15-410

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

Disks
March 14, 2011

Dave Eckhardt & Garth Gibson

Brian Railing & Steve Muckle

Contributions from

- **Eno Thereska, Rahul Iyer**
- **15-213**
- **"How Stuff Works" web site**

Overview

Anatomy of a Hard Drive

Common Disk Scheduling Algorithms

A brief mention of SSD's

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, etc)

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*

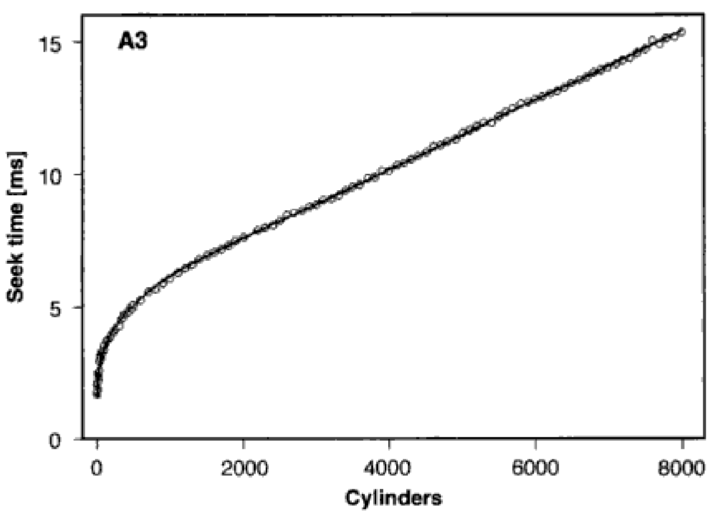
Taken from "How Hard Disks Work"
http://computer.howstuffworks.com/hard-disk2.htm

Anatomy of a Hard Drive

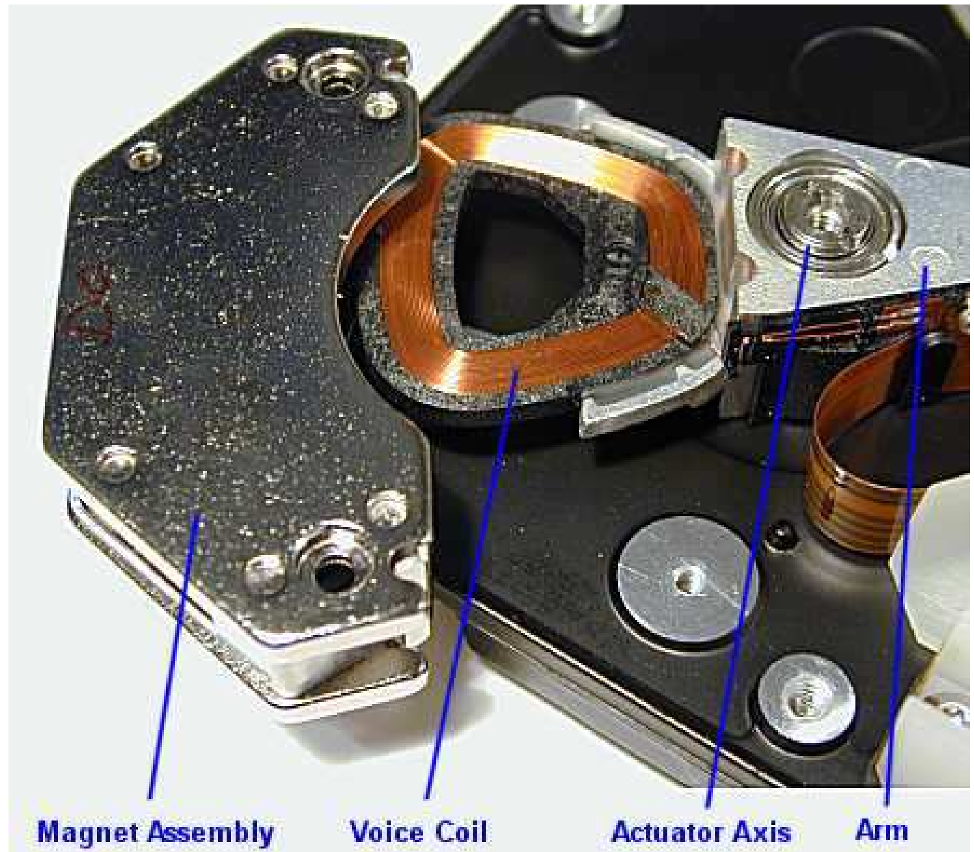
The arm is moved by a voice coil actuator

Slow, as computers go

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



Oklobdzija, Comp. Eng. Handbook, 2002



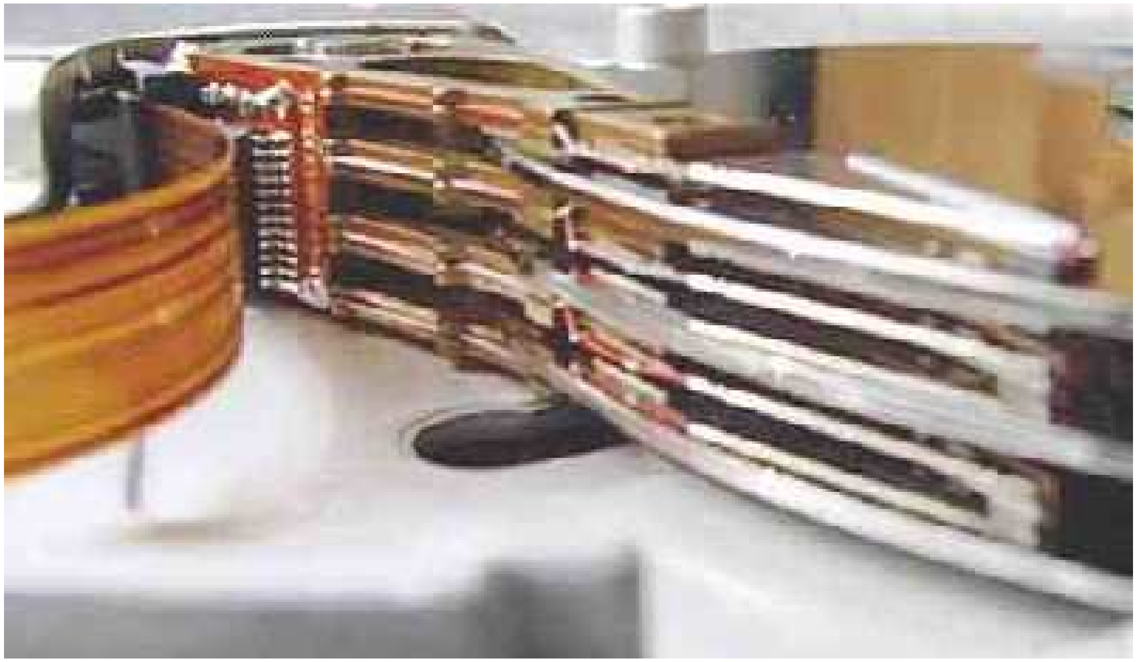
Taken from “Hard Disk Drives”
http://www.pcguide.com/ref/hdd

Anatomy of a Hard Drive

Both sides of each platter store information

Each side of a platter is called a *surface*

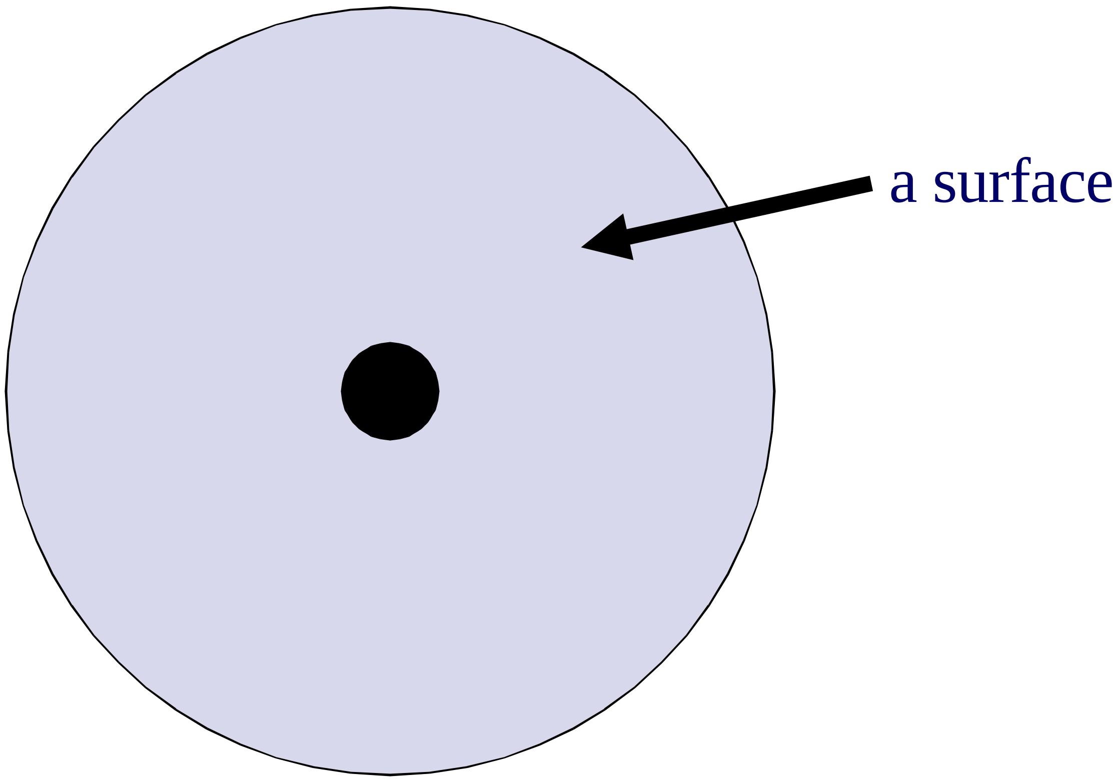
Each surface has its own read/write head



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

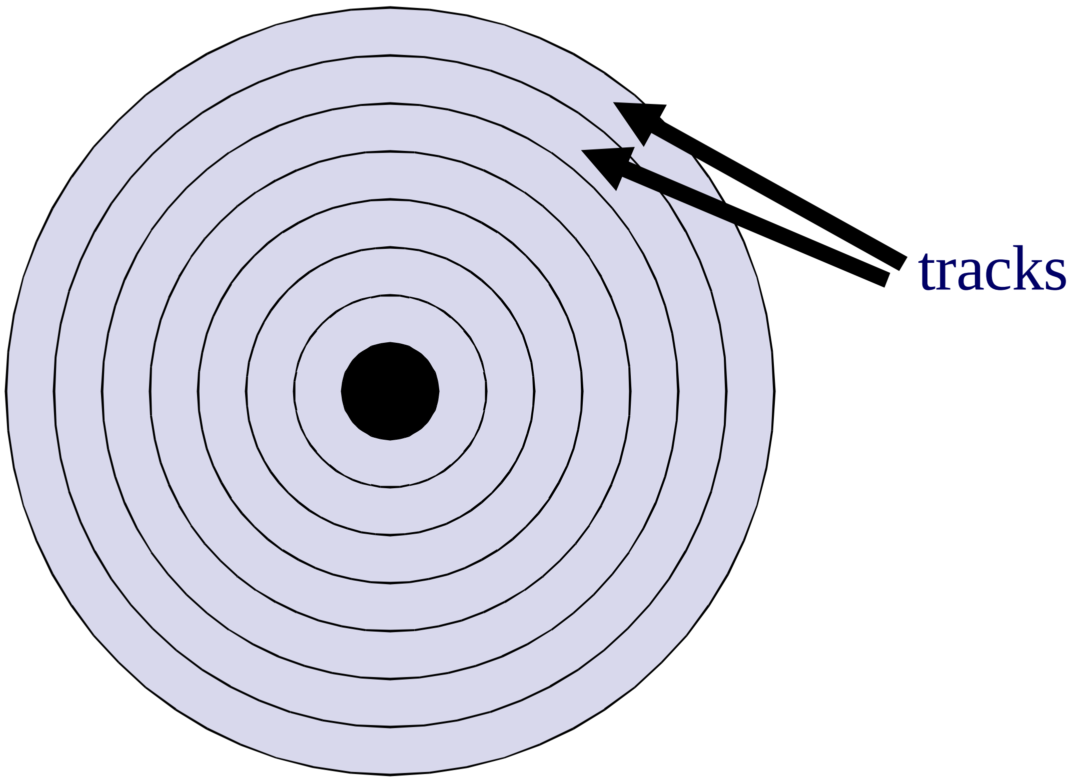
Anatomy of a Hard Drive

How are the surfaces organized?



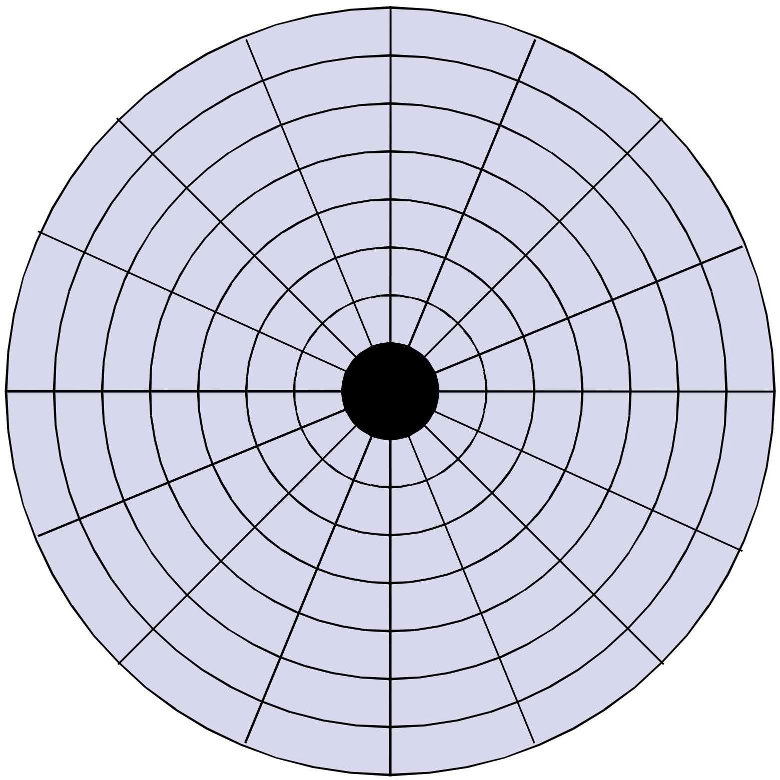
Anatomy of a Hard Drive

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



Anatomy of a Hard Drive

These tracks are further divided into *sectors*



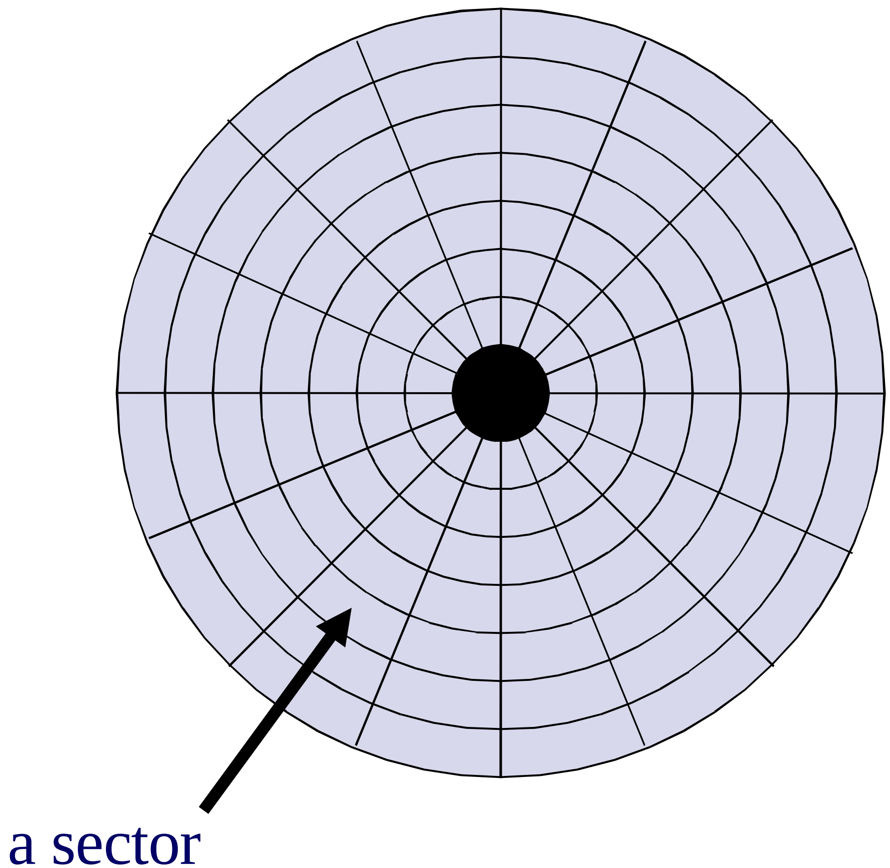
Anatomy of a Hard Drive

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!)**

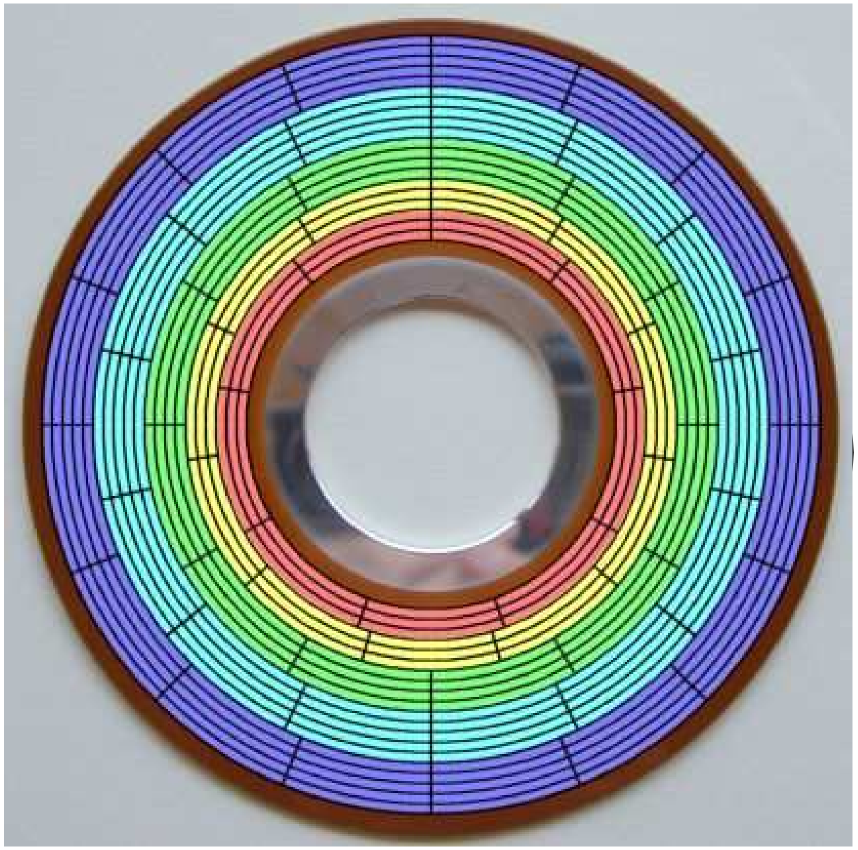
Gee, those outer sectors look bigger...?



