15-410

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

Disks
October 27th, 2008

Dave Eckhardt & Roger Dannenberg

Brian Railing & Steve Muckle

Contributions from

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

Synchronization

Speaker today

Shyam Sundar, Qualcomm

"Android and the Future of Open Source OS"

18:00, 7500

Computer Club Movie

Sneakers

Wednesday, 19:00, 7500

Pizza will be available for purchase

Checkpoint #3

- Look for a posting today/tomorrow
- Opportunity to do a conscious planning cycle
  - Estimate, then see what happens

Overview

Anatomy of a Hard Drive

Common Disk Scheduling Algorithms

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 *disk arm*

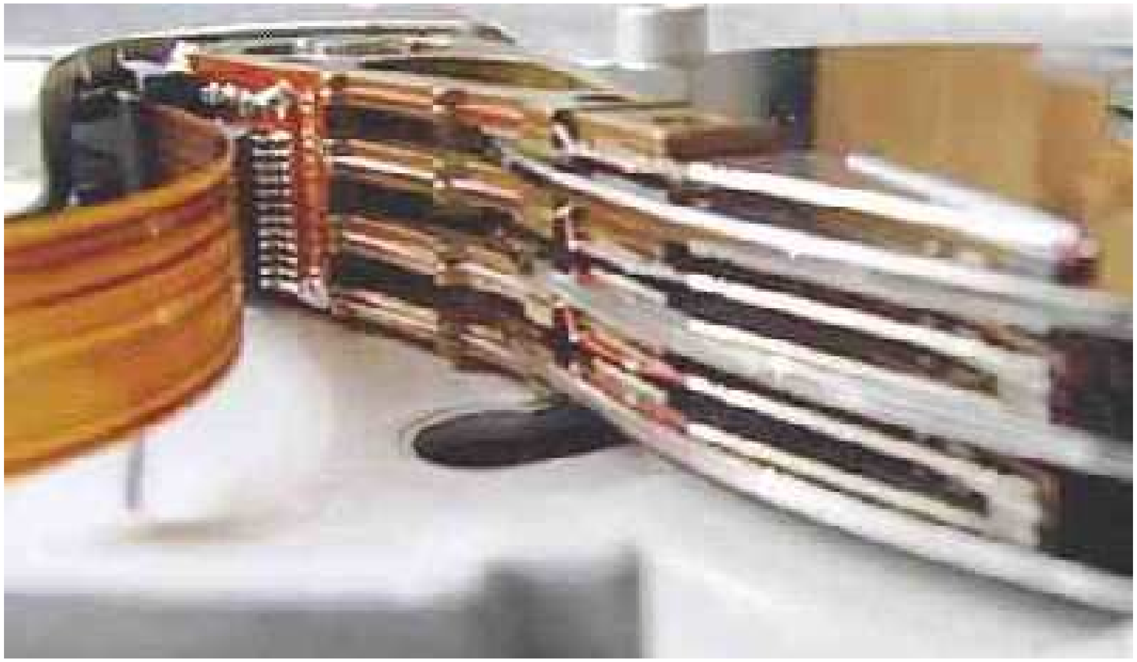
Taken from "How Hard Disks Work"
http://computer.howstuffworks.com/hard-disk2.htm

Anatomy of a Hard Drive

Both sides of each platter store information

Each side of a platter is called a ***surface***

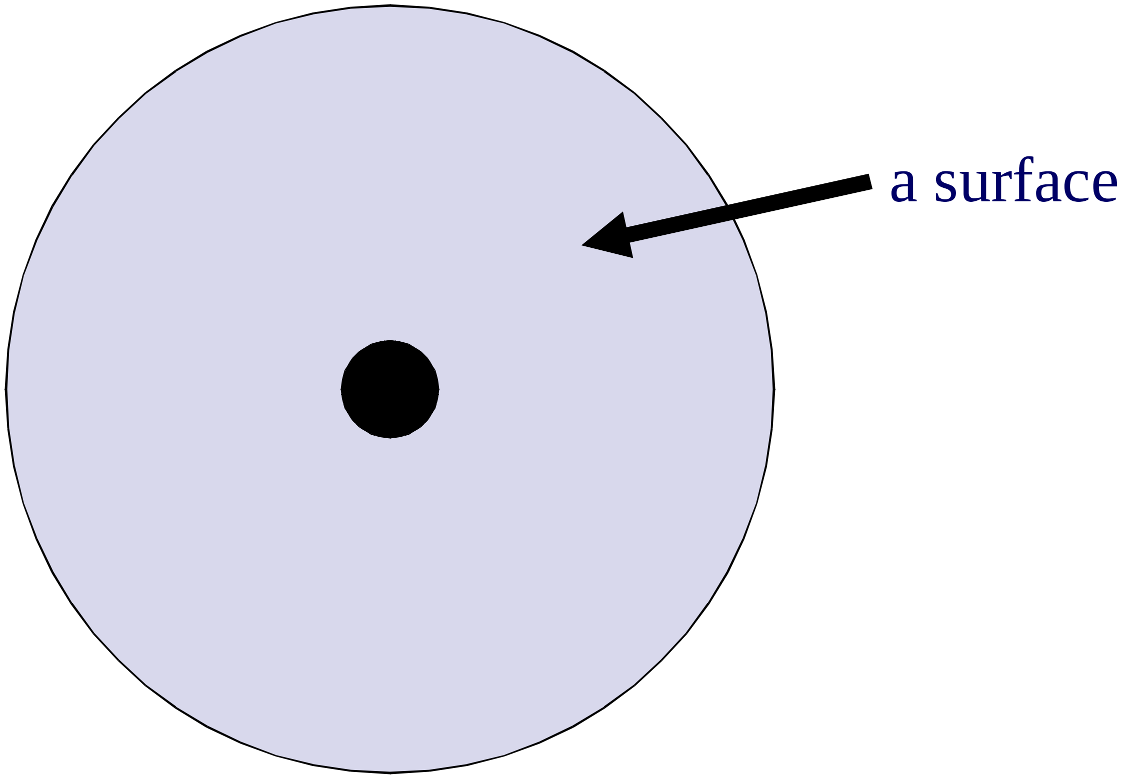
Each surface has its own read/write head



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

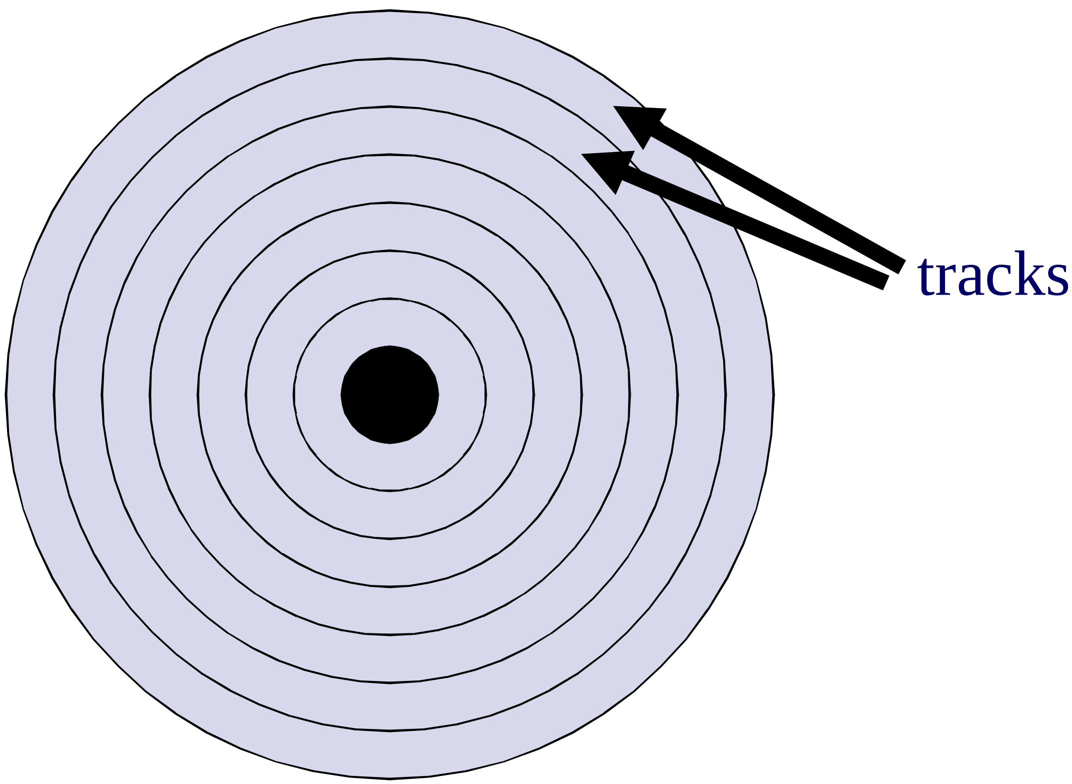
Anatomy of a Hard Drive

How are the surfaces organized?



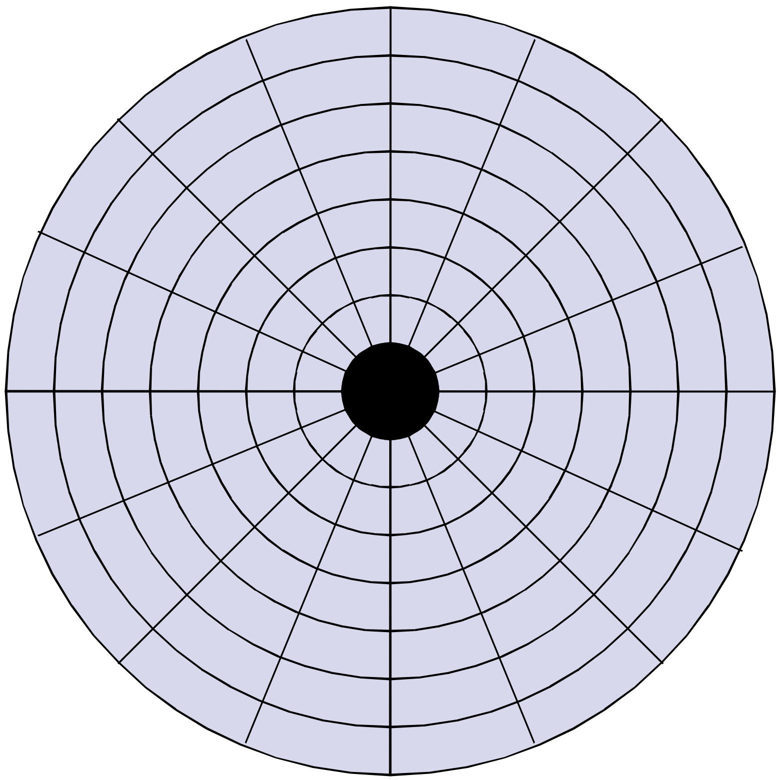
Anatomy of a Hard Drive

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



Anatomy of a Hard Drive

These tracks are further divided into *sectors*



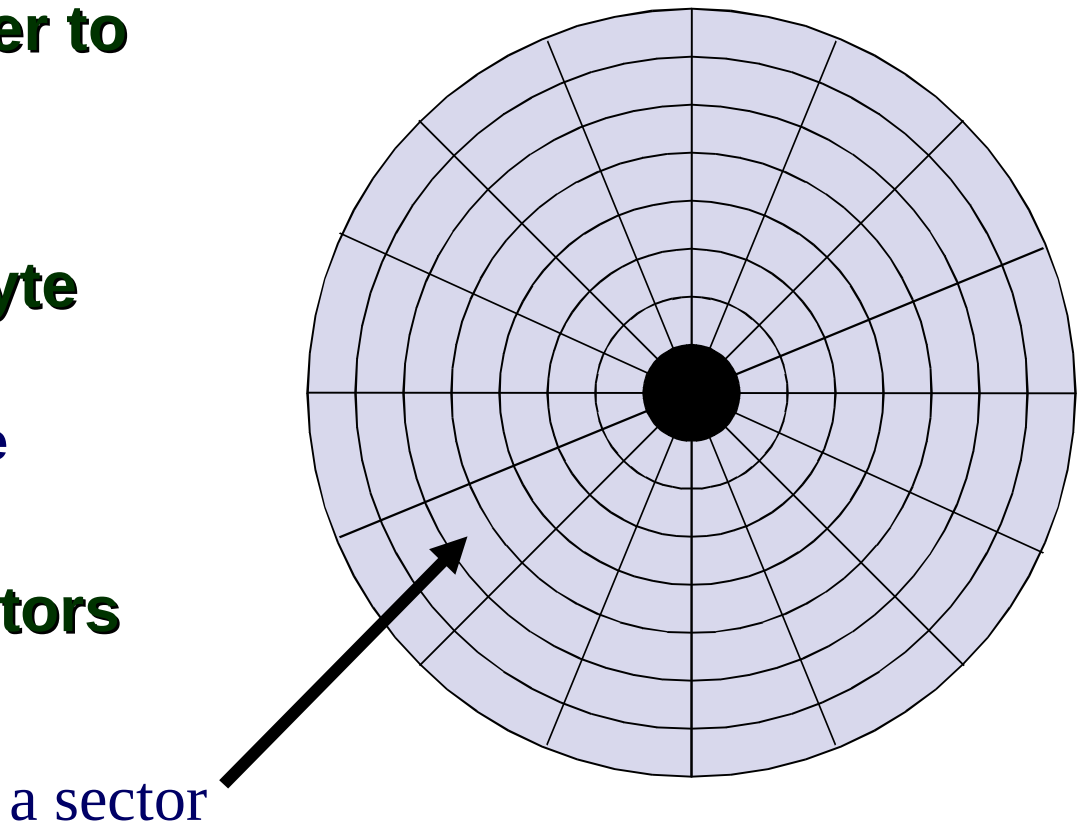
Anatomy of a Hard Drive

A sector is the smallest unit of data transfer to or from the disk

Most modern hard drives have 512-byte sectors

(CD-ROM sectors are 2048 bytes)

Gee, those outer sectors look bigger...

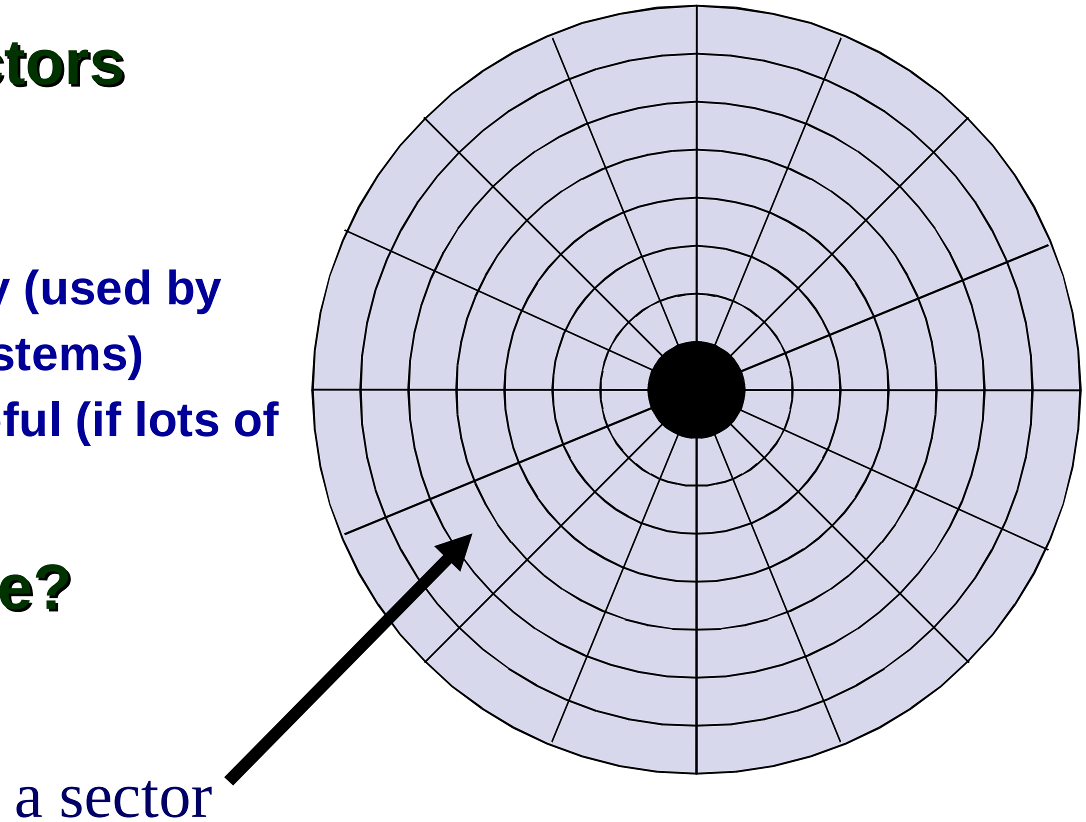


Anatomy of a Hard Drive

Gee, those outer sectors look bigger...

