15-410

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

Disks
March 15, 2010

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

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***

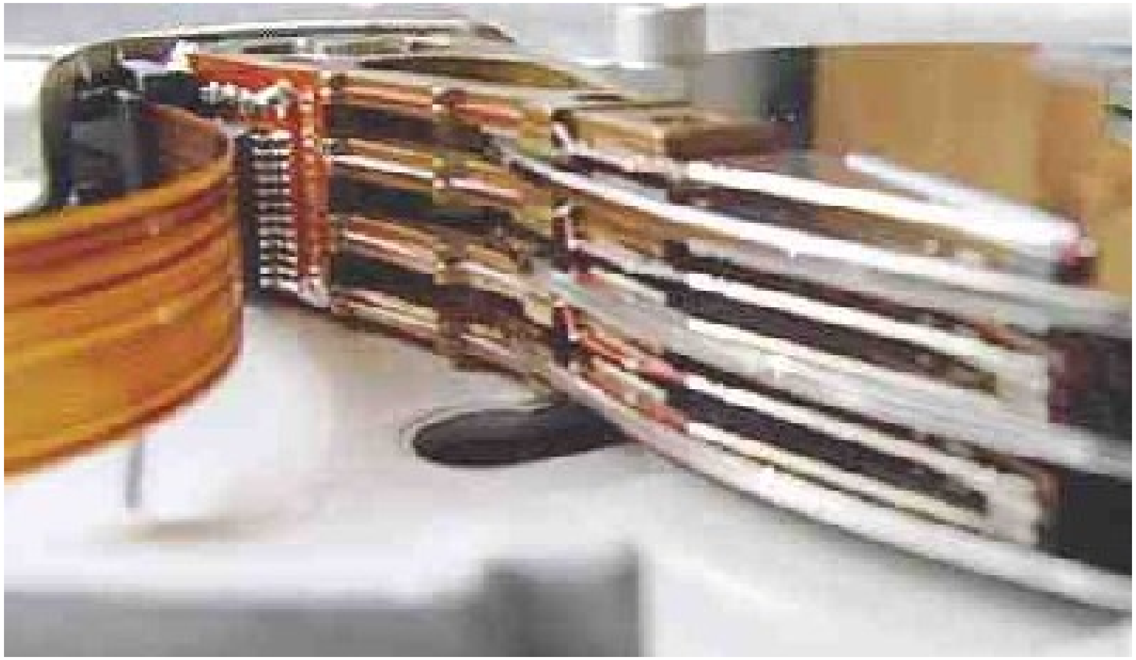
Taken from "How Hard Disks Work"
http://computer.howstuffworks.com/hard-disk2.htm

Anatomy of a Hard Drive

Both sides of each platter store information

Each side of a platter is called a *surface*

Each surface has its own read/write head



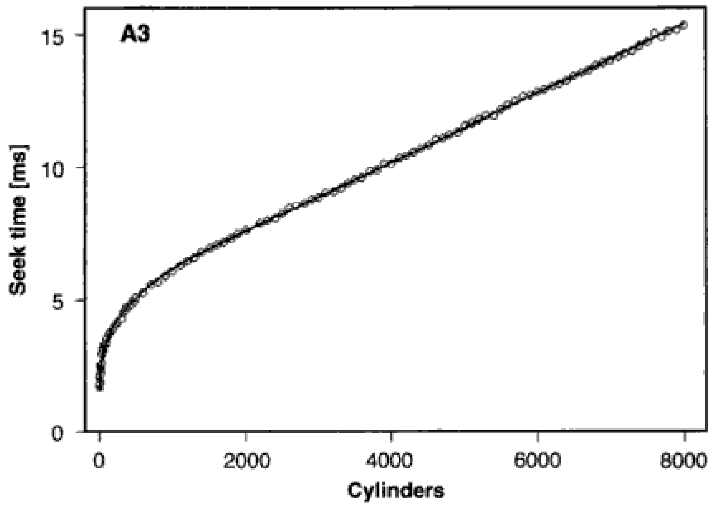
Taken from “How Hard Disks Work”
http://computer.howstuffworks.com/hard-disk2.htm

Anatomy of a Hard Drive

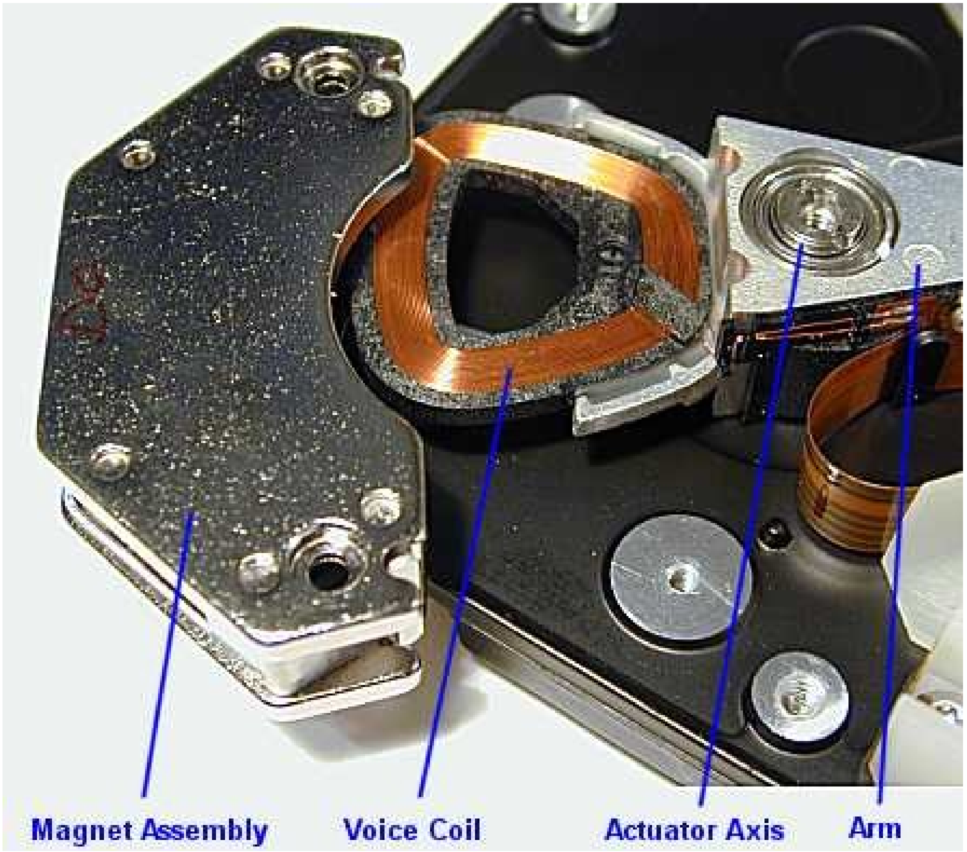
The arm is moved by a voice coil actuator

Slow, as computers go

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



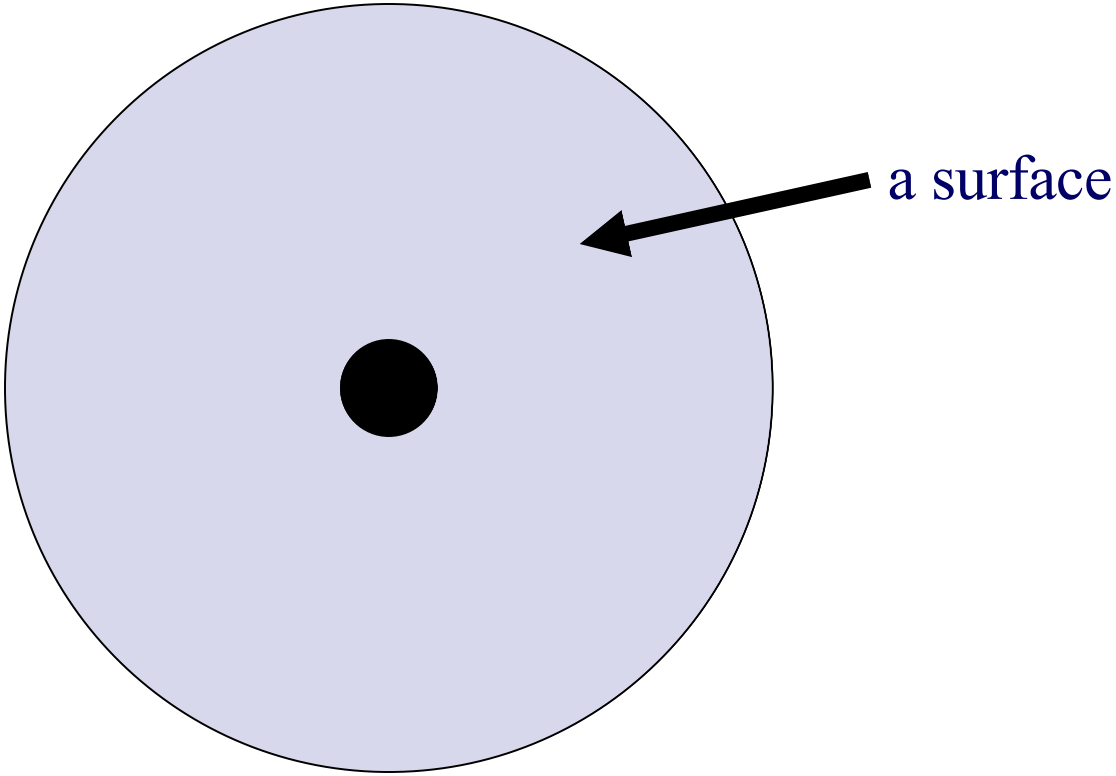
Oklobdzija, Comp. Eng. Handbook, 2002



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

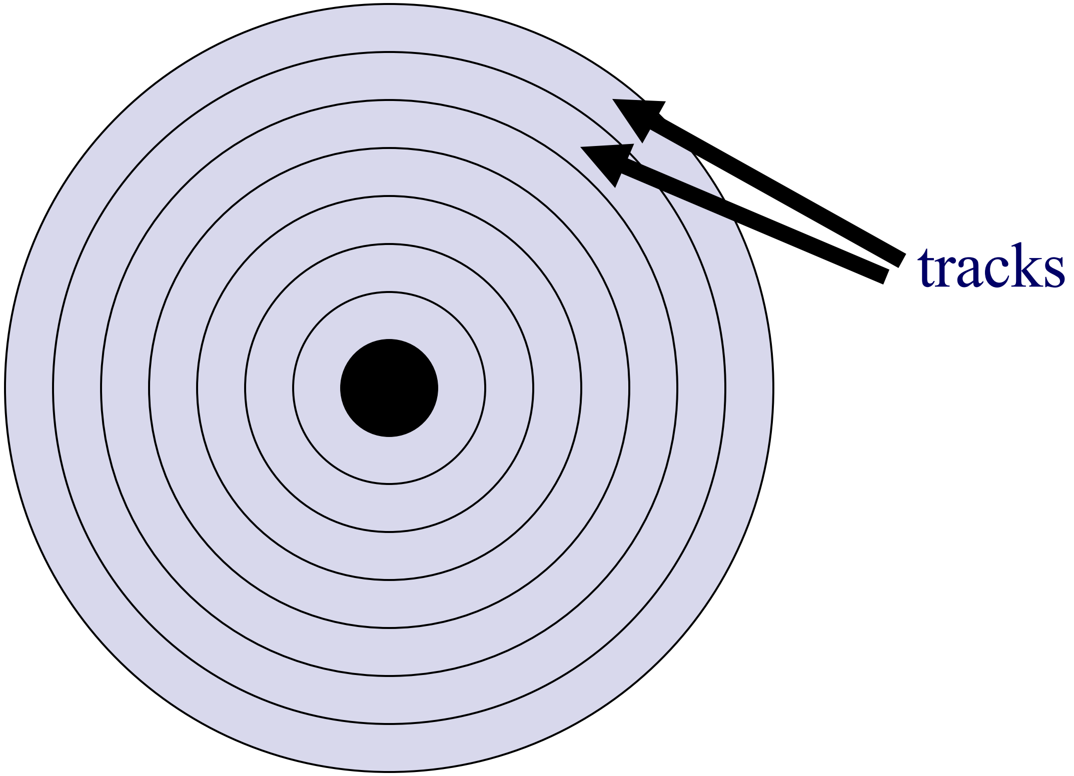
Anatomy of a Hard Drive

How are the surfaces organized?



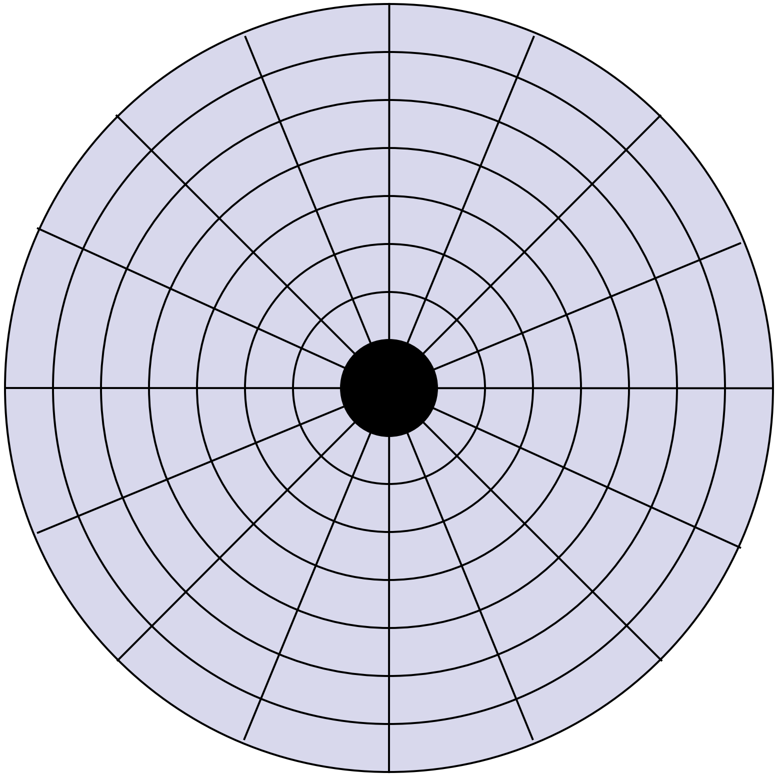
Anatomy of a Hard Drive

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



Anatomy of a Hard Drive

These tracks are further divided into *sectors*



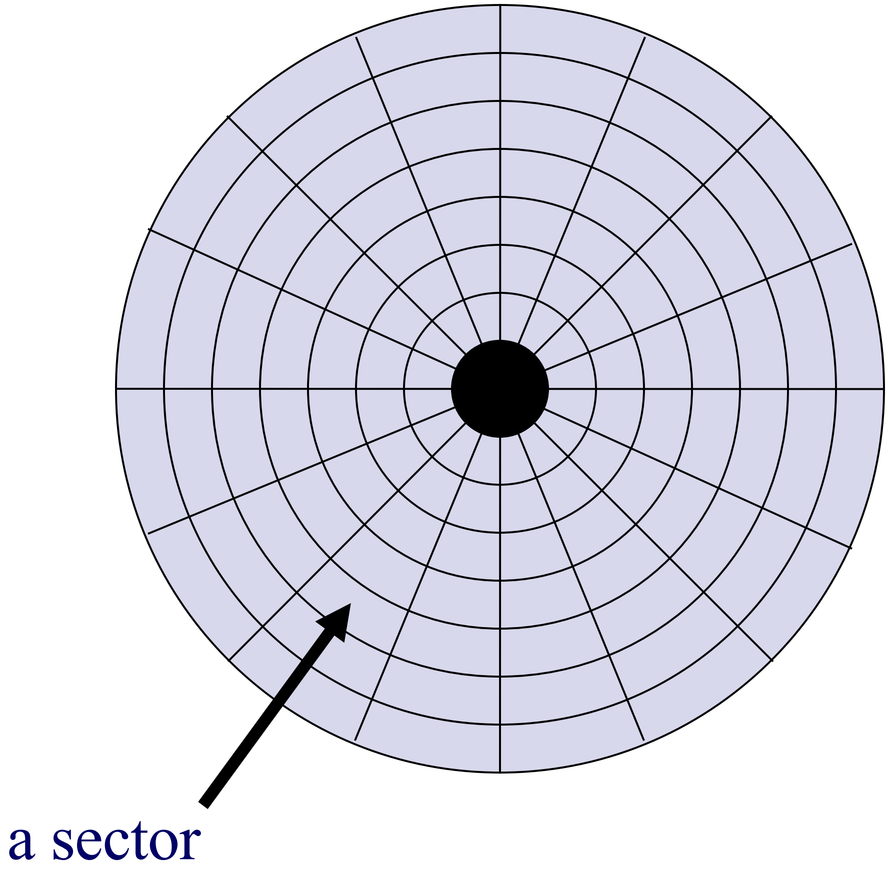
Anatomy of a Hard Drive

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!)

