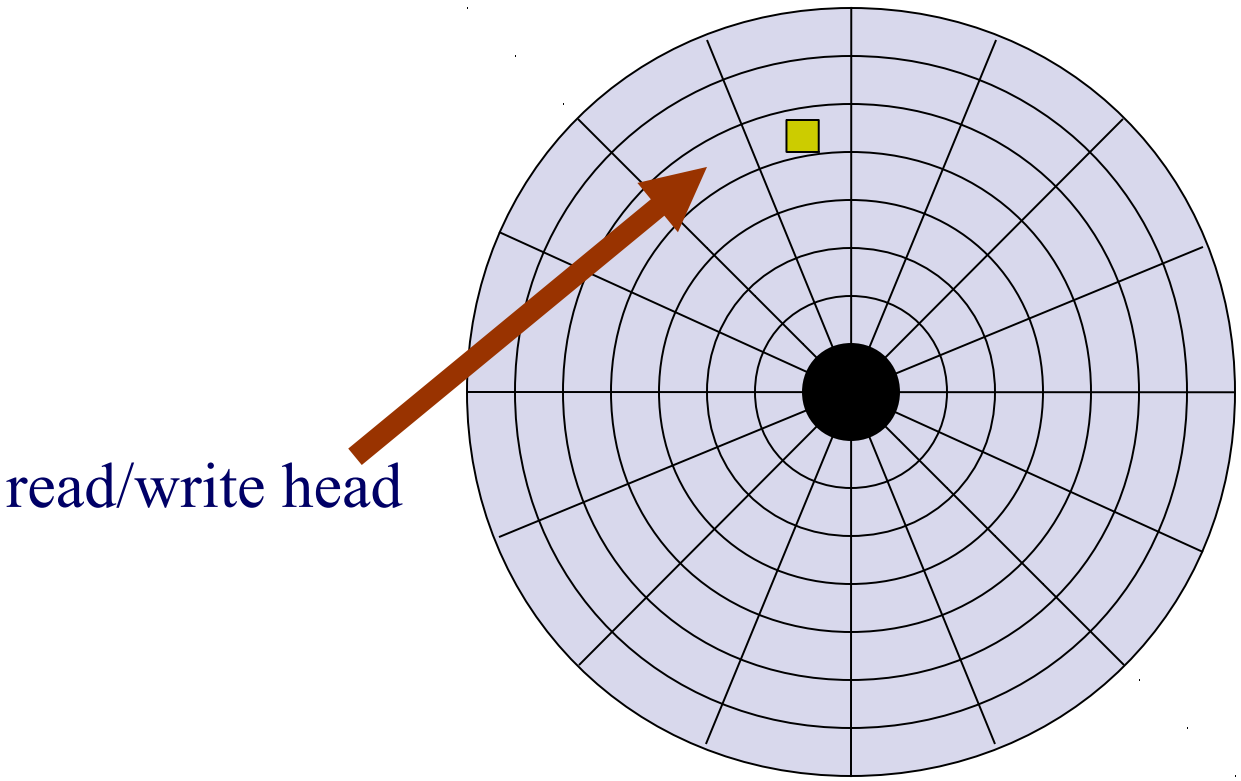
Anatomy of a Hard Drive

Let's read in a sector from the disk



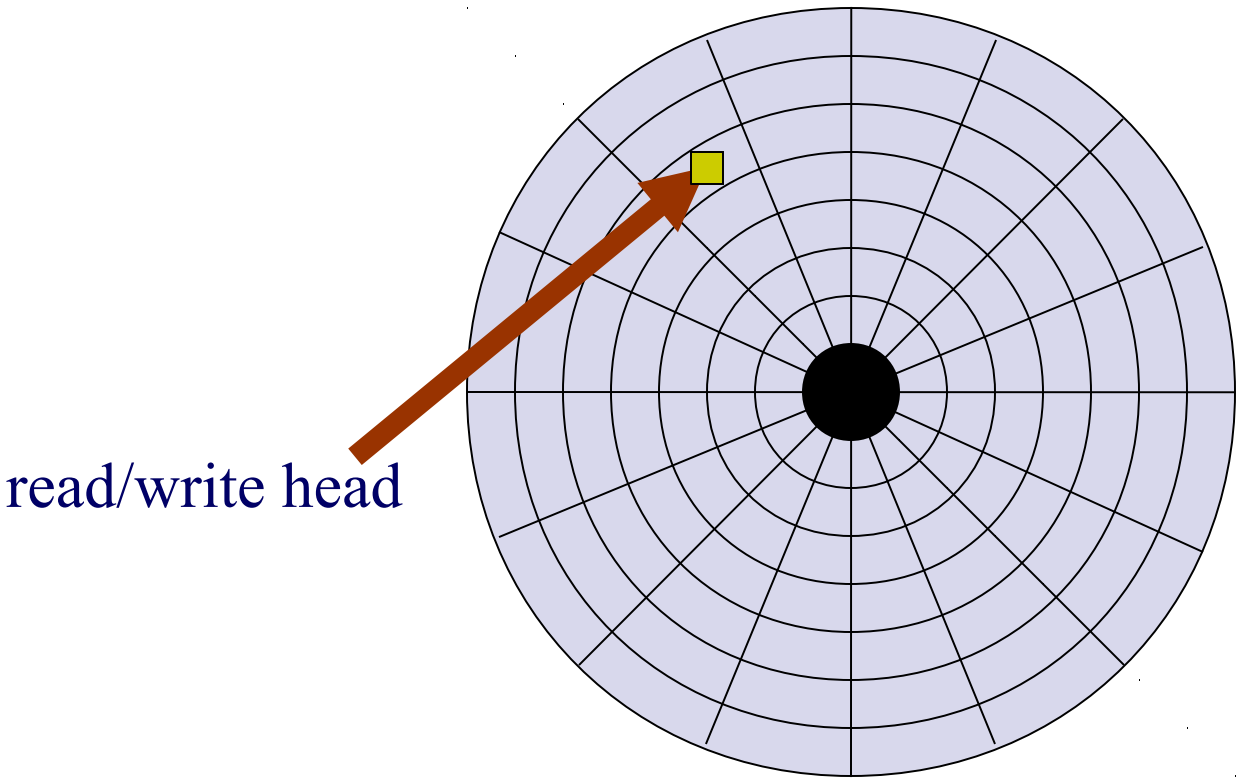
Anatomy of a Hard Drive

Let's read in a sector from the disk



Anatomy of a Hard Drive