Anatomy of a Hard Drive, Really

Modern hard drives use *zoned bit recording*

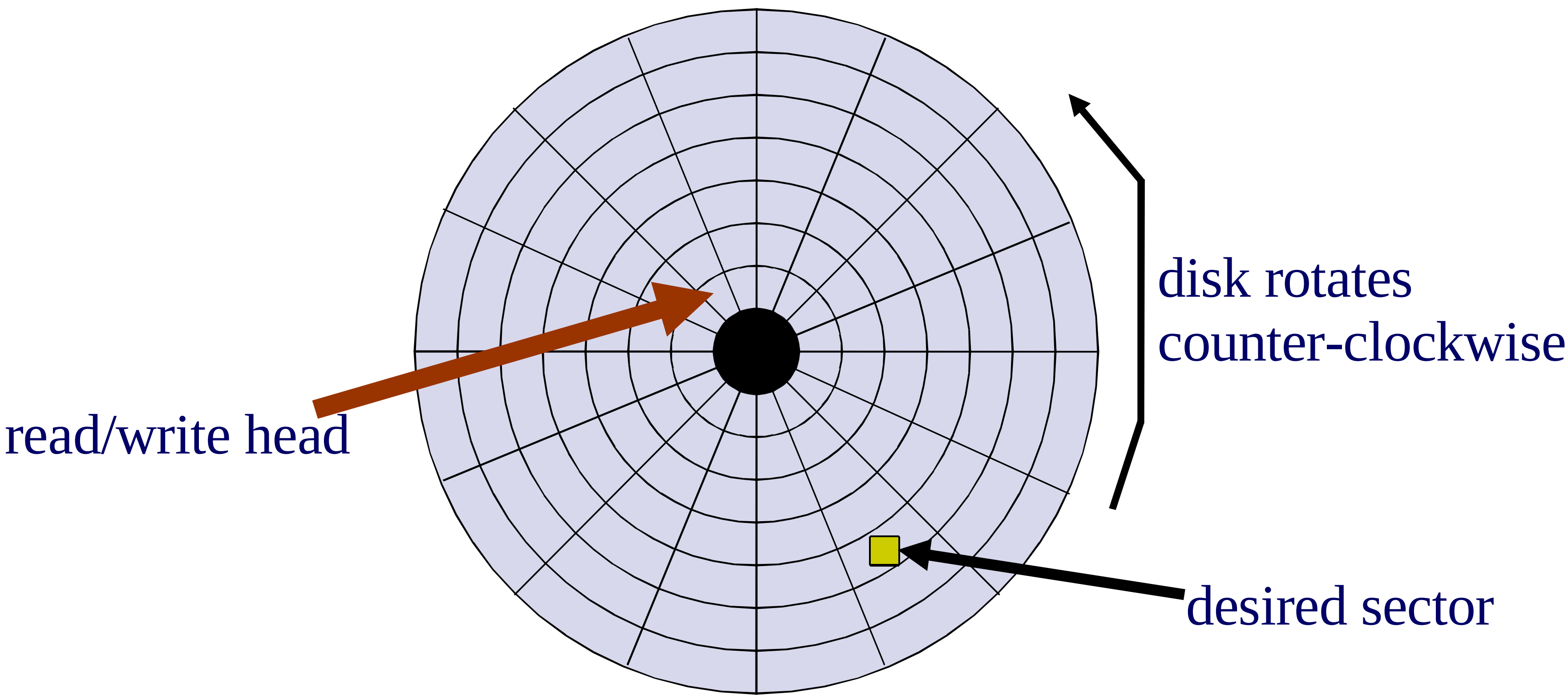
- **Disk has tables to map track# to #sectors**
- **Sectors are all roughly the same linear length**
- **Some old low-level code still thinks of "C/H/S addressing", but that's not what actually happens (for more than a decade!)**
  - **"sector address" names a sector like "page" names a frame**



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")

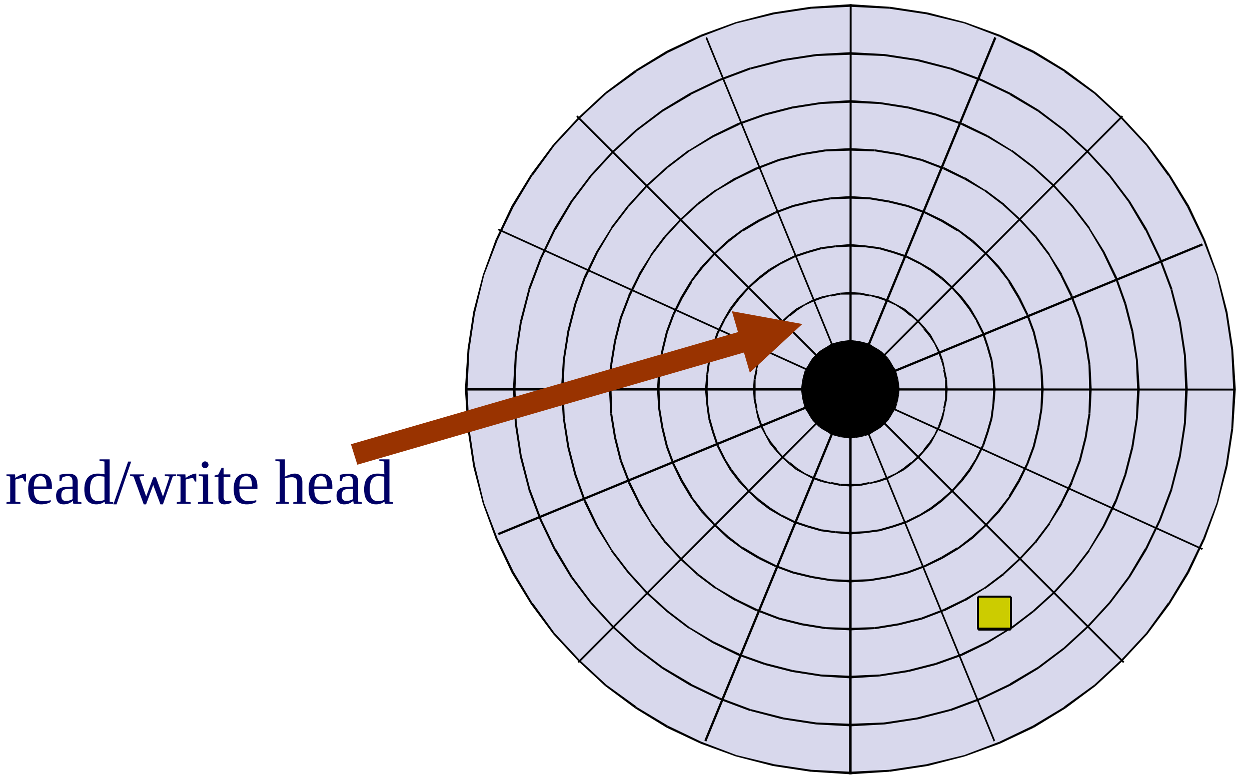
2. Wait until the desired sector spins around ("rotational delay"/"rotational latency")