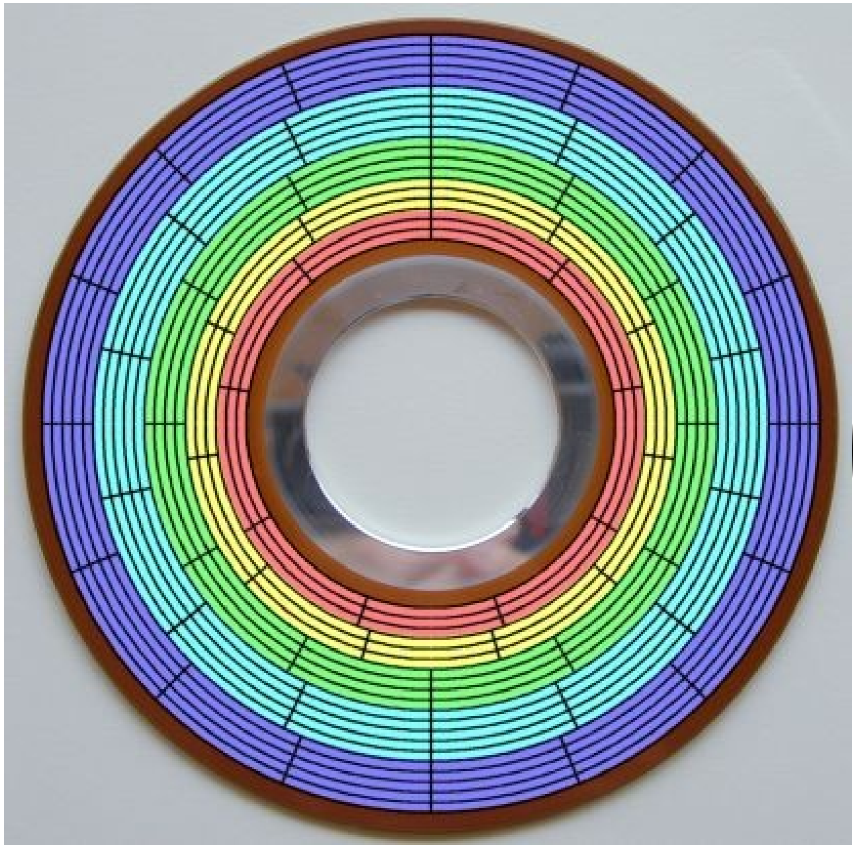
Gee, those outer sectors look bigger...?



Anatomy of a Hard Drive, Really

Modern hard drives use *zoned bit recording*

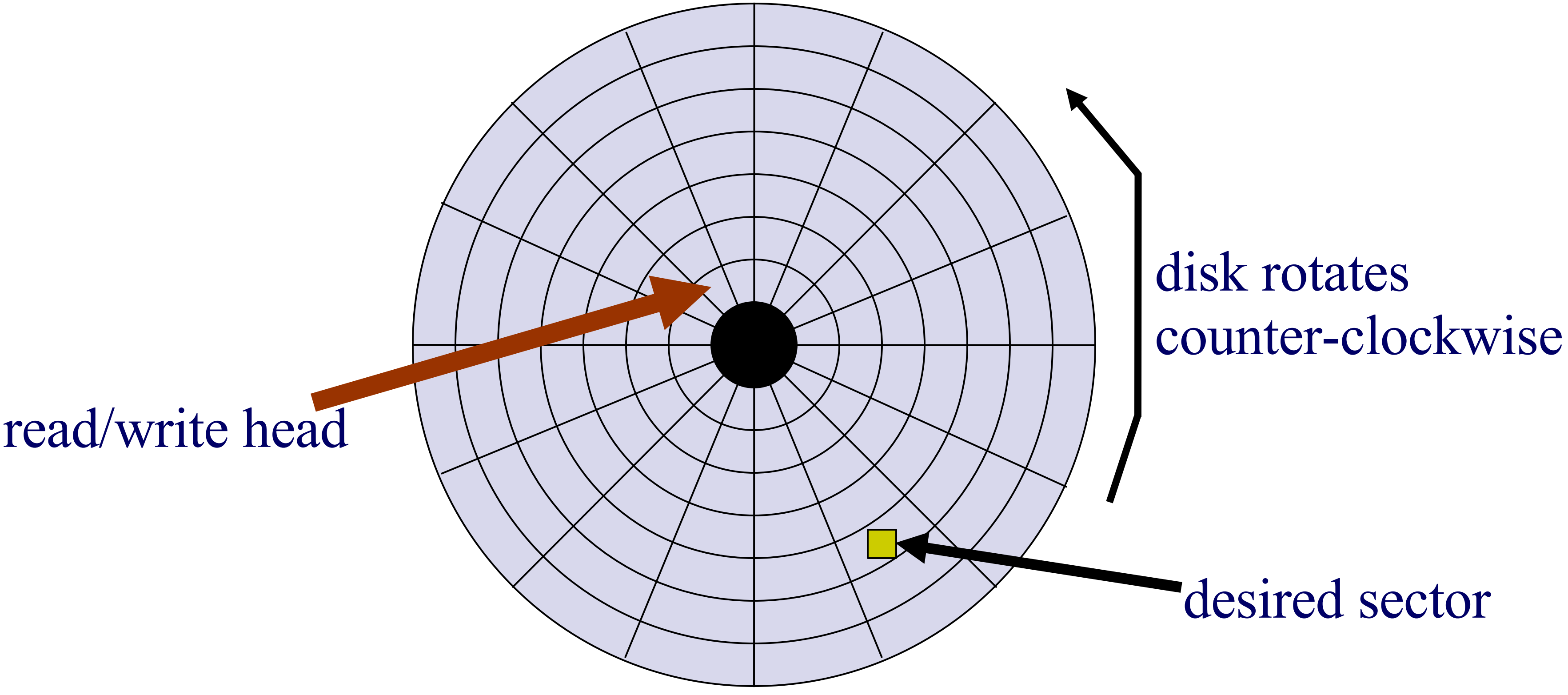
- **Disk has tables to map track# to #sectors**
- **Sectors are all roughly the same linear length**



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")**

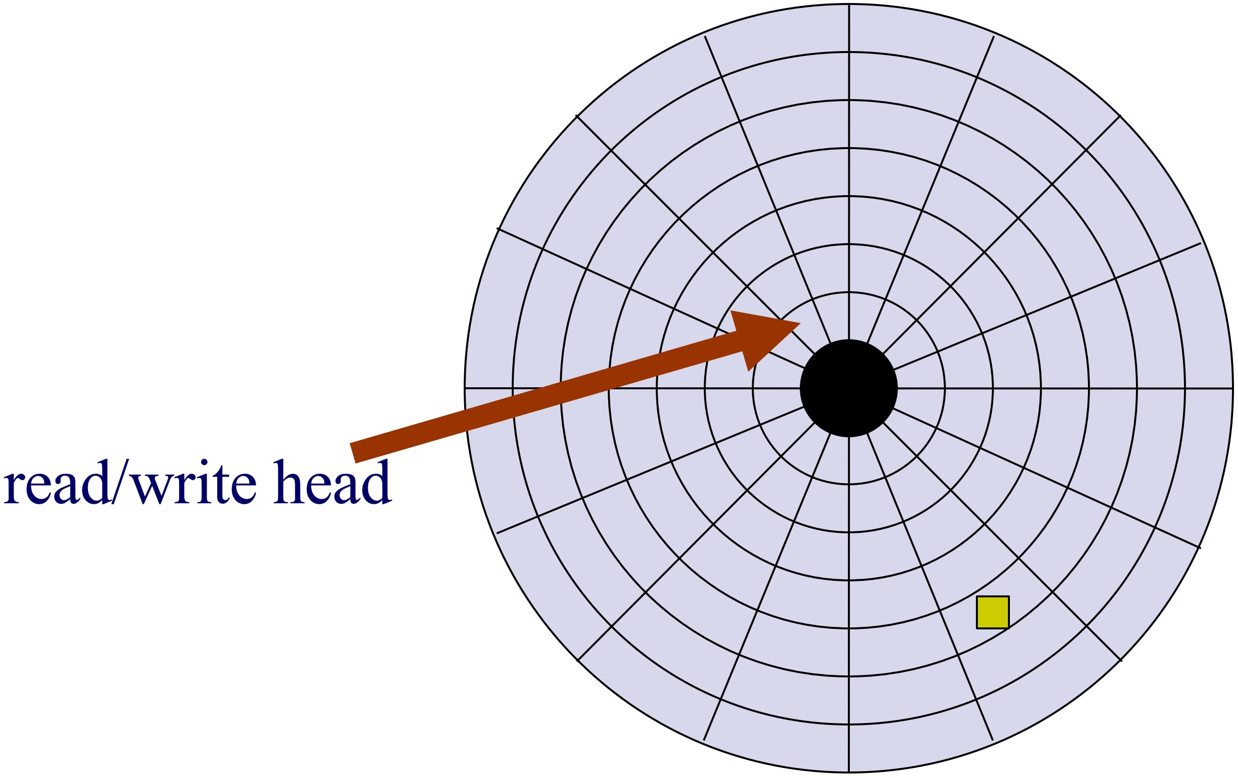
2. **Wait until the desired sector spins around ("rotational delay"/"rotational latency")**

Observe

- **Average seeks are 2 –10 msec**
- **Disk rotates 5,400...15,000 rpm, delay 11...4 msec**
- **Rotation dominates short seeks, matches average seeks**
- **We could say "seek delay" or "rotational time" - but experts usually don't (now you know how to sound like an expert)**