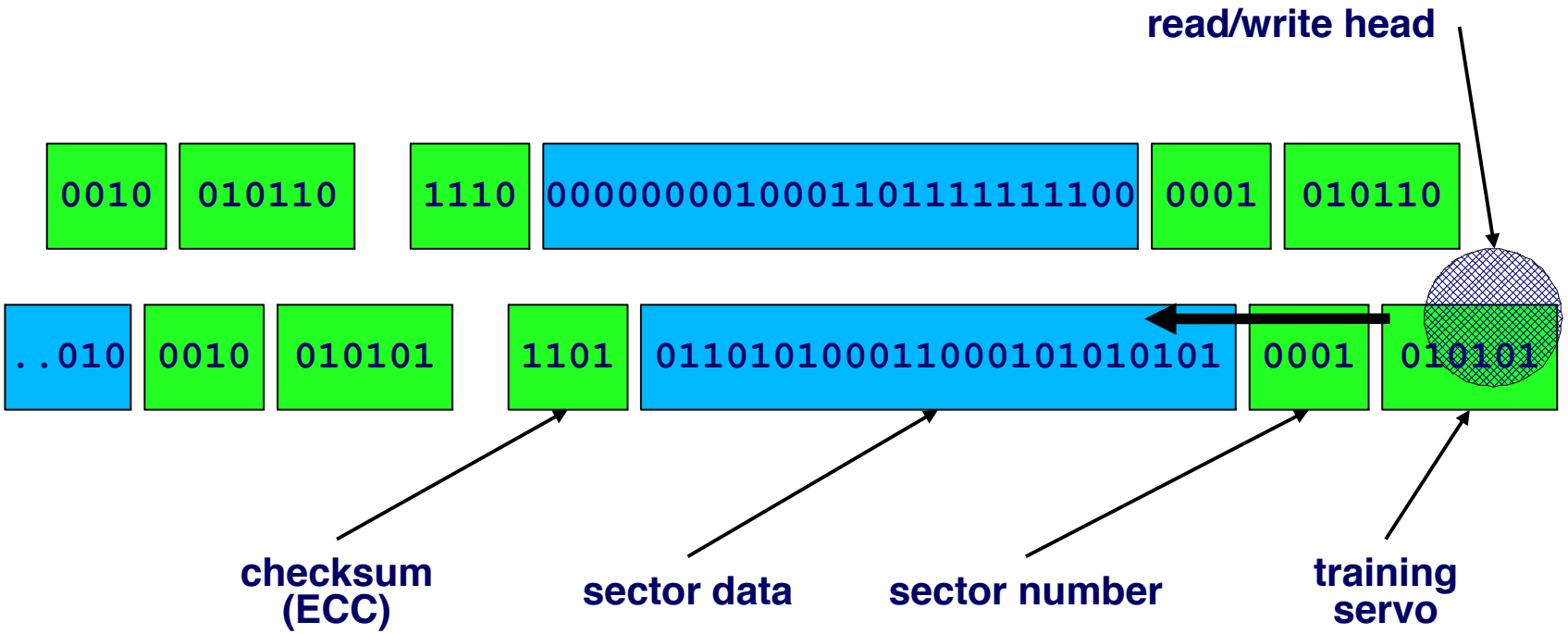
Let's read in a sector from the disk



Anatomy of a "Sector"

Finding a sector involves real work

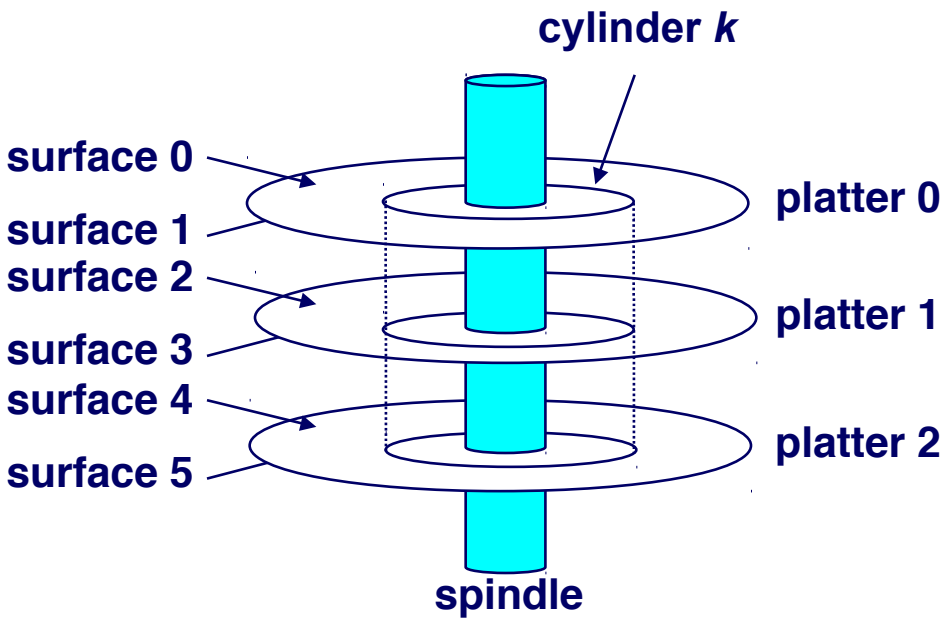
Locate correct track; scan sector headers for number



After sector is read, compare data to checksum