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**

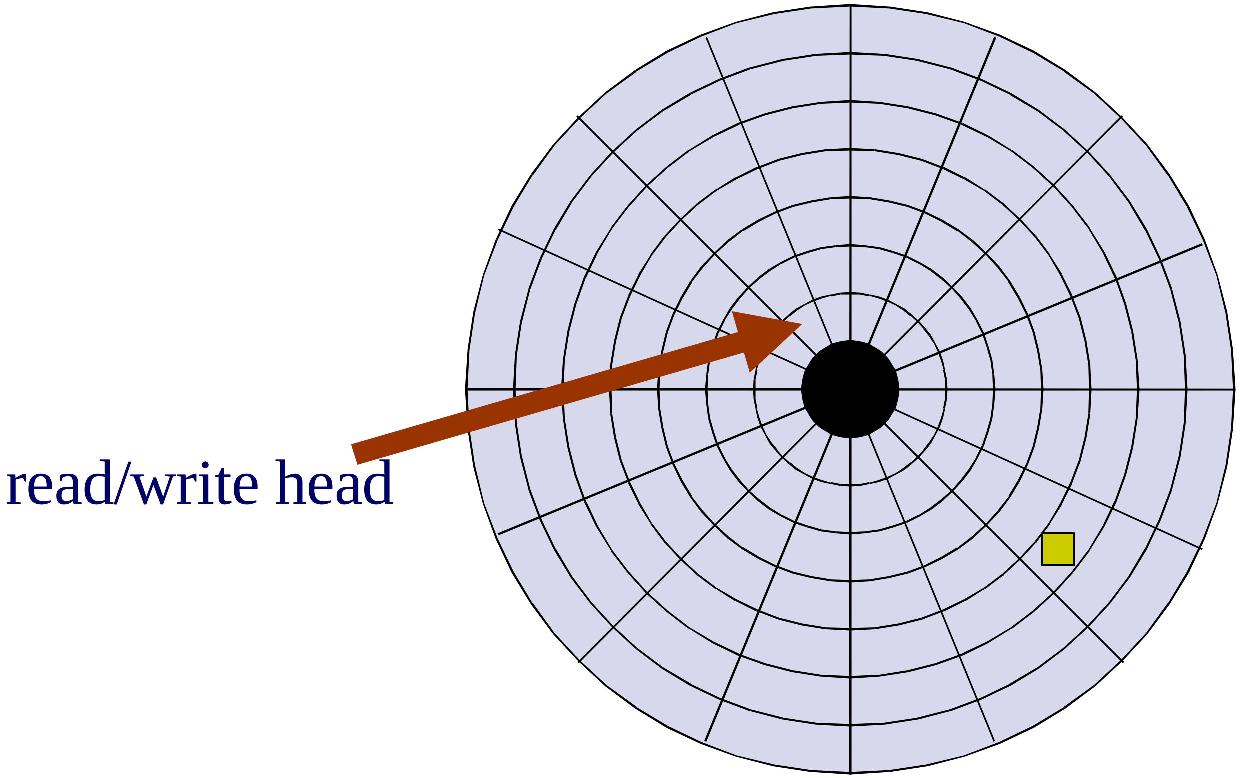
Anatomy of a Hard Drive

Let's read in a sector from the disk



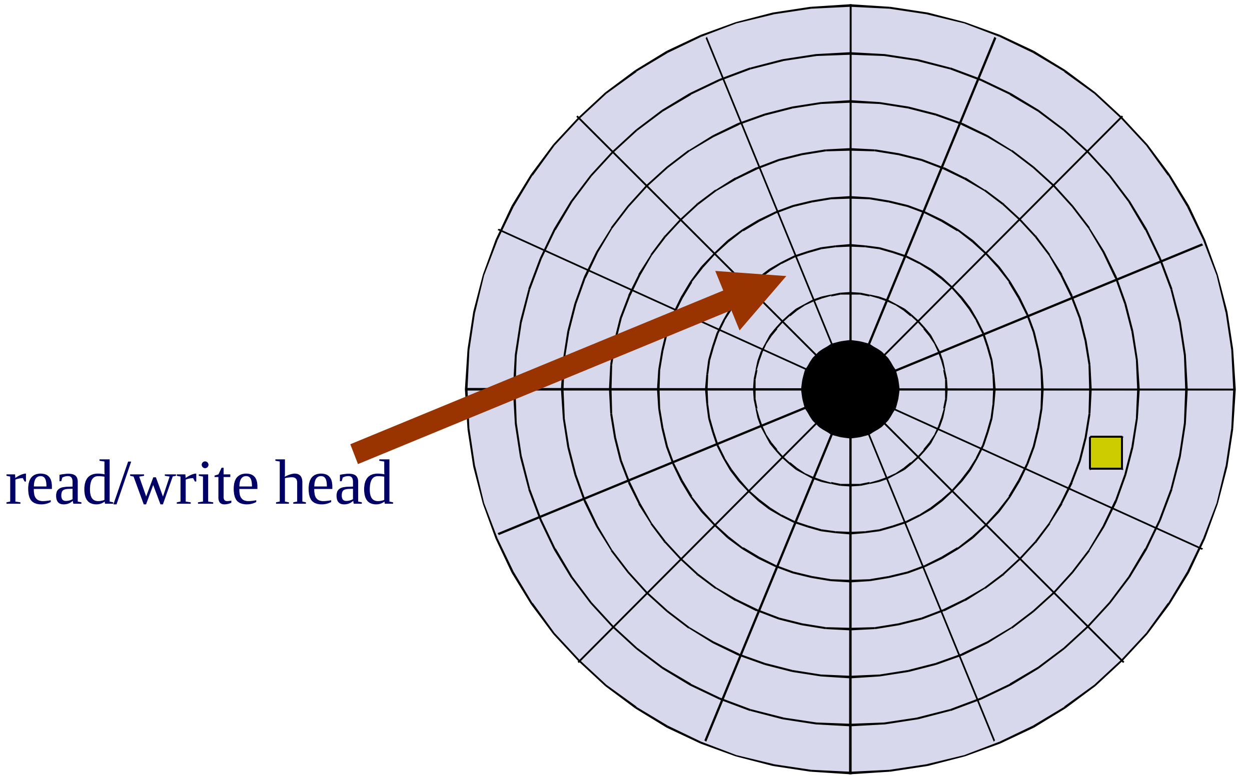
Anatomy of a Hard Drive

Let's read in a sector from the disk



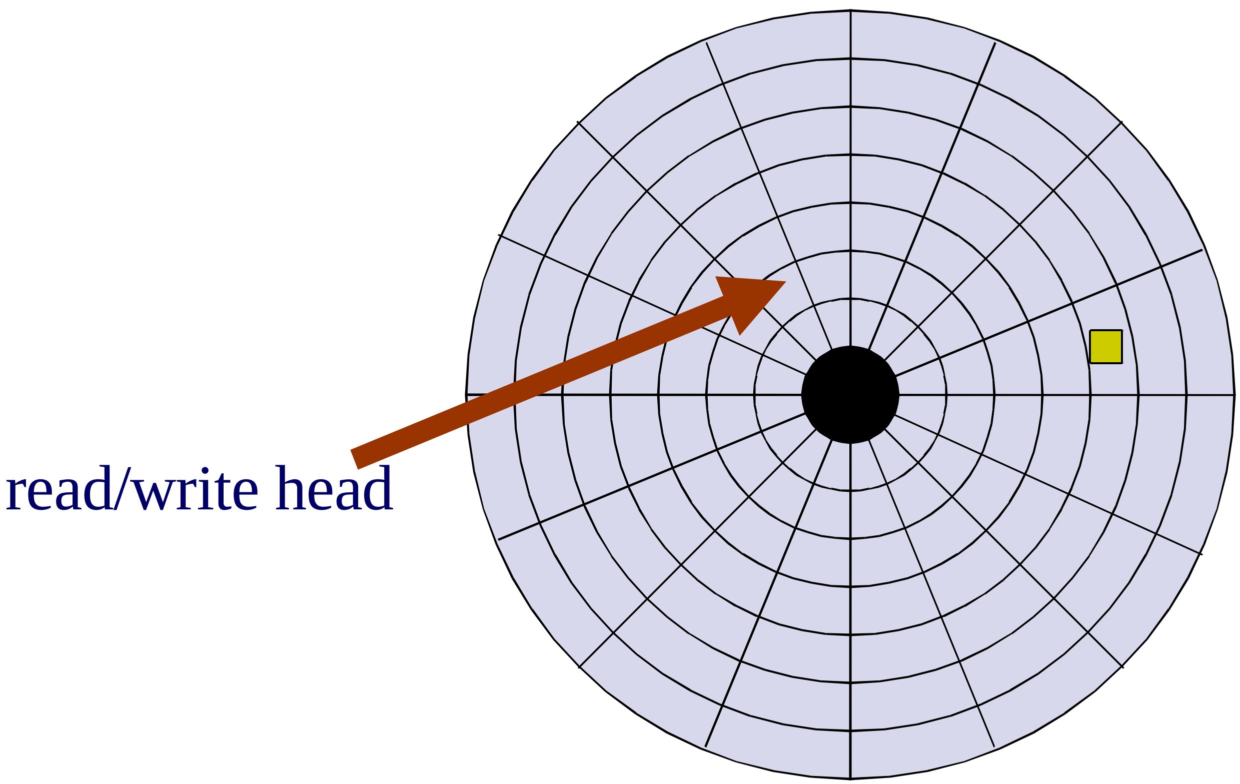
Anatomy of a Hard Drive

Let's read in a sector from the disk



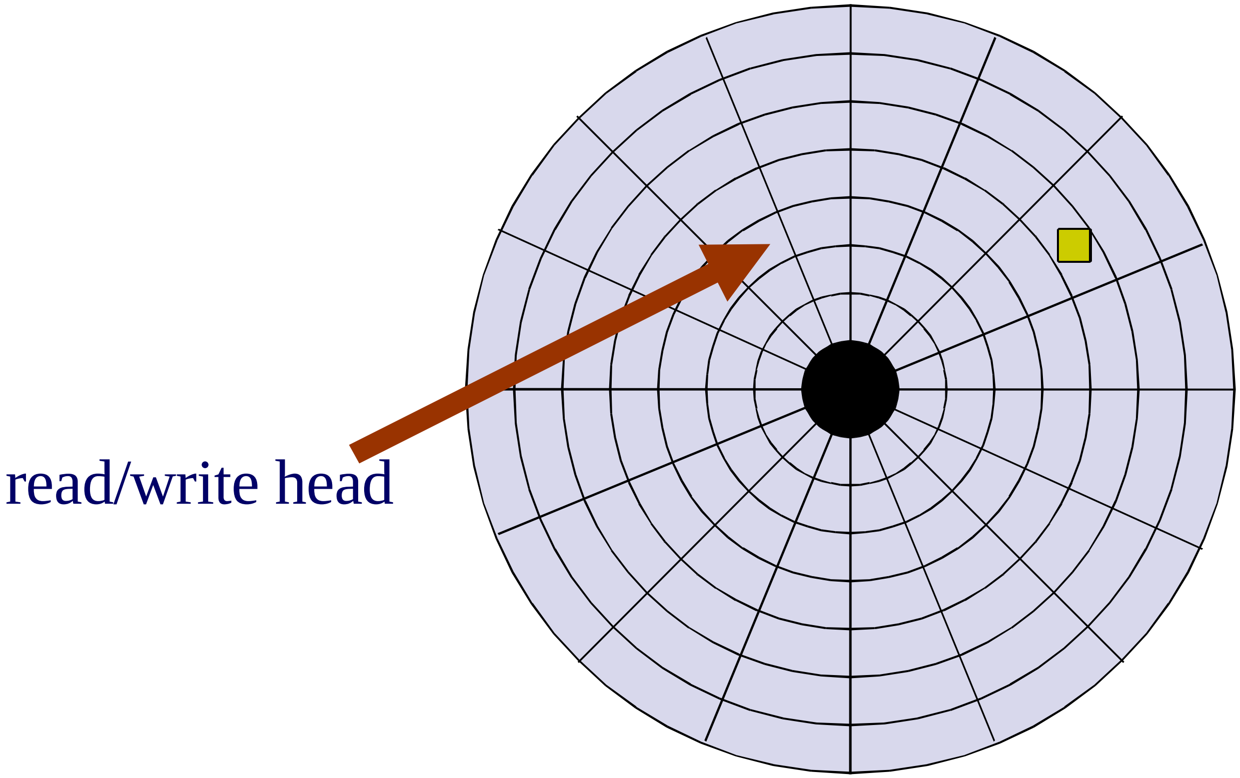
Anatomy of a Hard Drive

Let's read in a sector from the disk



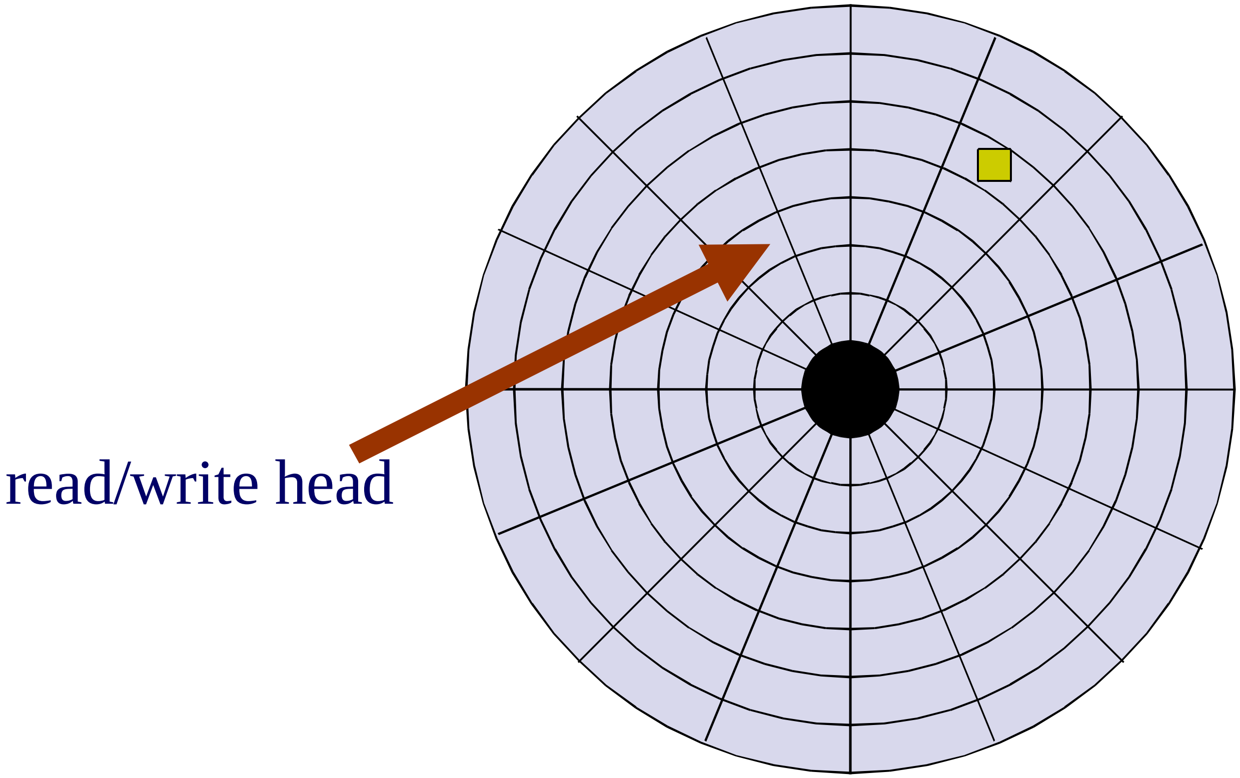
Anatomy of a Hard Drive

Let's read in a sector from the disk



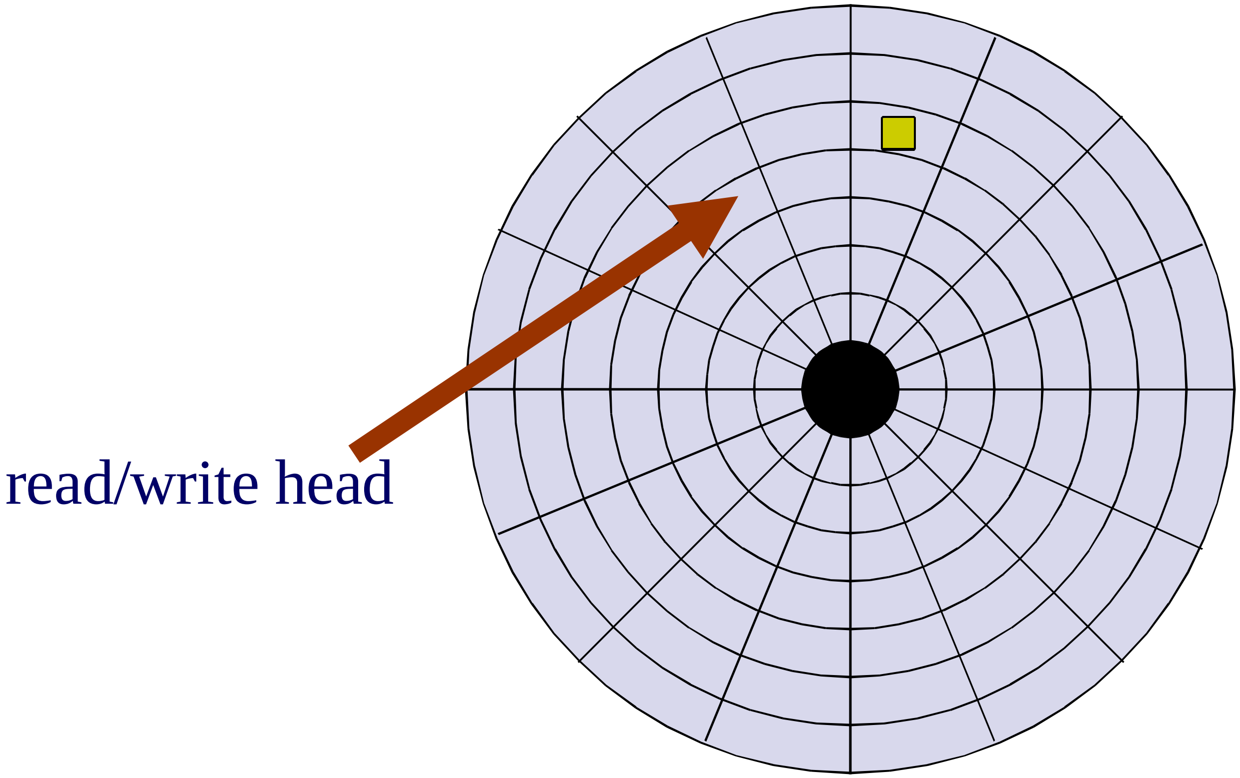
Anatomy of a Hard Drive

Let's read in a sector from the disk



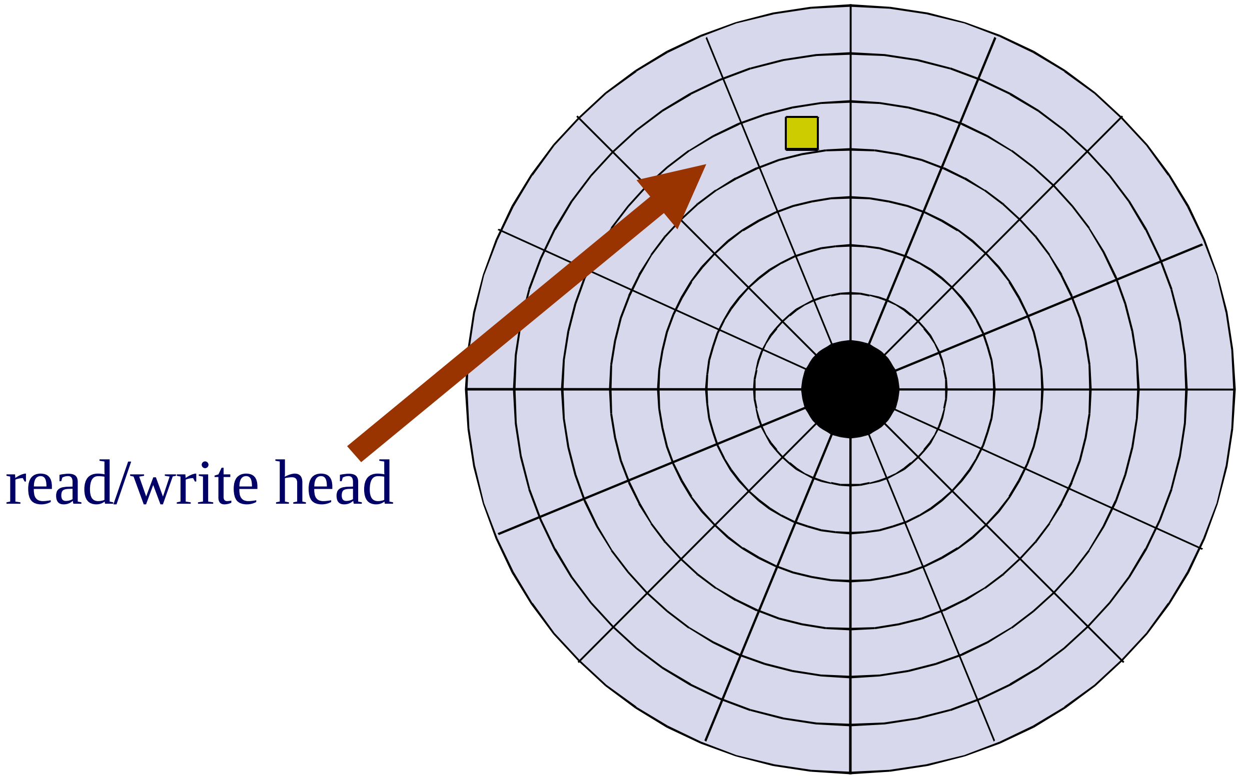
Anatomy of a Hard Drive

Let's read in a sector from the disk



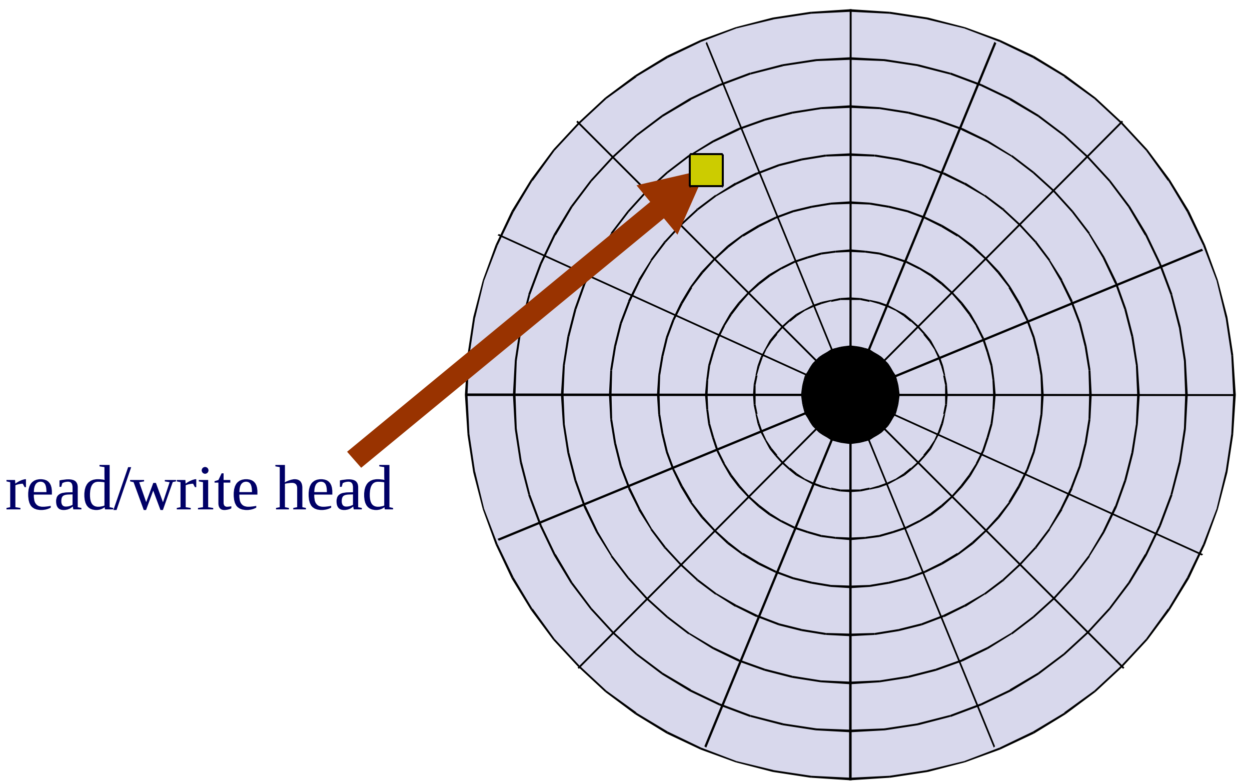
Anatomy of a Hard Drive

Let's read in a sector from the disk



Anatomy of a Hard Drive

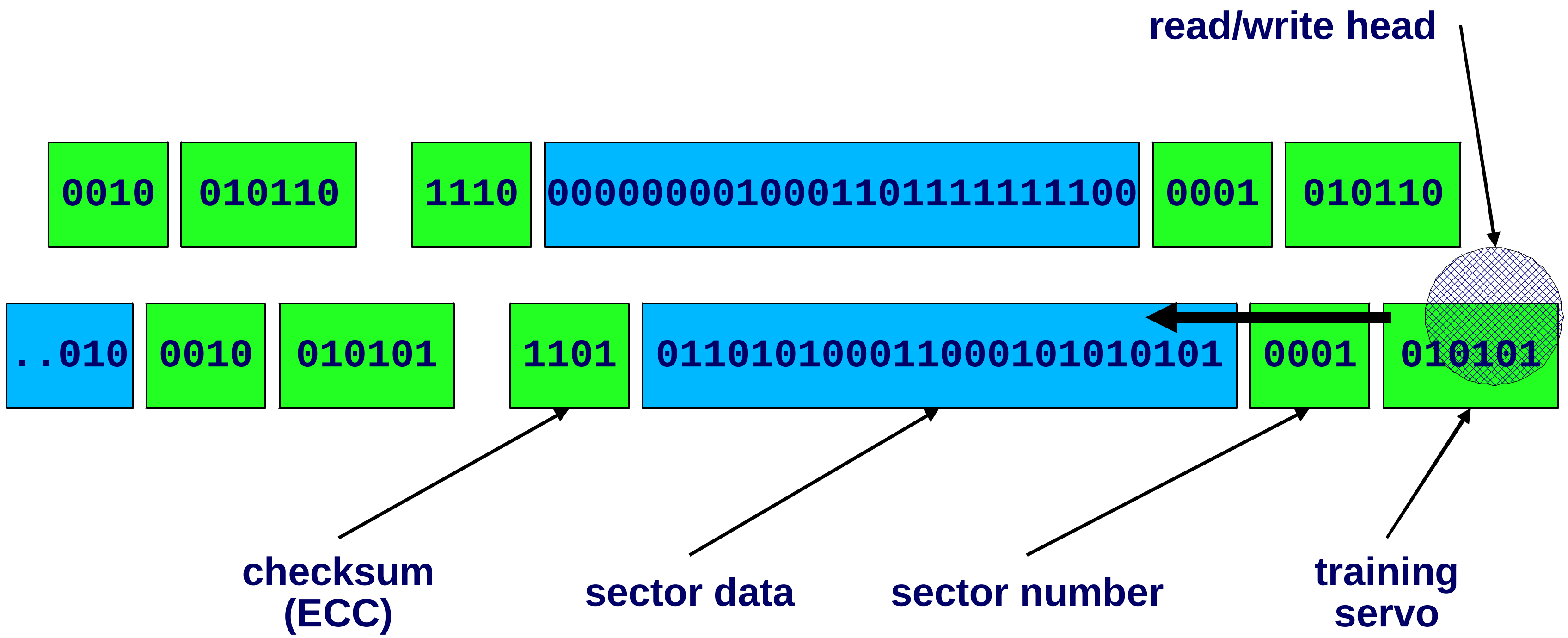
Let's read in a sector from the disk



Anatomy of a "Sector"

Finding a sector involves real work

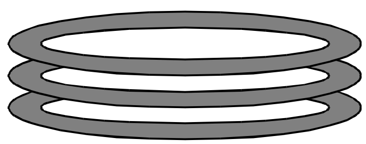
- Locate correct track; scan sector headers for number



After sector is read, compare data to checksum

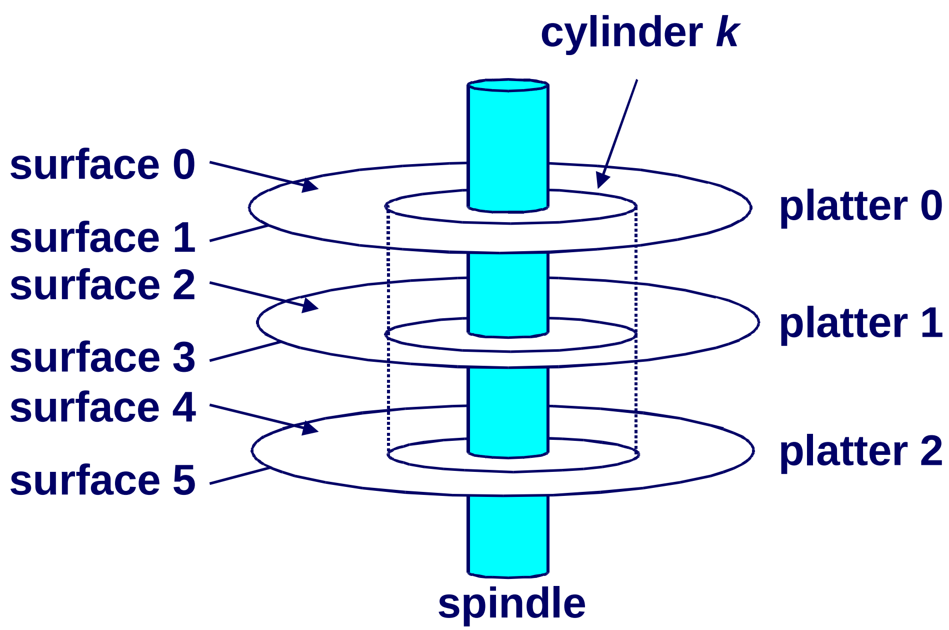
Disk Cylinder

Matching tracks across surfaces are collectively called a *cylinder*



Disk Cylinder

Matching tracks form a cylinder.