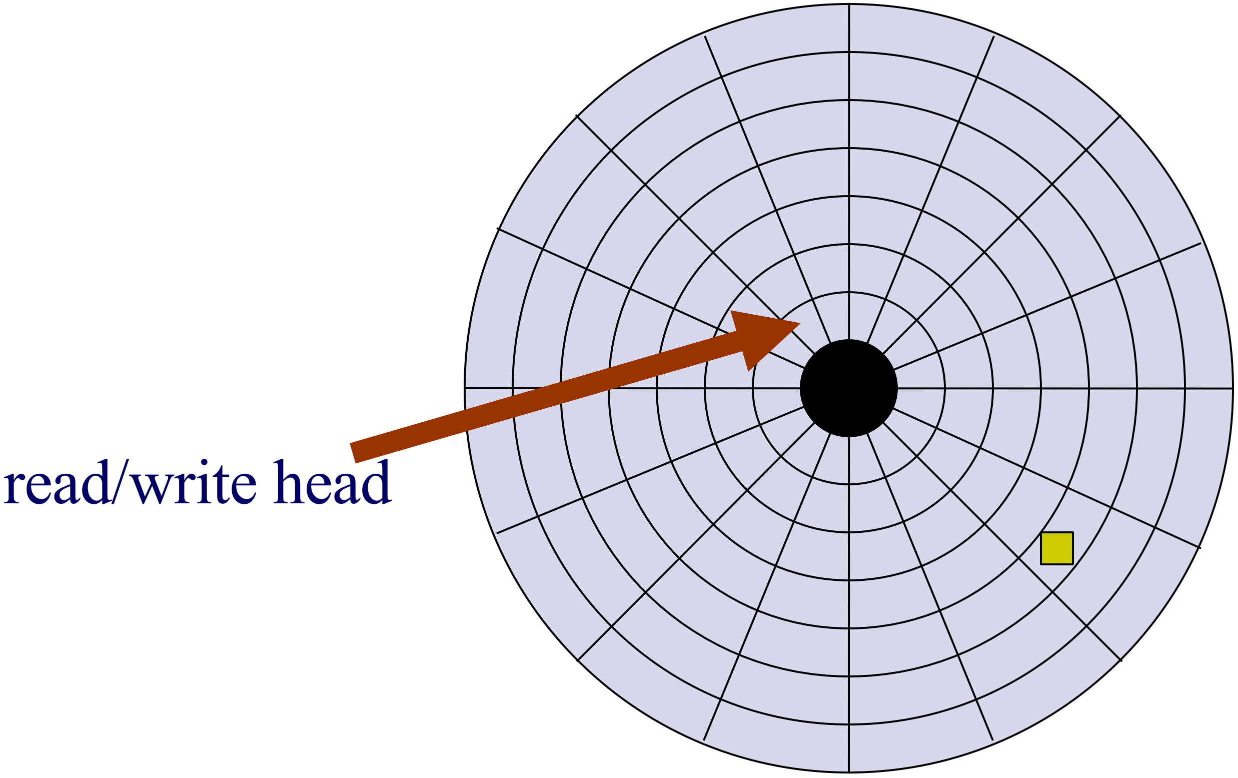
Anatomy of a Hard Drive

Let's read in a sector from the disk



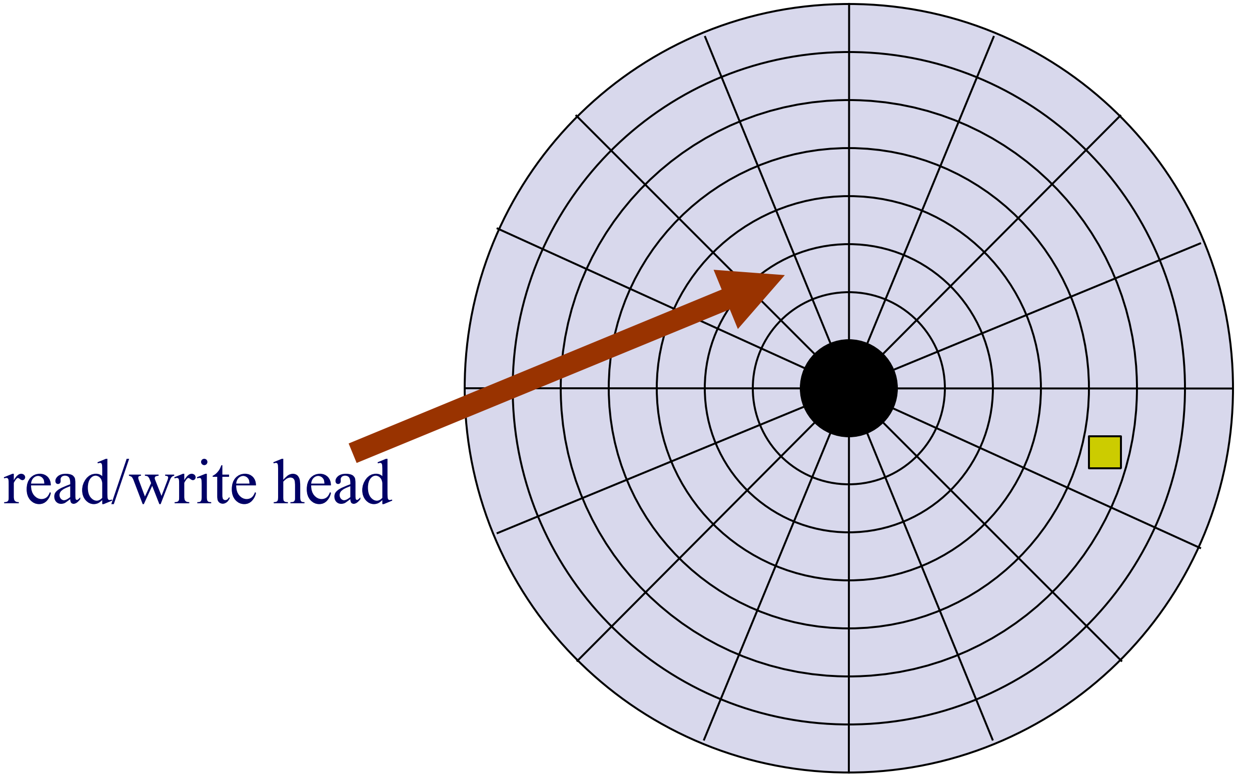
Anatomy of a Hard Drive

Let's read in a sector from the disk



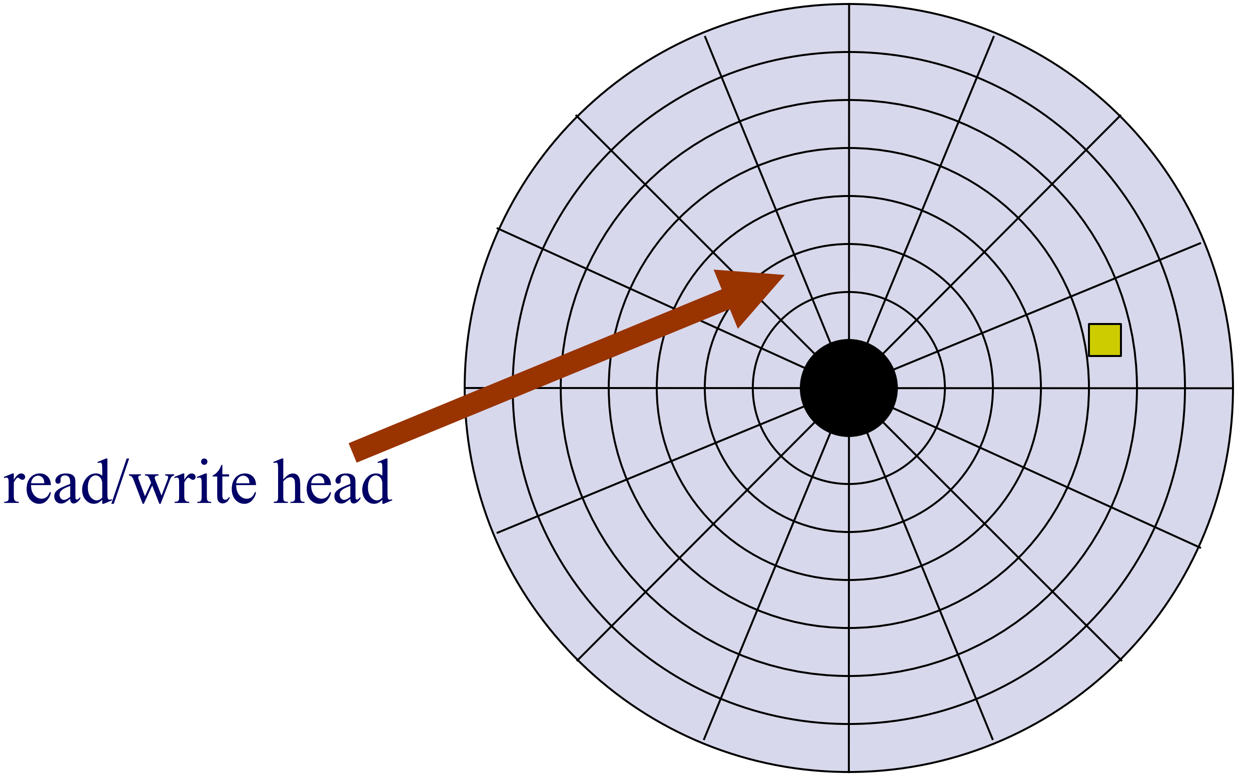
Anatomy of a Hard Drive

Let's read in a sector from the disk



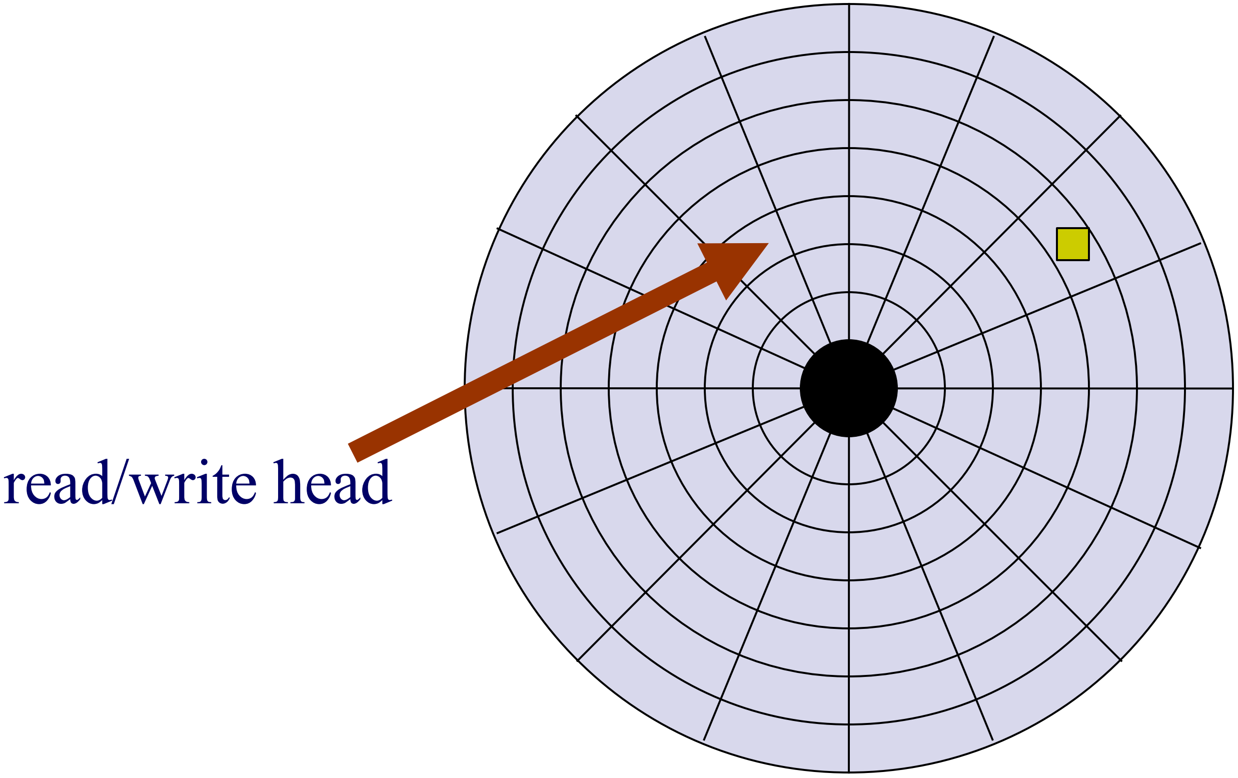
Anatomy of a Hard Drive

Let's read in a sector from the disk



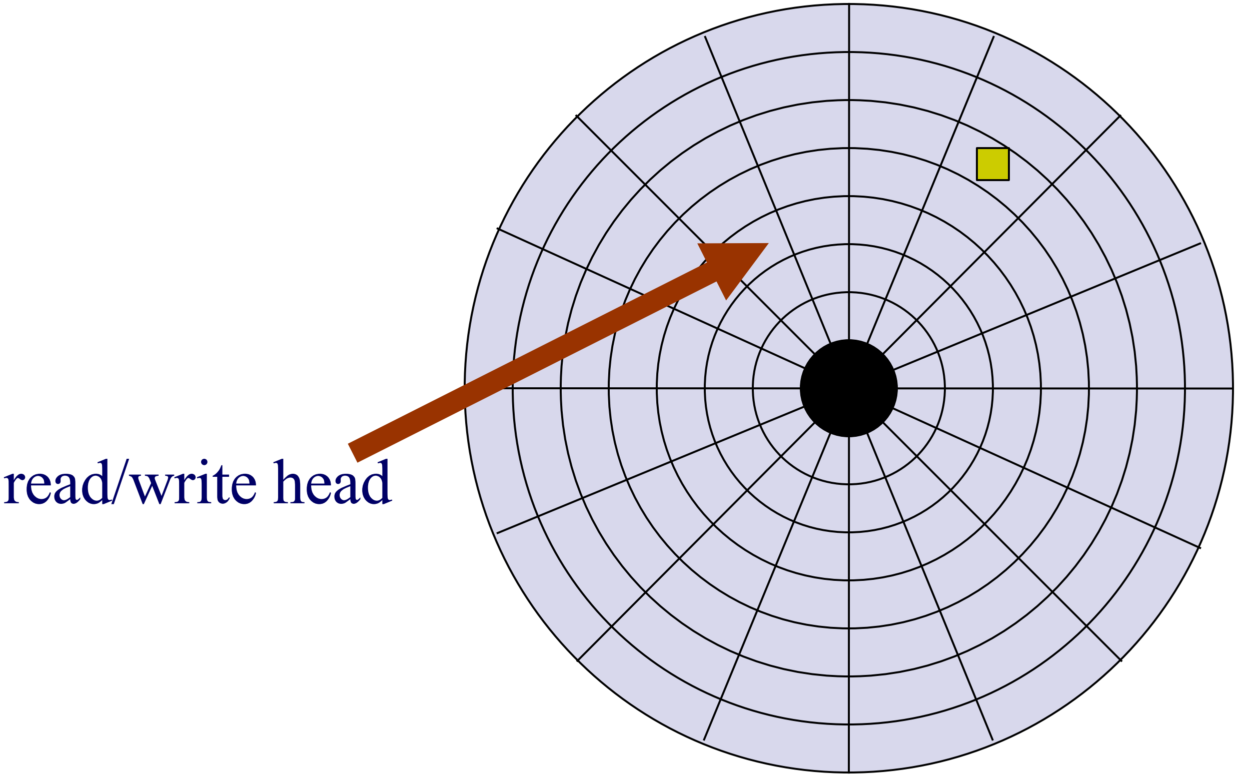
Anatomy of a Hard Drive

Let's read in a sector from the disk



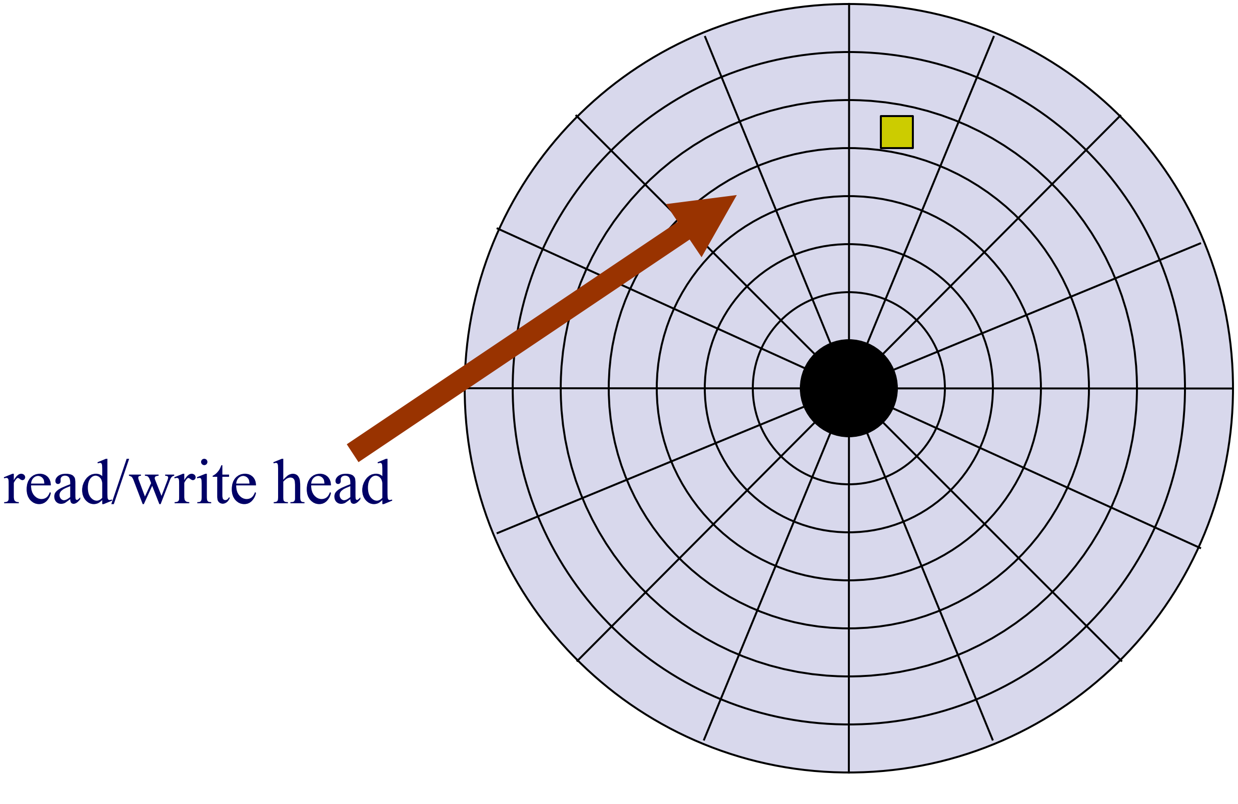
Anatomy of a Hard Drive

Let's read in a sector from the disk



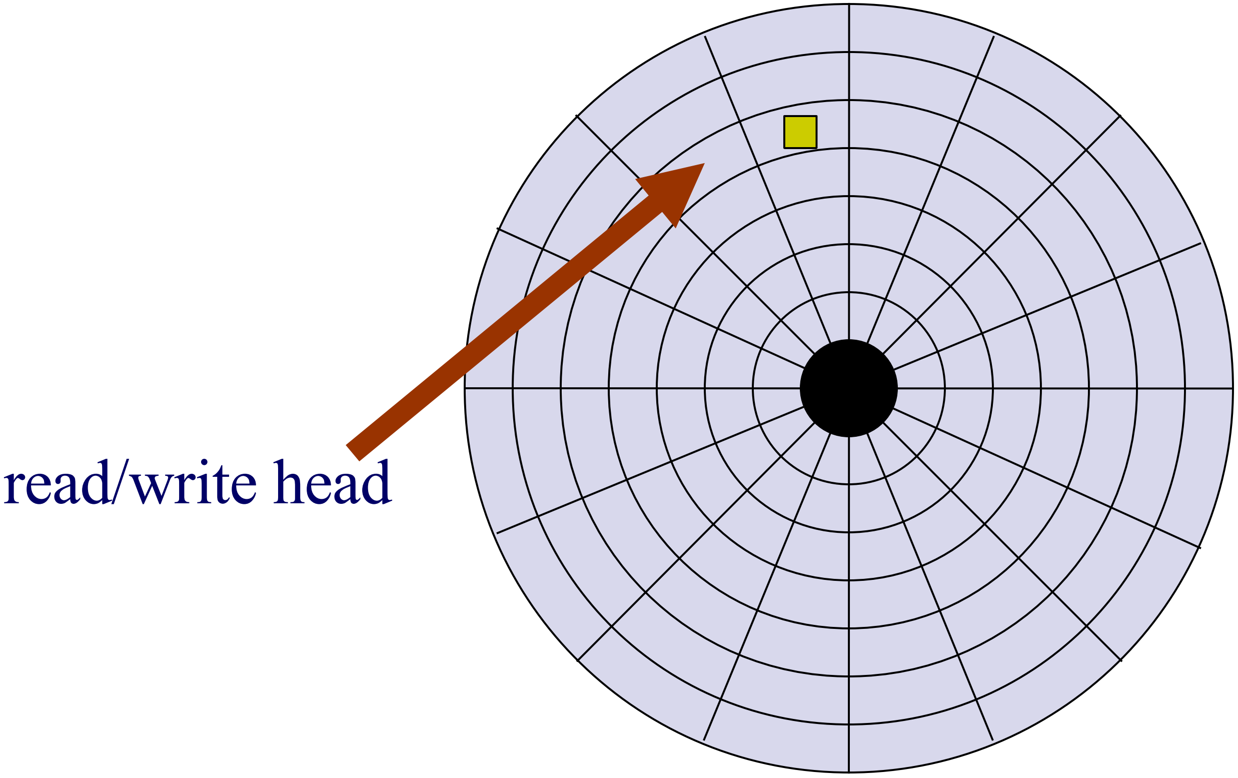
Anatomy of a Hard Drive

Let's read in a sector from the disk



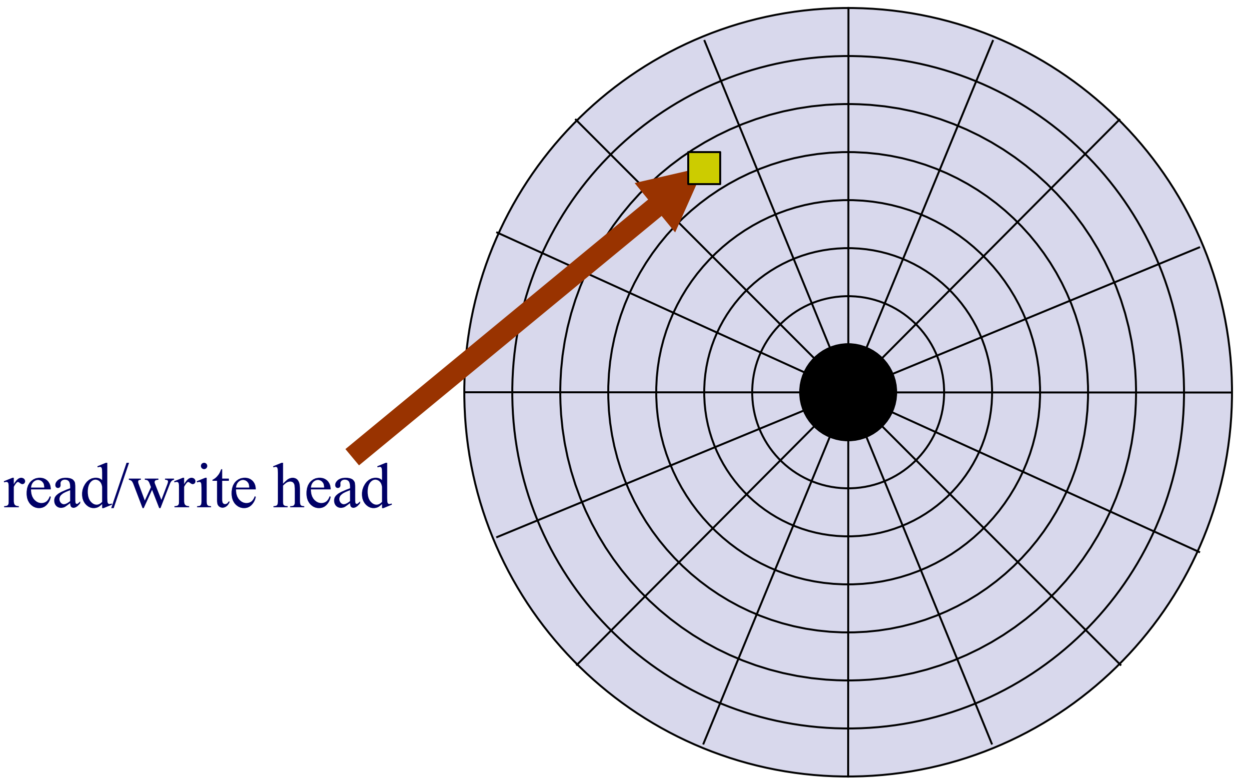
Anatomy of a Hard Drive

Let's read in a sector from the disk



Anatomy of a Hard Drive

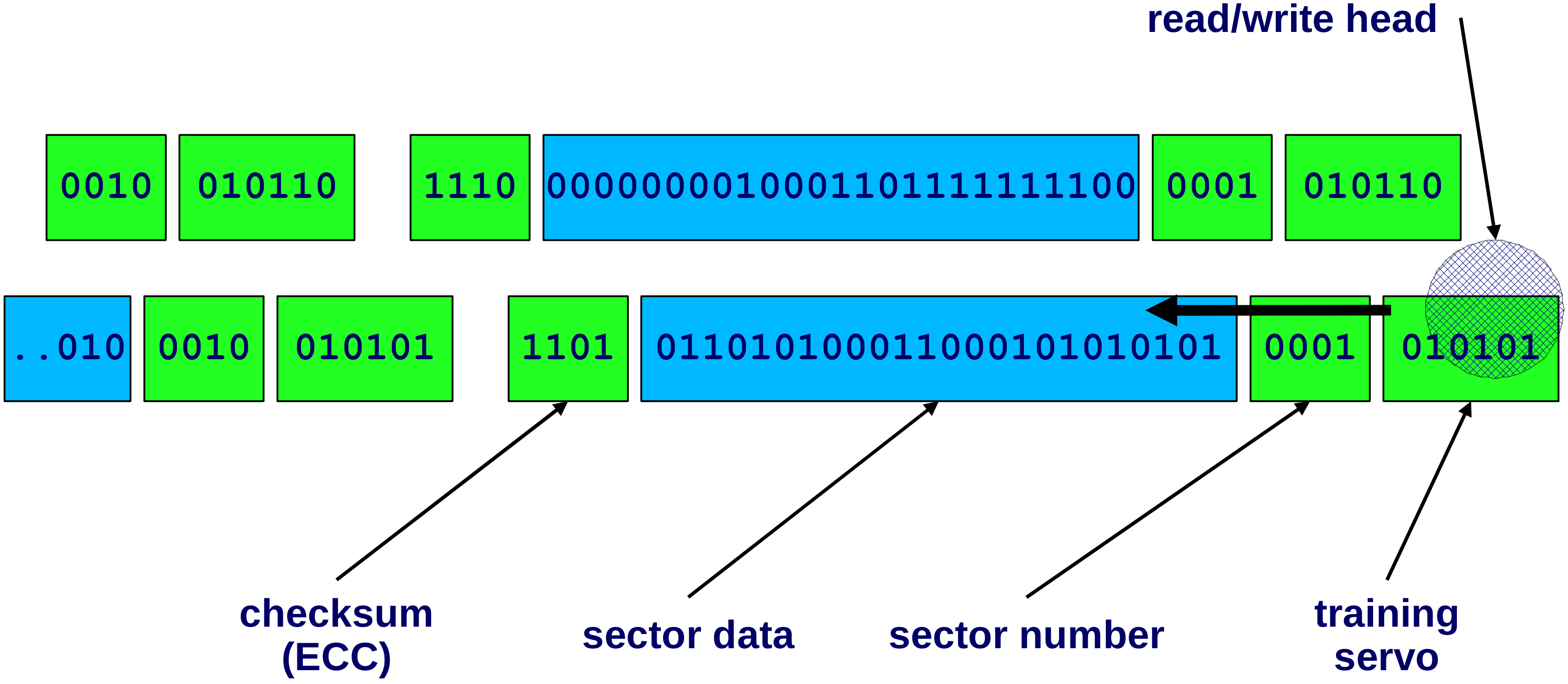
Let's read in a sector from the disk



Anatomy of a “Sector”

Finding a sector involves real work

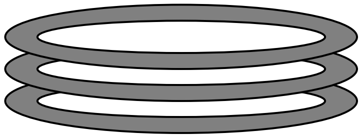
- Correct track; check sector header for number



After sector is read, compare data to checksum

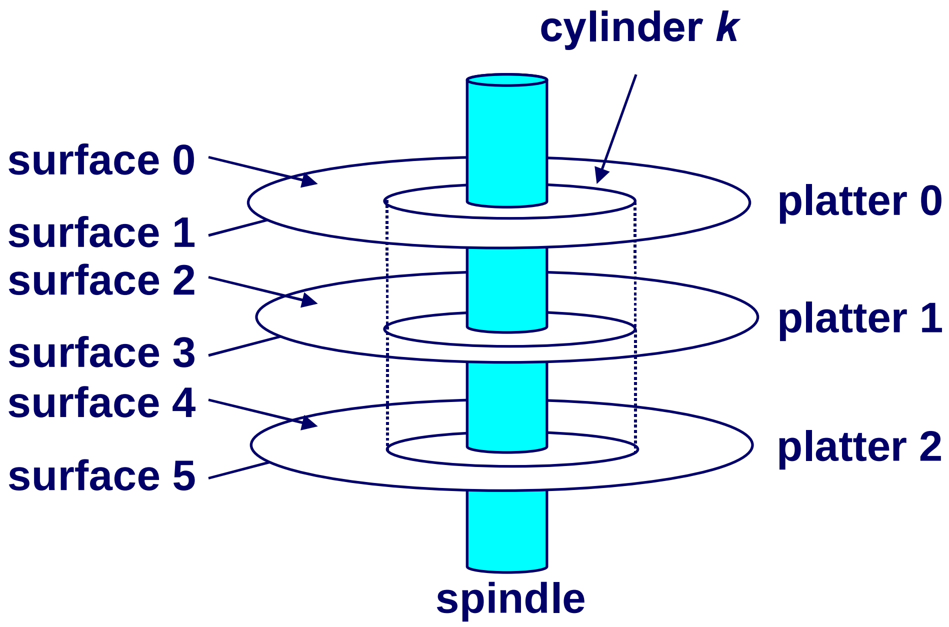
Disk Cylinder

- **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

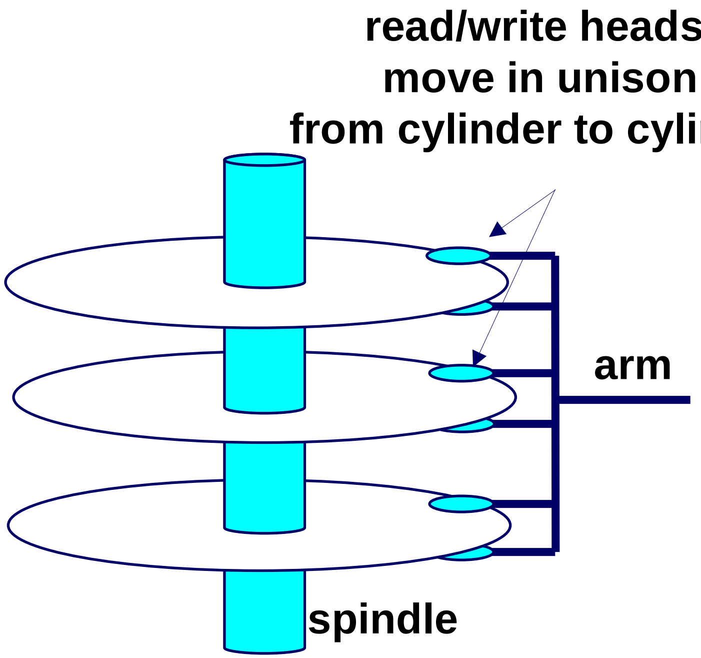
- All heads always on same cylinder

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



Some Disk Specs (2009)

Big is 2TB capacity, Cool is 1-2 W, Fast is 3.5 ms seeks

Specifications	2 TB[1]
Model Number	ST32000641AS
Interface Options	SATA 6Gb/s NCQ
Performance	
Transfer Rate, Max Ext (MB/s)	600
Sustained Data Rate OD (MB/s)	138
Cache (MB)	64
Average Latency (ms)	4.16
Spindle Speed (RPM)	7200
Configuration/Organization	
Heads/Disks	8/4
Bytes per Sector	512
Reliability/Data Integrity	
Load/Unload Cycles	300K
Nonrecoverable Read Errors per Bits Read, Max	1 per 10E14
Annualized Failure Rate	0.34%