Disk Cylinder

Matching tracks form a cylinder.



Access Within A Cylinder is Faster

Heads share one single arm

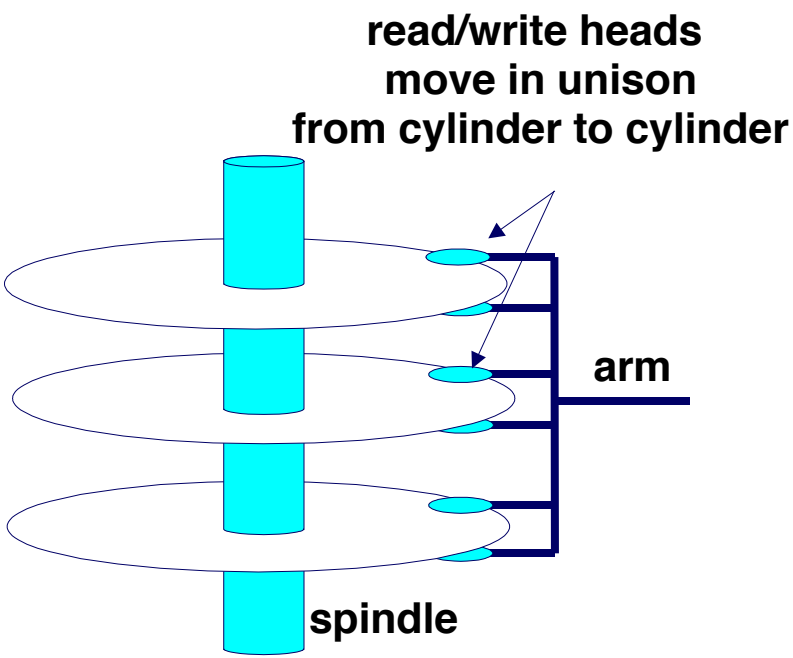
- All heads always on same cylinder
- Active head is aligned, others are "close"