- More area per bit
  - Greater reliability (used by some operating systems)
  - Eventually wasteful (if lots of tracks per disk)

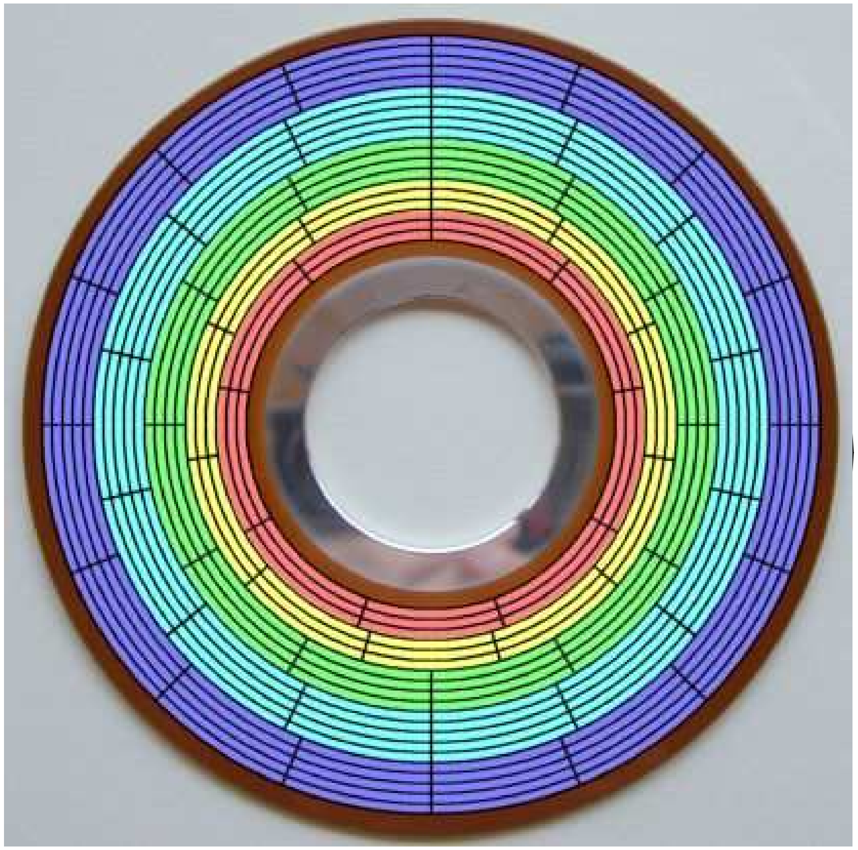
Is there an alternative?



Anatomy of a Hard Drive

Modern hard drives use *zoned bit recording*

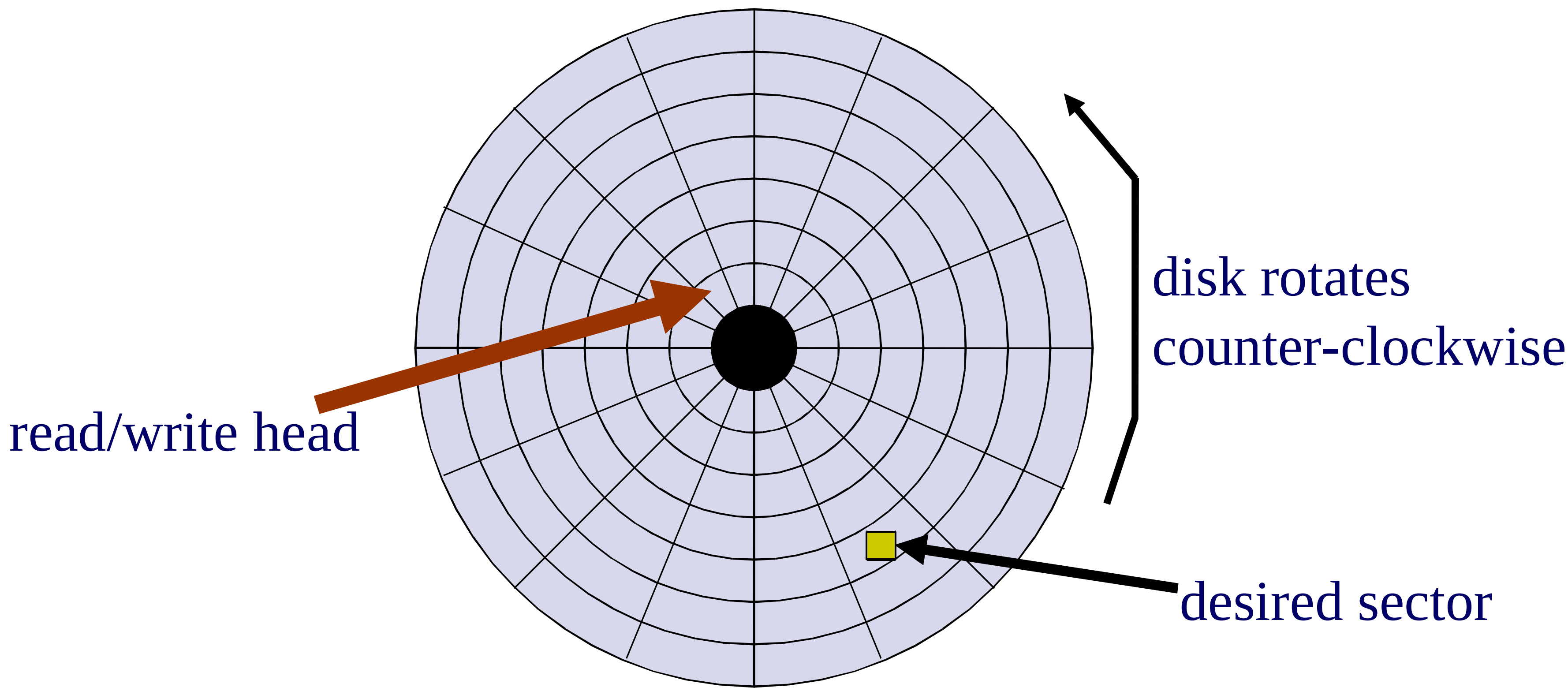
- **Table maps track# to #sectors**
- **Sectors are all roughly the same size**



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")**

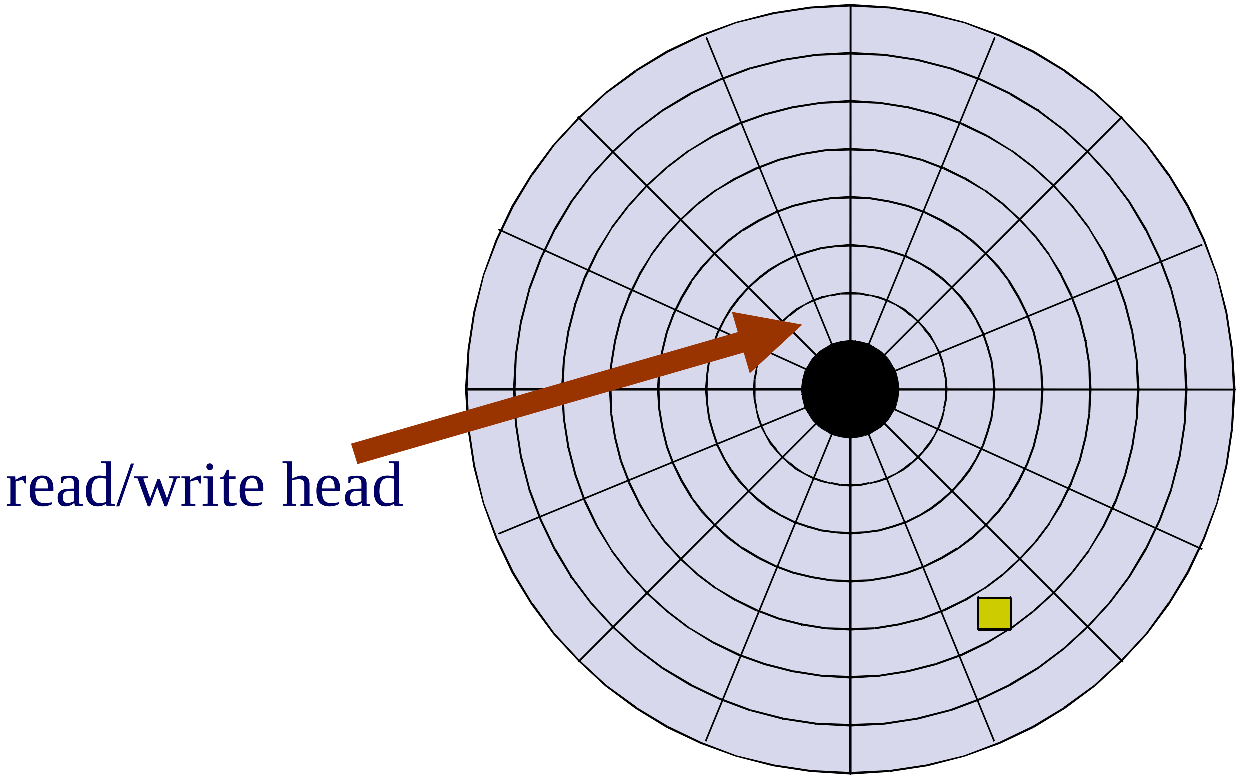
2. **Wait until the desired sector spins around ("rotational latency"/"rotational delay")**

Observe

- **We don't usually say "seek delay" or "rotational time" – careful use of jargon marks the true expert!**