Mean Time Between Failures (hours)	750,000
Limited Warranty (years)	5
Power Management	
Startup Current +12 Peak(A, ±10%)	2.8
Seek, Average (W)	7.3
Operating, Average (W)	9.23
Idle, Average (W)	6.39
Environmental	
Temperature (°C)	
Operating	5 to 60
Nonoperating	–40 to 70
Shock (Gs)	
Operating: 2 ms	63
Nonoperating: 2 ms	300
Acoustics (bels – sound power)	
Idle	2.8
Seek	3.2

Specifications	500 GB[1]
Model Number	ST9500325AS ST9500325ASG[2]
Interface Options	SATA 3Gb/s NCQ
Performance	
Transfer Rate	
Maximum Internal (Mb/s)	1175
Maximum External (MB/s)	300
Cache (MB)	8
Average Latency (ms)	5.6
Spindle Speed (RPM)	5400
Areal Density (Gb/in²)	394
Configuration/Organization	
Disks/Heads	2/4
Bytes per Sector	512
Reliability/Data Integrity	
G-Force Protection™	Available
Head-Rest Method	QuietStep™ Ramp Load
Load/Unload Cycles	>600,000
Nonrecoverable Read Errors per Bits Read	1 per 10E14
Annualized Failure Rate (AFR)	0.48%
Power Management	
Startup Current 5V (amps max)	1.0
Power Management (W)	
Seek	1.54
Read/Write Avg	2.6/2.85
Idle/Standby Avg	0.81/0.22

Specifications	600 GB[1]
Model Number	ST3600057SS ST3600957SS[2] ST3600057FC
Capacity	
Formatted 512 Kbytes/Sector (GB)	600
External Transfer Rate (MB/s)	
4-Gb/s Fibre Channel	400
6-Gb/s Serial Attached SCSI	600
Performance	
Spindle Speed (RPM)	15K
Average Latency (ms)	2.0
Seek Time	
Average Read/Write (ms)	3.4/3.9
Track-to-Track Read/Write (ms)	0.2/0.4
Transfer Rate	
Internal (Mb/s, OD–ID)	1450 to 2370
Sustained (MB/s, 1000 x 1000)	122 to 204
Cache, Multisegmented (MB/s)	16
Configuration/Organization	
Disks	4
Heads	8
Nonrecoverable Read Errors per Bits Read	1 sector per 10E16
Reliability Rating at Full 24x7 Operation (AFR)	0.55%
MTBF (hours)	1,600,000
Power Management	
Typical (W)	
Fibre Channel	16.31
SAS	16.35
Power Idle (W)	
Fibre Channel	11.61
SAS	11.68

Some Disk Specs (2009)

Big $330 ($.16/GB), Cool $85 ($.17/GB), Fast $670 ($1.12/GB)