Switching heads is "cheap"

- Deactivate head I, activate J
- Read a few sector headers to fine-tune arm position for J's track

Optimal transfer rate?

1. Transfer all sectors on a track
2. Transfer all tracks on a cylinder
3. *Then* move the arm



Access Time

- **On average, we will have to move the read/write head over one *third* of the tracks**
  - The time to do this is the "average seek time"
    - 5400 rpm: ~10 ms
    - 7200 rpm: ~8.5 ms
- **We will also must wait half a rotation, on average**
  - The time to do this is "average rotational delay"
    - 5400 rpm: ~5.5 ms
    - 7200 rpm: ~4 ms
- **These numbers don't exactly add**
  - While arm moves sideways, disk spins below it

Access Time

Total random access time is ~7 to 20 milliseconds

Access Time

Total random access time is ~7 to 20 milliseconds

- **1000 ms/second, 20 ms/access = 50 accesses/second**

Access Time

Total random access time is ~7 to 20 milliseconds

- 1000 ms/second, 20 ms/access = 50 accesses/second
- 50 ½-kilobyte transfers per second = 25 KByte/sec
- Oh man, disks are slow!
  - That's slower than DSL!!!

Access Time

Total random access time is ~7 to 20 milliseconds

- 1000 ms/second, 20 ms/access = 50 accesses/second
- 50 ½-kilobyte transfers per second = 25 KByte/sec
- Oh man, disks are slow!
  - That's slower than DSL!!!
  - But wait! Disk transfer rates are *hundreds of MBytes*/sec!

Access Time

Total random access time is ~7 to 20 milliseconds

- 1000 ms/second, 20 ms/access = 50 accesses/second
- 50 ½-kilobyte transfers per second = 25 KByte/sec
- Oh man, disks are slow!
  - That's slower than DSL!!!
  - But wait! Disk transfer rates are *hundreds of MBytes*/sec!

What can we, as OS programmers, do about this?

- Read/write more per seek (multi-sector transfers)
  - Disk cache can read ahead and delay/coalesce writes
- Don't seek so randomly
  - Place data near also-relevant data
  - Re-order requests
    - OS may do "disk scheduling" instead of a FIFO queue
    - Historically very important; recently, less
    - (Disks internally schedule too)

Solid-State Disks (SSD)

What is "solid state"?

- Original meaning: "no vacuum tubes"
- Modern meaning: "no moving parts"

Solid-State Disks (SSD)

What is "solid state" storage?

- RAM backed by a battery!
- "NOR flash"
- "NAND flash"
- Things under development

Solid-State Disks (SSD)

What is "solid state" storage?

- RAM backed by a battery!
  - Fast
  - Legato "Prestoserve", 1989 ($8,000 for 1 MB)
  - Allowed NFS servers to complete write RPCs without waiting for disk
- "NOR flash"
  - Word-accessible
  - Writes are slow, density is low
  - Used to boot embedded devices, store configuration
- "NAND flash"
  - Read/write "pages" (512 B), erase "blocks" (16 KB)
  - Most SSDs today are NAND flash
- Things under development
  - "Phase-change" memory (melting)
  - Magnetic RAM
  - "Memristor" memory

Solid-State Disks (SSD)