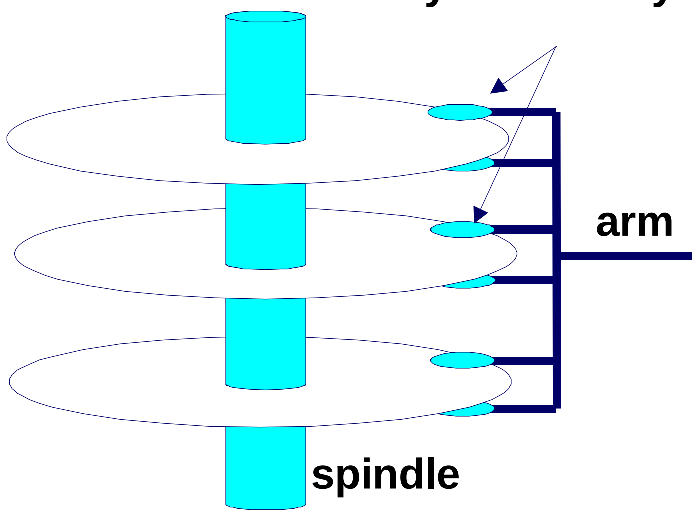
Access Within A Cylinder is Faster

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

Other factors influence overall disk access time

- Settle time, the time to stabilize the read/write head after a seek
- Command overhead, the time for the disk to process a command and start doing something

Minor compared to seek time and rotational delay

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 O.S. programmers, do about this?

- **Read more per seek (multi-sector transfers)**
- **Don't seek so randomly ("disk scheduling")**

Disk Scheduling Algorithms

The goal of a disk scheduling algorithm is to be nice to the disk

We can help the disk by giving it requests that are located close to each other

- This minimizes seek time, and possibly rotational latency

There exist a variety of ways to do this

Addressing Disks

What the OS knows about the disk

- Interface type (SATA/SCSI), unit number, number of sectors

What happened to sectors, tracks, etc?

- Old disks were addressed by cylinder/head/sector (CHS)
- Modern disks are addressed by abstract sector number
  - LBA = logical block addressing

Who uses sector numbers?

- File systems assign logical blocks to files

Terminology

- To disk people, “block” and “sector” are the same
- To file system people, a “block” is some number of sectors

Disk Addresses vs. Scheduling

Goal of OS disk-scheduling algorithm

- Maintain queue of requests
- When disk finishes one request, give it the "best" request
    - E.g., whichever one is closest in terms of disk geometry

Goal of disk's logical addressing

- Hide messy details of which sectors are located where
    - Disk change fast – more than once a year
    - OSs change slowly – up to 5 years for Windows

A good approximation

- Older OS's tried to understand disk layout
- Modern OS's just assume nearby sector numbers are close

Scheduling Algorithms

“Don't try this at home”

- FCFS
- SSTF

Arguably less wrong

- SCAN, C-SCAN

Plausible

- LOOK, C-LOOK

Useful, but hard

- SPTF

First Come First Served (FCFS)

Send requests to disk as they are generated by the OS

Trivial to implement – FIFO queue in device driver

Fair

- **What could be more fair?**

“Unacceptably high mean response time”

- **File “abc” in sectors 1, 2, 3, ...**
- **File “def” in sectors 16384, 16385, 16386, ...**
- **Sequential reads: 1, 16384, 2, 16385, 3, 16386, …**
    - **(disk shakes so much it “walks” across the room)**

Shortest Seek Time First (SSTF)

Maintain "queue" of disk requests

Serve the request nearest to the disk arm

- **Estimate nearness by subtracting block numbers**

Great!

- **Excellent throughput (most seeks are short)**
- **Very good average response time**

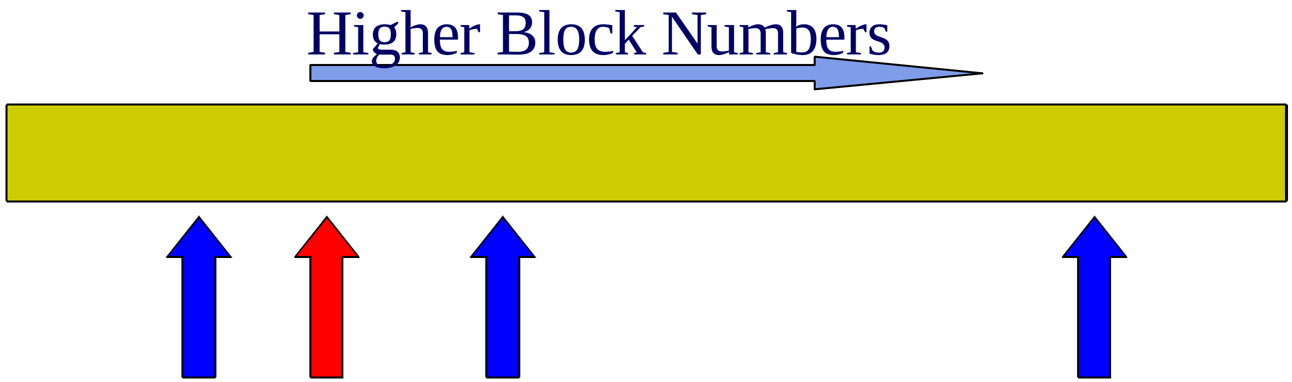
Intolerable response time *variance*, however

Why?