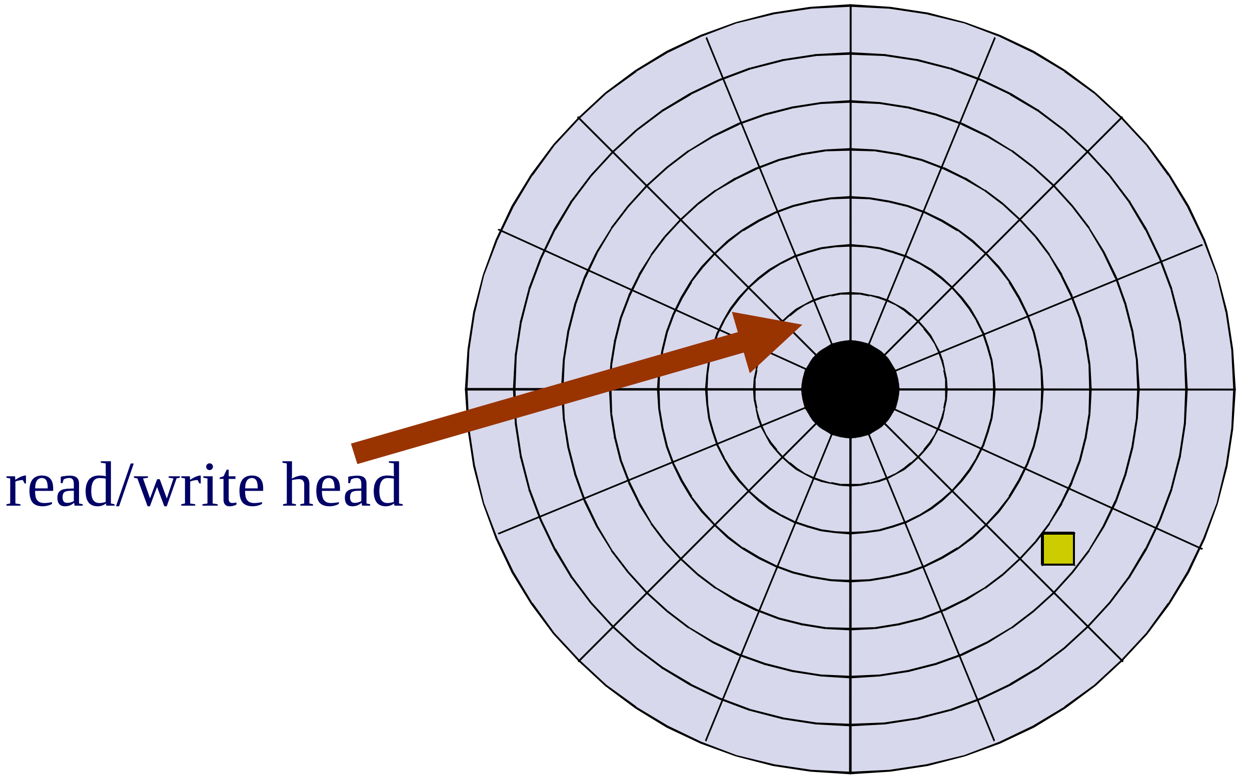
Anatomy of a Hard Drive

Let's read in a sector from the disk



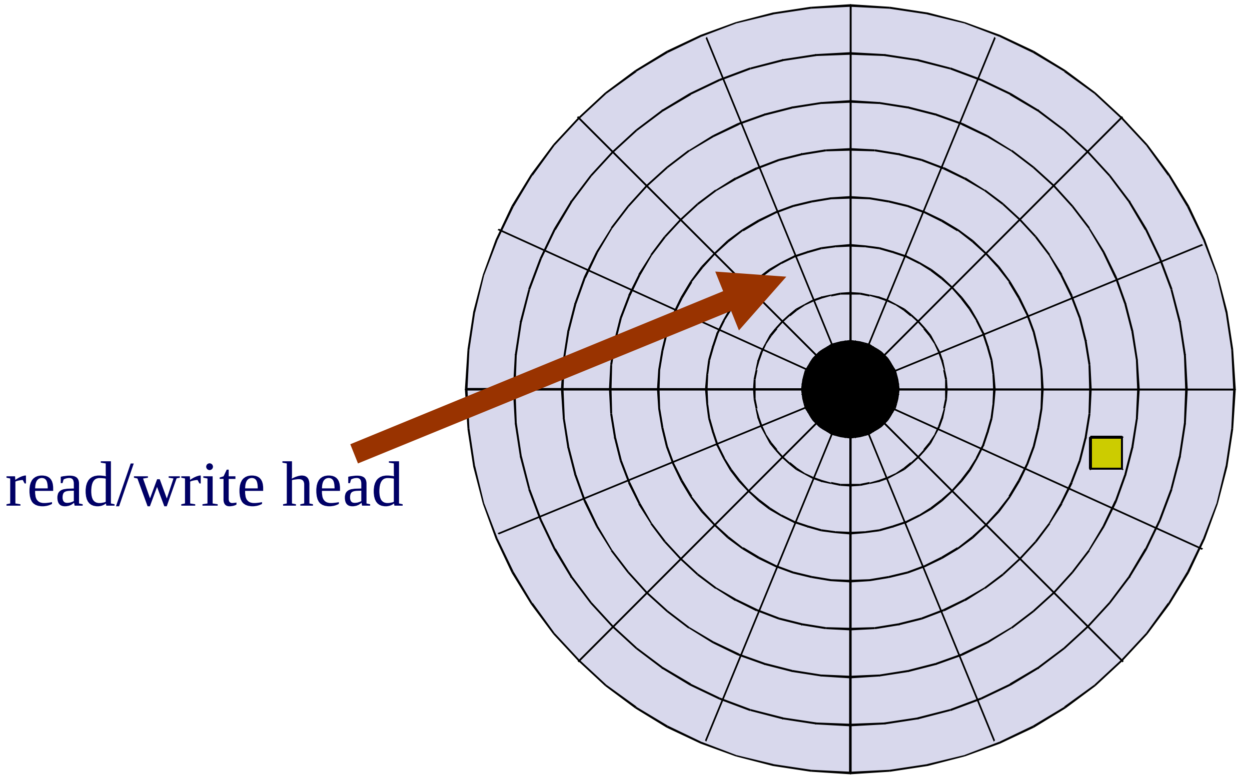
Anatomy of a Hard Drive

Let's read in a sector from the disk



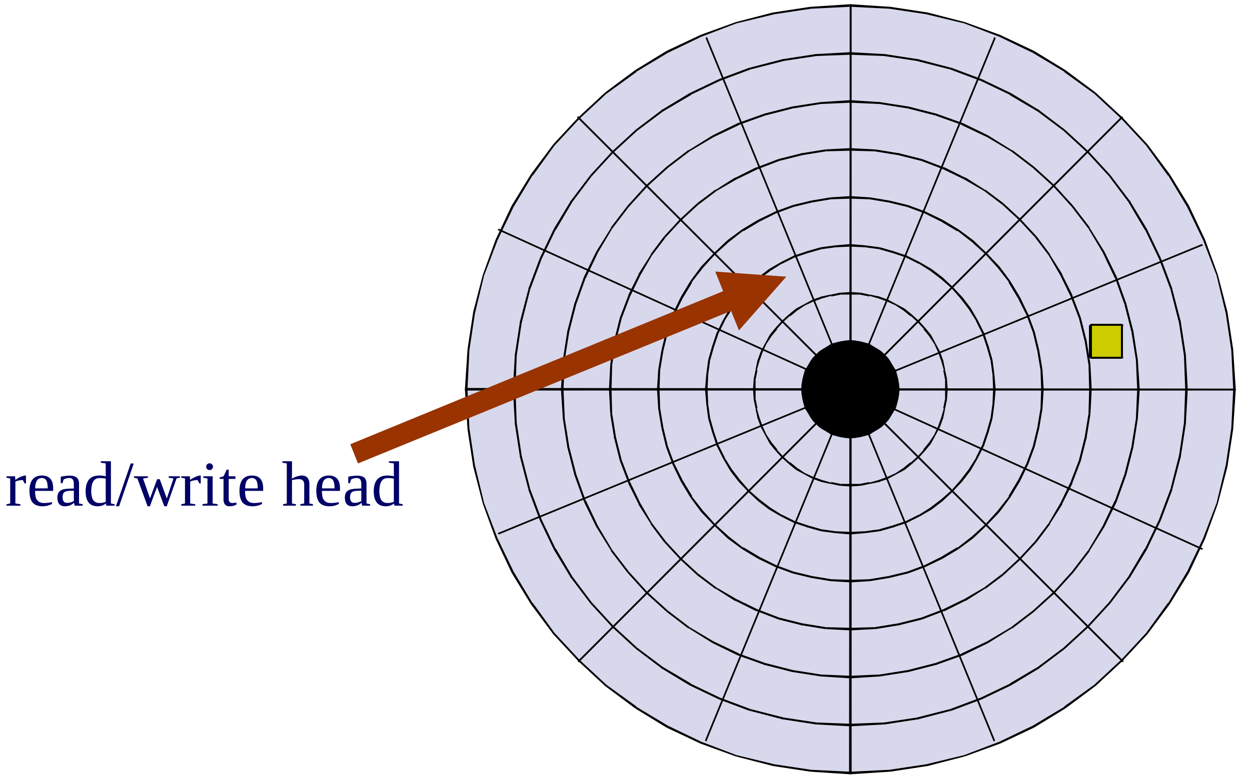
Anatomy of a Hard Drive

Let's read in a sector from the disk



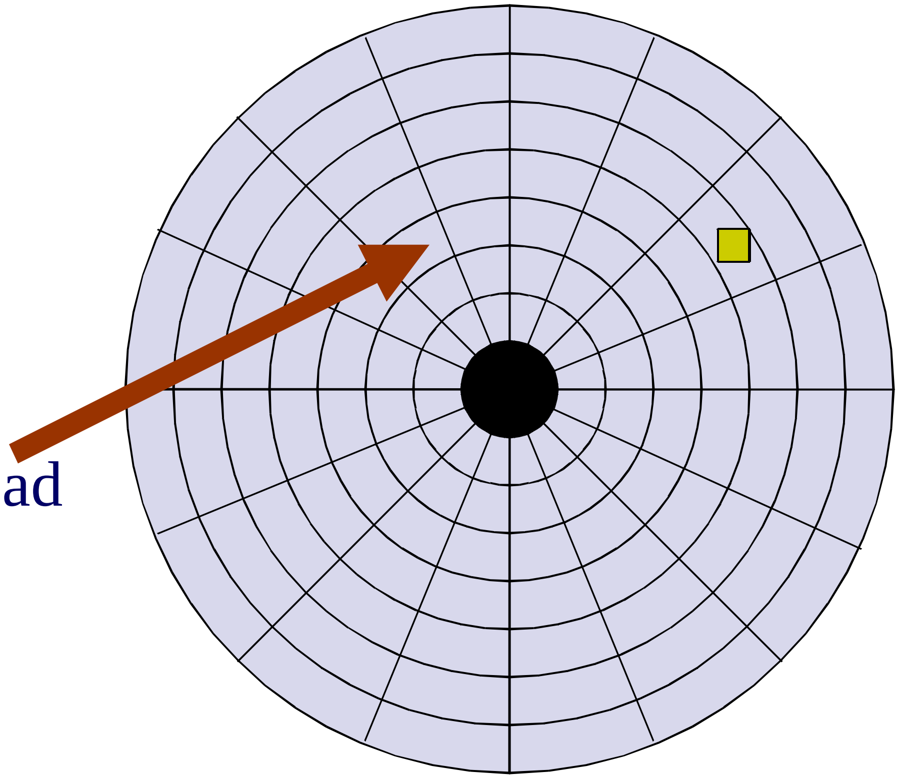
Anatomy of a Hard Drive

Let's read in a sector from the disk



Anatomy of a Hard Drive

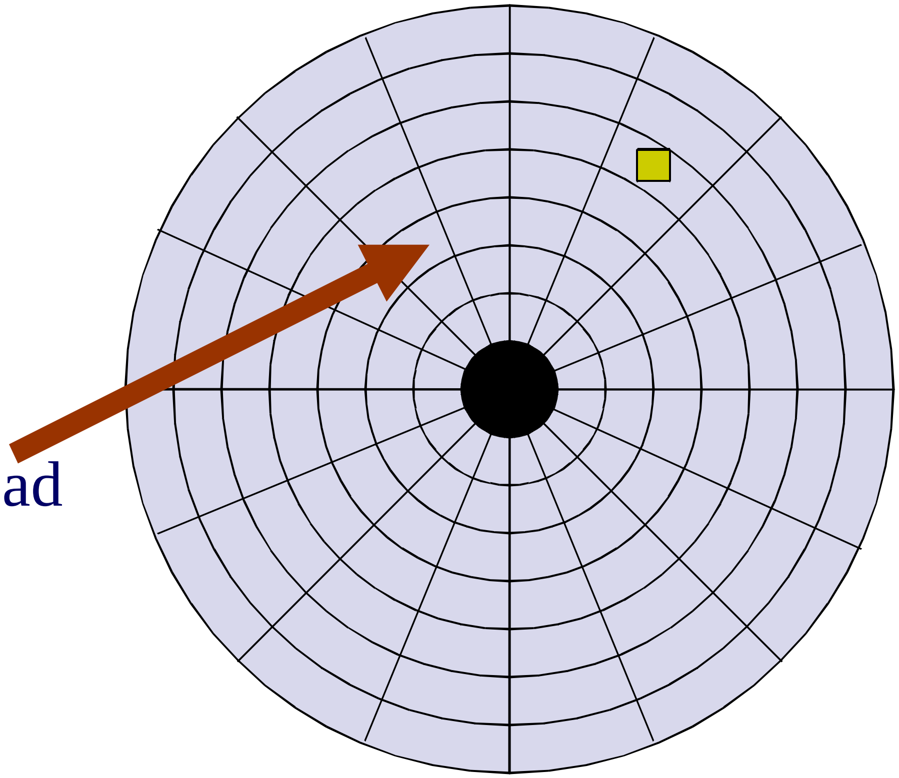
Let's read in a sector from the disk



read/write head

Anatomy of a Hard Drive

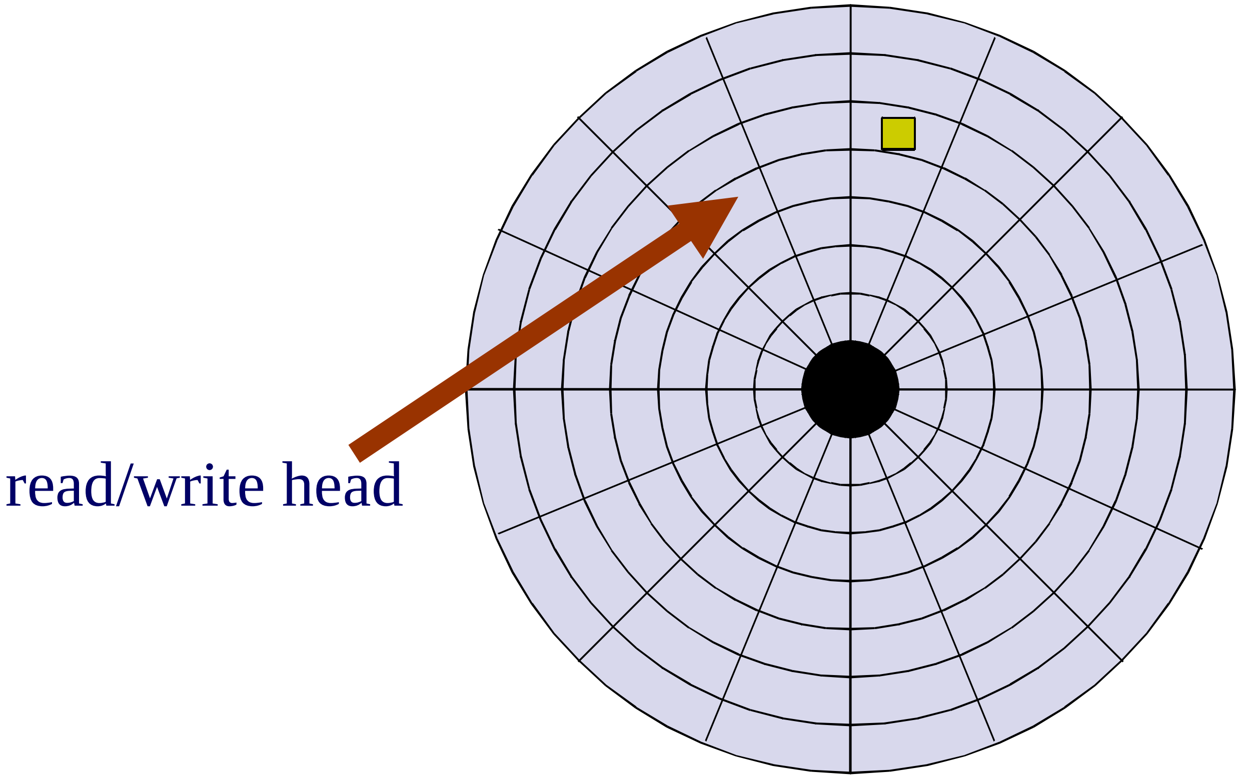
Let's read in a sector from the disk



read/write head

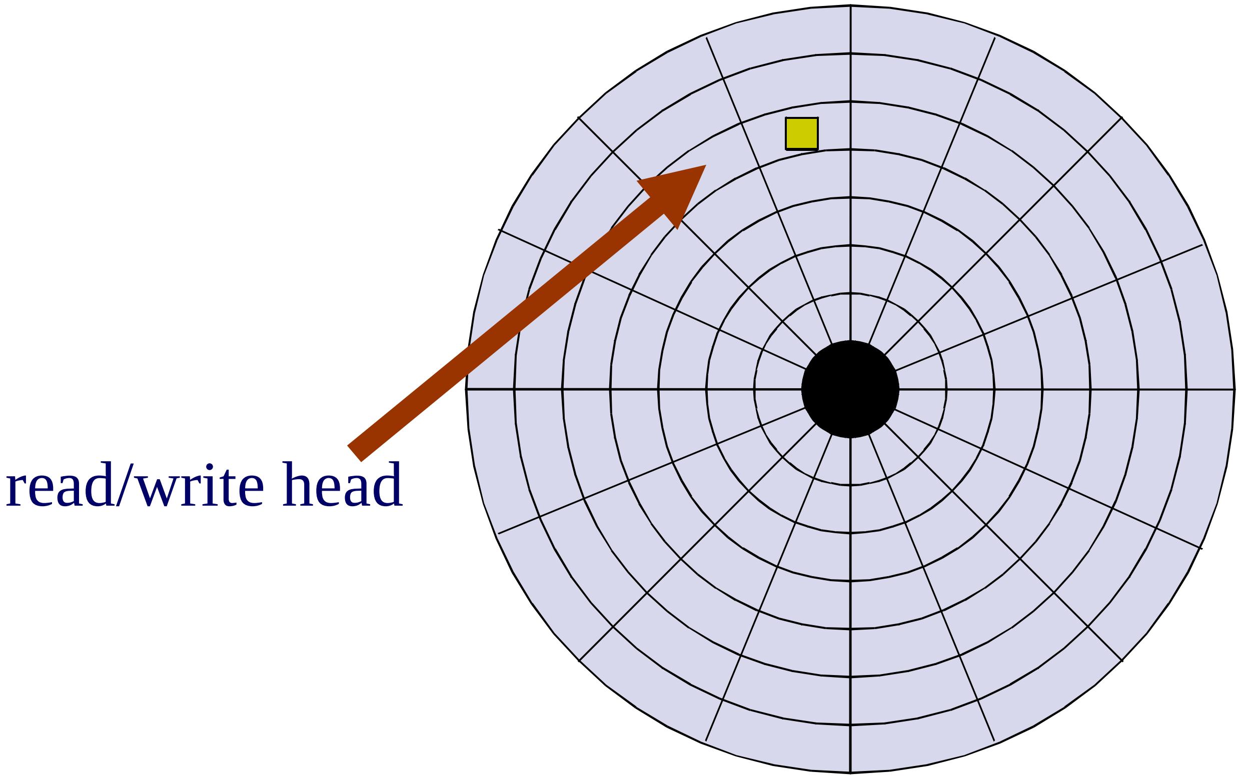
Anatomy of a Hard Drive

Let's read in a sector from the disk



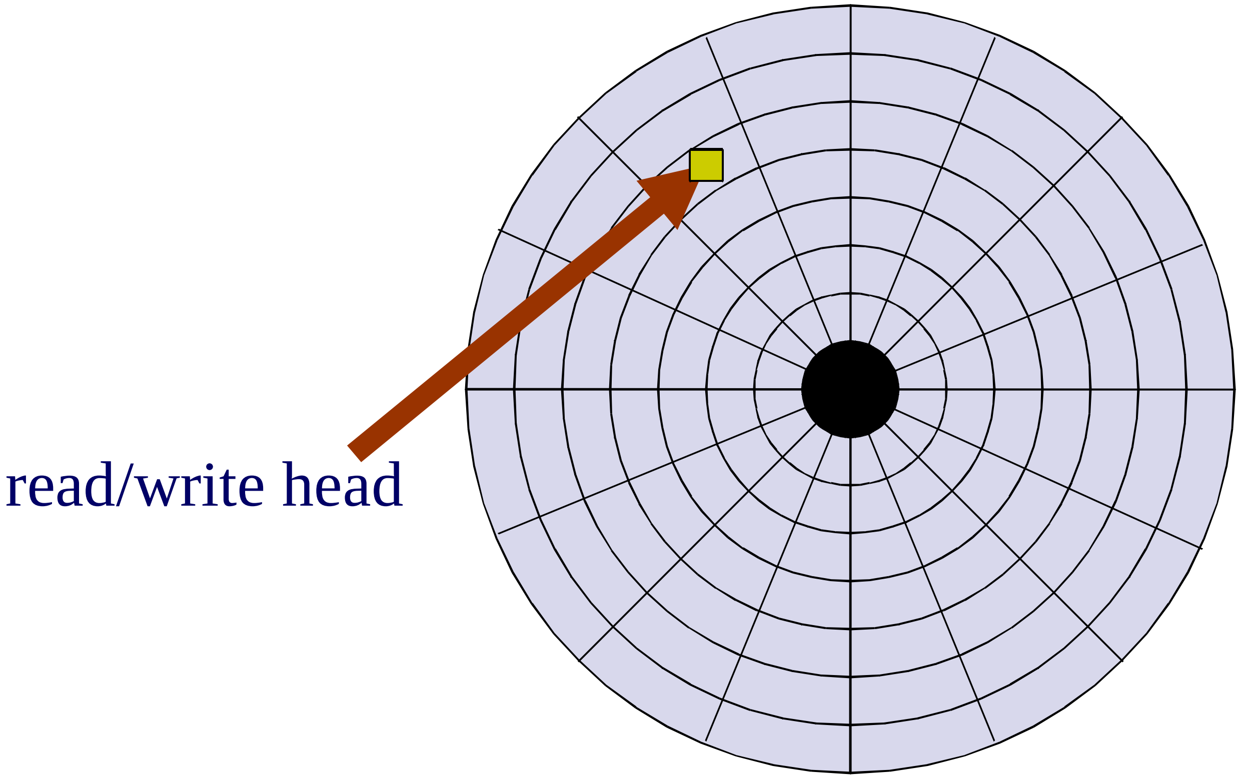
Anatomy of a Hard Drive

Let's read in a sector from the disk



Anatomy of a Hard Drive

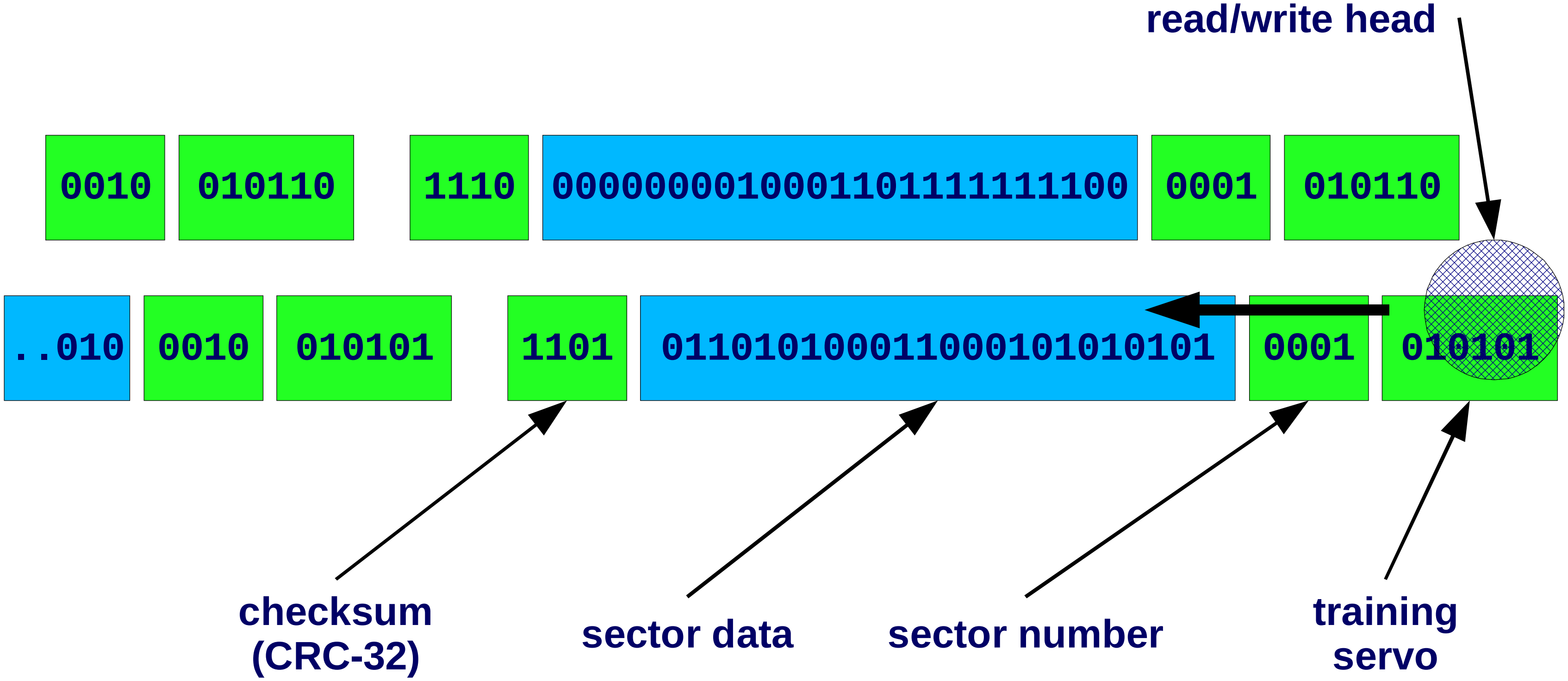
Let's read in a sector from the disk



Anatomy of a "Sector"

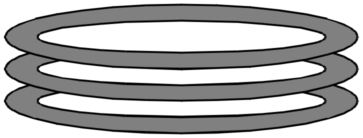
Finding a sector involves real work

Correct track; check sector header for number; check CRC...