Architectural features of NAND flash

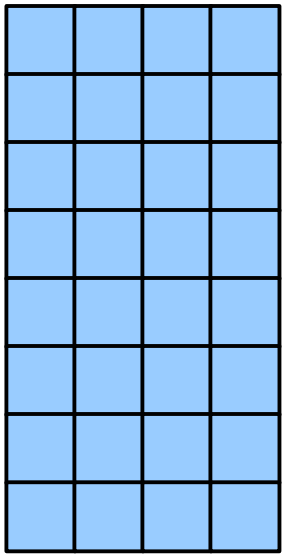
- No moving parts means no “seek time” / “rotational delay”
- Read is faster than write
- Write and “erase” are different
  - A blank page can be written to (once)
  - A written page must be erased before rewriting
  - But pages can't be individually erased!
    - “Erase” works on multi-page *blocks* (16 KB)
    - “Erase” is very slow
    - “Erase” *damages the block* each time

Implications

- “Write amplification”
- “Wear leveling”

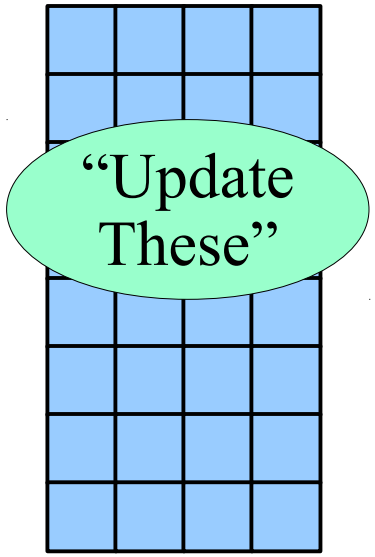
“Write Amplification”

Goal: update 8 pages (4 KB) in a block (16 KB)



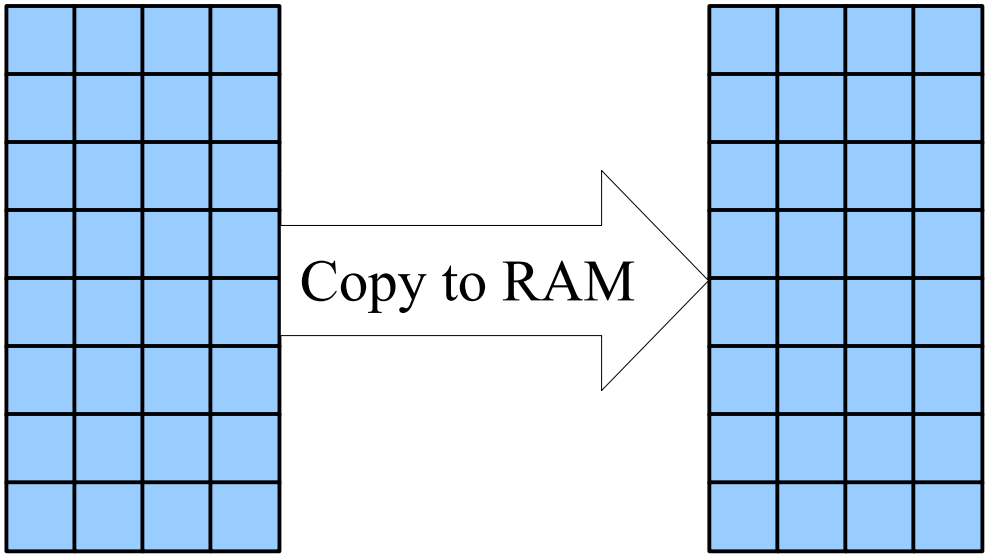
“Write Amplification”

Goal: update 8 pages (4 KB) in a block (16 KB)