SSTF

Blue are requests

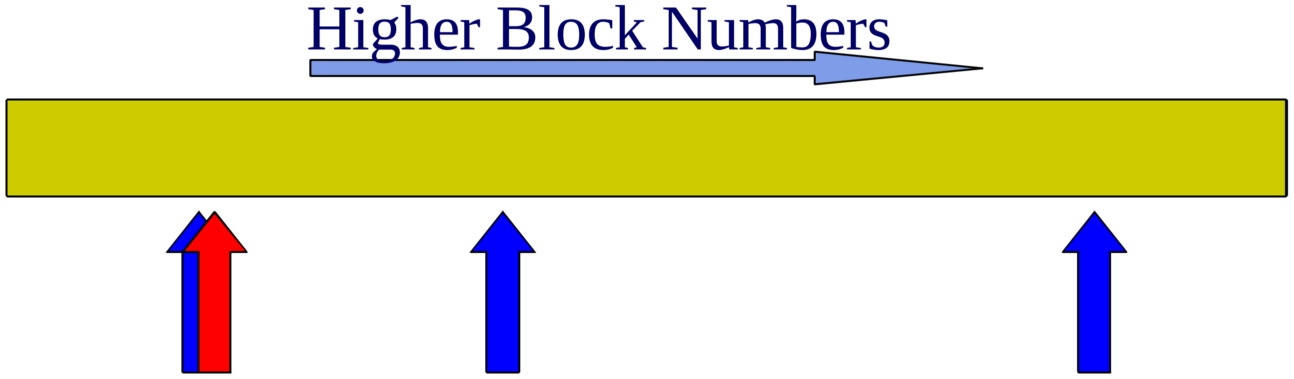
Yellow is disk



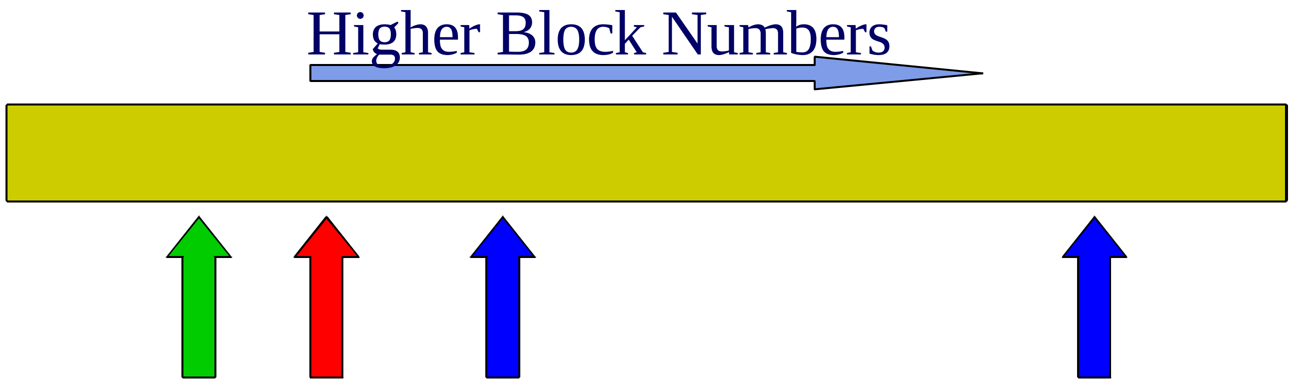
Red is disk head

Green is completed requests

SSTF

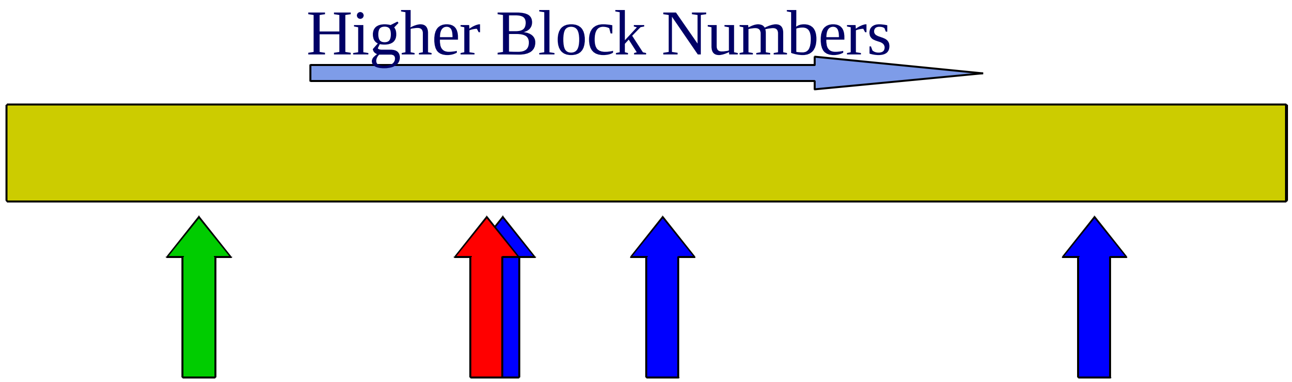


SSTF

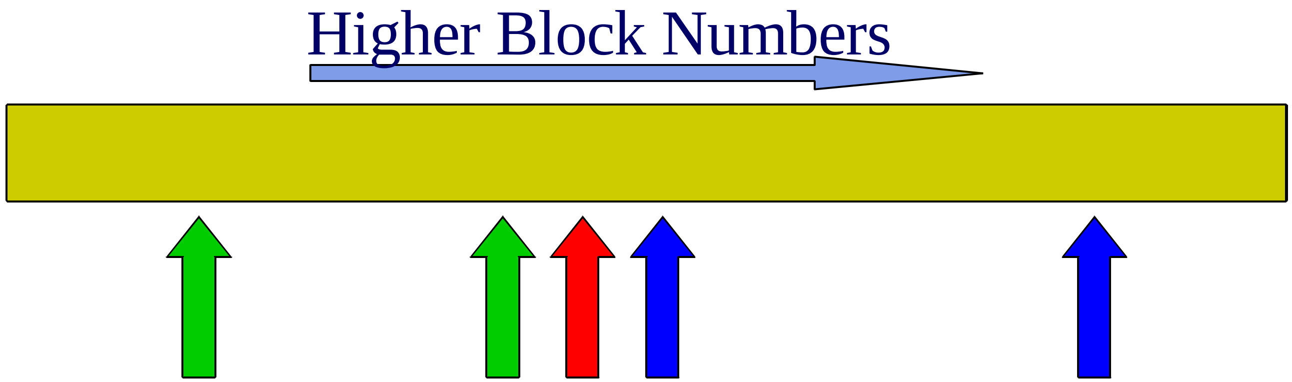


SSTF

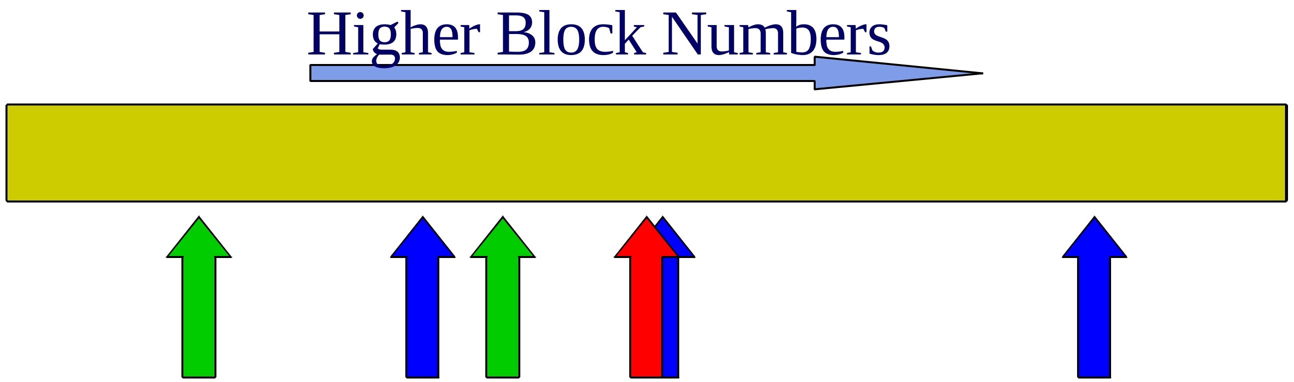
New Requests arrive…



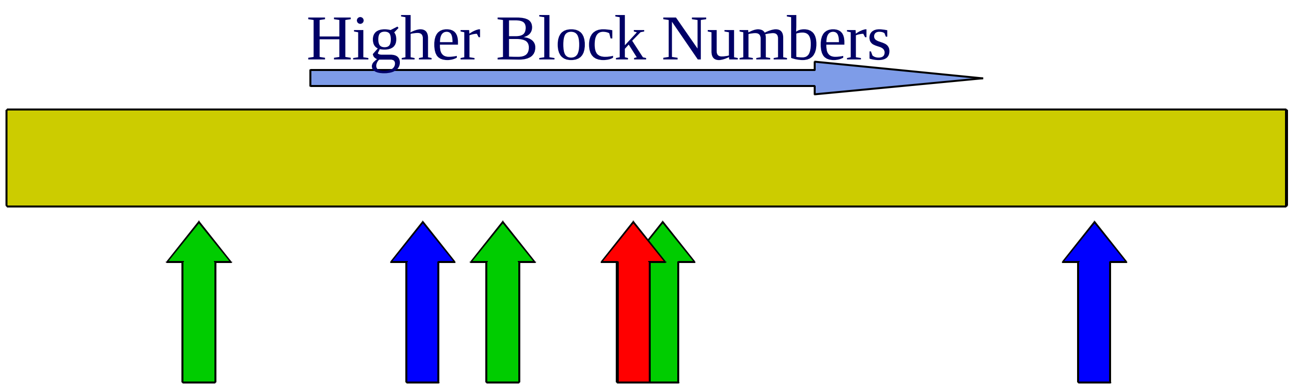
SSTF



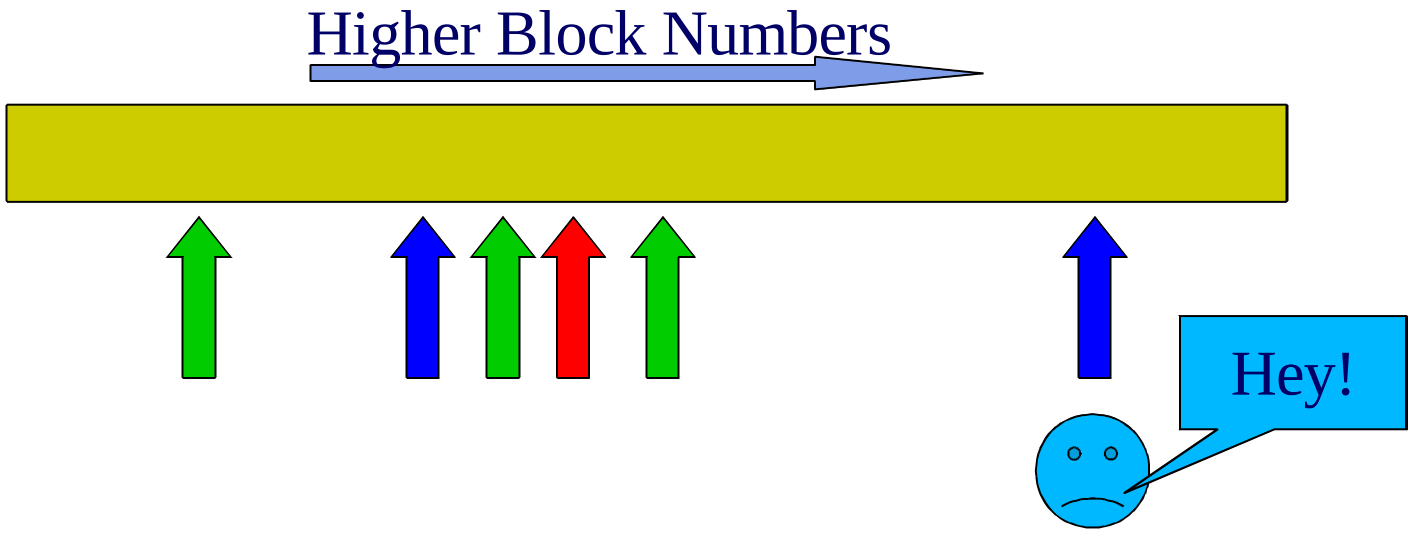
SSTF



SSTF

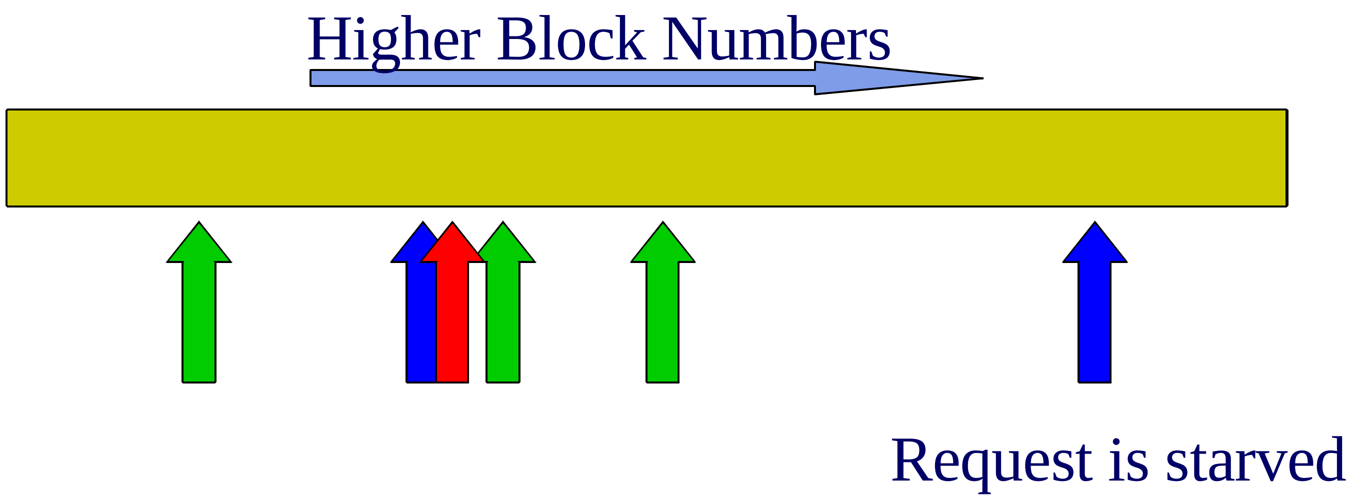


SSTF



SSTF

Starves requests that are “far away” from the head



What Went Wrong?

FCFS - “fair, but slow”

- Ignores position of disk arm, so it's slow

SSTF – good throughput, very unfair

- Pays too much attention to requests near disk arm
- Ignores necessity of eventually scanning entire disk

What Went Wrong?

FCFS - "fair, but slow"

- Ignores position of disk arm, so its slow

SSTF – good throughput, very unfair

- Pays too much attention to requests near disk arm
- Ignores necessity of eventually scanning entire disk

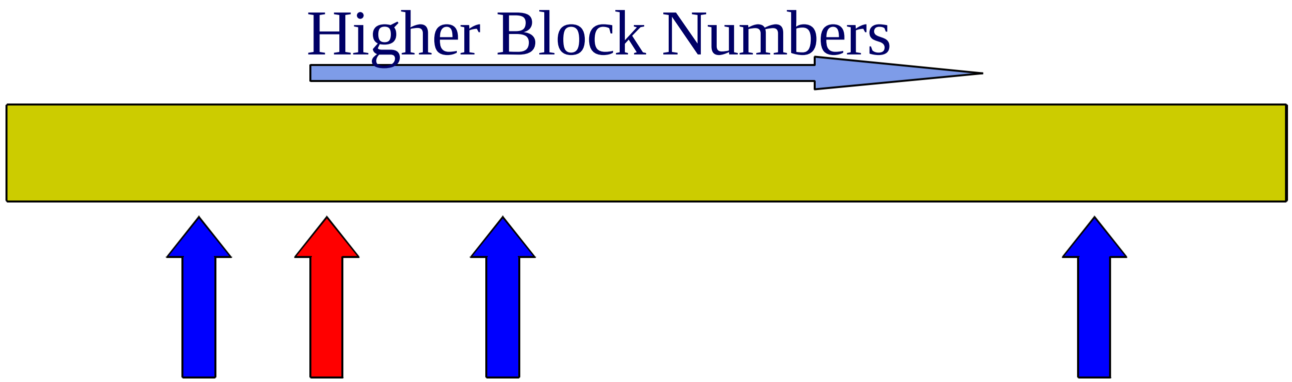
"Scan entire disk" - now that's an idea!

- Start disk arm moving in one direction
- Serve requests as the arm moves past them
  - No matter when they were queued
- When arm bangs into stop, reverse direction