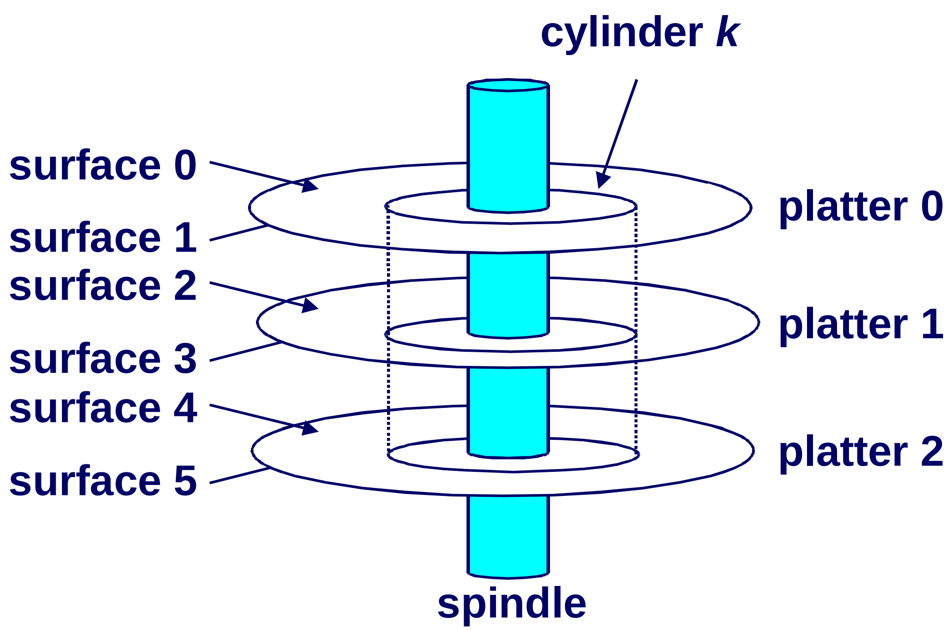
Disk Cylinder

Matching tracks across surfaces are collectively called a *cylinder*



Disk Cylinder

Matching tracks form a cylinder.



Cheap Access Within A Cylinder

Heads share one single arm

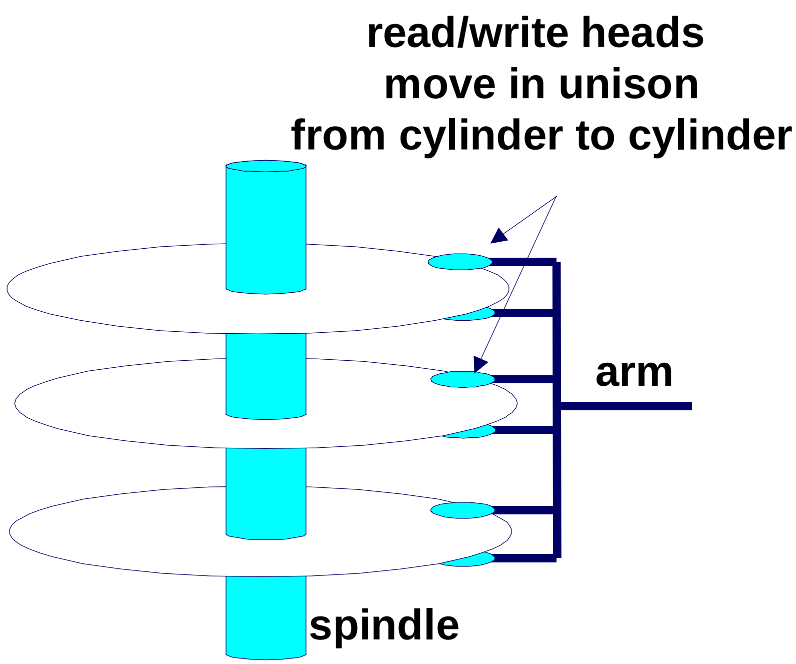
- All heads always on same cylinder

Switching heads is "cheap"

- Deactivate head 3
- Activate head 4
- Wait for 1st sector header

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”, and is ~10ms for a 5400 rpm disk

We will also must wait half a rotation, on average

- The time to do this is rotational delay, and on a 5400 rpm drive is ~5.5ms

Seagate 7200.7, a modern-ish 7200 RPM SATA drive

- Average seek time 8.5 ms
- Average rotational delay 4.16 ms

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 ~10 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 quoted at *tens 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 quoted at *tens 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 (IDE/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

Oh, well

- Older OS's tried to understand disk layout
- Modern OS's just assume nearby sector numbers are close
- Experimental OS's try to understand disk layout again
- Next few slides assume “modern”, not “old”/“experimental”

Scheduling Algorithms

"Don't try this at home"

- FCFS
- SSTF

Arguably less wrong

- SCAN, LOOK

Plausible

- C-SCAN, C-LOOK

Useful, but hard

- SPTF, WSPTF

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, ...**

“Fair, but cruel”

- **“Don't try this at home”**

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**

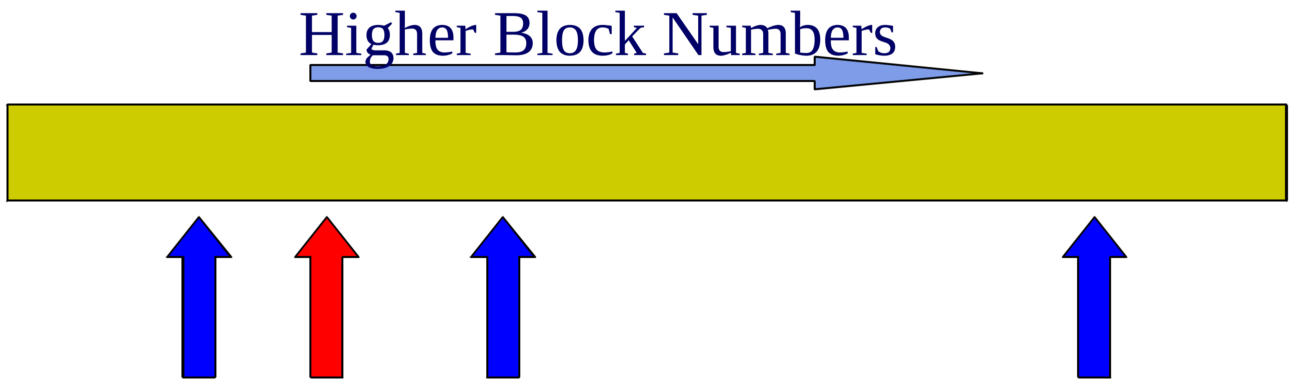
Intolerable response time *variance*, however

Why?