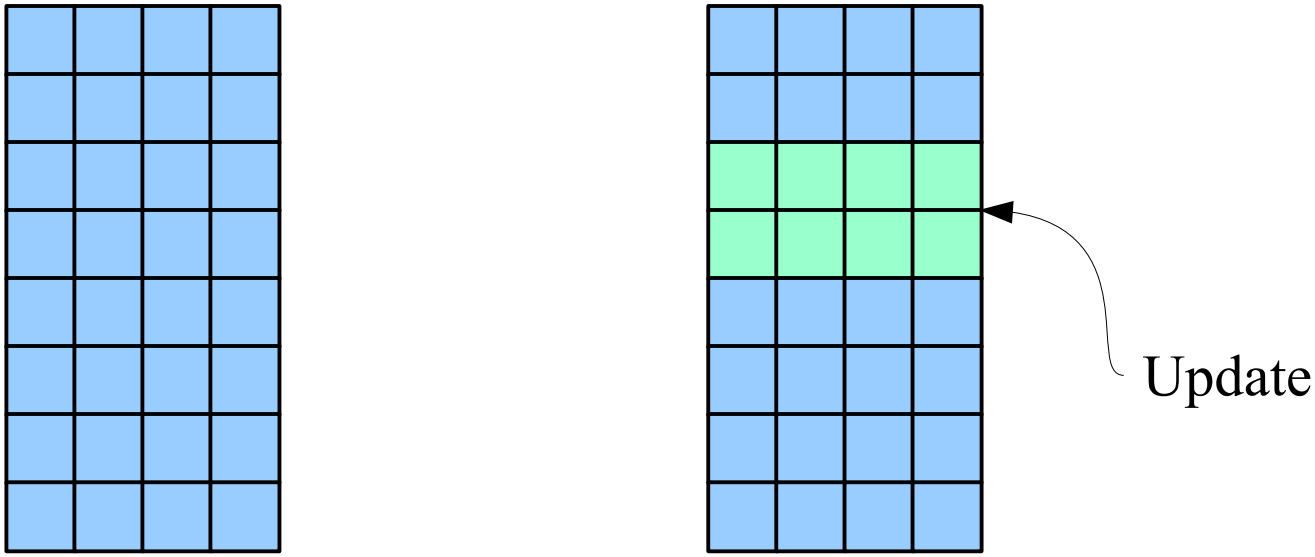
“Write Amplification”

Goal: update 8 pages (4 KB) in a block (16 KB)



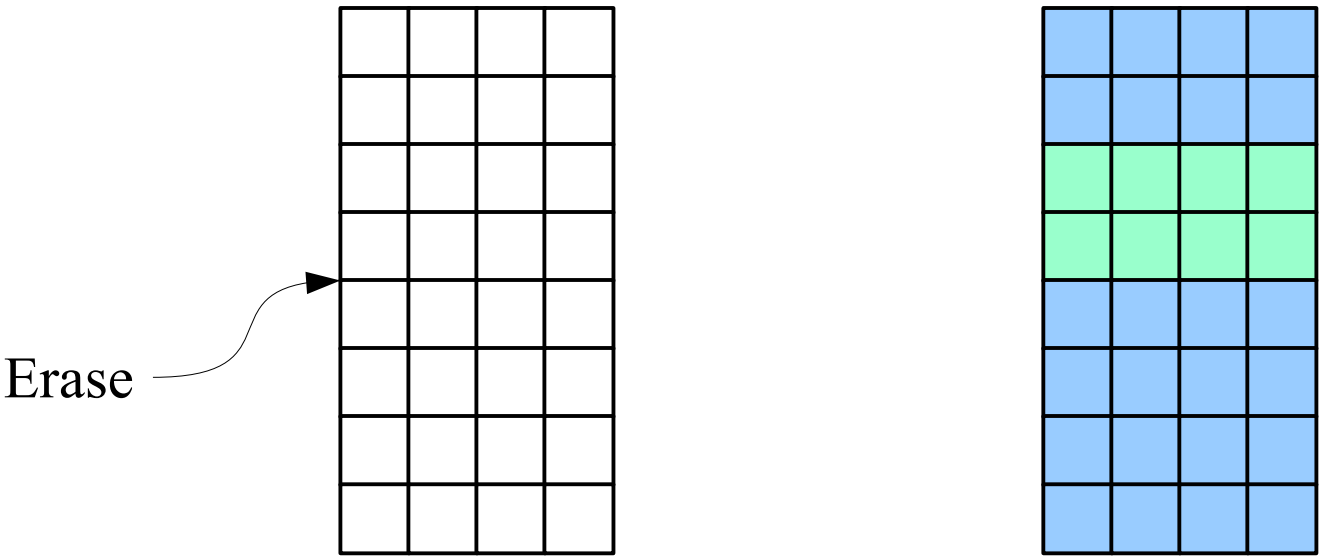
“Write Amplification”

Goal: update 8 pages (4 KB) in a block (16 KB)