SCAN

Blue are requests

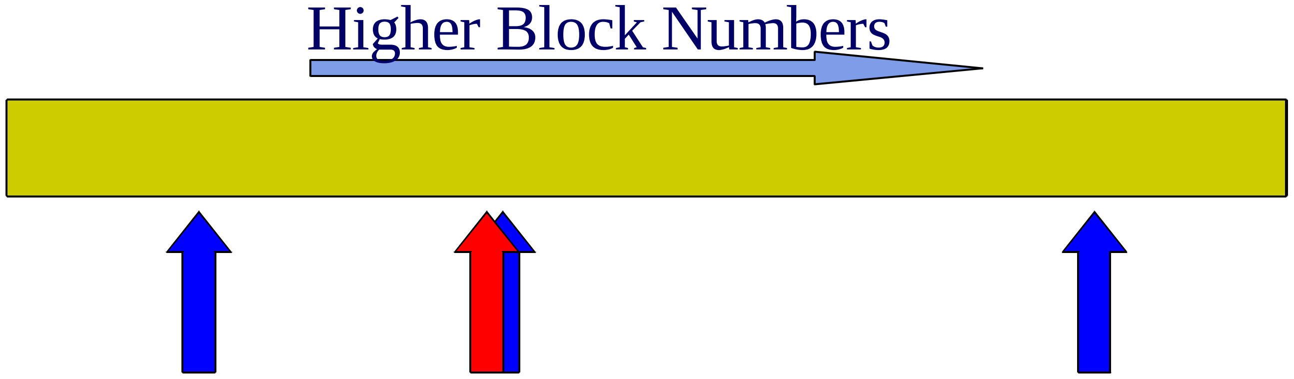
Yellow is disk



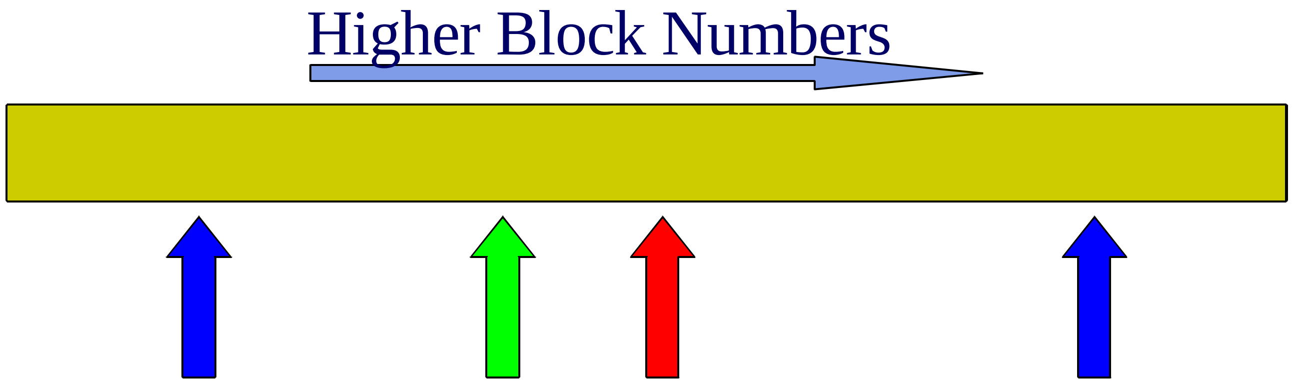
Red is disk head

Green is completed requests

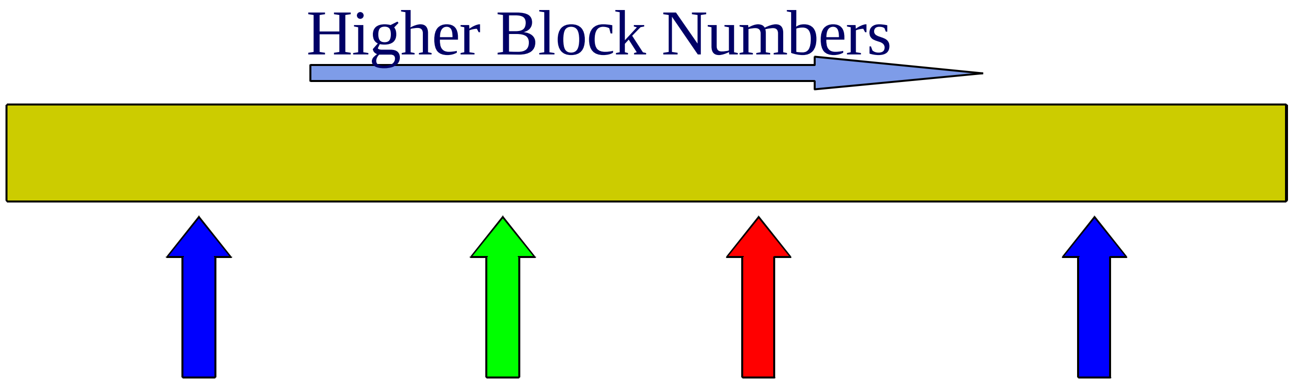
SCAN



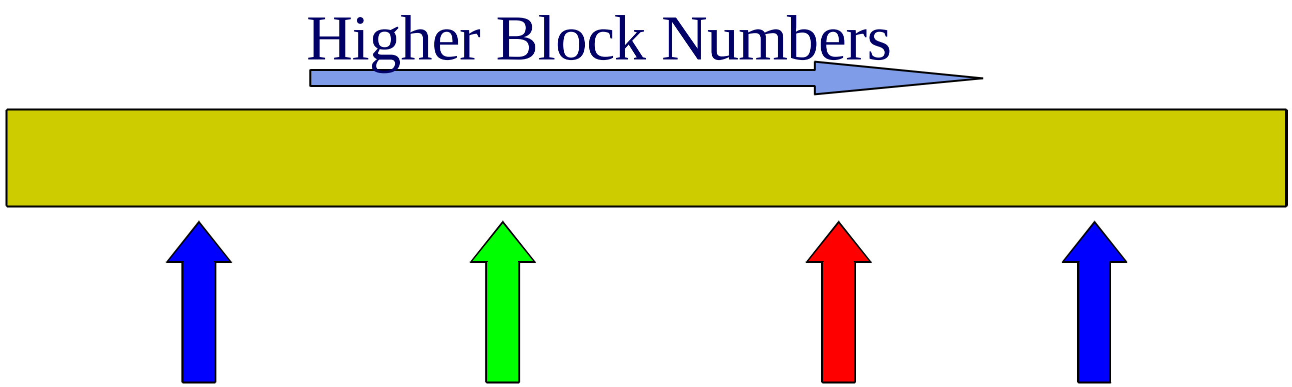
SCAN



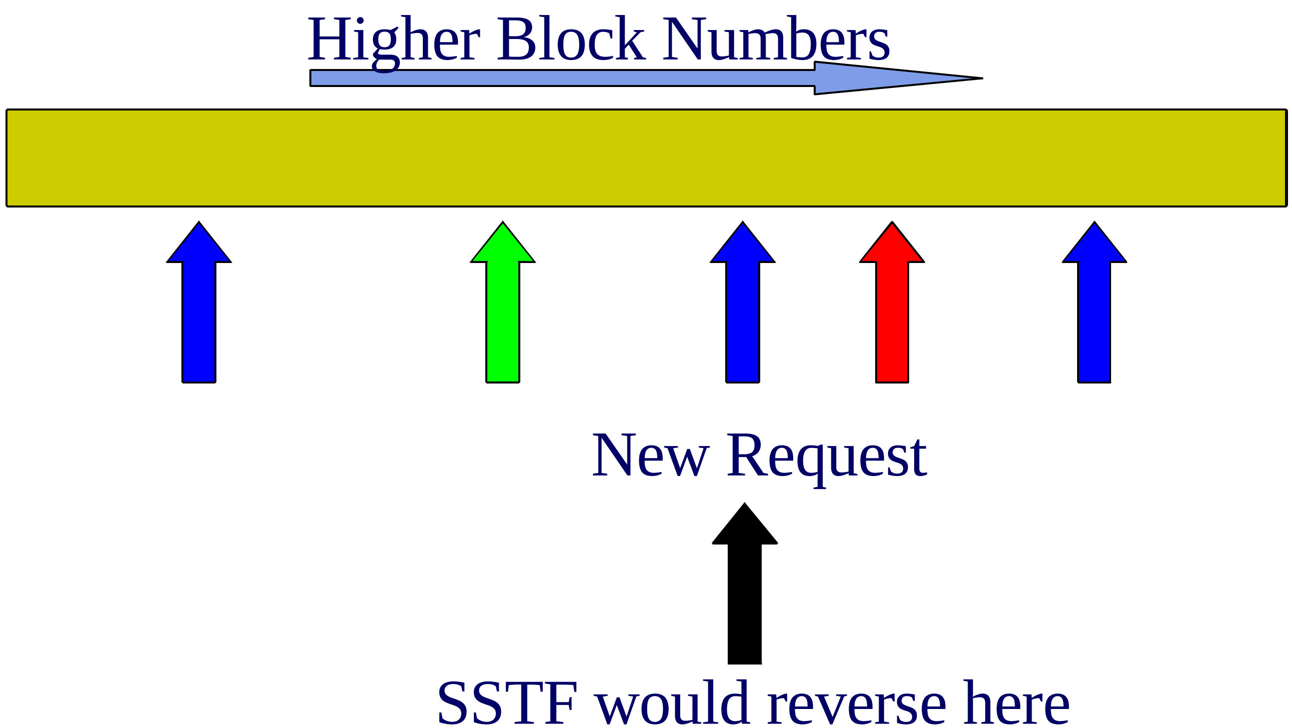
SCAN



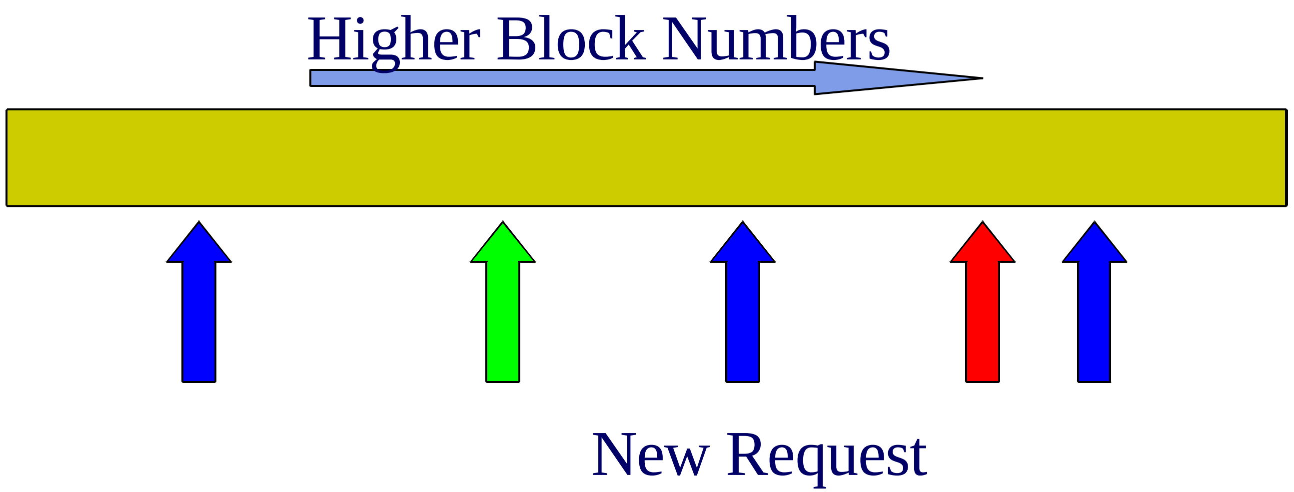
SCAN



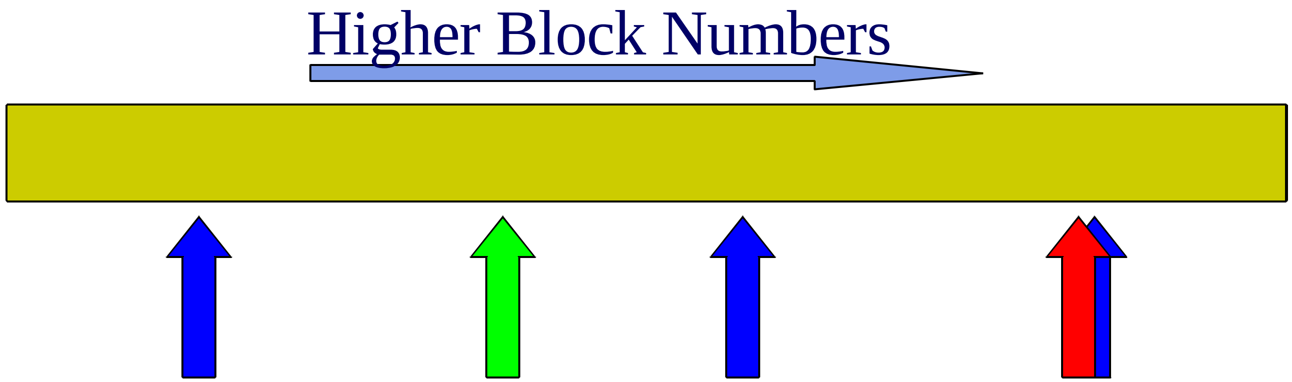
SCAN



SCAN

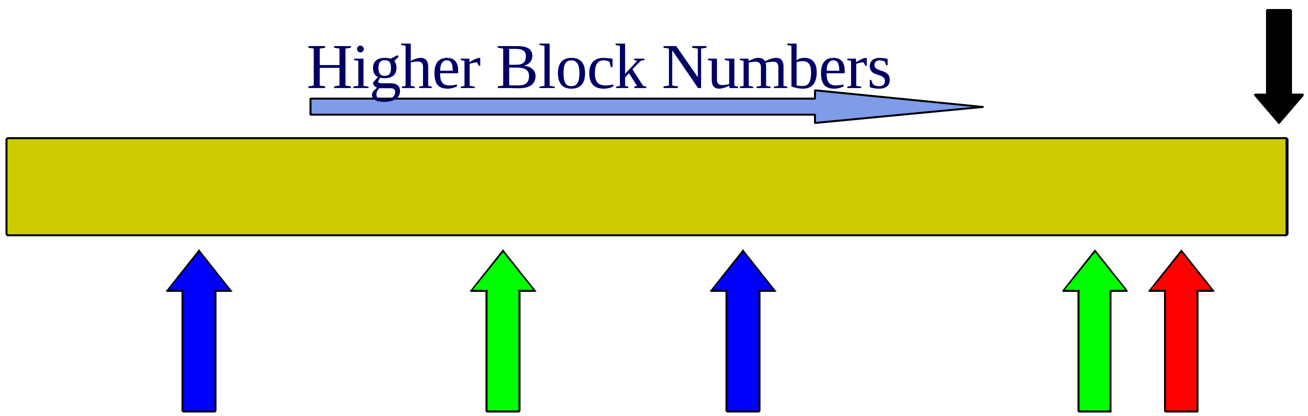


SCAN

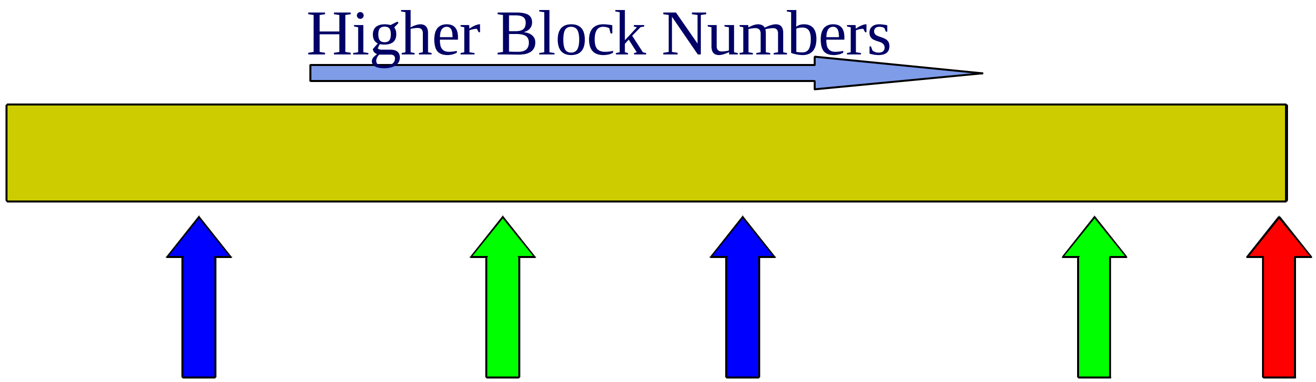


SCAN

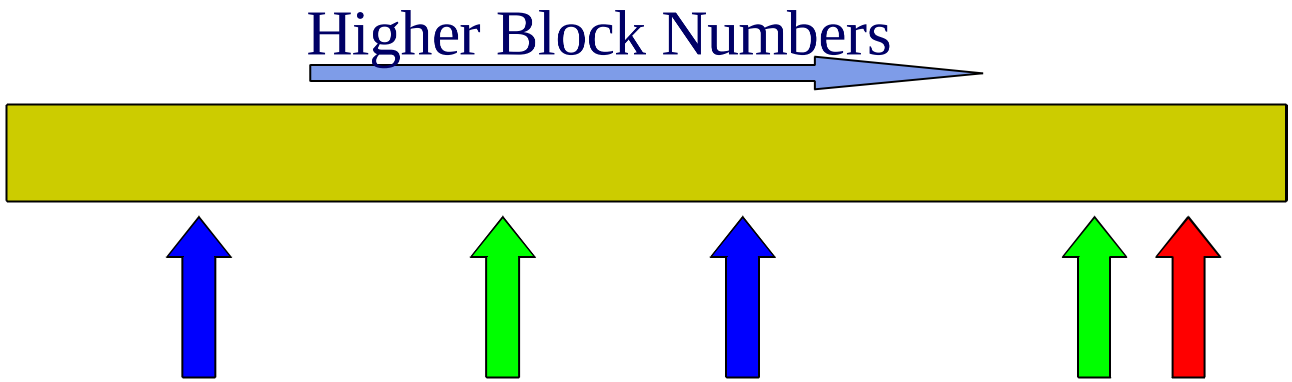
In SCAN, we continue to the end of the disk



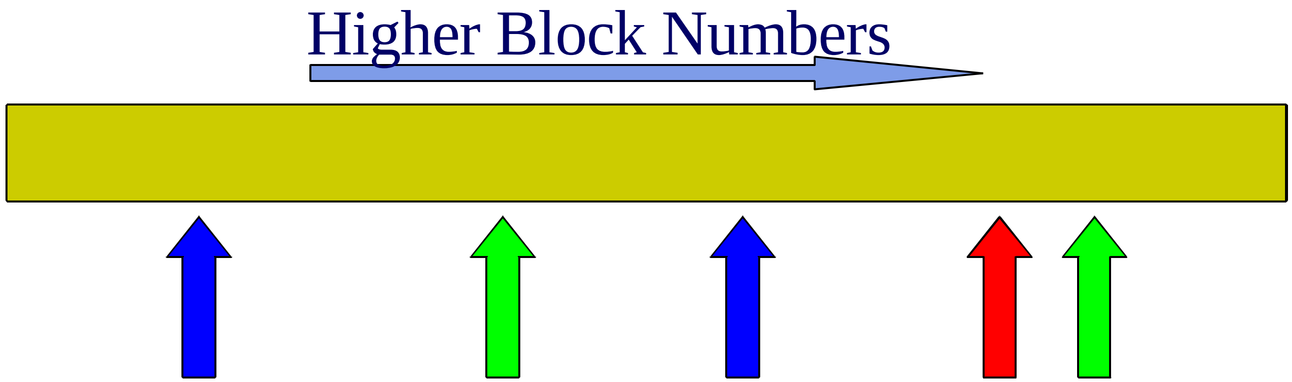
SCAN



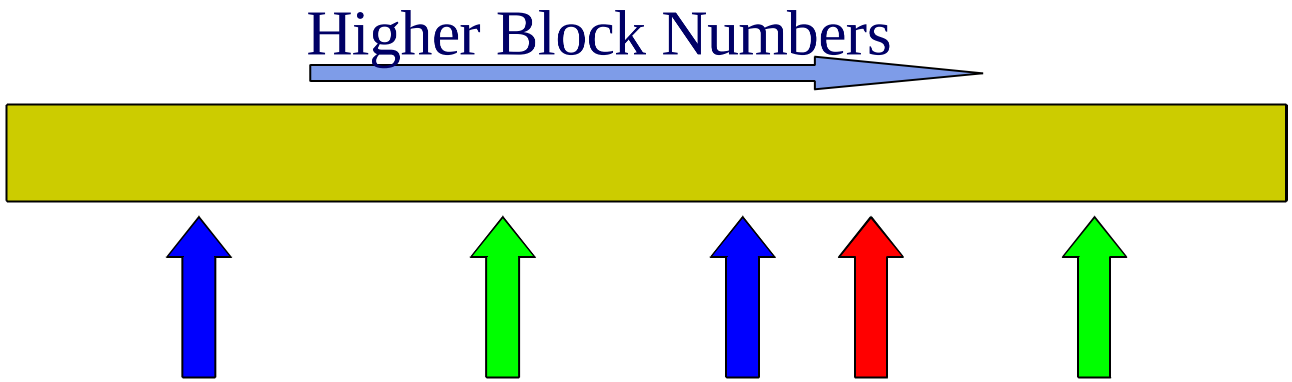
SCAN



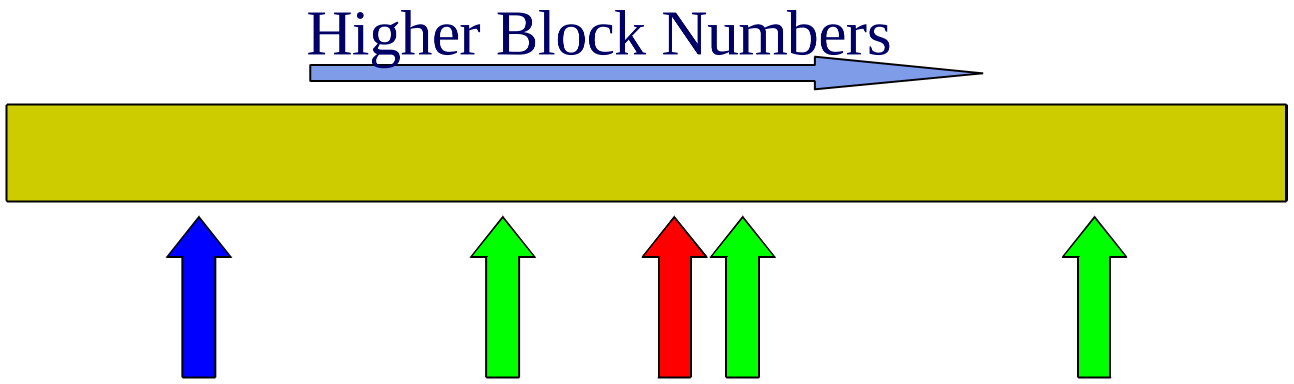
SCAN



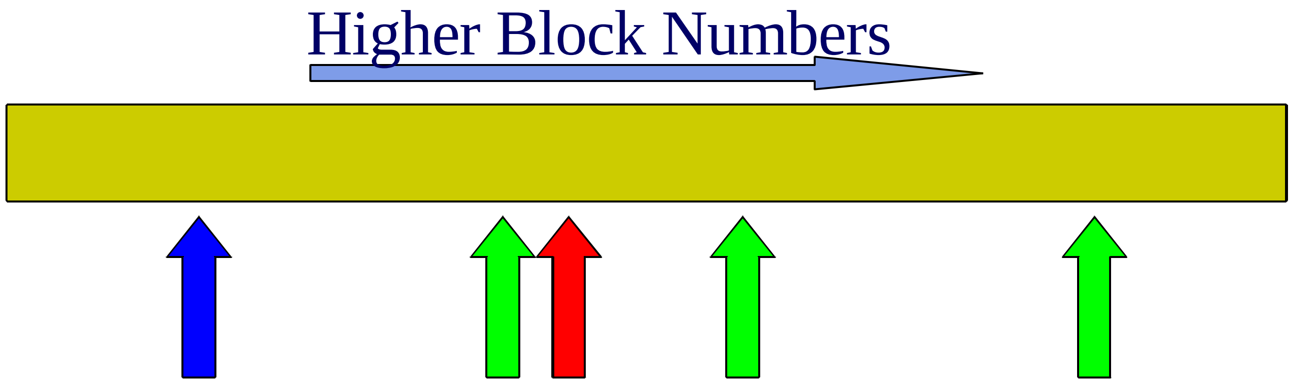
SCAN



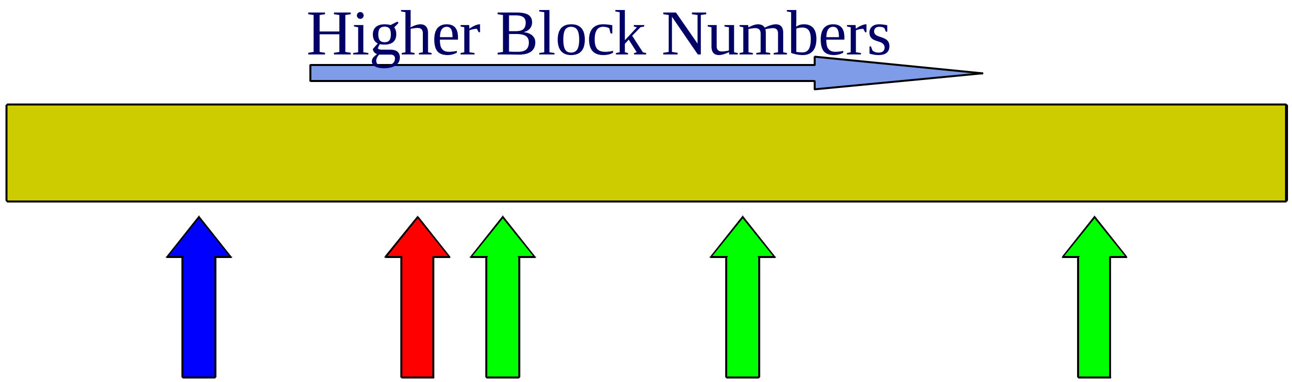
SCAN



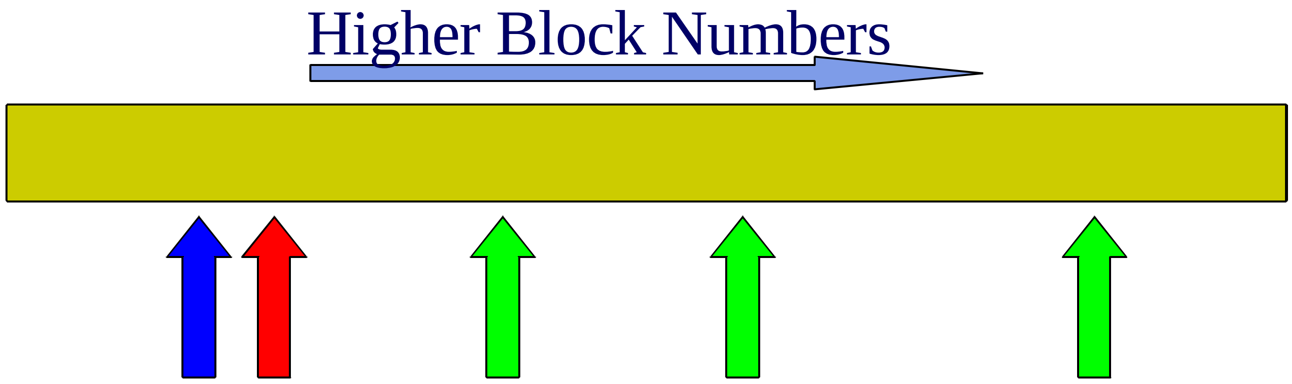
SCAN



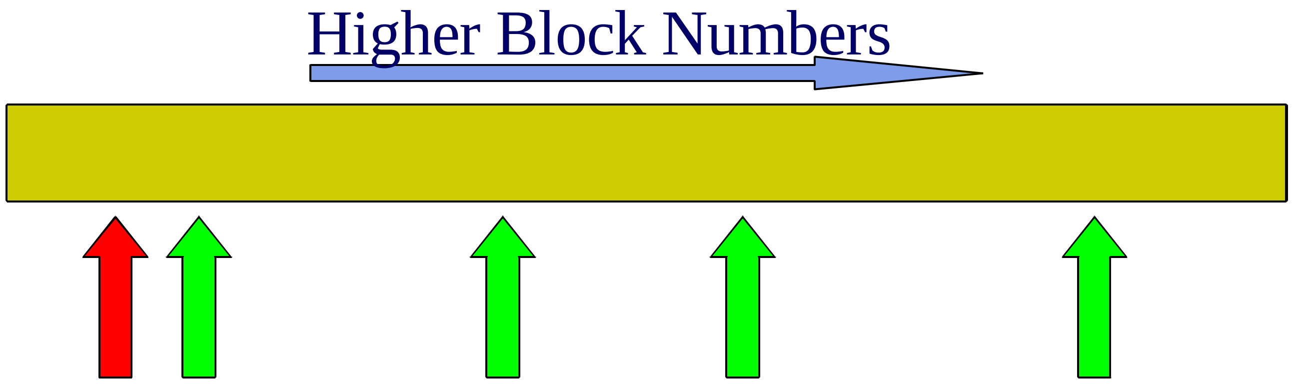
SCAN



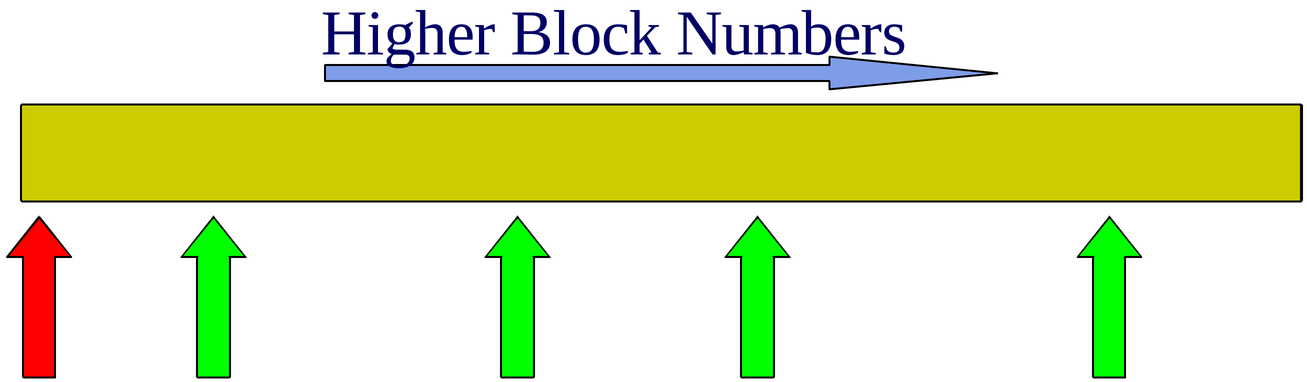
SCAN



SCAN



SCAN



Evaluating SCAN

Mean response time

- **Worse than SSTF, better than FCFS**
- **You should be able to say why**

Response time *variance*

- **Better than SSTF**

Do we need to go all the way to the end of the disk?