SSTF

Blue are requests

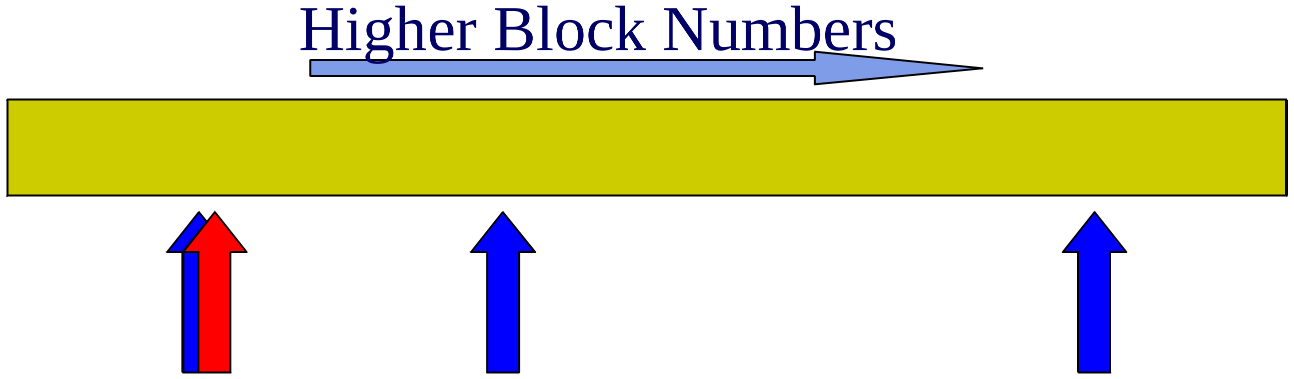
Yellow is disk



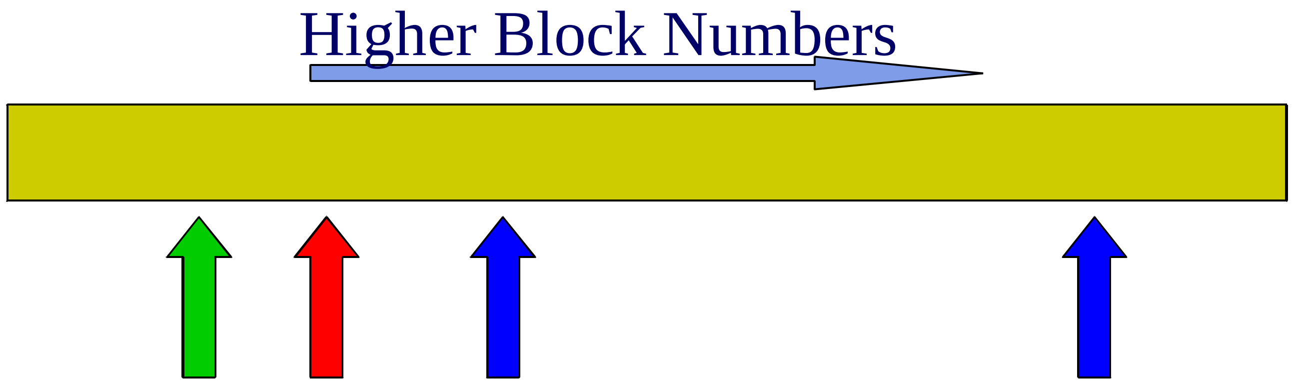
Red is disk head

Green is completed requests

SSTF

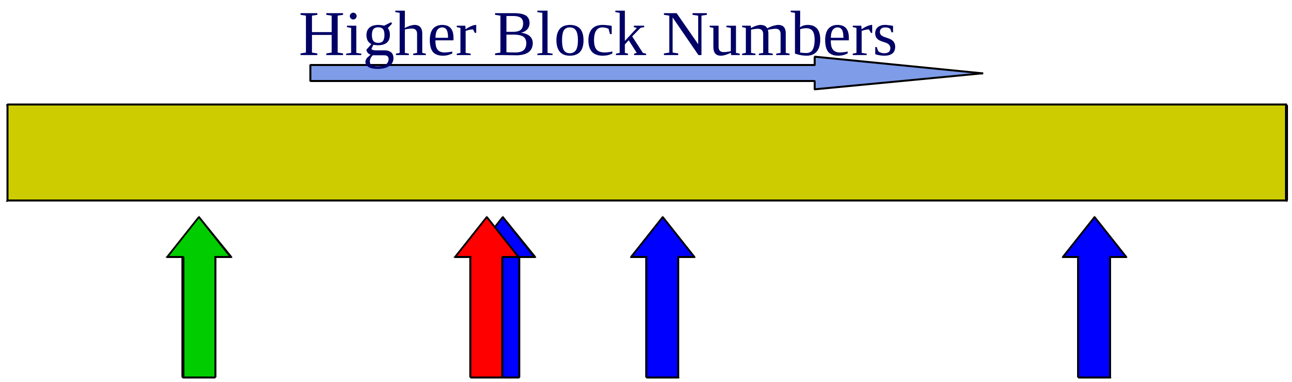


SSTF

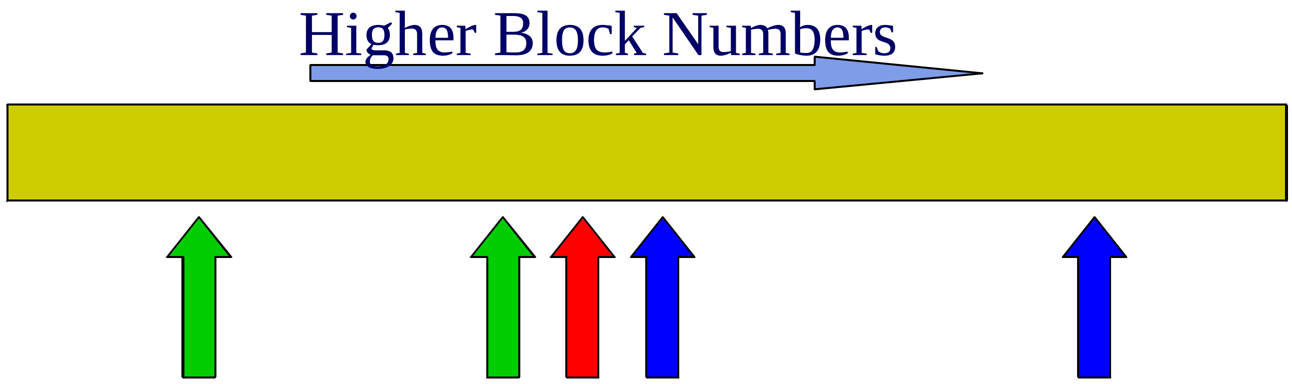


SSTF

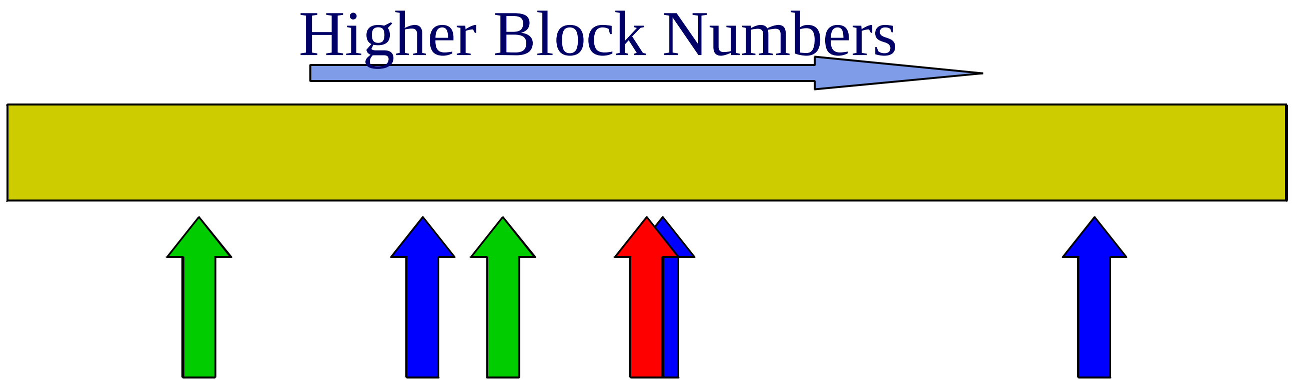
New Requests arrive…



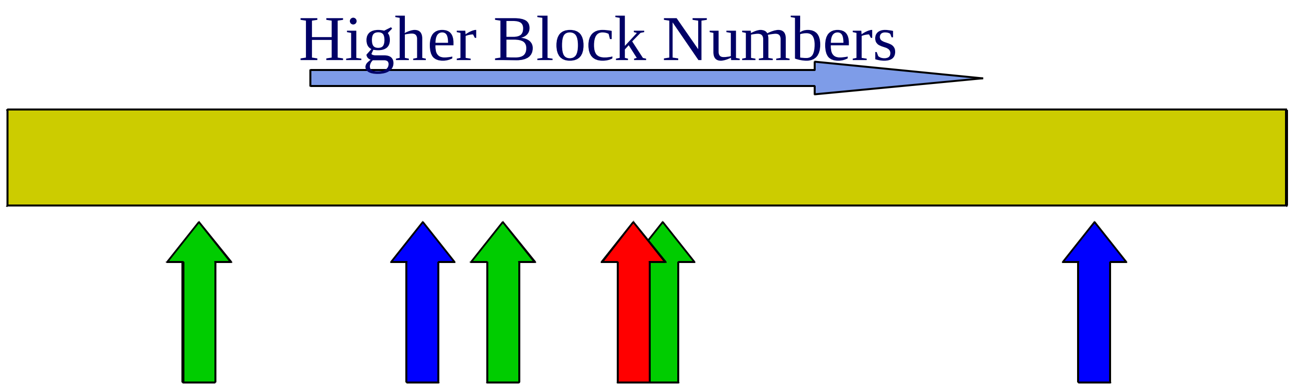
SSTF



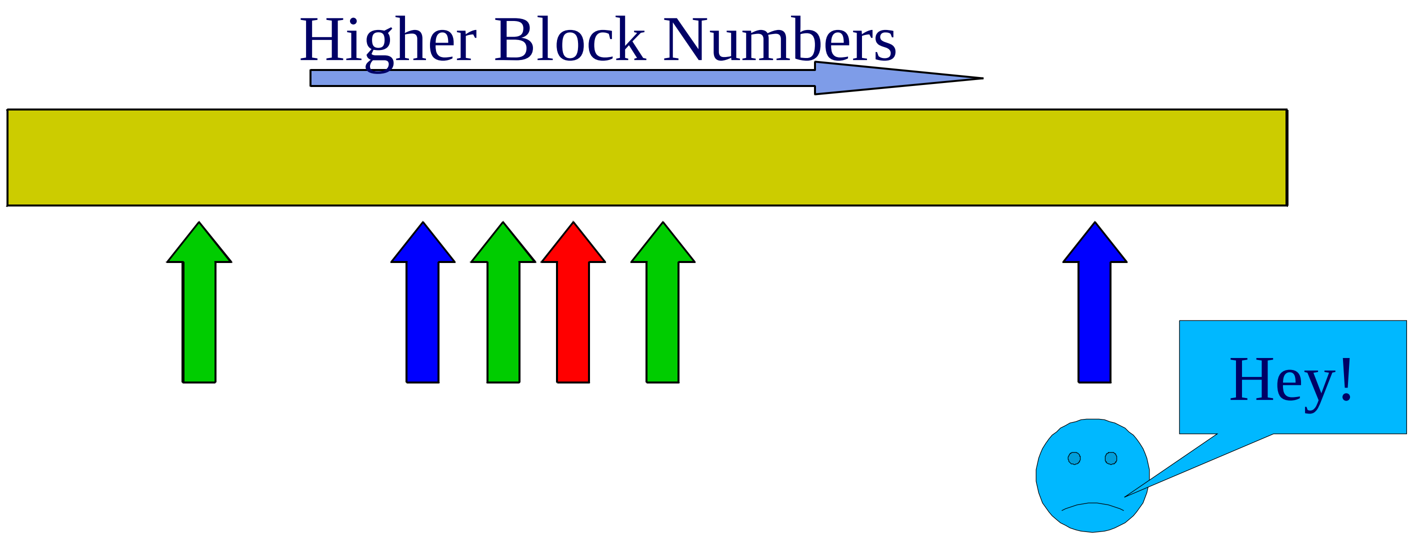
SSTF



SSTF

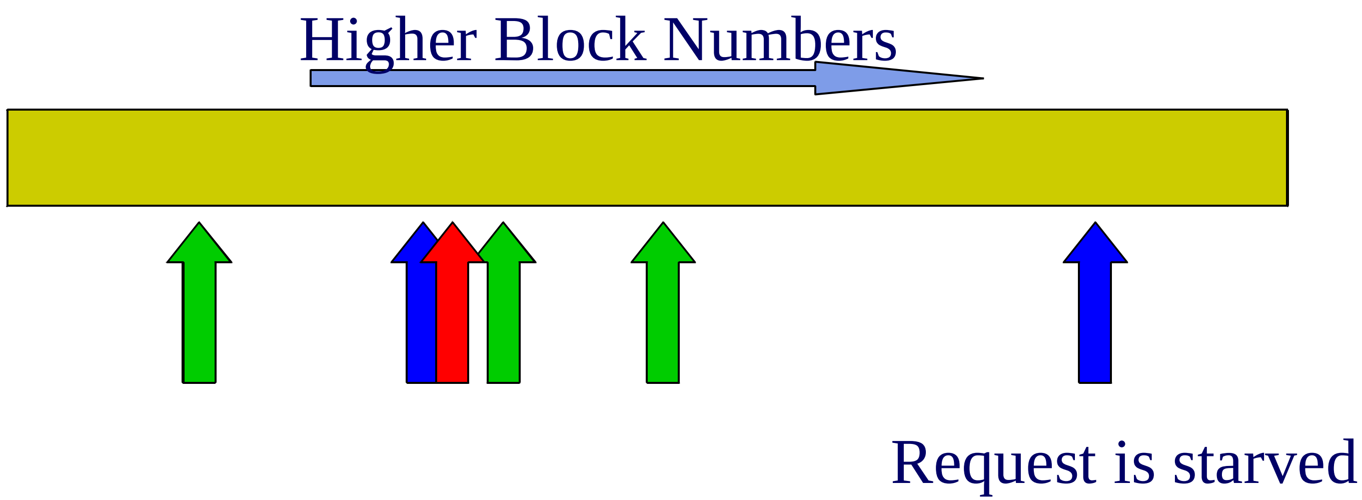


SSTF



SSTF

Starves requests that are “far away” from the head



What Went Wrong?

FCFS - “fair, but cruel”

- Ignores position of disk arm, very 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 cruel”

- Ignores position of disk arm, very 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 –Queue Management

Doubly-linked ordered list of requests

- Insert according to order

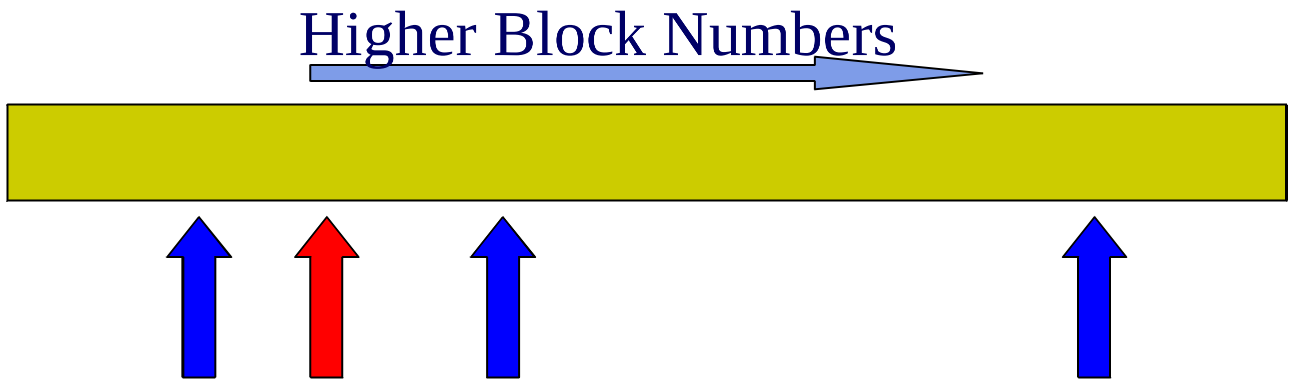
Bi-directional scanning

- Direction = +1 or -1
- Tell disk: “seek to cylinder X=current+direction”
- Examine list for requests in cylinder X, serve them
- If X == 0 or X == max
  - direction = -direction
- Else
  - current = X