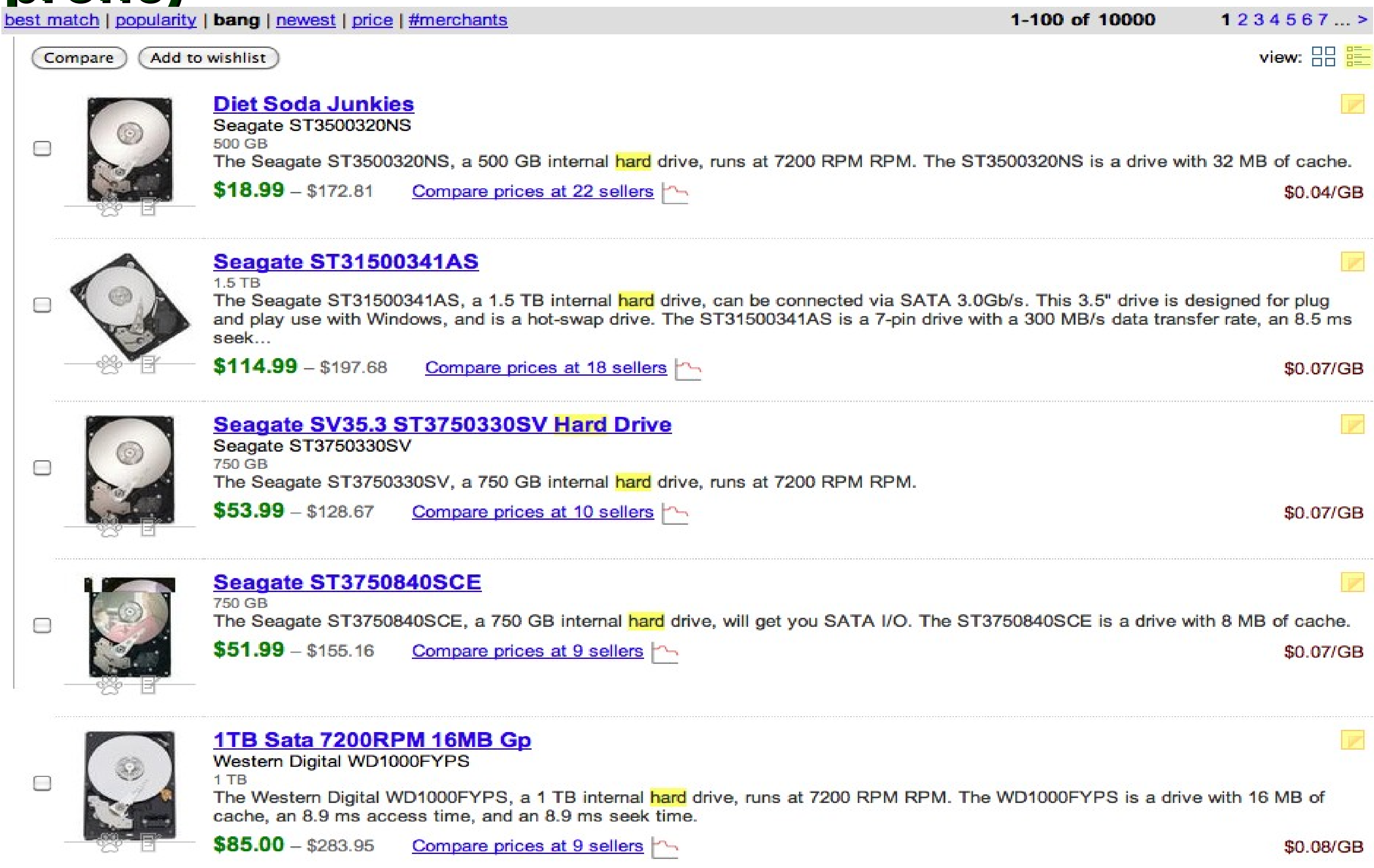
Specifications	2 TB[1]
Model Number	ST32000641AS
Interface Options	SATA 6Gb/s NCQ
Performance	
Transfer Rate, Max Ext (MB/s)	600
Sustained Data Rate OD (MB/s)	138
Cache (MB)	64
Average Latency (ms)	4.16
Spindle Speed (RPM)	7200
Configuration/Organization	
Heads/Disks	8/4
Bytes per Sector	512
Reliability/Data Integrity	
Load/Unload Cycles	300K
Nonrecoverable Read Errors per Bits Read, Max	1 per 10E14
Annualized Failure Rate	0.34%
Mean Time Between Failures (hours)	750,000
Limited Warranty (years)	5
Power Management	
Startup Current +12 Peak(A, ±10%)	2.8
Seek, Average (W)	7.3
Operating, Average (W)	9.23
Idle, Average (W)	6.39
Environmental	
Temperature (°C) Operating Nonoperating	 5 to 60 –40 to 70
Shock (Gs) Operating: 2 ms Nonoperating: 2 ms	 63 300
Acoustics (bels – sound power) Idle Seek	 2.8 3.2

Specifications	500 GB[1]
Model Number	ST9500325AS ST9500325ASG[2]
Interface Options	SATA 3Gb/s NCQ
Performance	
Transfer Rate Maximum Internal (Mb/s) Maximum External (MB/s)	 1175 300
Cache (MB)	8
Average Latency (ms)	5.6
Spindle Speed (RPM)	5400
Areal Density (Gb/in²)	394
Configuration/Organization	
Disks/Heads	2/4
Bytes per Sector	512
Reliability/Data Integrity	
G-Force Protection™	Available
Head-Rest Method	QuietStep™ Ramp Load
Load/Unload Cycles	>600,000
Nonrecoverable Read Errors per Bits Read	1 per 10E14
Annualized Failure Rate (AFR)	0.48%
Power Management	
Startup Current 5V (amps max)	1.0
Power Management (W) Seek Read/Write Avg Idle/Standby Avg	 1.54 2.6/2.85 0.81/0.22

Specifications	600 GB[1]
Model Number	ST3600057SS ST3600957SS[2] ST3600057FC
Capacity	
Formatted 512 Kbytes/Sector (GB)	600
External Transfer Rate (MB/s) 4-Gb/s Fibre Channel 6-Gb/s Serial Attached SCSI	 400 600
Performance	
Spindle Speed (RPM)	15K
Average Latency (ms)	2.0
Seek Time Average Read/Write (ms) Track-to-Track Read/Write (ms)	 3.4/3.9 0.2/0.4
Transfer Rate Internal (Mb/s, OD–ID) Sustained (MB/s, 1000 x 1000)	 1450 to 2370 122 to 204
Cache, Multisegmented (MB/s)	16
Configuration/Organization	
Disks	4
Heads	8
Nonrecoverable Read Errors per Bits Read	1 sector per 10E16
Reliability Rating at Full 24x7 Operation (AFR)	0.55%
MTBF (hours)	1,600,000
Power Management	
Typical (W) Fibre Channel SAS	 16.31 16.35
Power Idle (W) Fibre Channel SAS	 11.61 11.68

Some Disk Specs (2009)

Price lead is $.07/GB (smaller, slower, hotter, failure prone)



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 average rotational delay, and on a 5400 rpm drive is ~5.5ms

Seagate 7200.7, a 2005-era 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 ~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

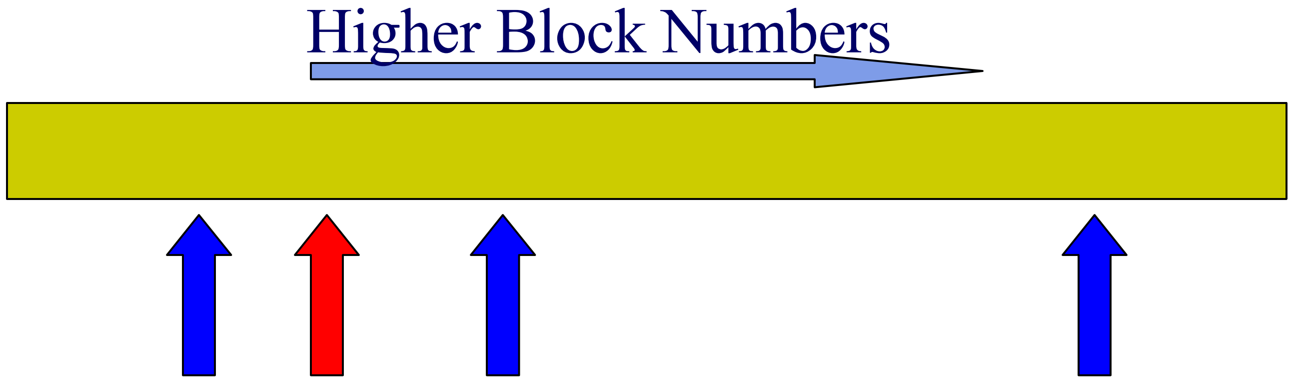
Intolerable response time *variance*, however

Why?