The LOOK Optimization

Just like SCAN – sweep back and forth through cylinders

Don't wait for the "thud" to reverse the scan

- **Reverse when there are no requests "ahead" of the arm**

Improves mean response time, variance

Both SCAN and LOOK are unfair – why?

C-SCAN - “Circular SCAN”

Send requests in ascending cylinder order

When the last cylinder is reached, seek all the way back to the first cylinder

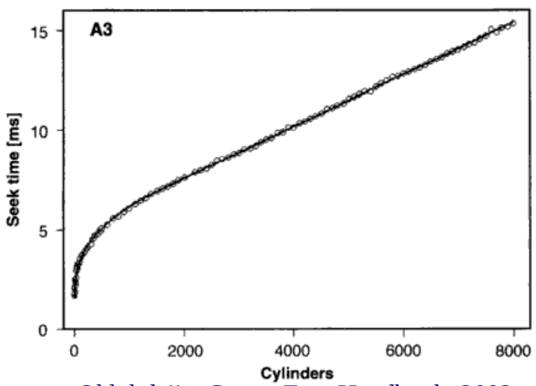
Long seek is amortized across all accesses

- **Key implementation detail**
  - **Seek time is a *non-linear* function of seek distance**
  - **One big seek is faster than N smaller seeks**

Variance is improved

Fair

Still missing something though…



Oklobdzija, Comp. Eng. Handbook, 2002

C-LOOK

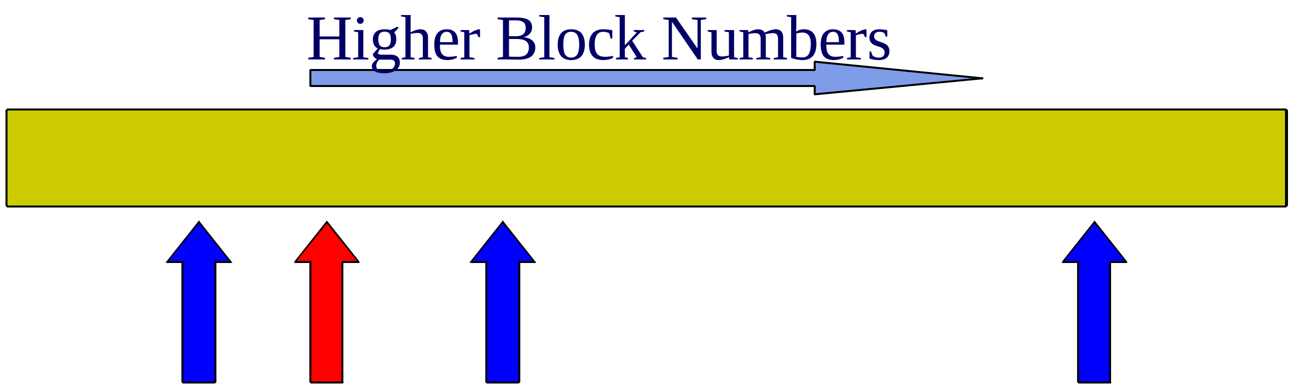
CSCAN + LOOK

Scan in one direction, as in CSCAN

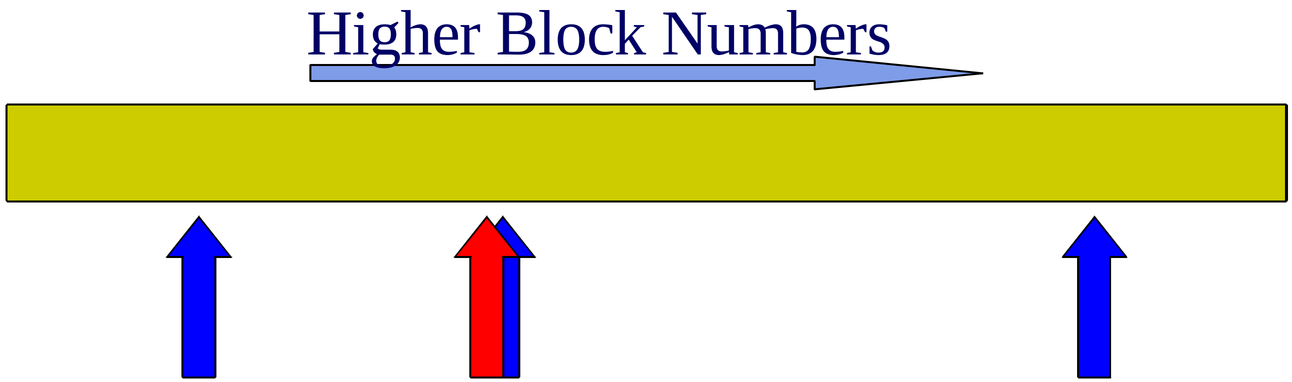
If there are no more requests in current direction go back to furthest request