SCAN

Blue are requests

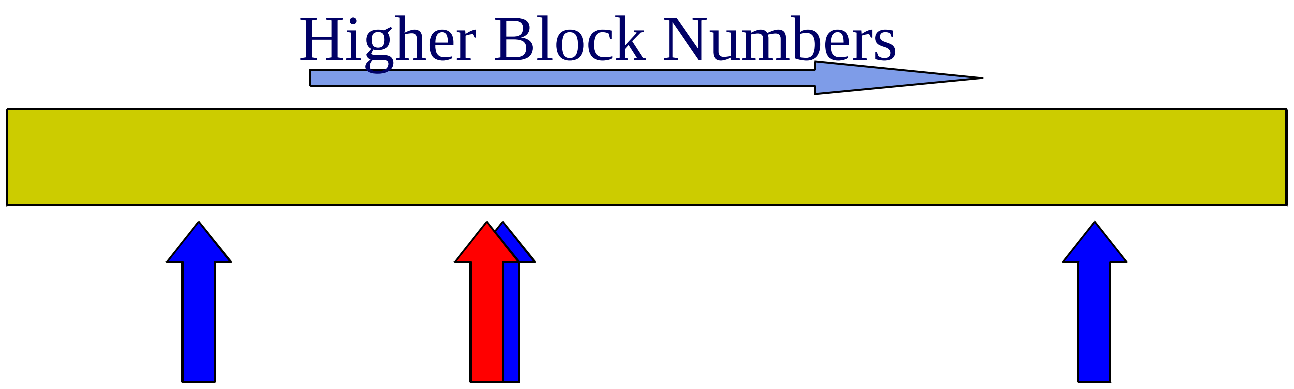
Yellow is disk



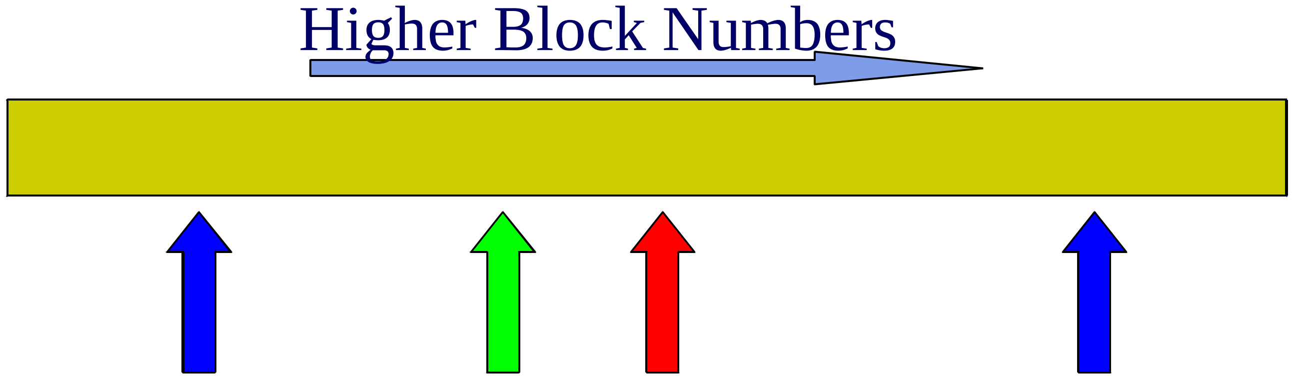
Red is disk head

Green is completed requests

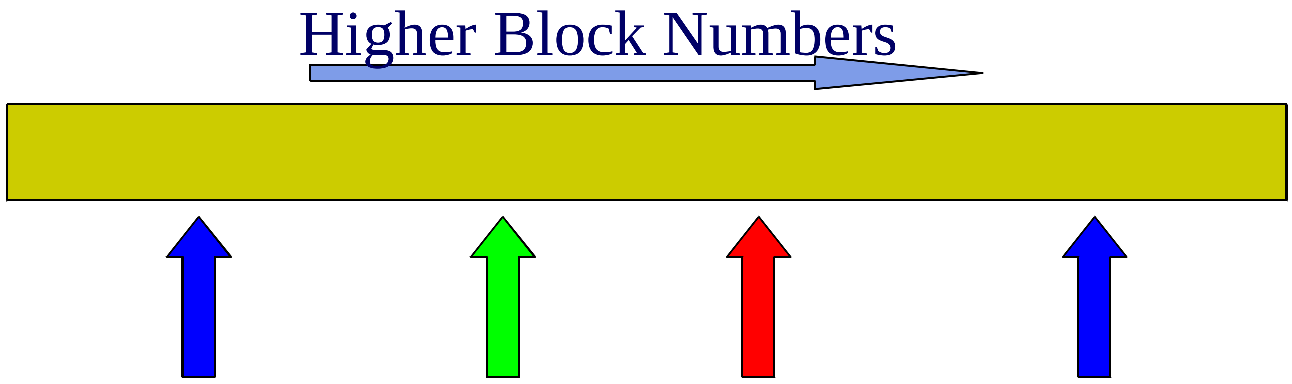
SCAN



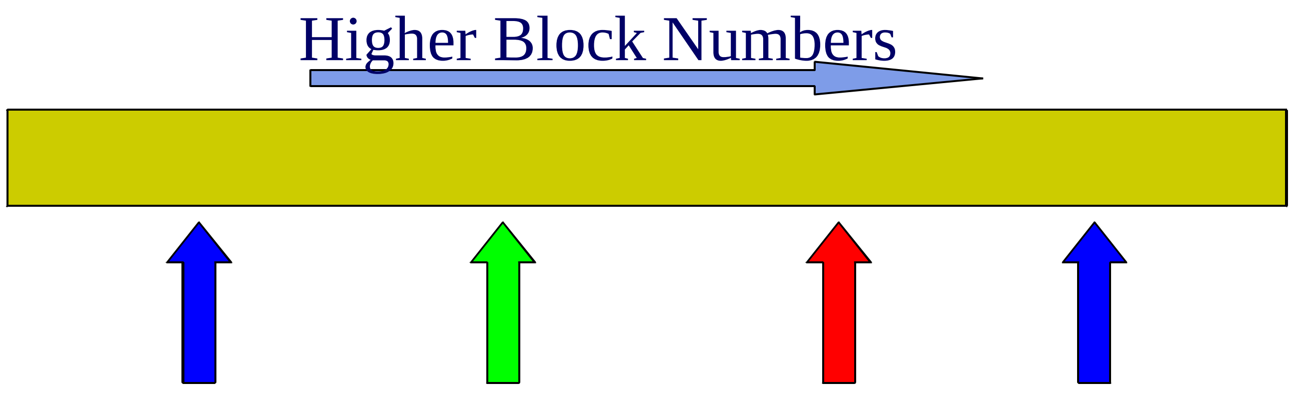
SCAN



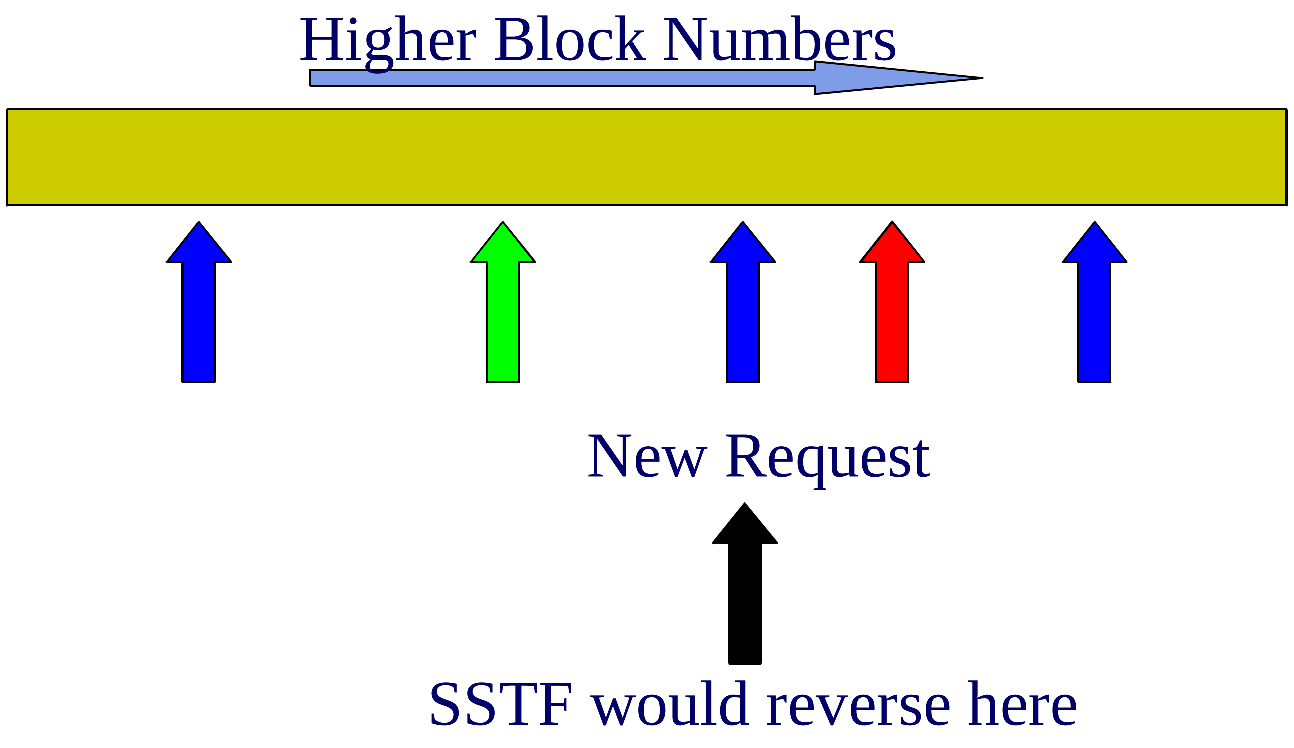
SCAN



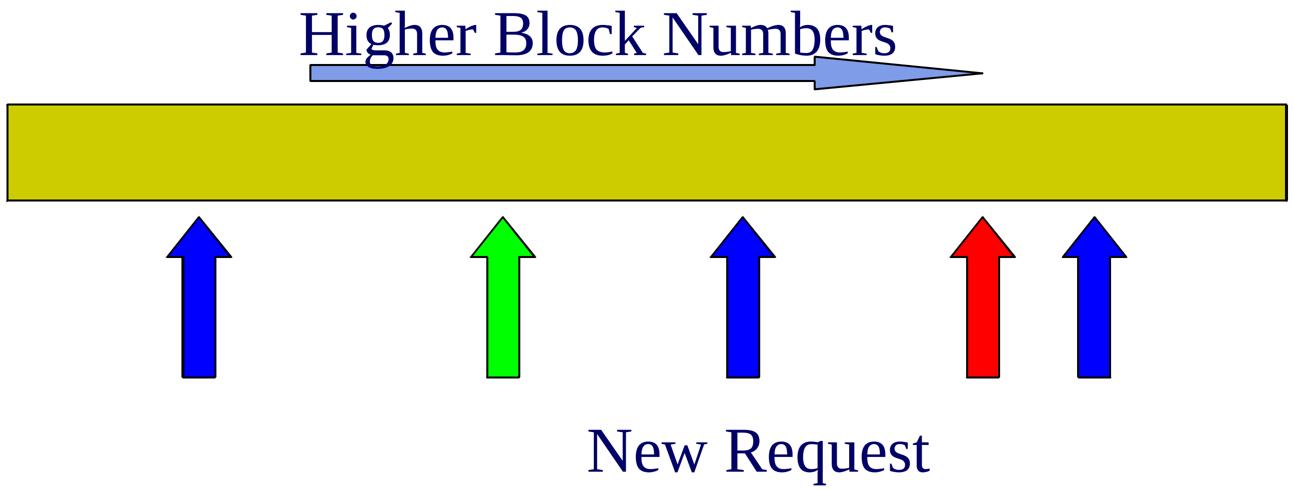
SCAN



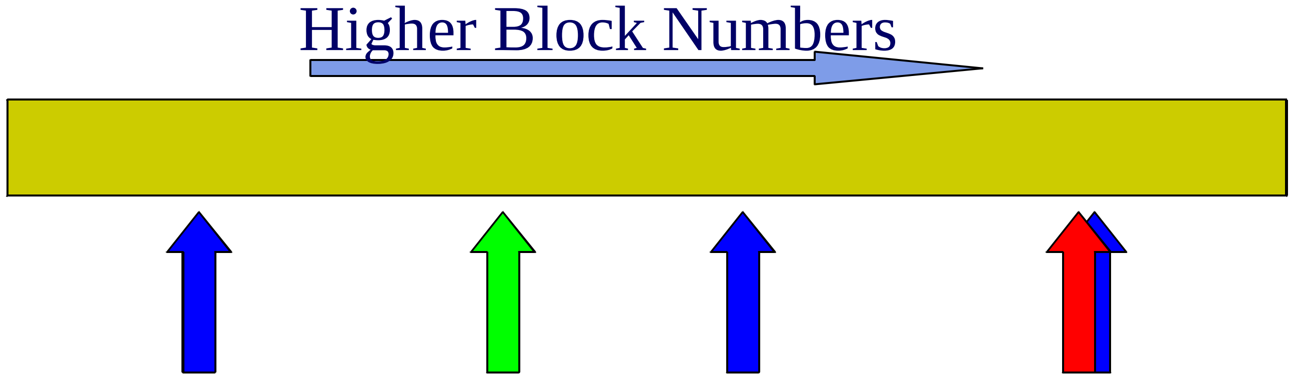
SCAN



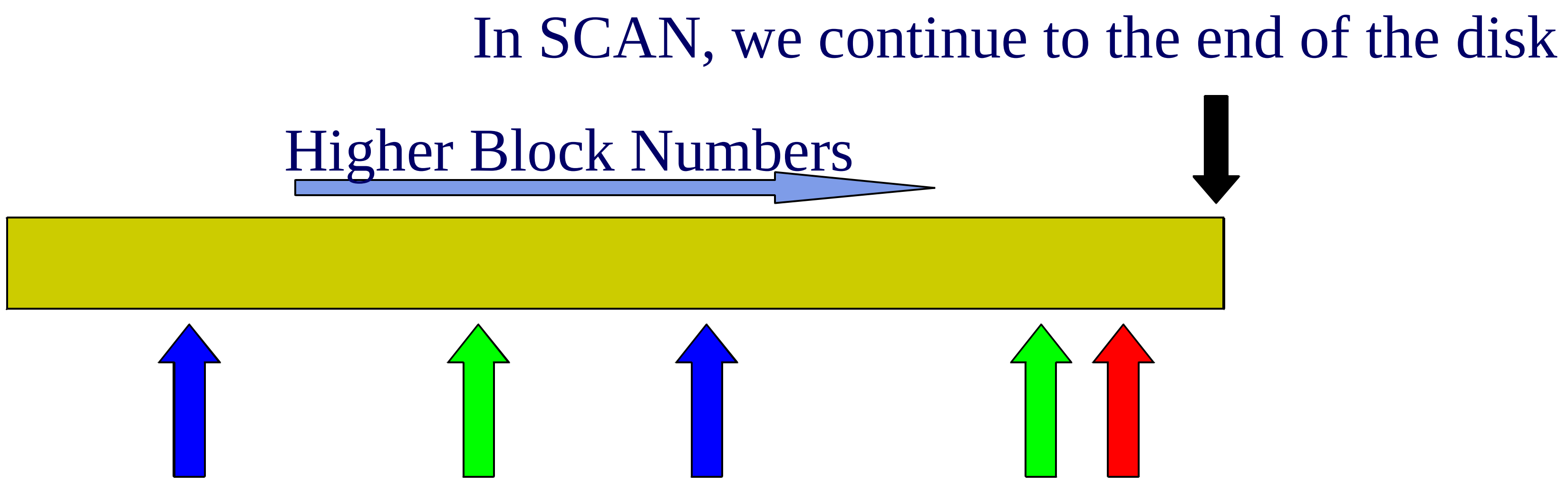
SCAN



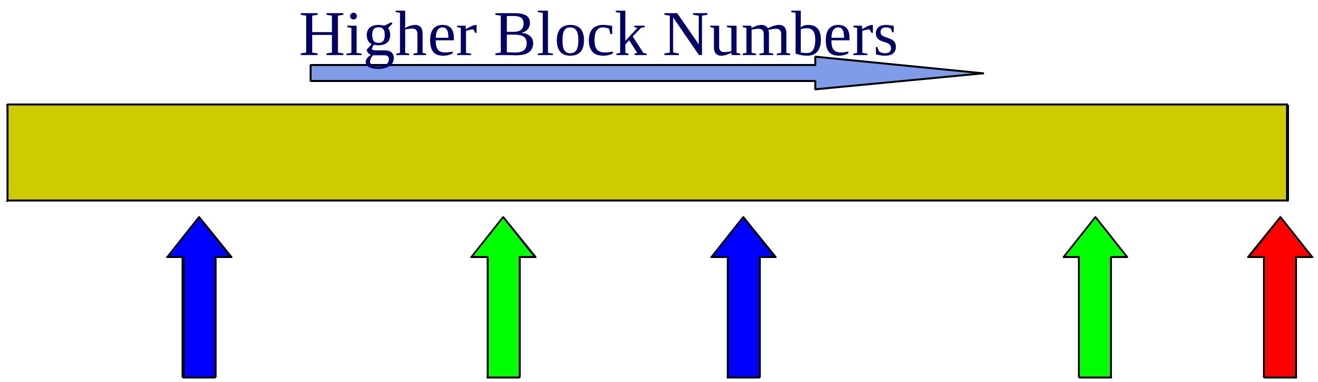
SCAN



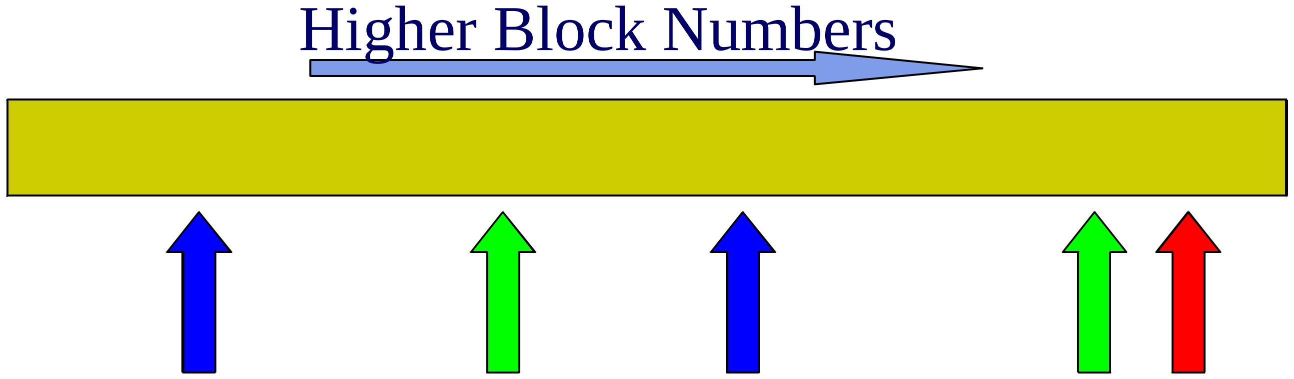
SCAN


In SCAN, we continue to the end of the disk
Higher Block Numbers

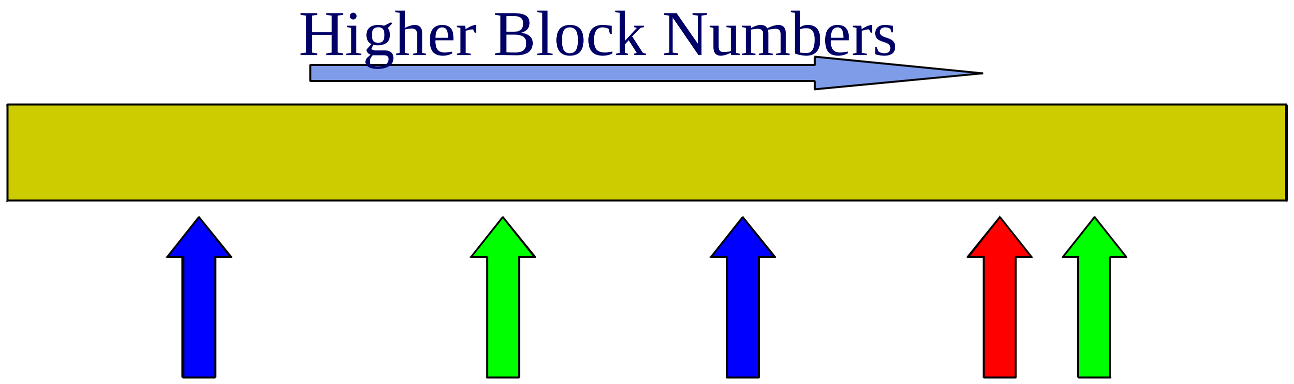
SCAN



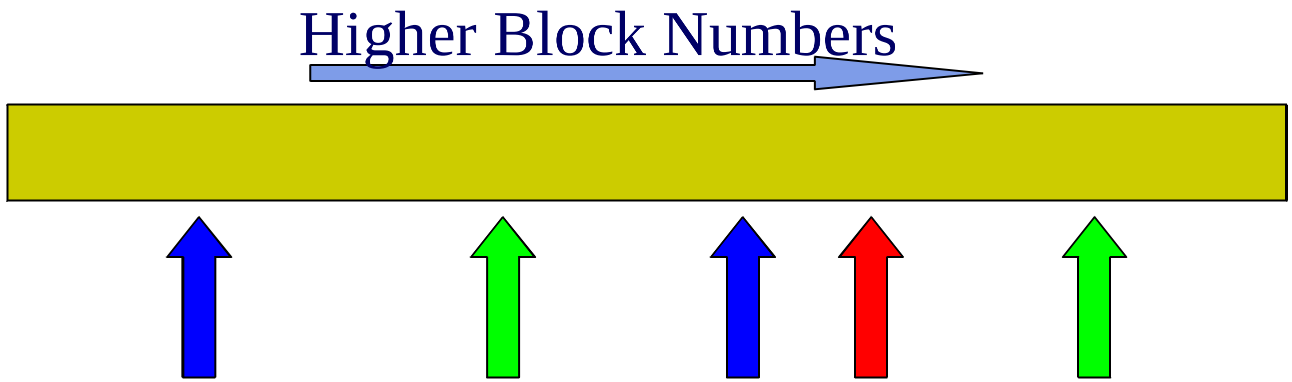