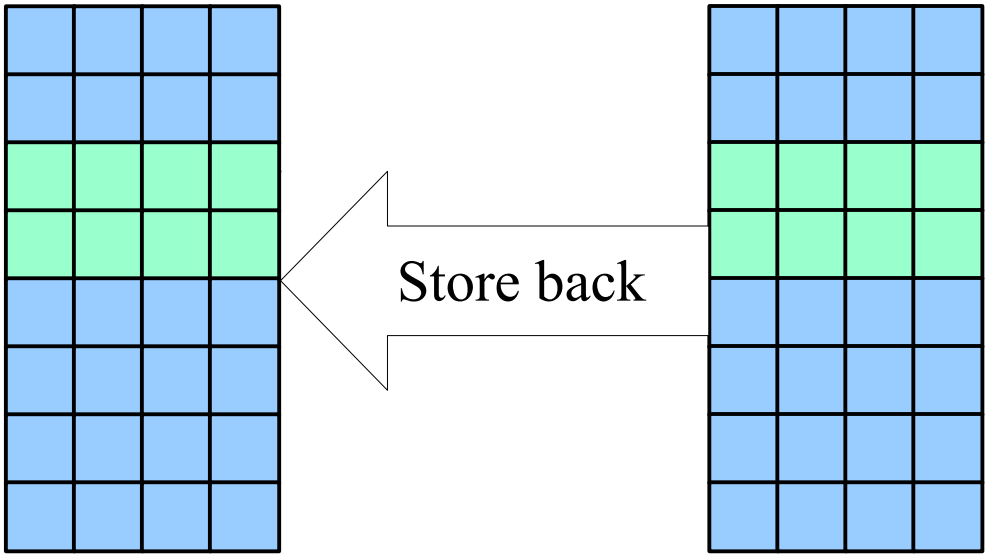
“Write Amplification”

Goal: update 8 pages (4 KB) in a block (16 KB)



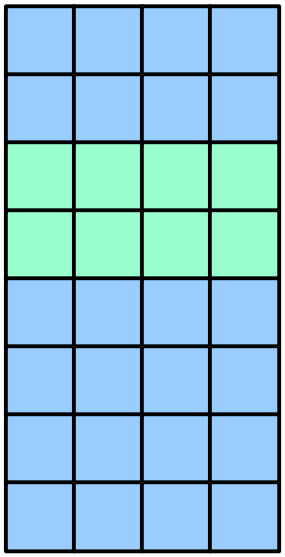
“Write Amplification”

Goal: update 8 pages (4 KB) in a block (16 KB)



"Write Amplification"

Goal: update 8 pages (4 KB) in a block (16 KB)



Result

- Logical: wrote 4 KB
- Physical: erased and write 16 KB
- "Amplification factor": 4
  - Why do we care? Device will wear out 4X faster!

Hot-Spot Wear

The bad case

- File systems like to write the same block repeatedly
- Erasing damages part of the flash
  - ~10,000 erases destroys a block

Strategy: ?

Managing - Wear Leveling

The bad case

- File systems like to write the same block repeatedly
- Erasing damages part of the flash
  - ~10,000 erases destroys a block

Strategy: lie to the OS!

- Host believes it is writing to specific “disk blocks” - LBA
- Store the information somewhere else!
  - Secretly re-map host address onto NAND address
  - FTL - “flash translation layer”

Managing - Wear Leveling

The bad case

- File systems like to write the same block repeatedly
- Erasing damages part of the flash
  - ~10,000 erases destroys a block

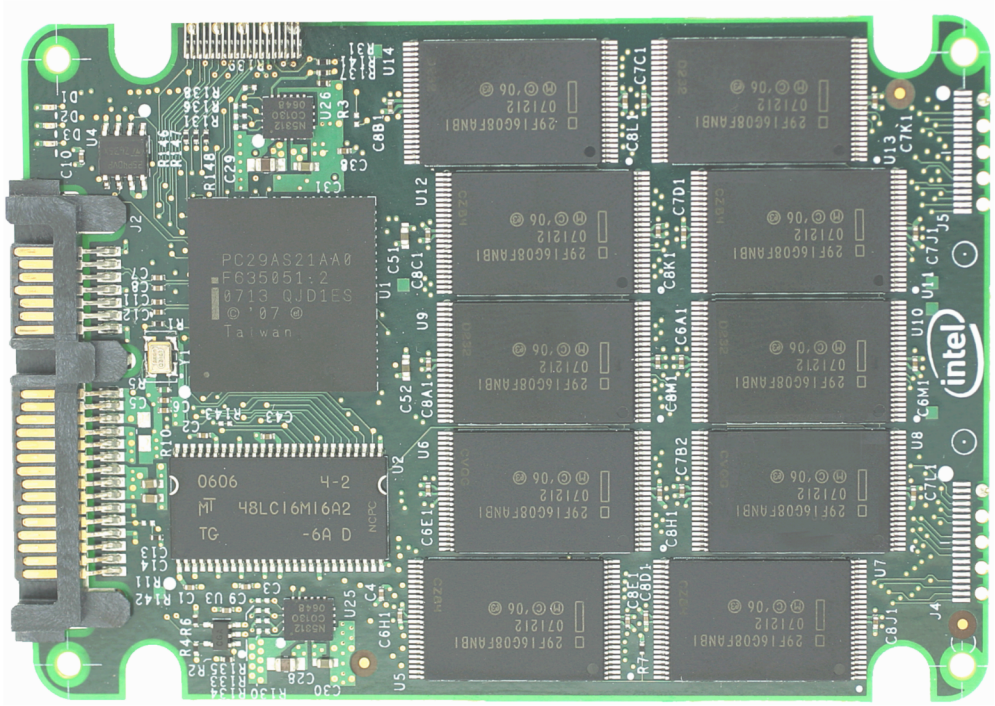
Strategy: lie to the OS!