SSTF

Blue are requests

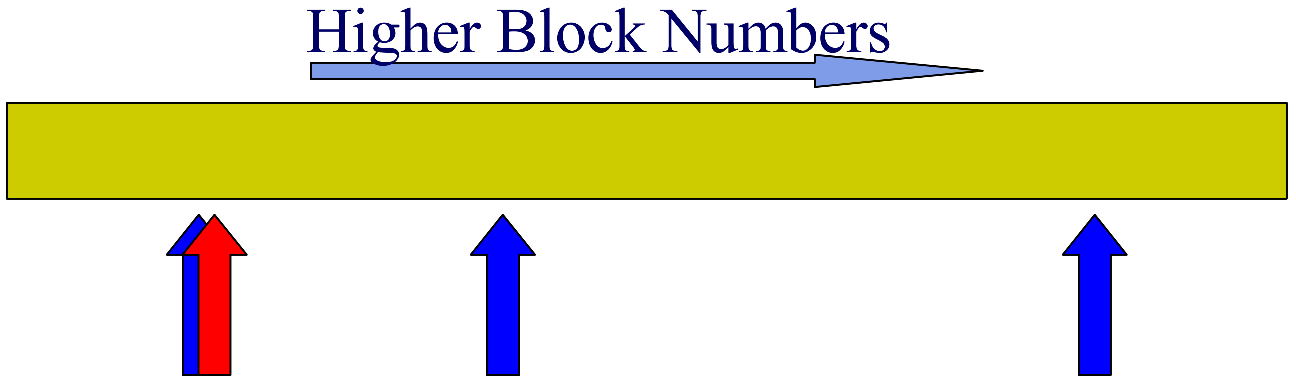
Yellow is disk



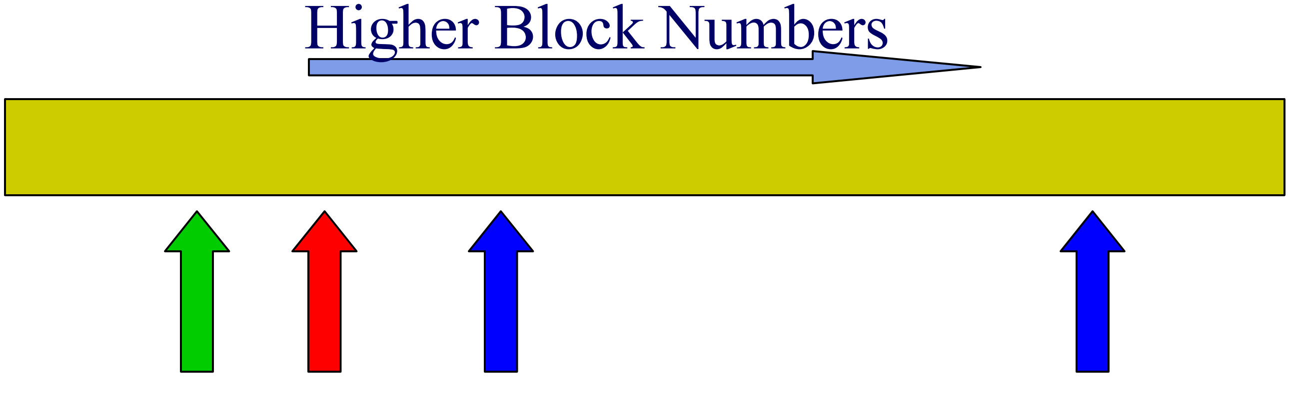
Red is disk head

Green is completed requests

SSTF

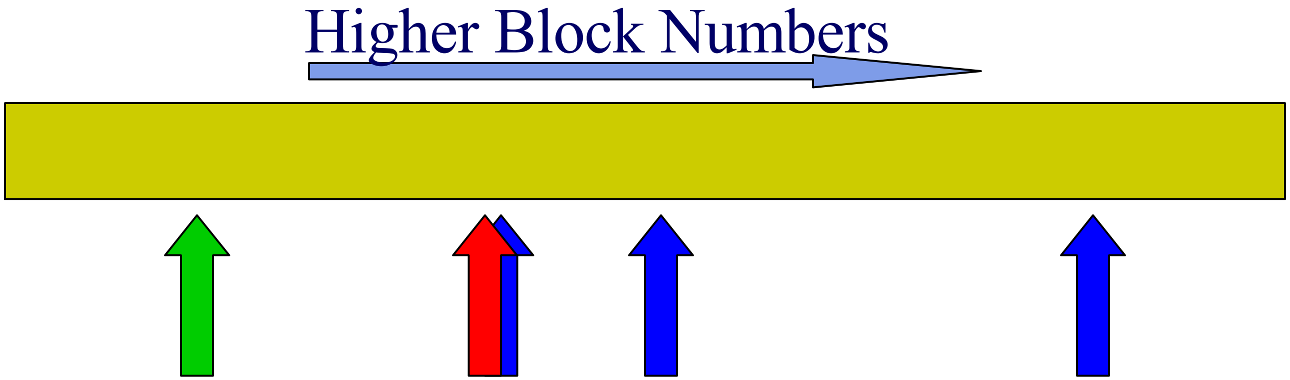


SSTF

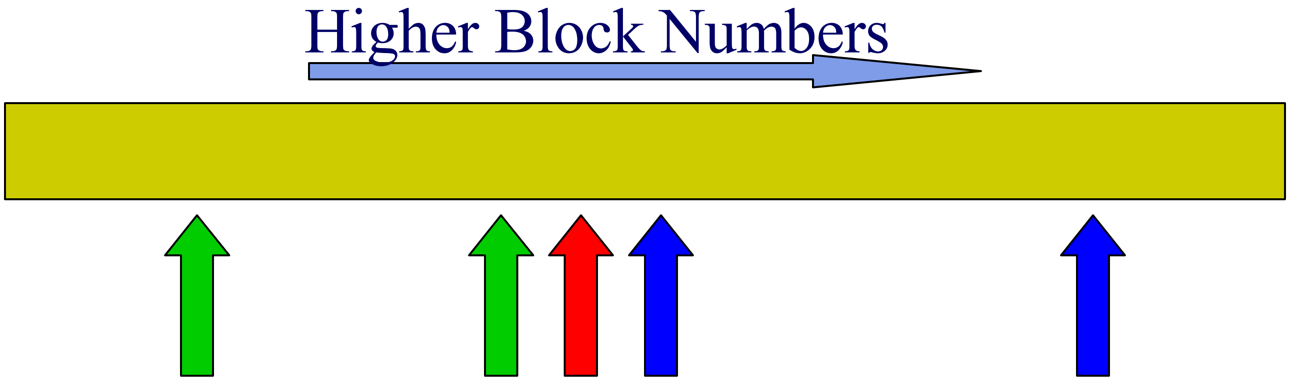


SSTF

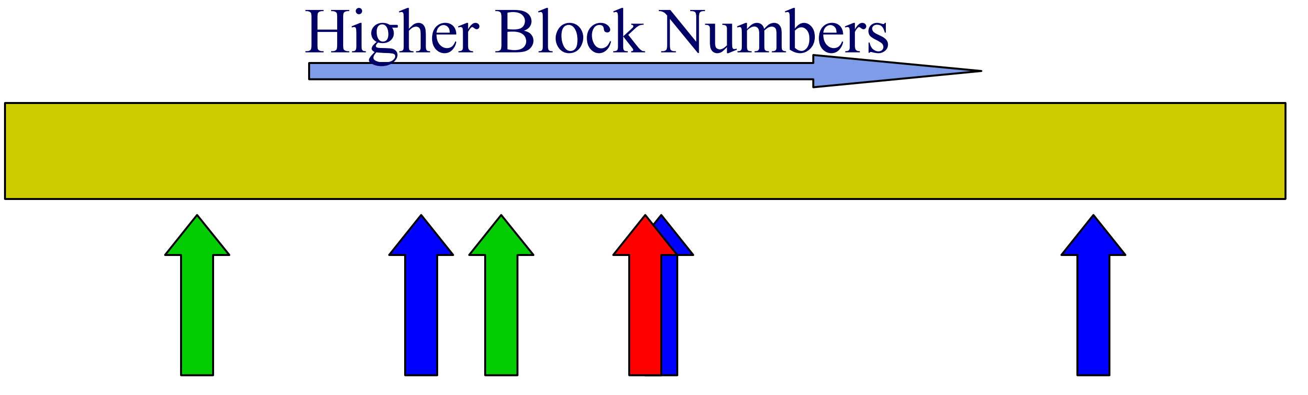
New Requests arrive…



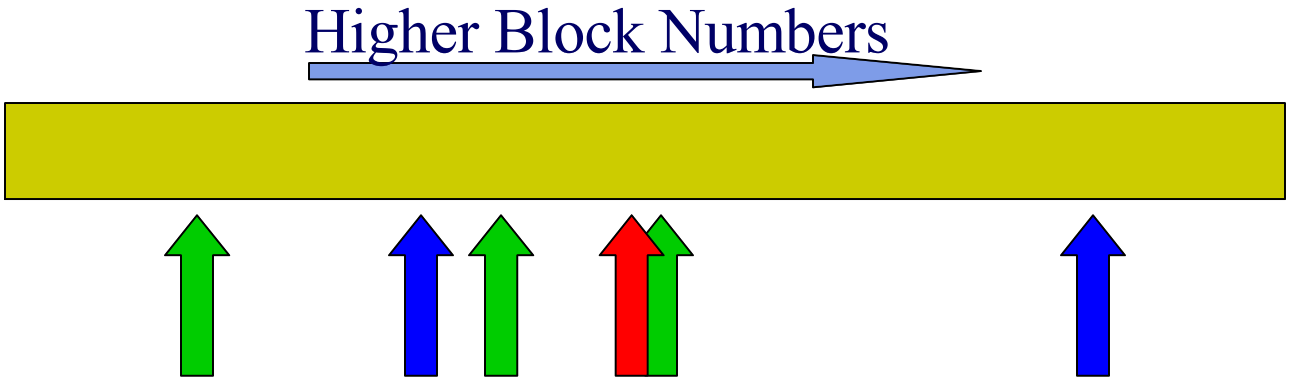
SSTF



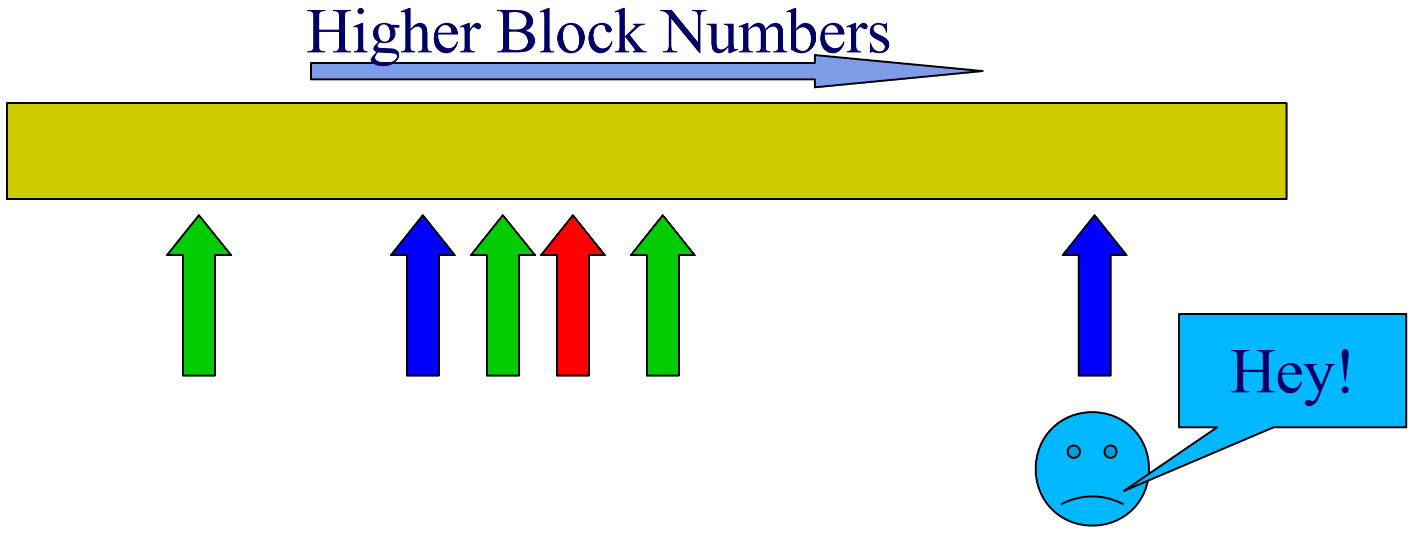
SSTF



SSTF

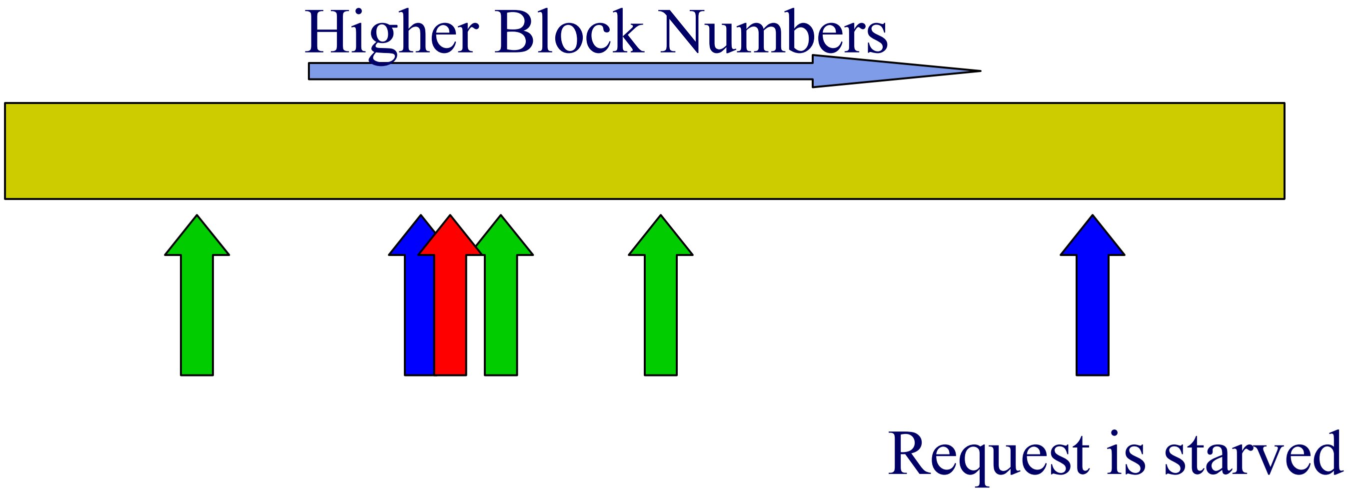


SSTF



SSTF

Starves requests that are "far away" from the head



What Went Wrong?

FCFS - “fair, but slow”

- Ignores position of disk arm, so its 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

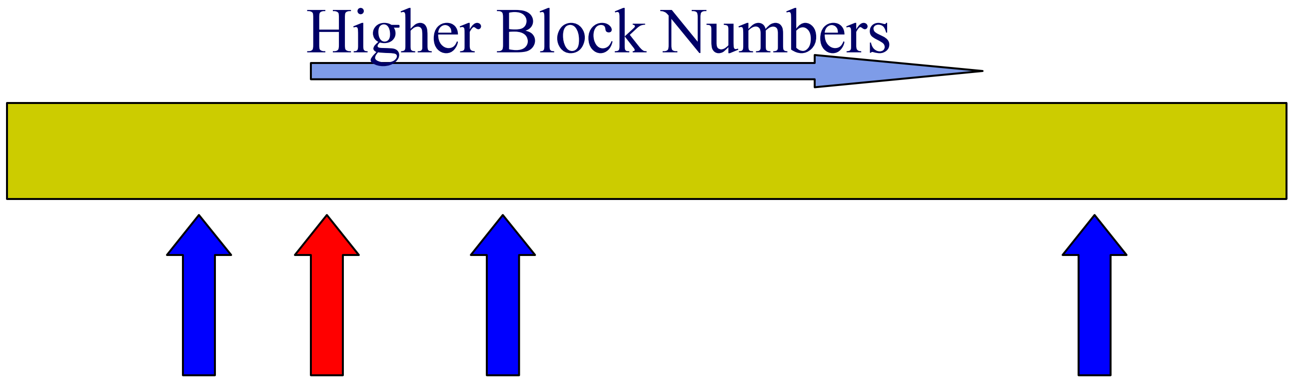
"Scan entire disk" - now that's an idea!