SCAN



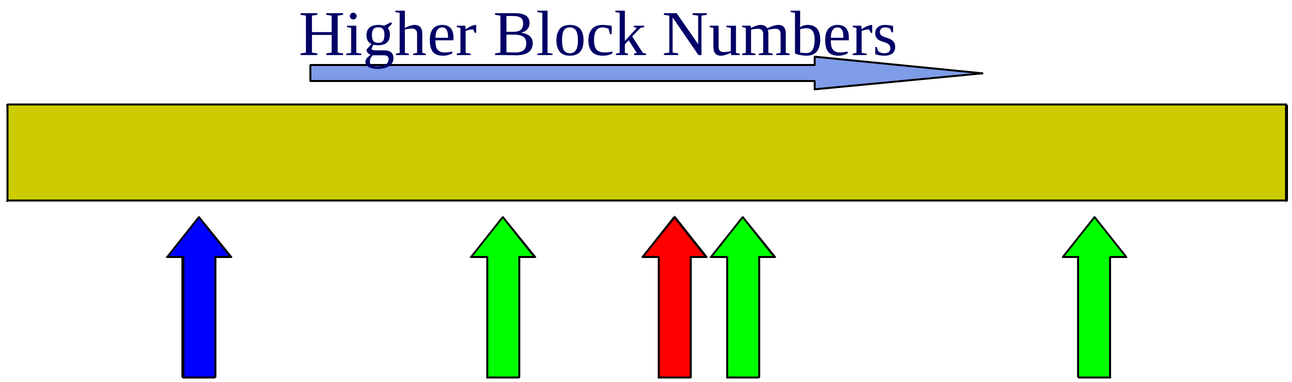
SCAN



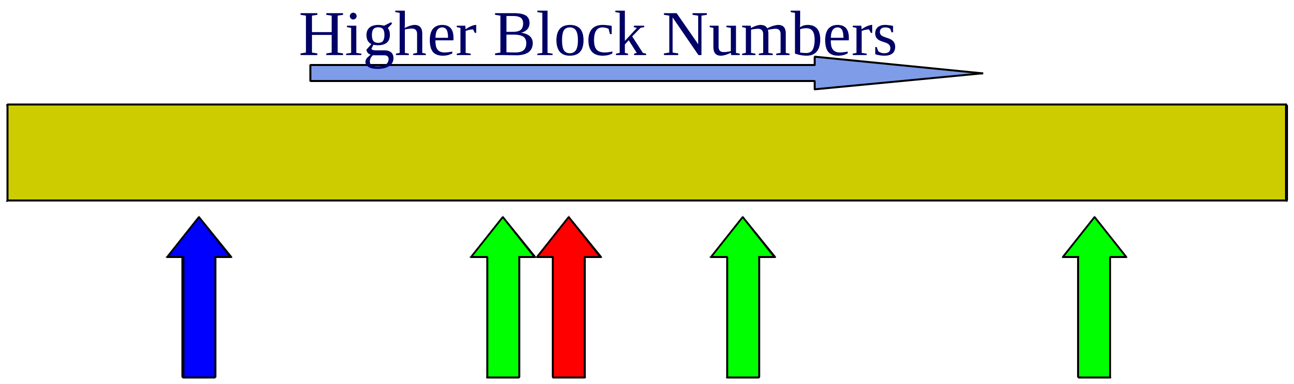
SCAN



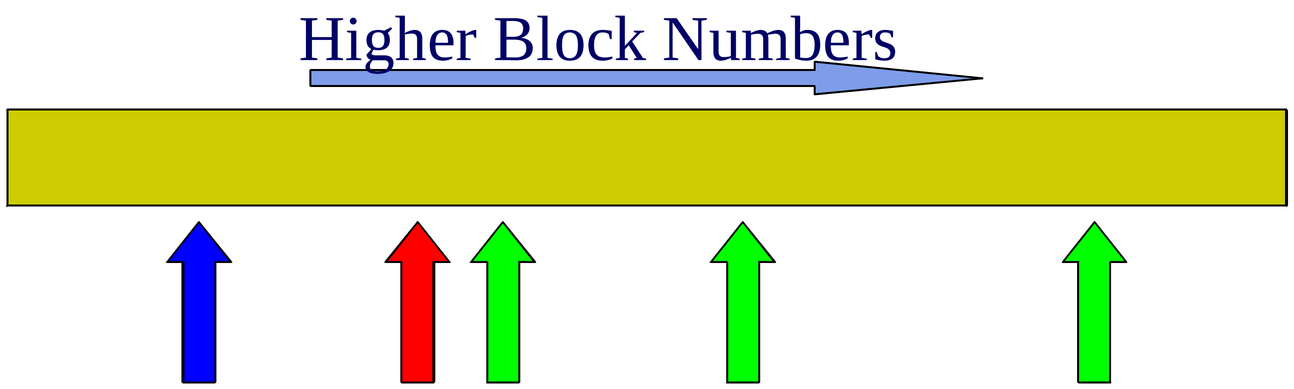
SCAN



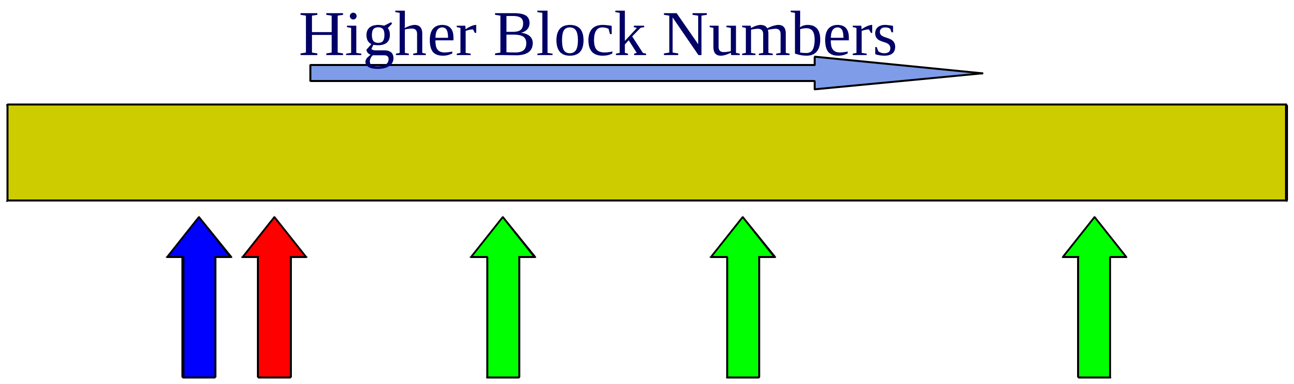
SCAN



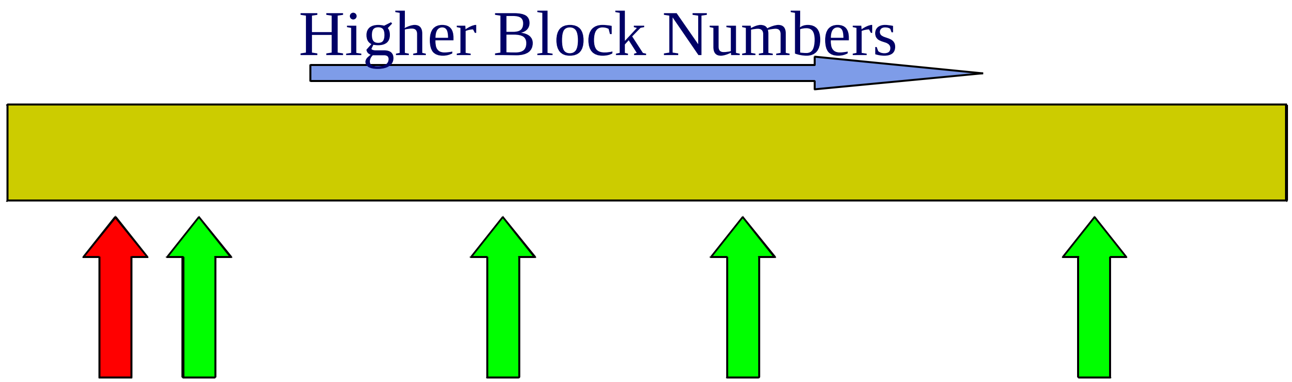
SCAN



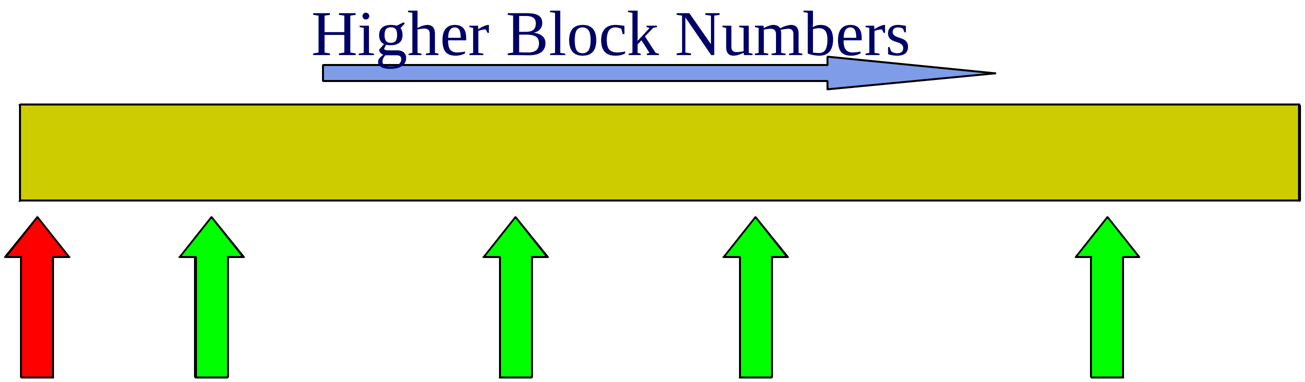
SCAN



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

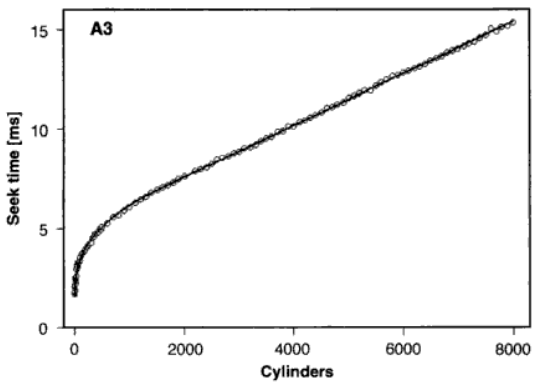
Long seek is amortized across all accesses

- **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

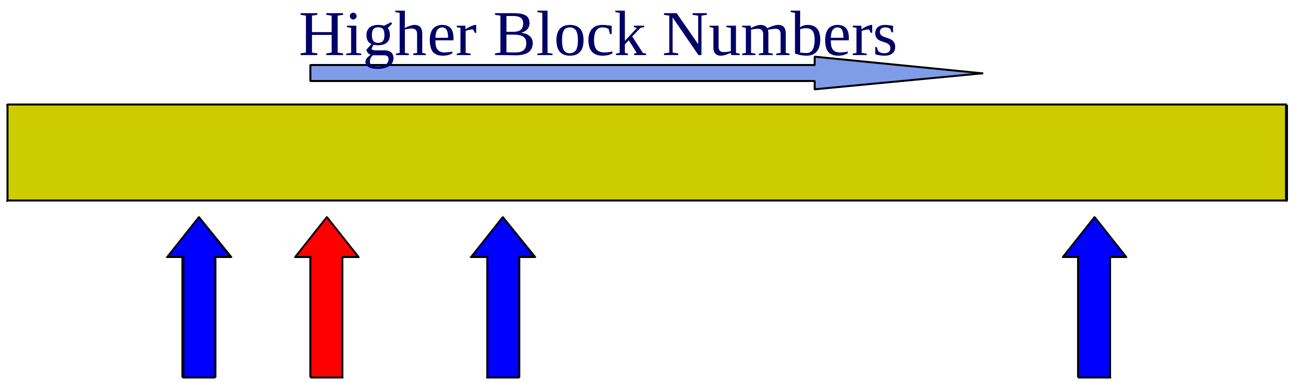
CSCAN + LOOK

Scan in one direction, as in CSCAN

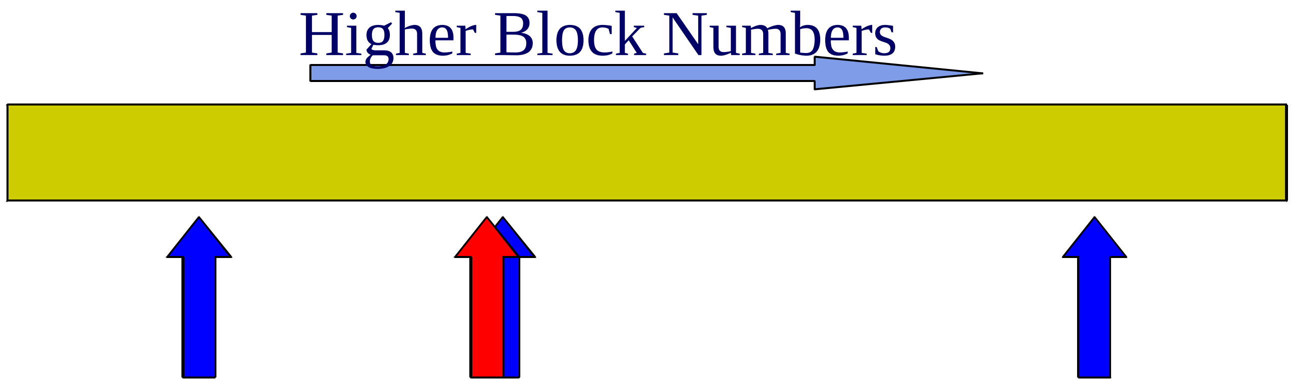
If there are no more requests in current direction go back to furthest request