- Host believes it is writing to specific "disk blocks" - LBA
- Store the information somewhere else!
  - Secretly re-map host address onto NAND address
  - FTL - "flash translation layer"
- Each part of the "disk" moves from one part of the flash to another over time
- "Over-provision"
  - Promise less space than there really is
  - Spare space to replace worn-out blocks
- Use up overprovisioning as blocks wear out
  - Device eventually gets slower and then fails

Wear Leveling - FTL

FTL is a *computer*

- **CPU, RAM**
- **Access to lots of flash for code & data structures & user data**



Managing - Write Amplification

The bad case

- **Small random writes**

Strategy: ?

Managing - Write Amplification

The bad case

- Small random writes

Strategy: lie to the OS!

- Host believes it is writing to specific “disk blocks” - LBA
- Store the information somewhere else!
  - Secretly re-map host address onto NAND address
  - FTL - “flash translation layer”
- Group multiple small writes into full blocks
  - Write at sequential write rates
- To update a “disk block”, store a new copy *somewhere else*
  - Leaves “holes” in other blocks (stale old block versions)
  - At some point, “clean out” the holes by reading a bunch of old blocks and writing back a smaller number of whole pages
- Rate of cleaning depends amount of unallocated space
  - Controller reserves X% hidden space (ie. 10, 20, 50%)

SSD Summary

SSD vs. disk

- ☺ SSD's implement “regular disk” model
  - LBA sectors
  - Write-sector, read-sector, “park heads”, etc.
- ☺ Read operations are extremely fast (100X faster), no “seek time” or “rotational delay” (every sector is “nearby”)
- ? Write operations “vary widely” (maybe 100X faster, maybe not faster at all)
- ☺ SSD's use less power than actual disks (~1/5?)
- ☺ SSD's are shock-resistant
- ☹ Writing to an SSD wears it out much faster than a disk
- ☹ SSD's are *expensive* (20X or more)

SSD Summary

Opportunity & threat

- "TRIM" command speeds up writes!
  - "Dear FTL, logically zero-fill these blocks"
- "Securely erase disk" may or may not be possible

The future?

- Lots more SSD's
- Lots more disks too
- Hybrid systems to take advantage of best features of both

What You Should Know

Storage is *slow*

- Whatever you want to do takes *milliseconds*

Storage lies

- You get some number of "disk blocks"
- There is no way to know where on the "disk" they are
- LBA is a faint approximation of proximity

Failure model

- Sometimes a read fails (sorry!)
- Writing to that block will cause the device to re-map
  - Both spinning-disk and SSD
- When re-map space is exhausted, device refuses to write

Security

- Actually erasing information from flash is uncertain
- Suggestion: encrypt

Summary

What is “storage”?

Anatomy of a Hard Drive

SSD's

What you should know

Further Reading

Reliably Erasing Data from Flash-based Solid State Drives

Wei et al., UCSD

FAST '11

http://www.usenix.org/legacy/events/fast11/tech/full_papers/Wei.pdf

A Conversation with Jim Gray

Dave Patterson

ACM Queue, June 2003

http://queue.acm.org/detail.cfm?id=864078

Terabyte Territory

Brian Hayes

American Scientist, May/June 2002

http://www.americanscientist.org/issues/pub/terabyte-territory