- 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

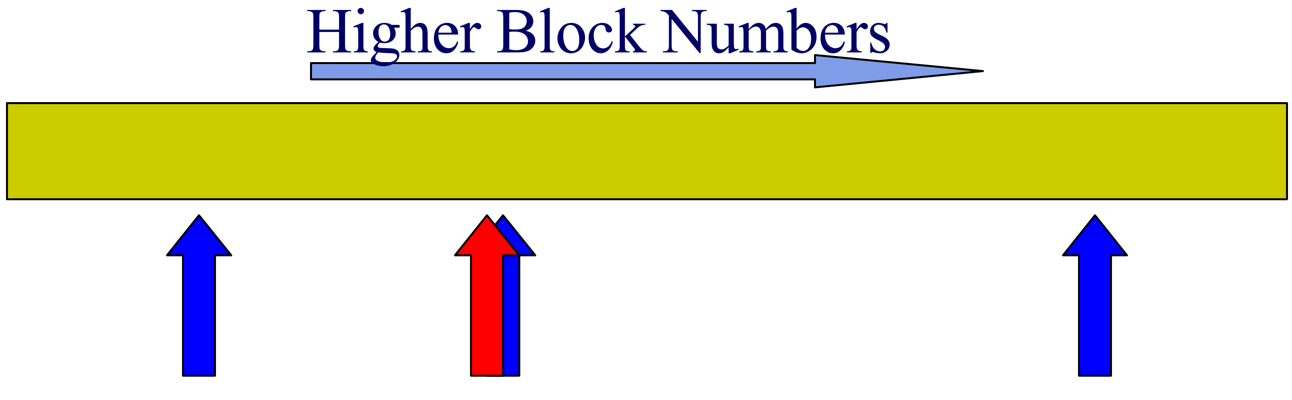
Yellow is disk



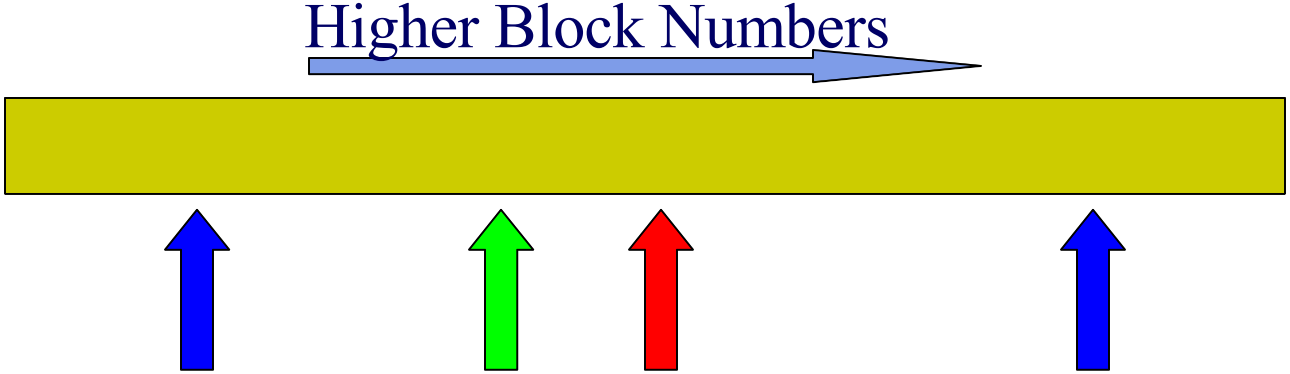
Red is disk head

Green is completed requests

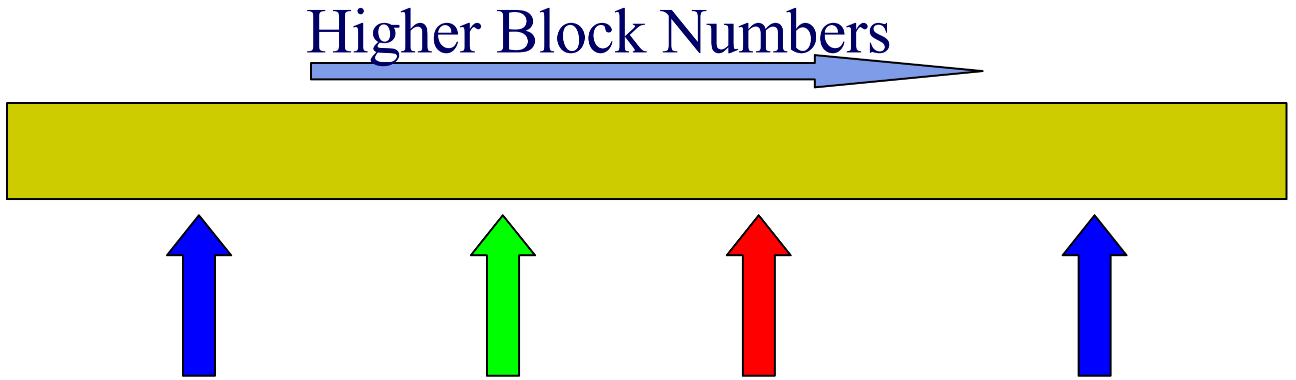
SCAN



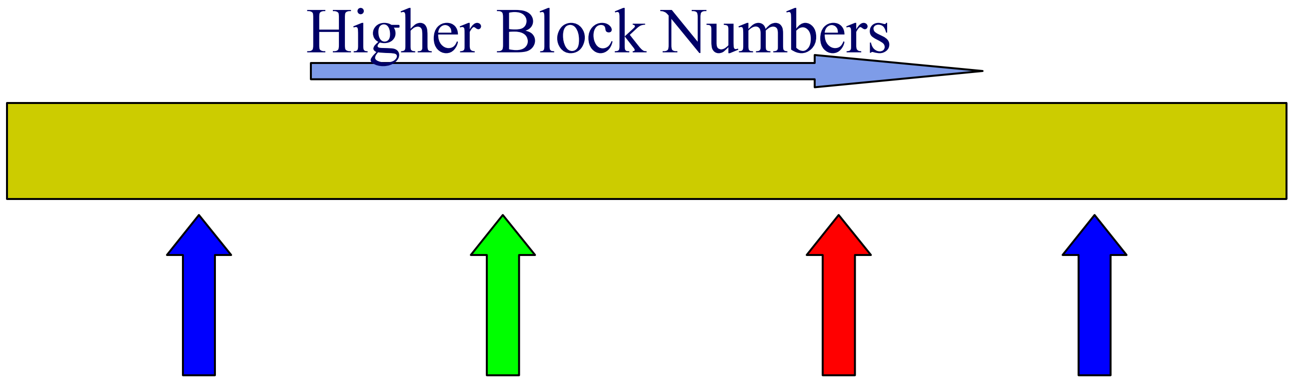
SCAN



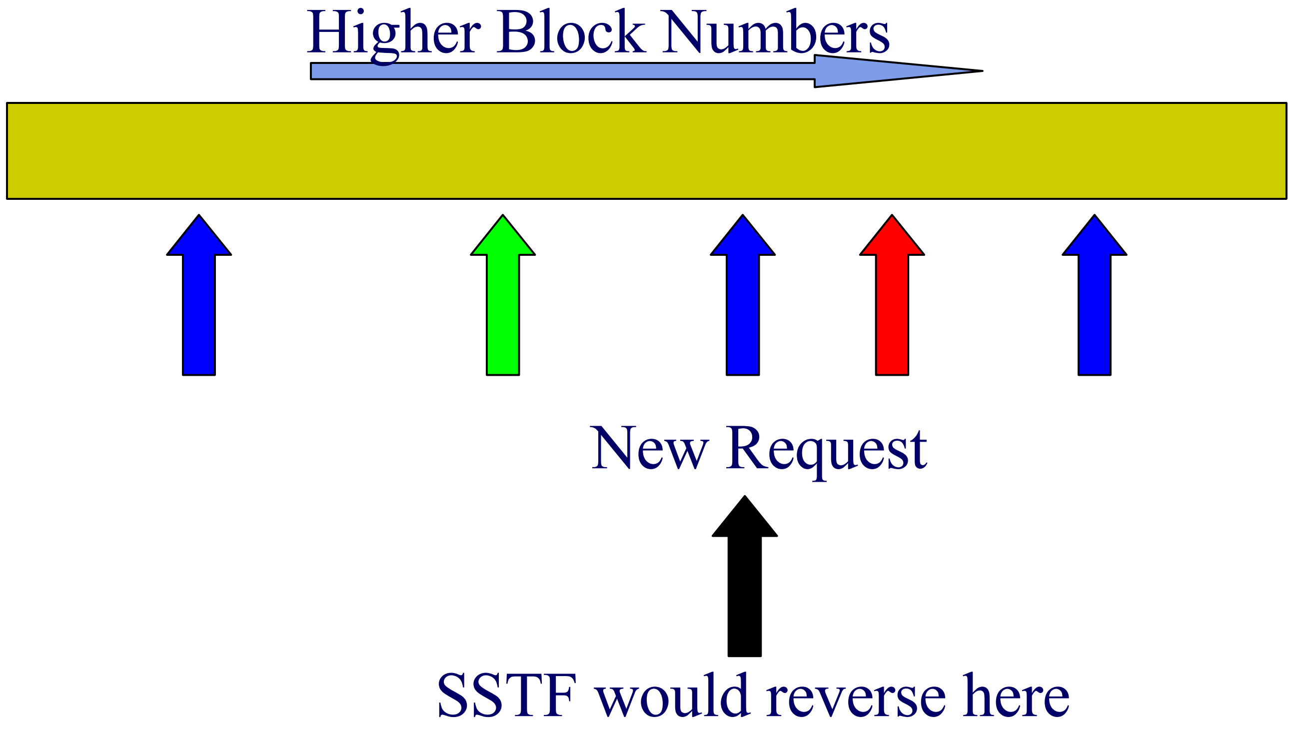
SCAN



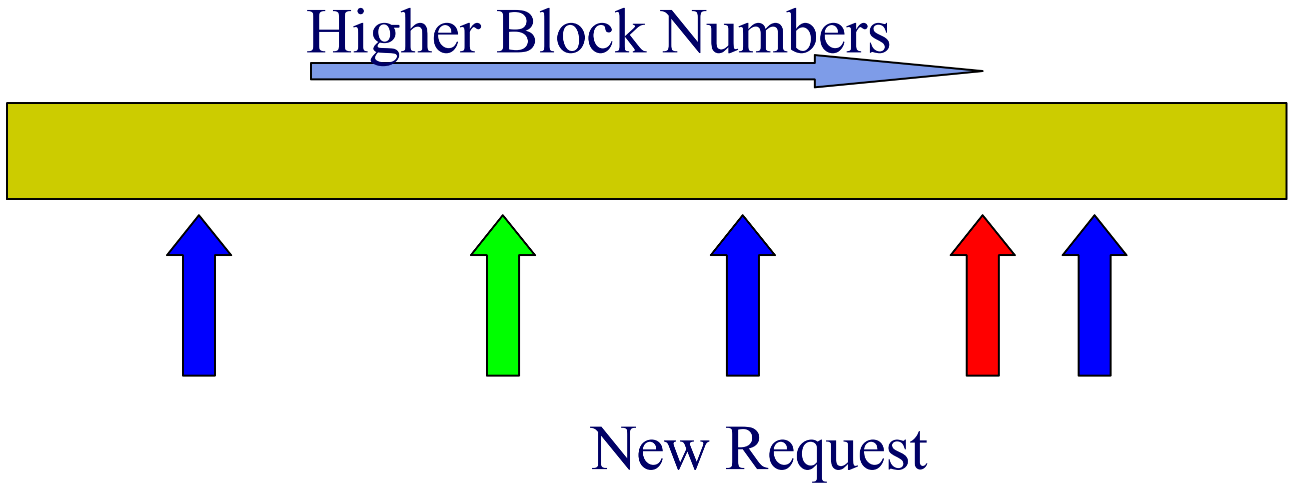
SCAN



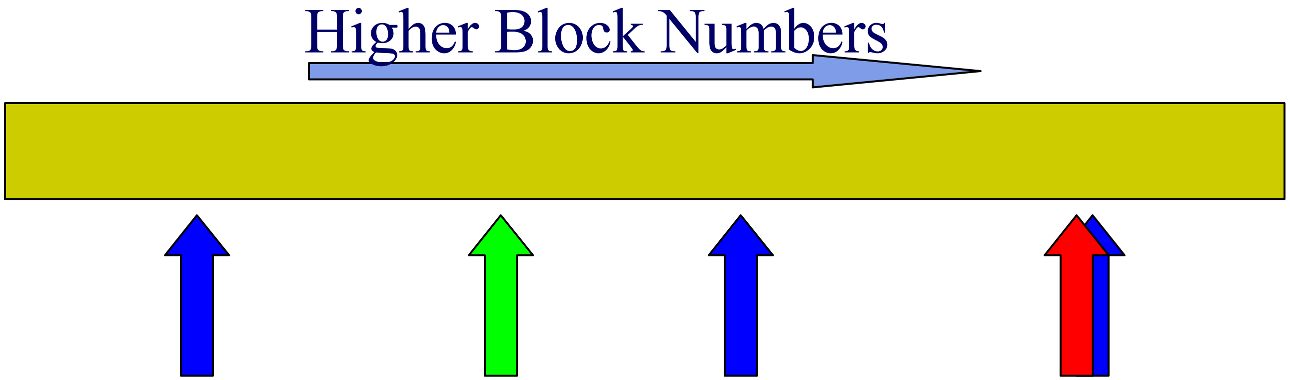
SCAN



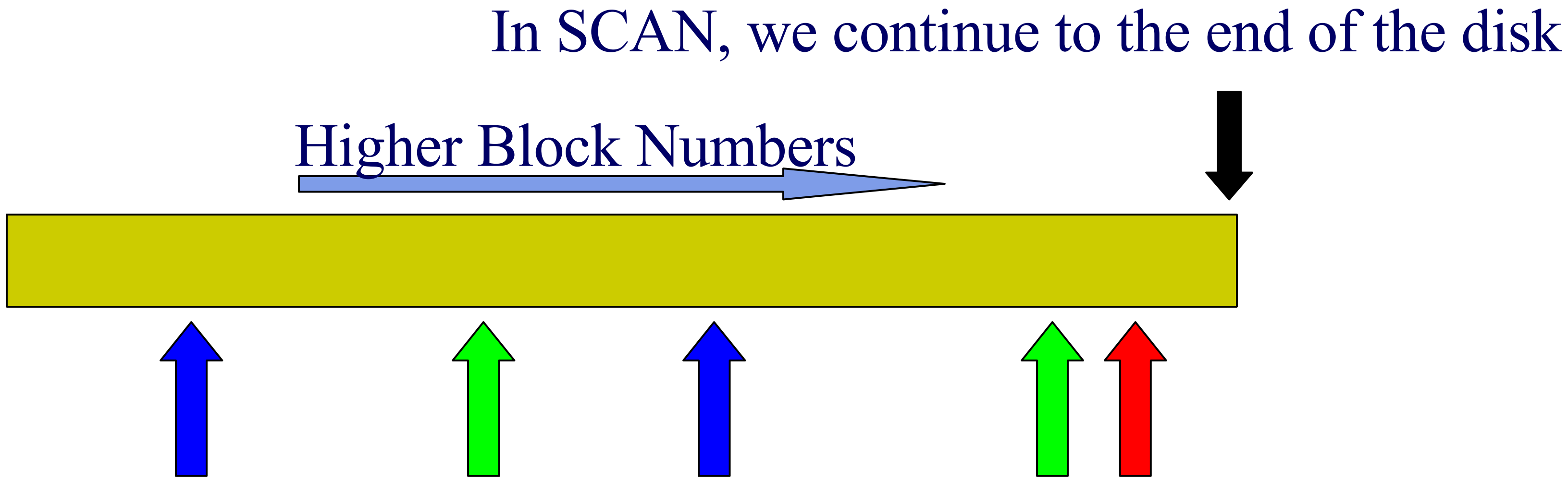
SCAN



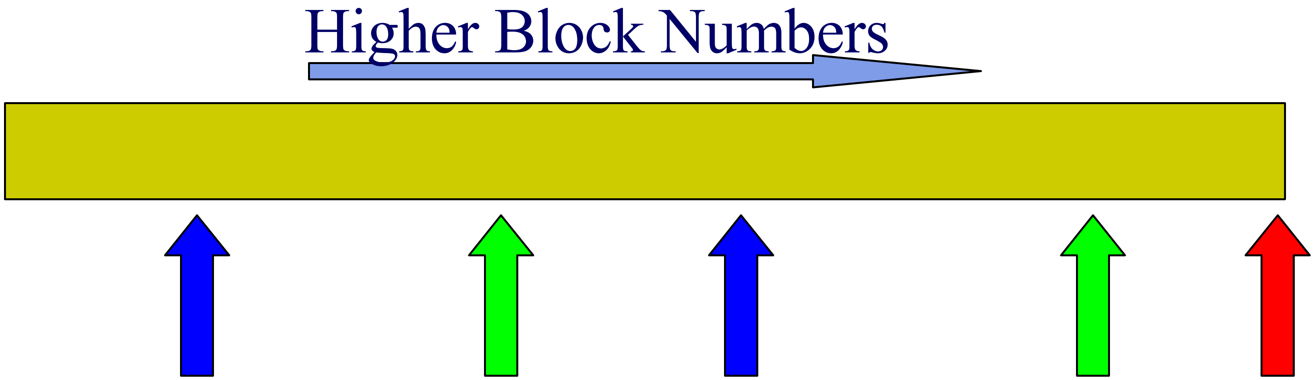
SCAN



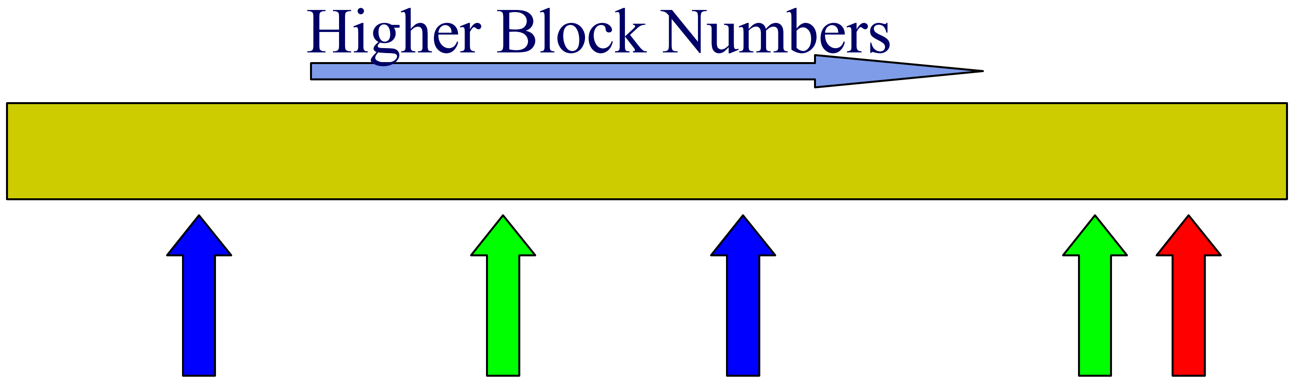
SCAN


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

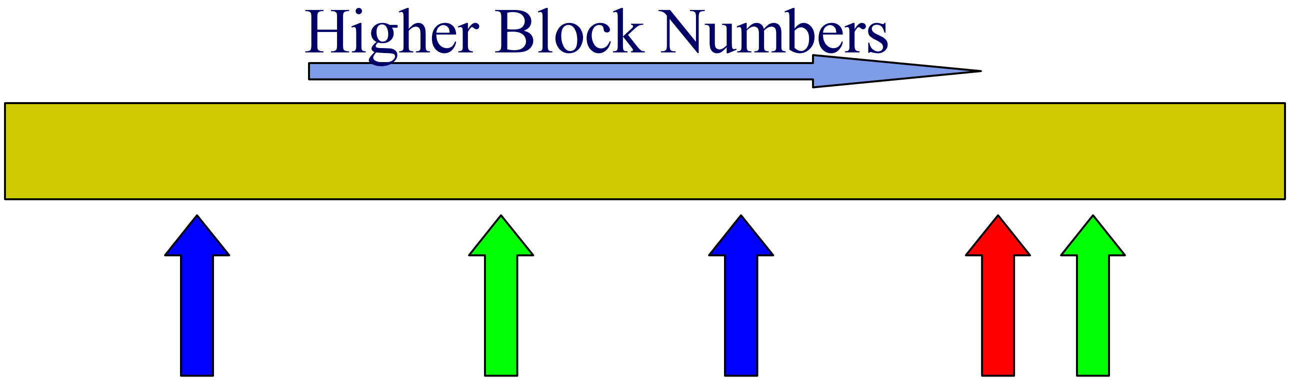
SCAN



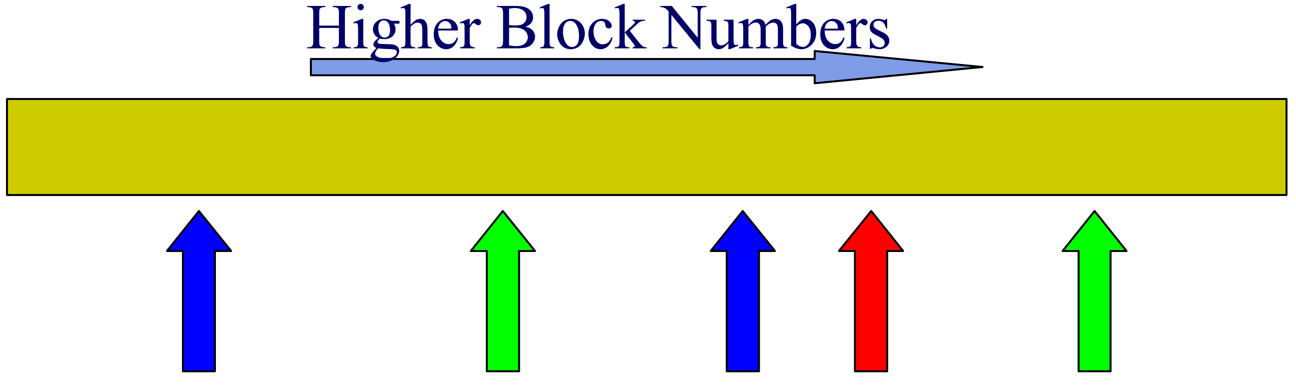
SCAN



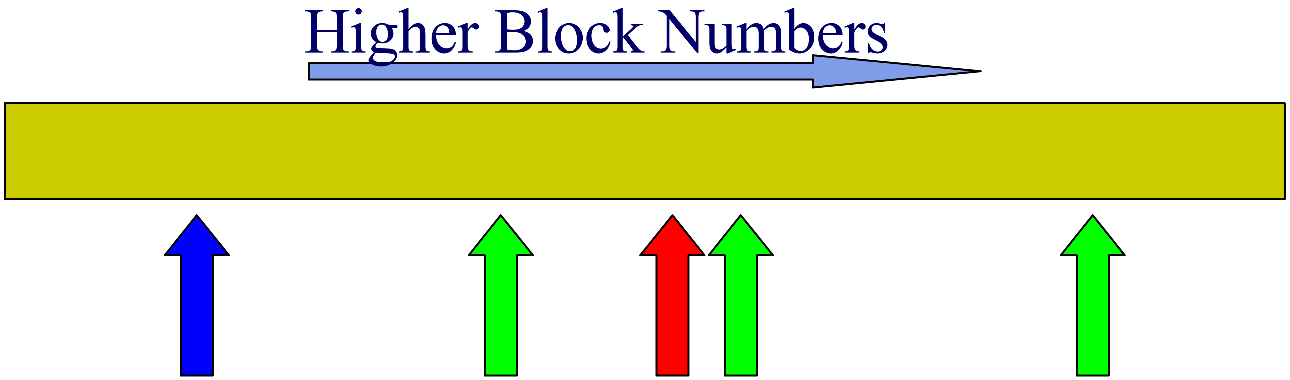