Very popular

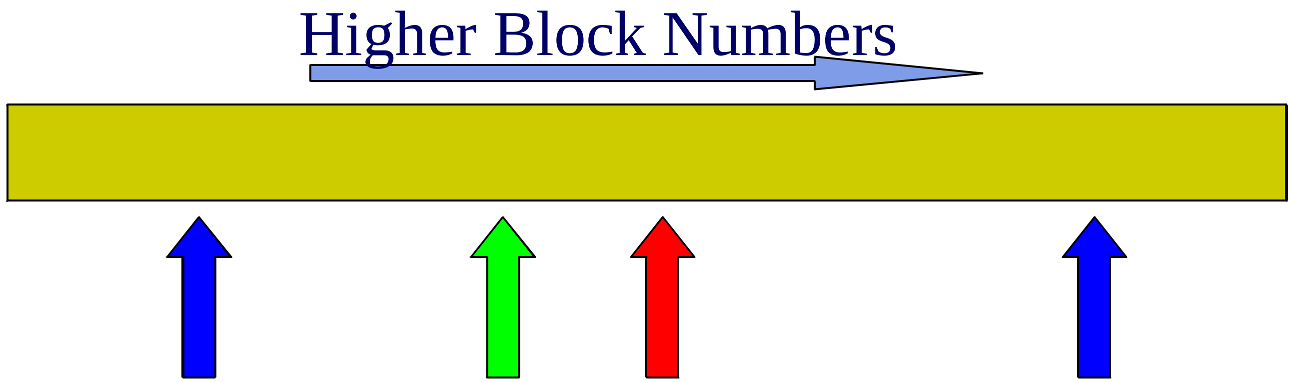
C-LOOK



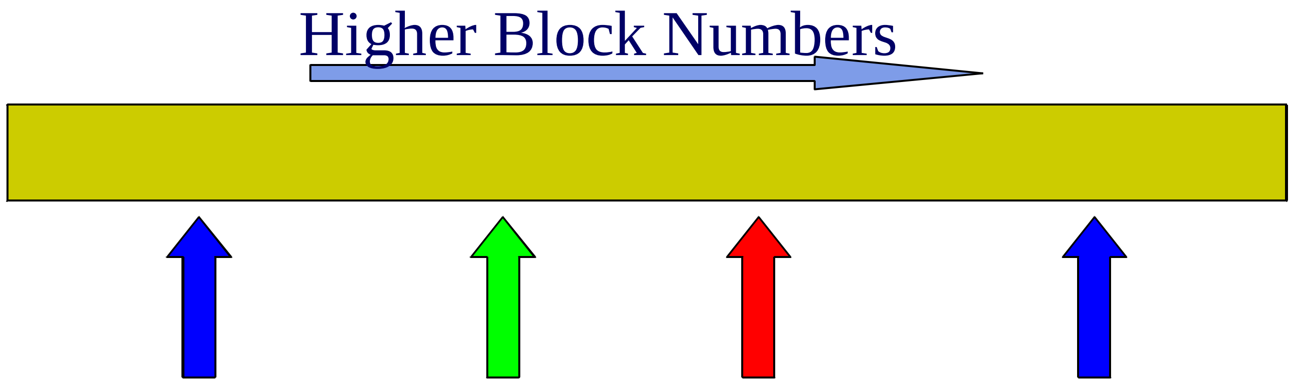
C-LOOK



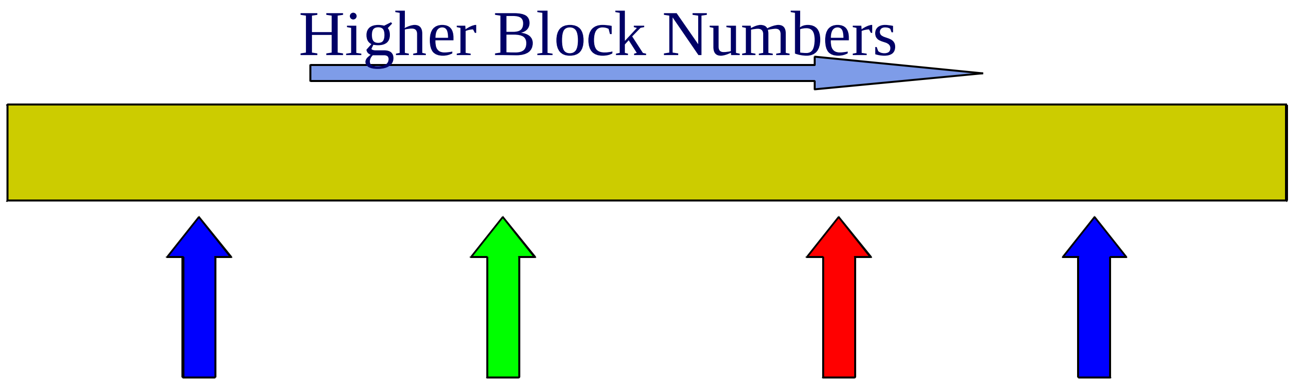
C-LOOK



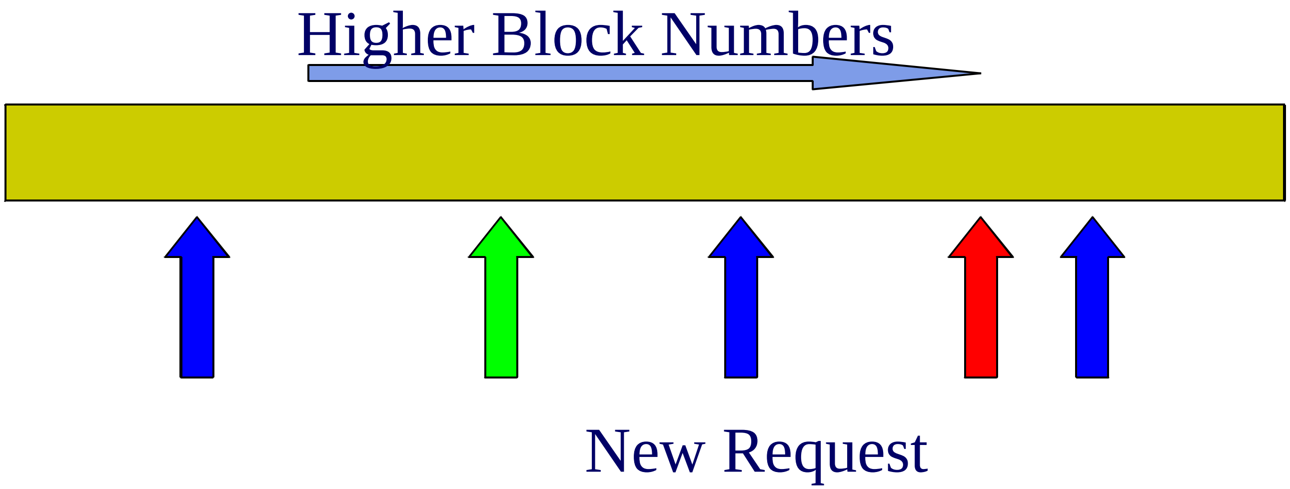
C-LOOK



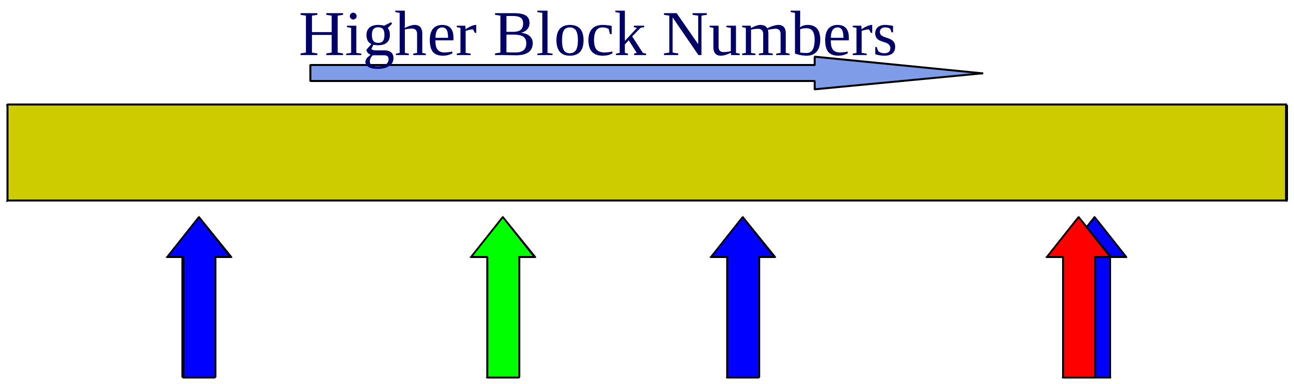
C-LOOK



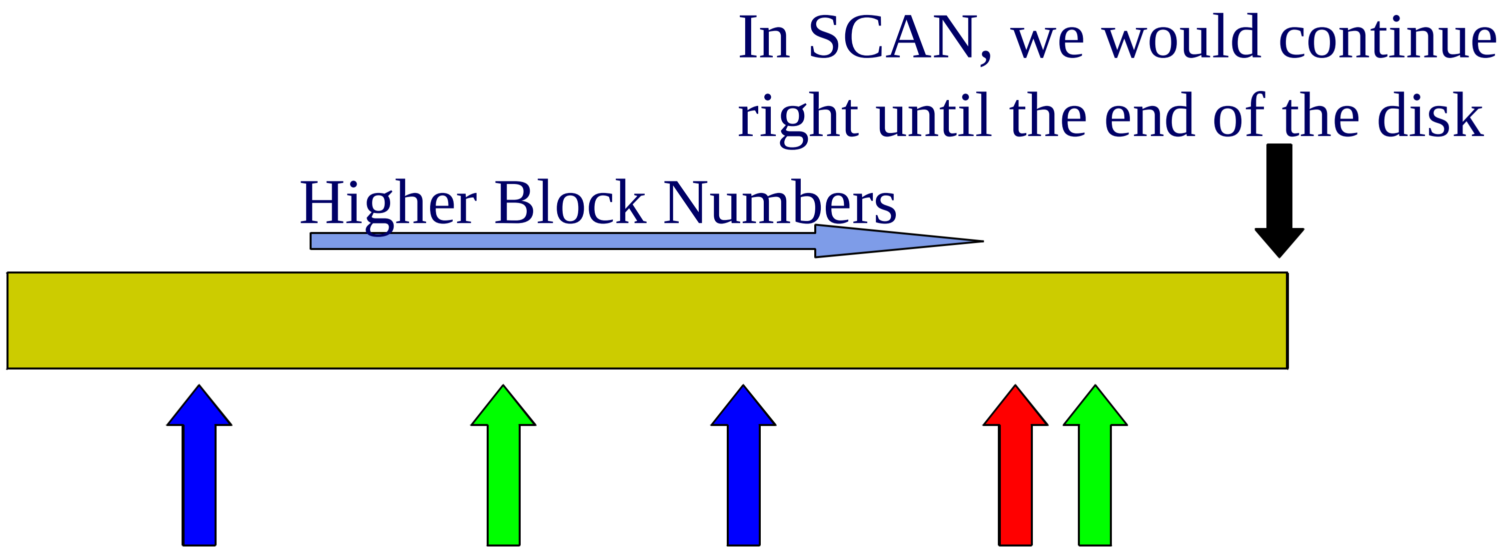
C-LOOK



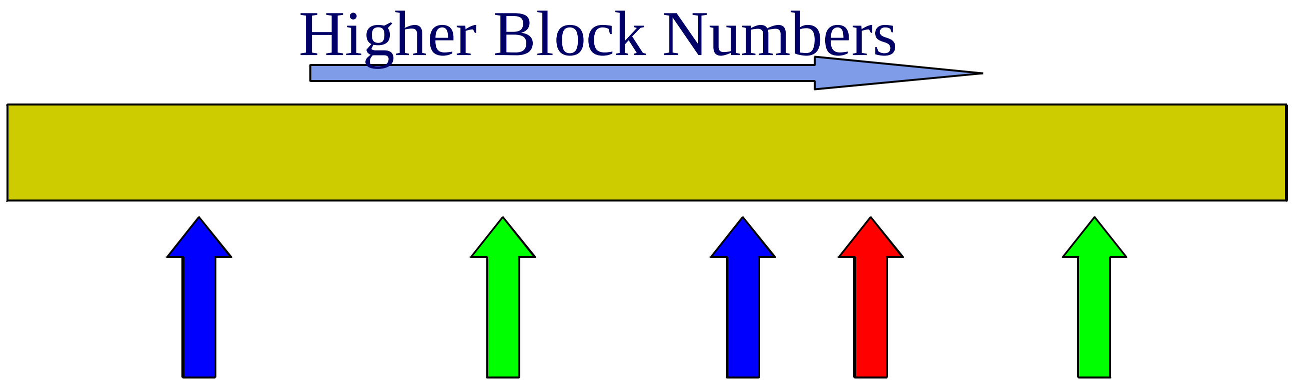
C-LOOK



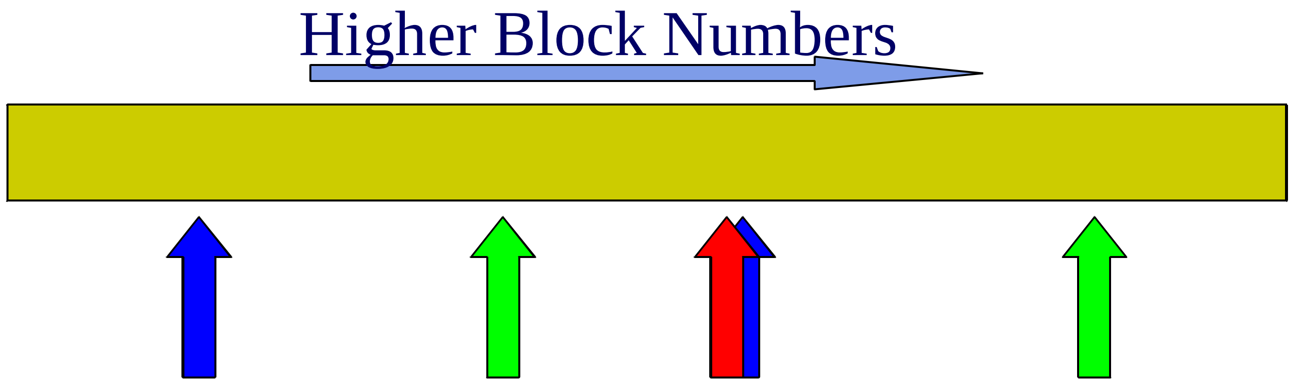
C-LOOK


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

C-LOOK

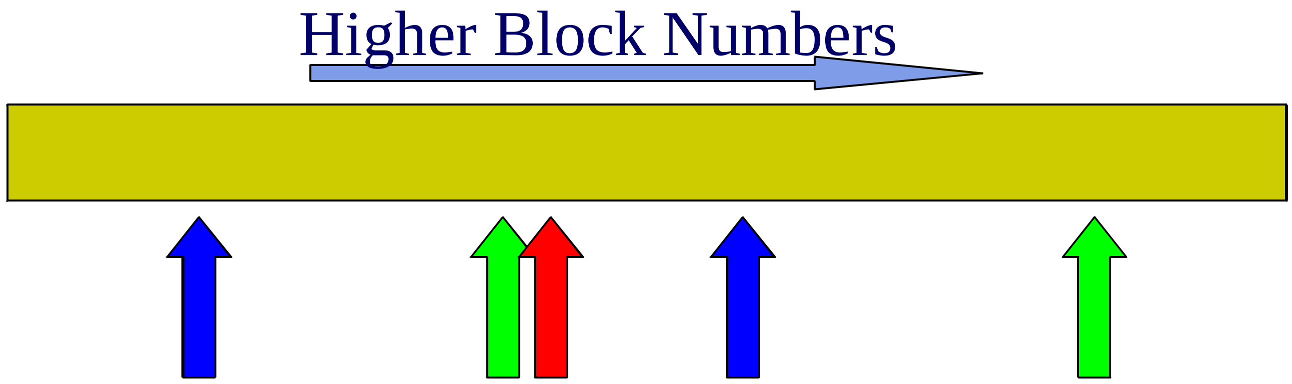


C-LOOK

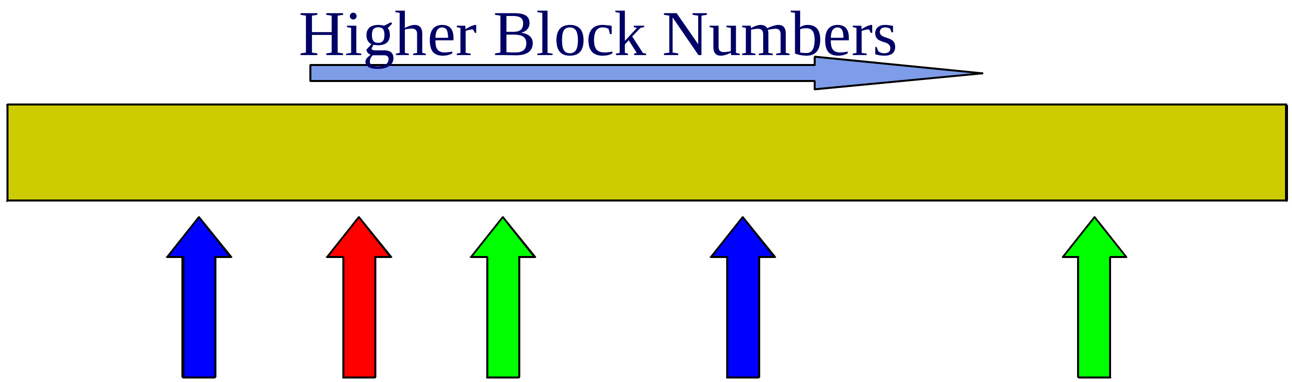


In LOOK, we would have read this request (unfair extra service ~~so~~ we'll skip it)

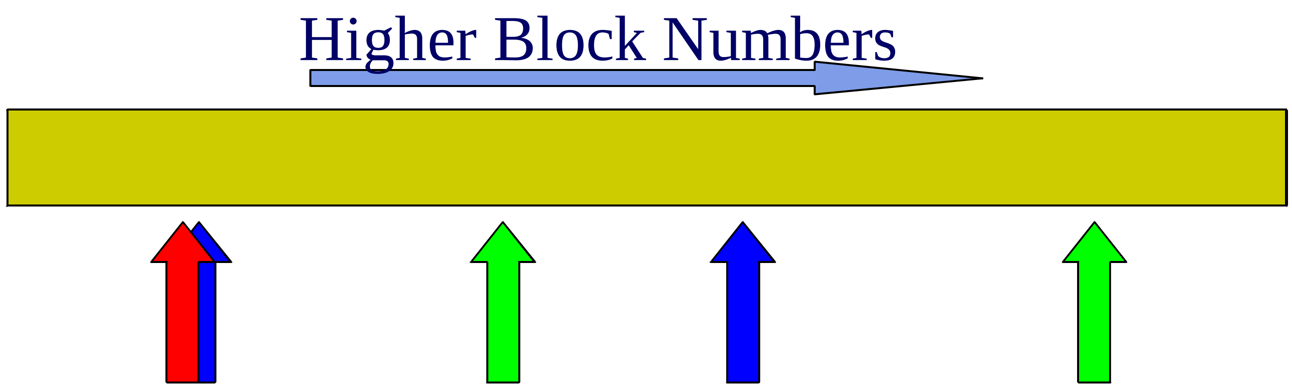
C-LOOK



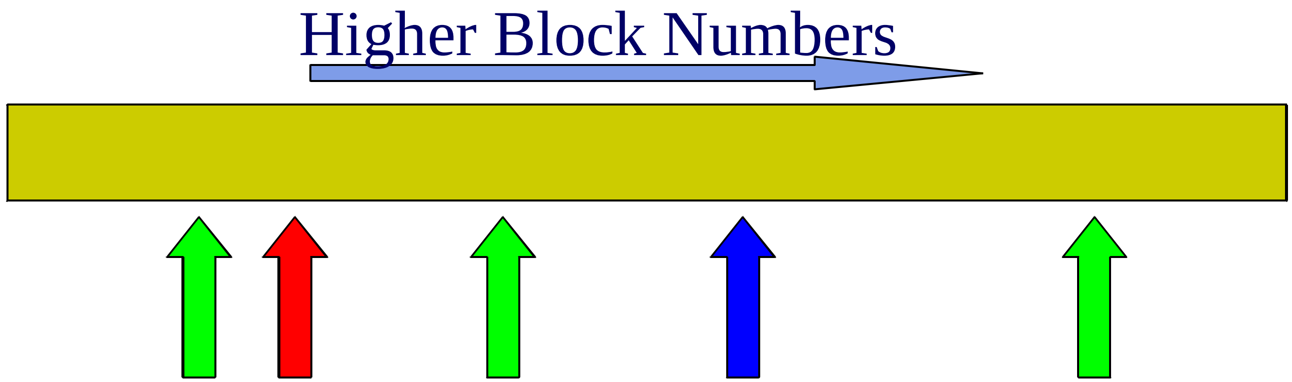
C-LOOK



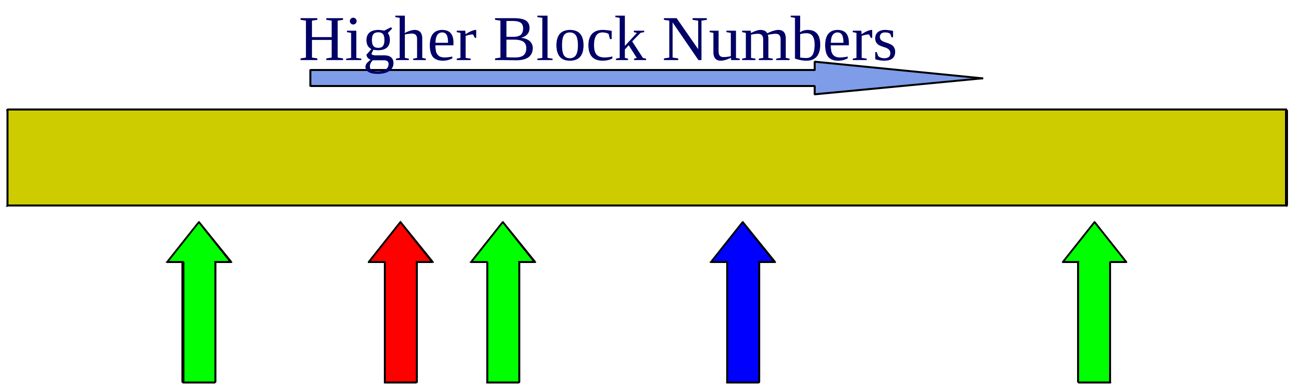
C-LOOK



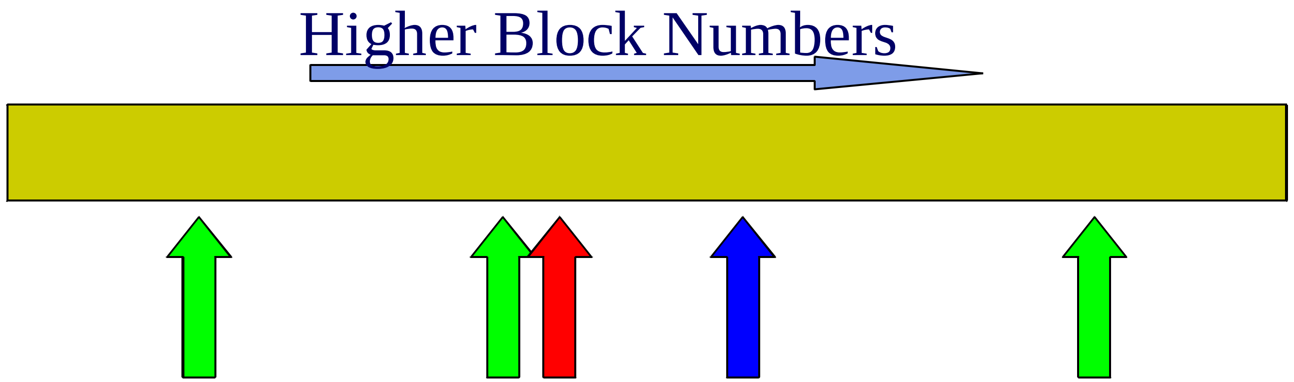
C-LOOK



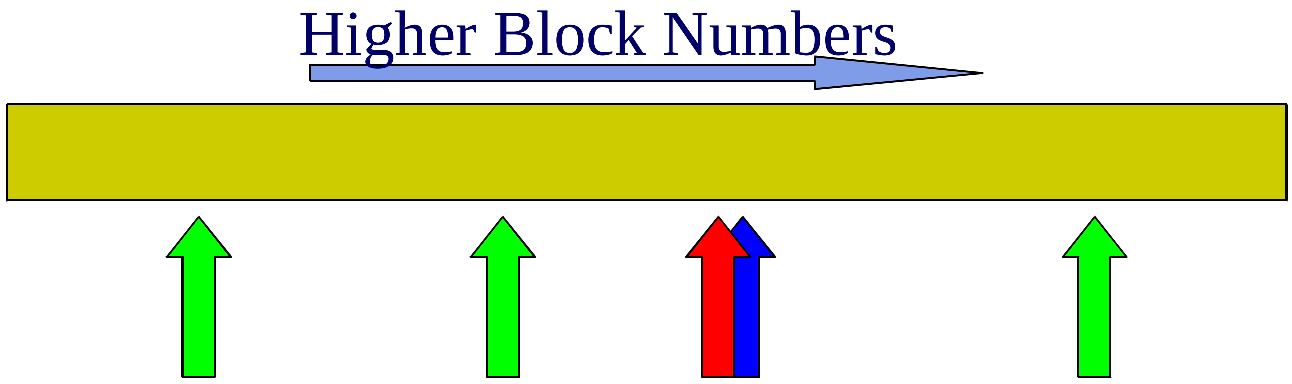
C-LOOK



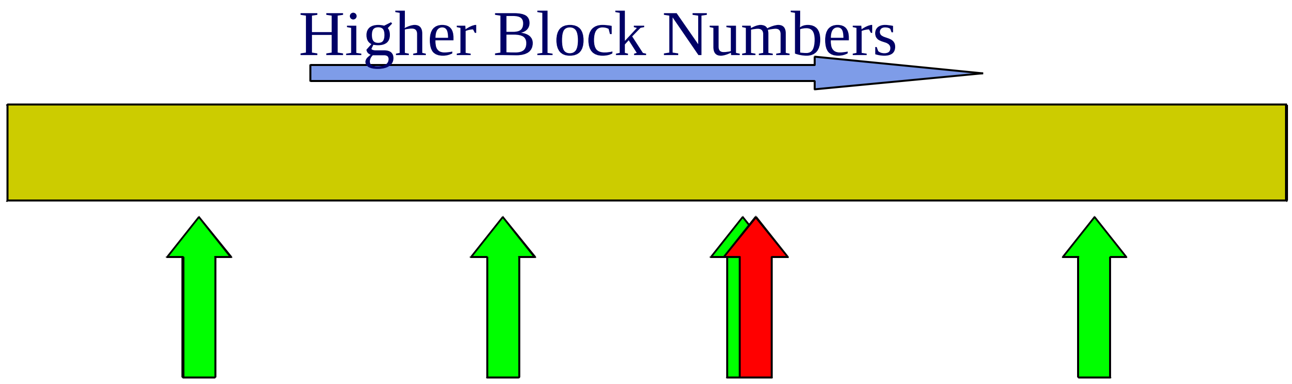
C-LOOK



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...**
  - **Some disks (SCSI, newer IDE) accept a request queue**
  - **Sector complete ⇒ give OS both data and sector number**

Head to Head

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
- Circular starts back at the “starting” position
- 2-way is unfair
  - Services requests at the center twice as often

Weighting

- “High Throughput” algorithms can starve requests
- Making them fair costs us in terms of performance
- Add aging to requests to prevent starvation

Lies Disks Tell

Disks re-order I/O requests

- You ask “read 37”, “read 83”, “read 2”
  - Disk gives you 37, 2, 83
    - » Great! That's why disks accept multiple requests.

Disks lie about writes

- You ask “read 37”, “write 23”, “read 2”
  - Disk writes 23, gives you 2, 37
    - » Odd, but seems ok...
- You ask “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?

Lies Disks Tell

Disks lie about lies

- **Special commands**
  - **“Flush all pending writes”**
    - » **Think “my disk is 'modern'”, think “disk barrier”**
  - **“Disable write cache”**
    - » **Think “please don't be quite so modern”**

Lies Disks Tell

Disks lie about lies

- Special commands
  - "Flush all pending writes"
    - » Think "my disk is 'modern'", think "disk barrier"
  - "Disable write cache"
    - » Think "please don't be quite so modern"
- Some disks ignore the special commands
  - "Flush all pending writes" ⇒ "Uh huh, sure, no problem"
  - "Disable write cache" ⇒ "Uh huh, sure, no problem"
- Result
  - Great performance on benchmarks!!!
  - Really bizarre file system corruption after power failures

Further Reading

Terabyte Territory

Brian Hayes

American Scientist, May/June 2002

http://www.americanscientist.org/template/AssetDetail/assetid/14750?&print=yes

A Conversation with Jim Gray

Dave Patterson

ACM Queue, June 2003

http://www.acmqueue.org/modules.php?name=Content&pa=showpage&pid=43

Conclusions

Disks are very slow

Disks are very complicated

FCFS is a very bad idea

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

Disks lie

- **Some are vicious**