Very popular

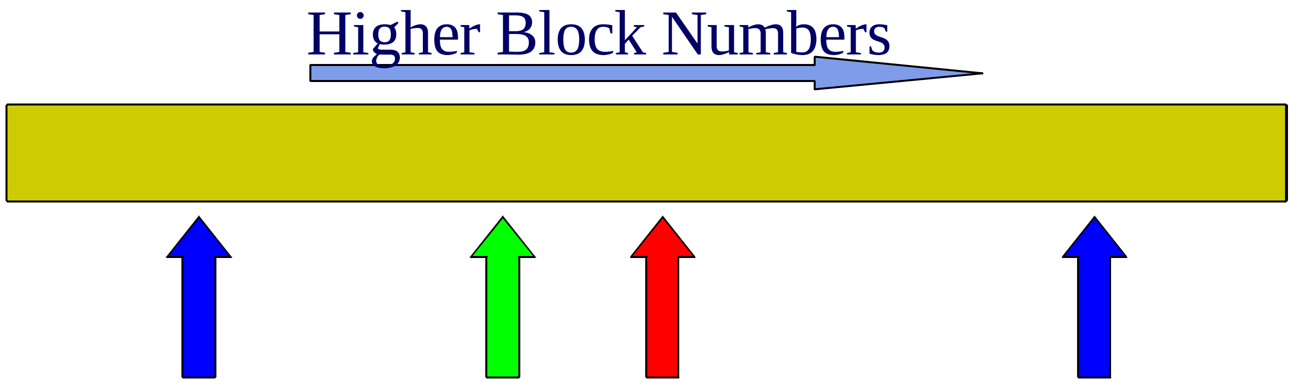
C-LOOK



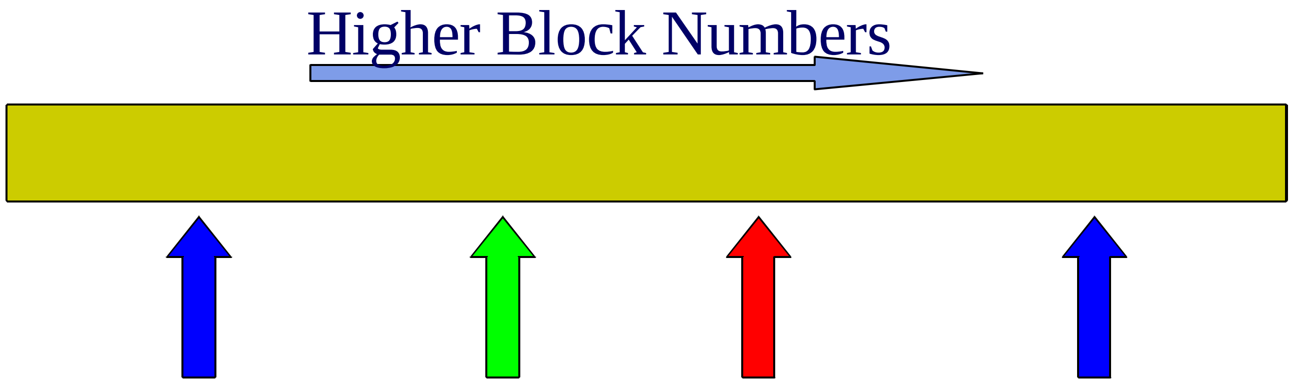
C-LOOK



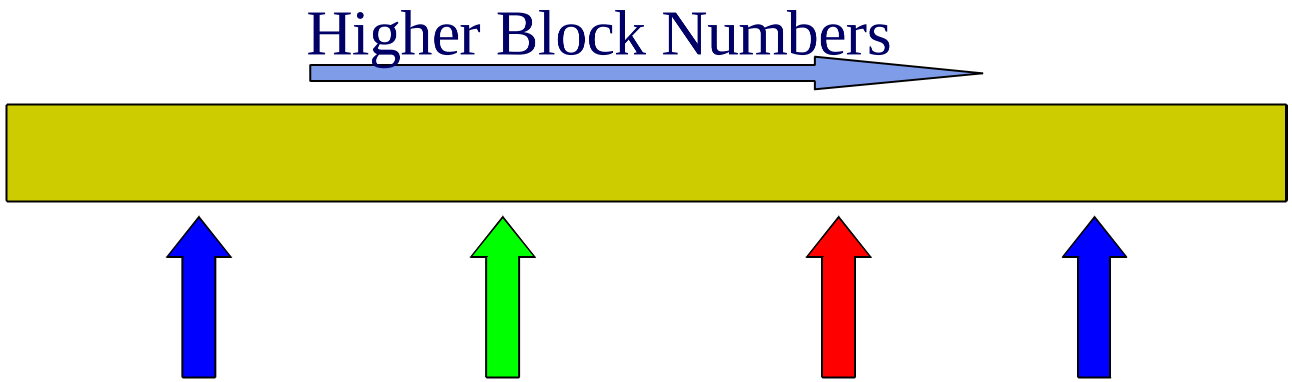
C-LOOK



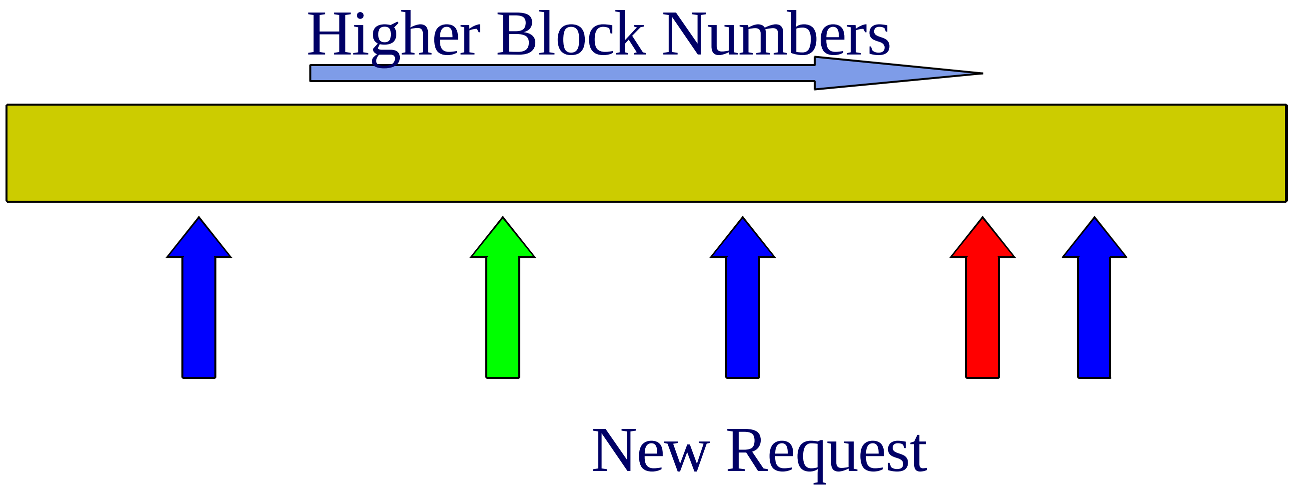
C-LOOK



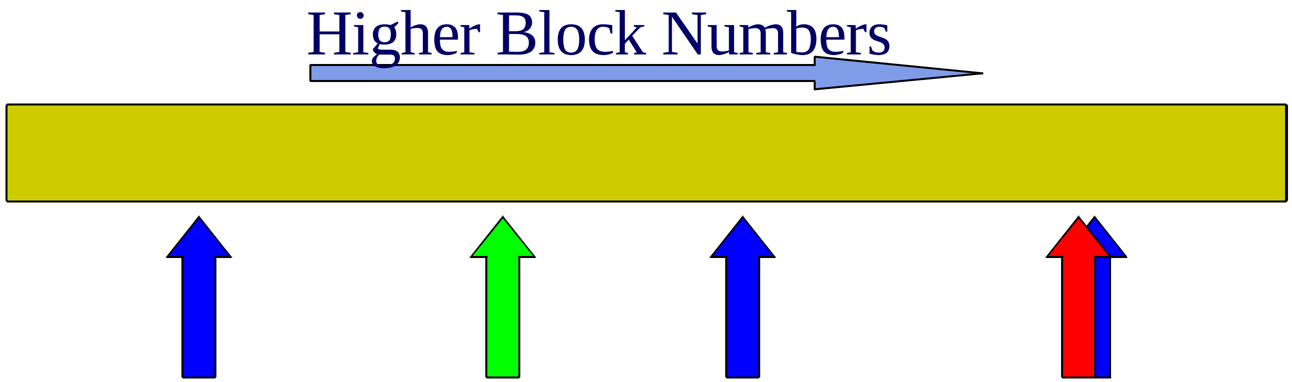
C-LOOK



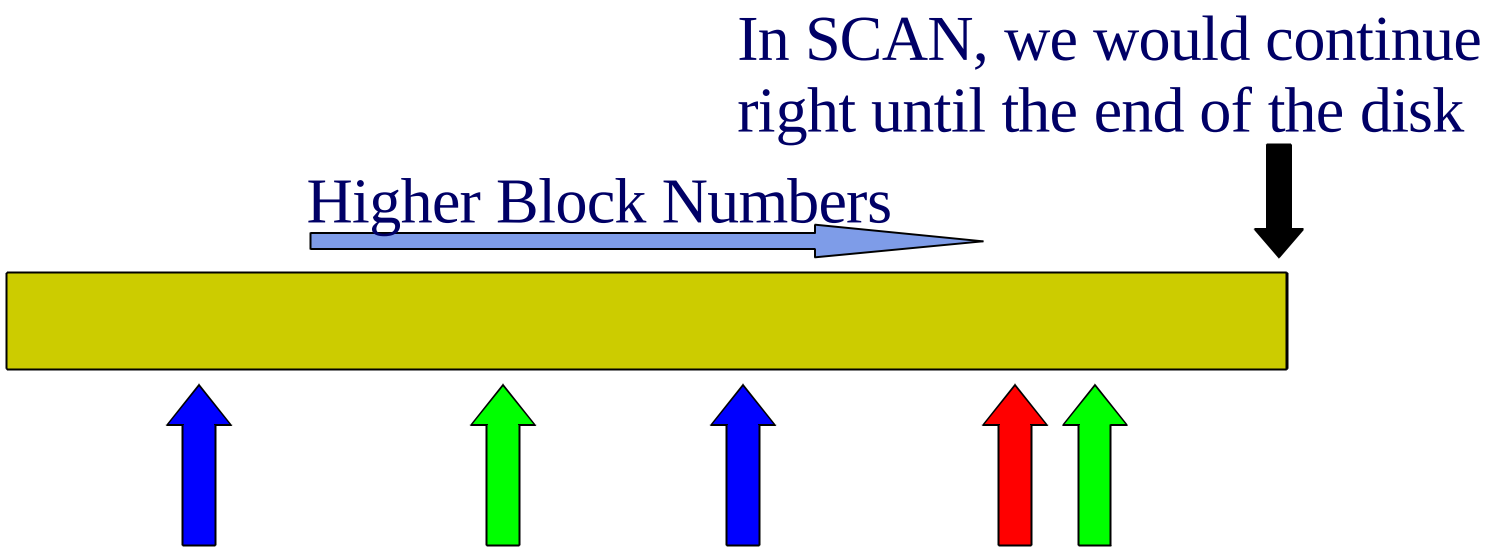
C-LOOK



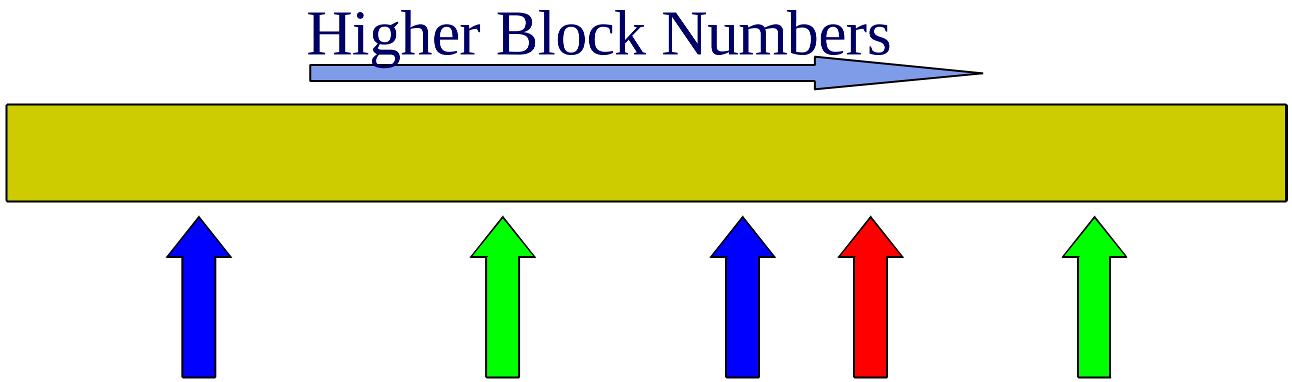
C-LOOK



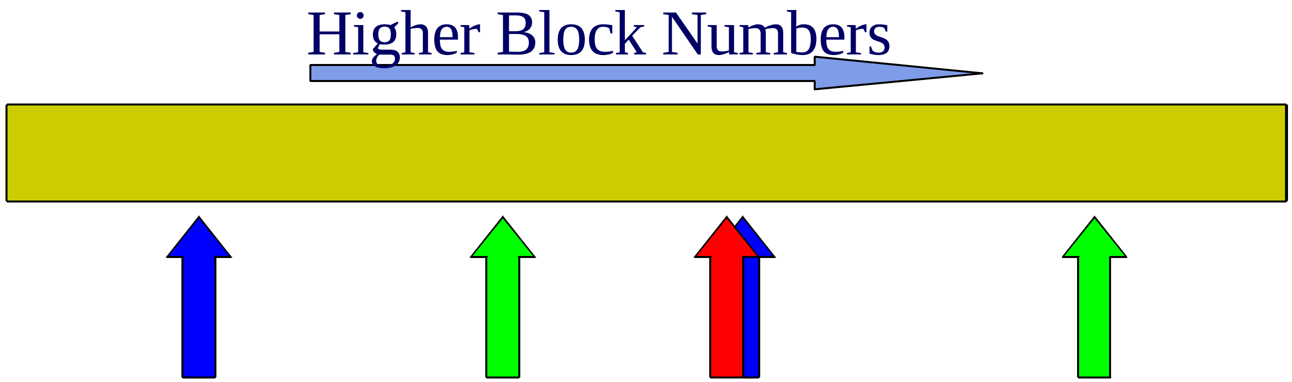
C-LOOK


In SCAN, we would continue right until the end of the disk
Higher Block Numbers

C-LOOK

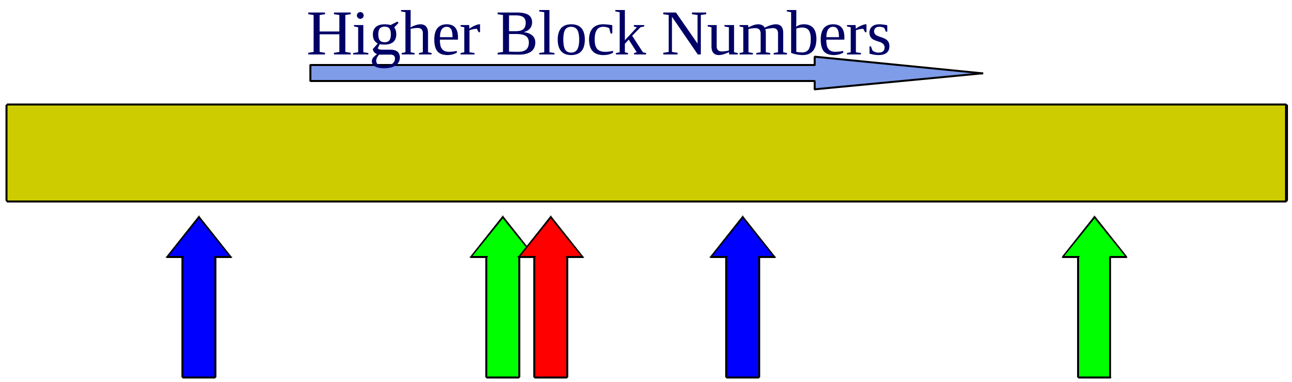


C-LOOK

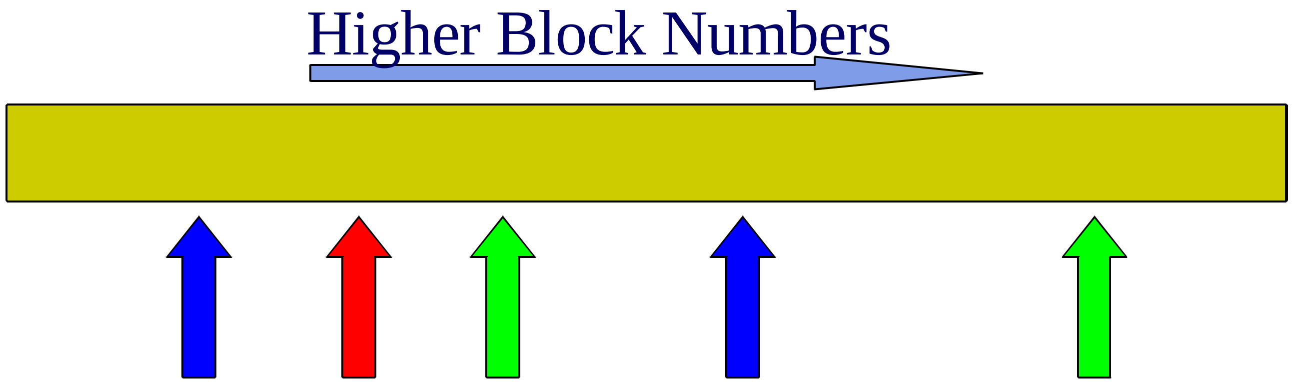


In LOOK, we would have read this request
(unfair extra service—so we'll skip it)

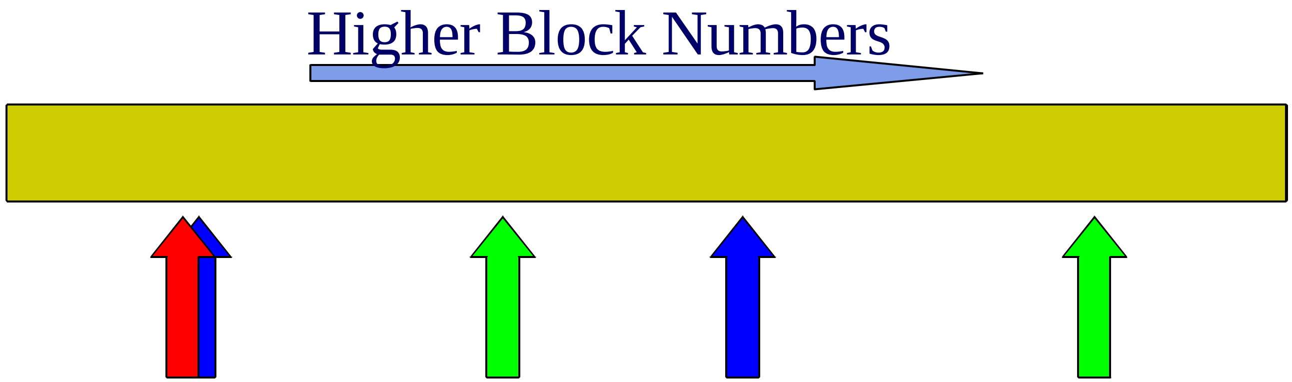
C-LOOK



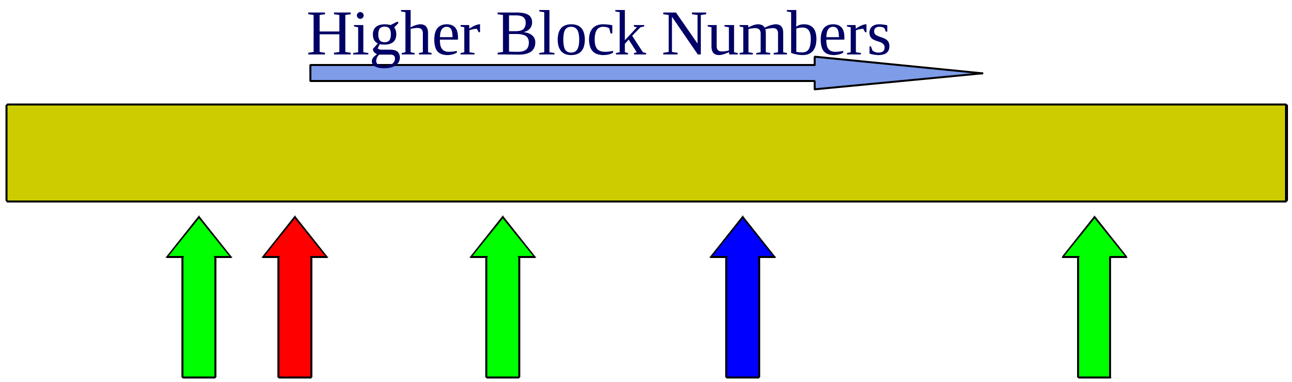
C-LOOK



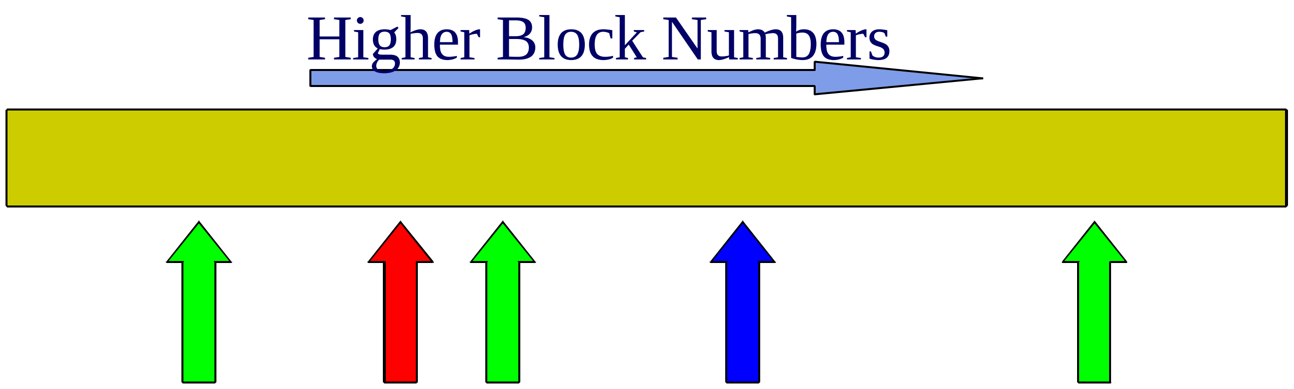
C-LOOK



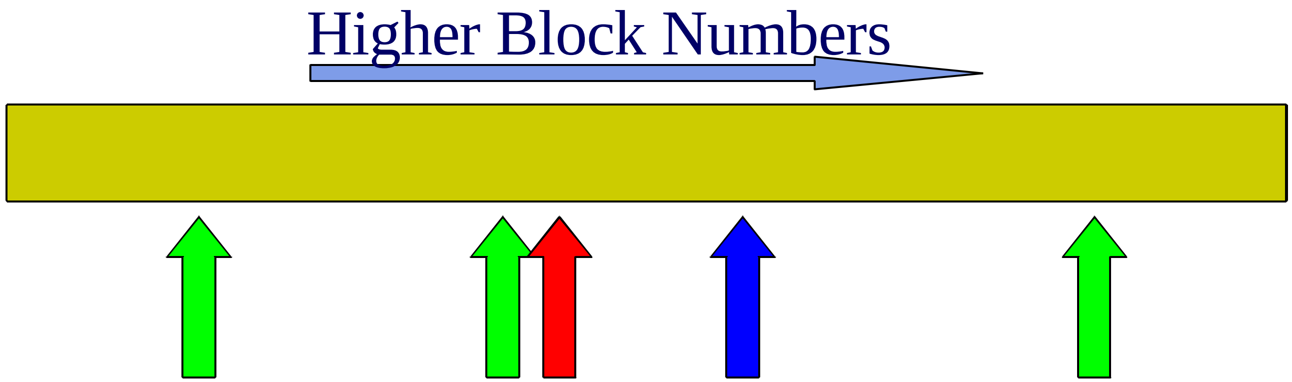
C-LOOK



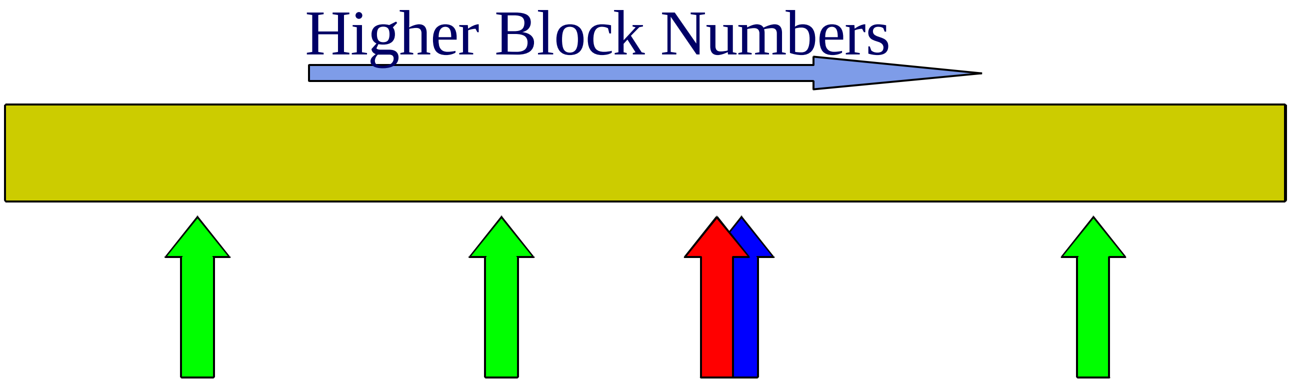
C-LOOK



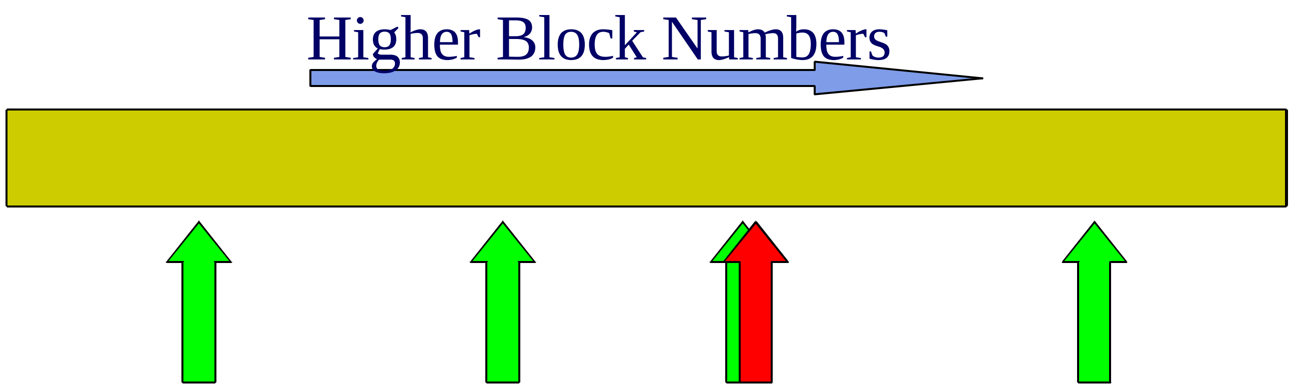
C-LOOK



C-LOOK



C-LOOK



Algorithm Classification

SCAN vs. LOOK

- LOOK doesn't visit far edges of disk unless there are requests

LOOK vs. C-LOOK

- C for “circular” - don't double-serve middle sectors

We are now excellent disk-arm schedulers

- Done, right?

Shortest ***Positioning*** Time First

Key observation

- **Seek time takes a while, C-LOOK is a reasonable response**
- **But rotational delay is comparable!**
  - **More: short seeks are *faster* than whole-disk rotations**
- **What matters is *positioning* time, not seek time**

SPTF is like SSTF

- **Serve “temporally nearest” sector next**

Challenge

- **Driver can't estimate positions from sector numbers**
- **Must know layout, plus rotation position of disk in real time!**

Performs better than SSTF, but still starves requests

Weighted Shortest Positioning Time First (WSPTF)

SPTF plus fairness

Requests are “aged” to prevent starvation

- **Compute “temporal distance” to each pending request**
- **Subtract off “age factor” - old requests are artificially close**
- **Result: sometimes serve old request, not closest request**

Various aging policies possible, many work fine

Excellent performance

As SPTF, hard for OS to know disk status in real time

- **On-disk schedulers can manage this, though...**
  - **Modern disks (SATA, SCSI) accept a *request queue***
  - **Sector complete ⇒ give OS both data and sector number**

Scheduling Concept Summary

LOOK vs SCAN

- SCAN goes to the very end of the disk
- LOOK goes only as far as the farthest request

2-way vs circular

- 2-way reverses directions at the extremes, unfair
- Circular starts back at the "starting" position

Modern disks queue internally, using positioning time

- Head of request queue managed by disk – two-level scheduler

Fairness

- "High-throughput" algorithms can starve requests
- "Complete fairness" is slow
- Balance somehow... "aging" is one option

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 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”**

Solid-State Disks (SSD)

SSD vs. disk

- ☺ SSD's Implement 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)

Solid-State Disks (SSD)

SSD pretends to be a disk

- Identify, read/write, “park heads”, SMART, ...

It's a big lie

- “Write amplification”
  - Flash must be erased in big blocks (256KB)
  - So storing 512B could blow up by 512X!
- Wear leveling
  - When a block is written 10K/100K times, it breaks
  - File systems tend to write “favorite blocks” over and over...
- “Flash translation layer” (computer) maps LBA to arbitrary flash locations
  - This is a tough, complicated, messy job
  - Approaches and algorithms still in flux

Solid-State Disks (SSD)

Opportunity & threat

- "TRIM" command speeds up writes!
  - "Dear FTL, logically zero-fill these blocks"
- "Securely erase disk" is impossible

The future?

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

Conclusions

Disks are mechanical (voice coil == speakers)

Disks are slow, best if accesses are big & sequential

Disks are complicated (there's a computer inside)

FCFS is a very bad idea

- C-LOOK is ok in practice
- Disks probably do something like SPTF internally

Flexible queuing is good for performance

- Data-structure integrity and performance pull opposite ways

SSD's are "hot"

- (cool)

Further Reading

Terabyte Territory

Brian Hayes

American Scientist, May/June 2002

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

A Conversation with Jim Gray

Dave Patterson

ACM Queue, June 2003

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

Reliably Erasing Data from Flash-based Solid State Drives

Wei et al., UCSD

FAST '11

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