SCAN



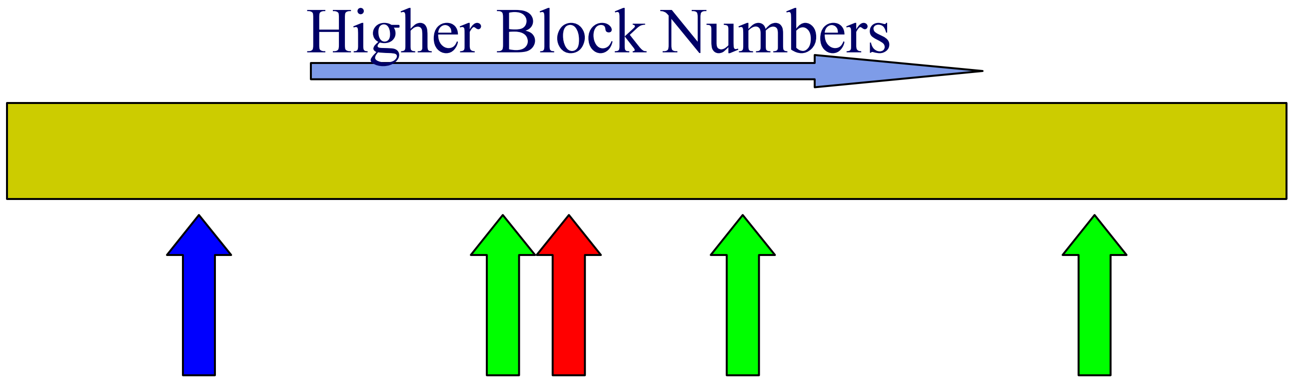
SCAN



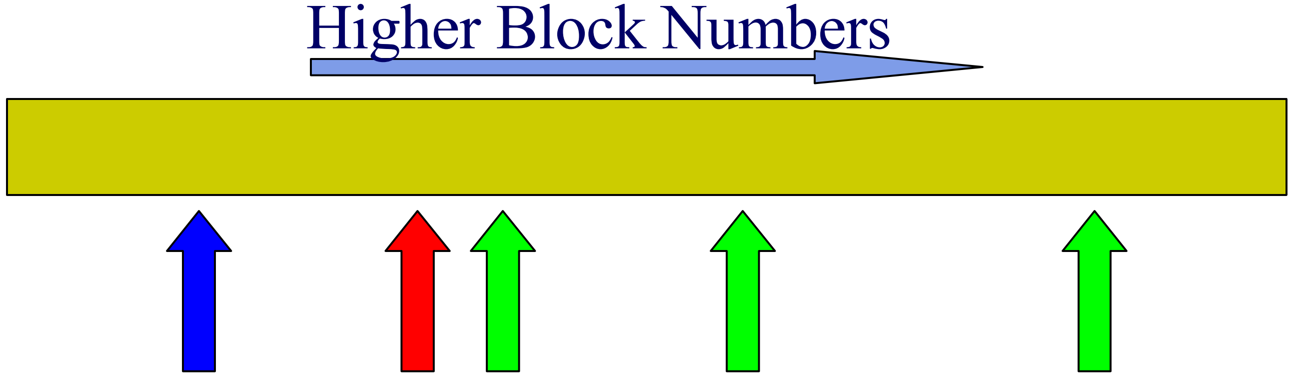
SCAN



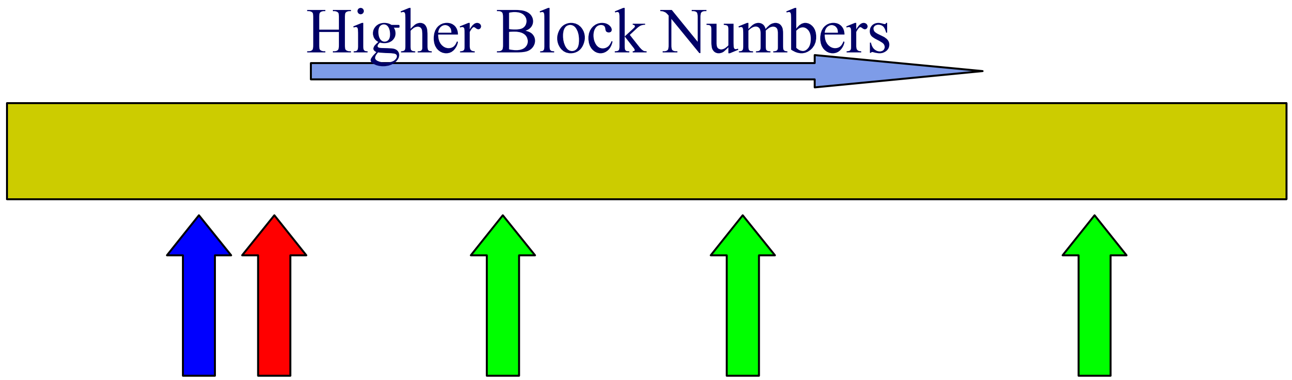
SCAN



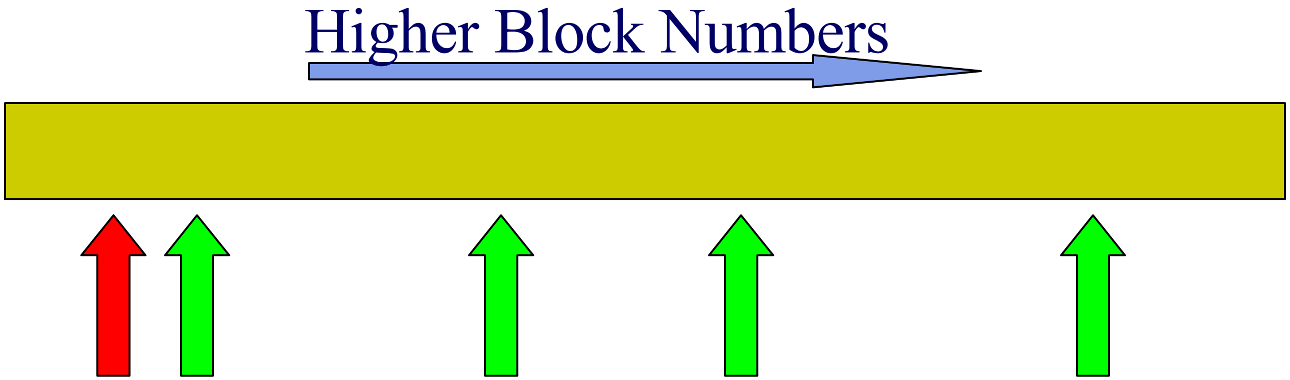
SCAN



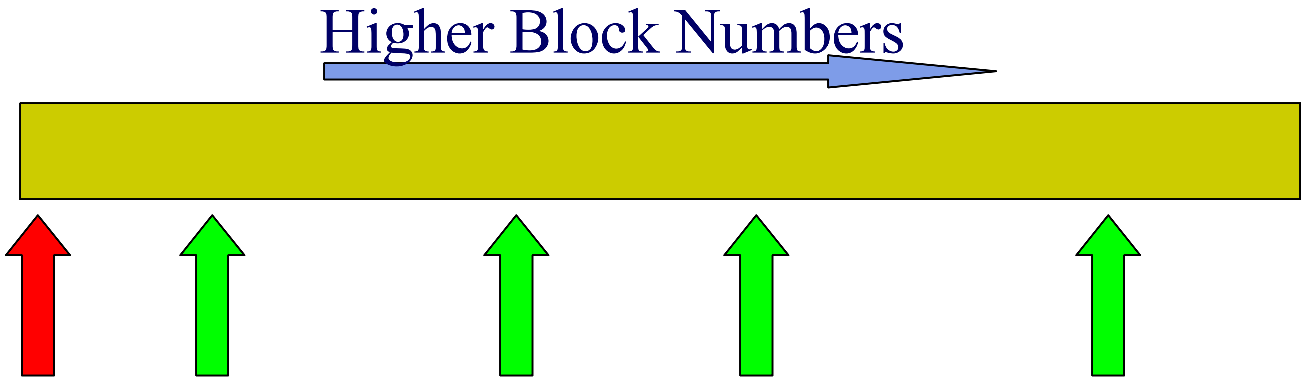
SCAN



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

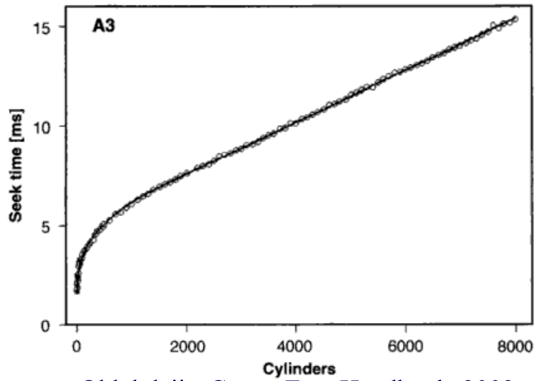
Long seek is amortized across all accesses

- 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

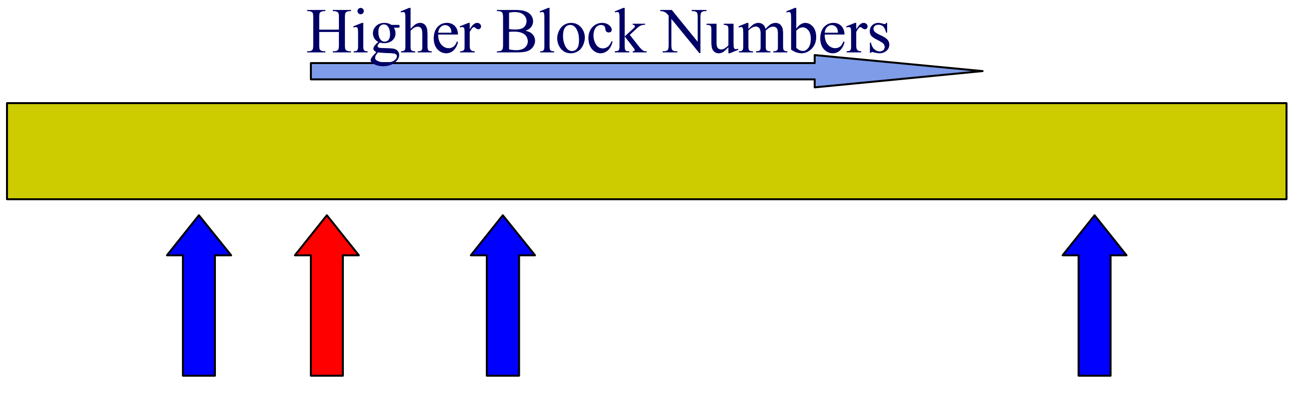
CSCAN + LOOK

Scan in one direction, as in CSCAN

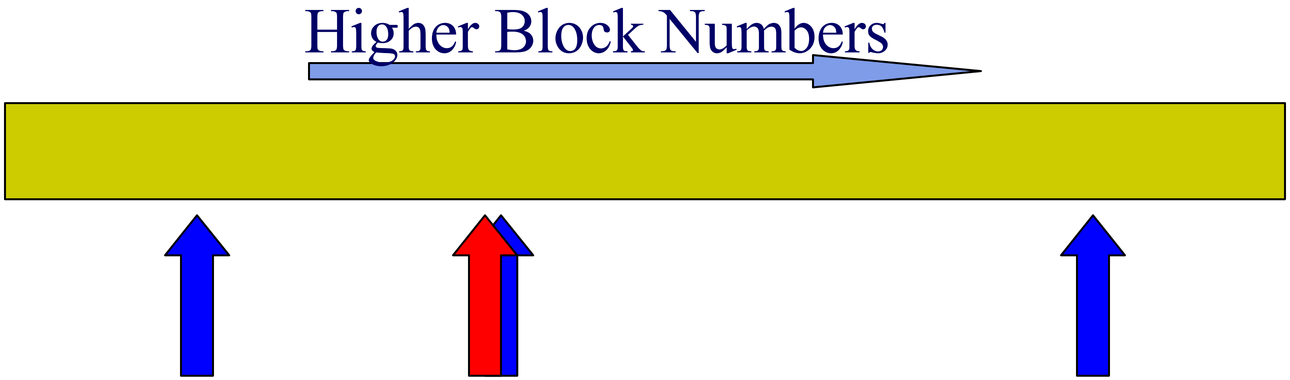
If there are no more requests in current direction go back to furthest request

Very popular

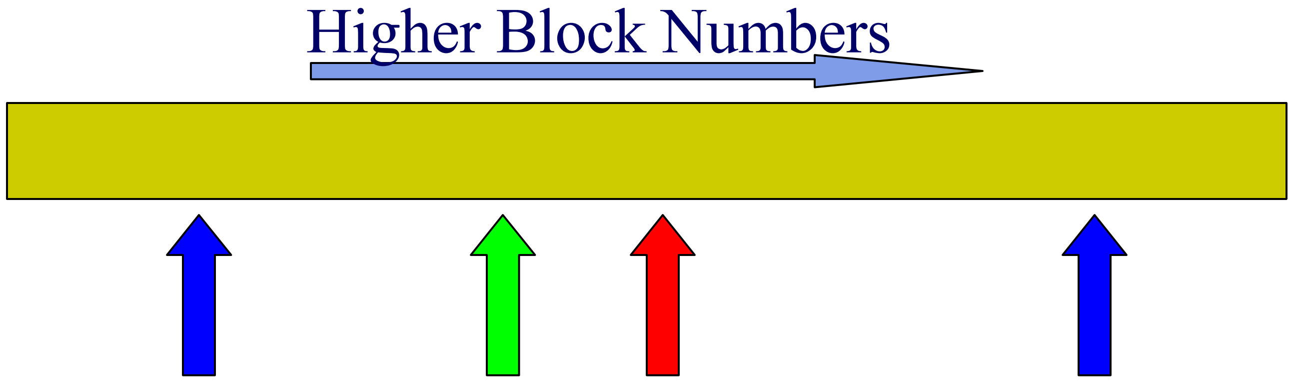
C-LOOK



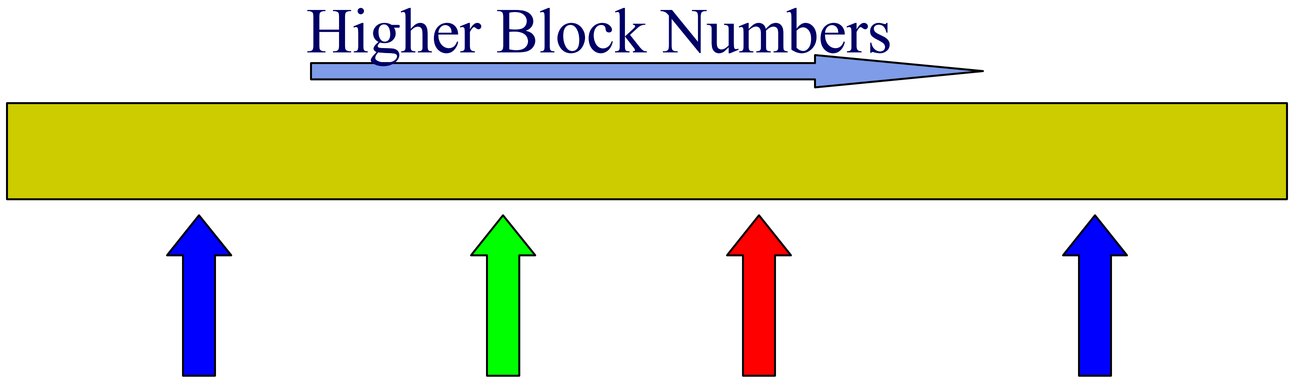
C-LOOK



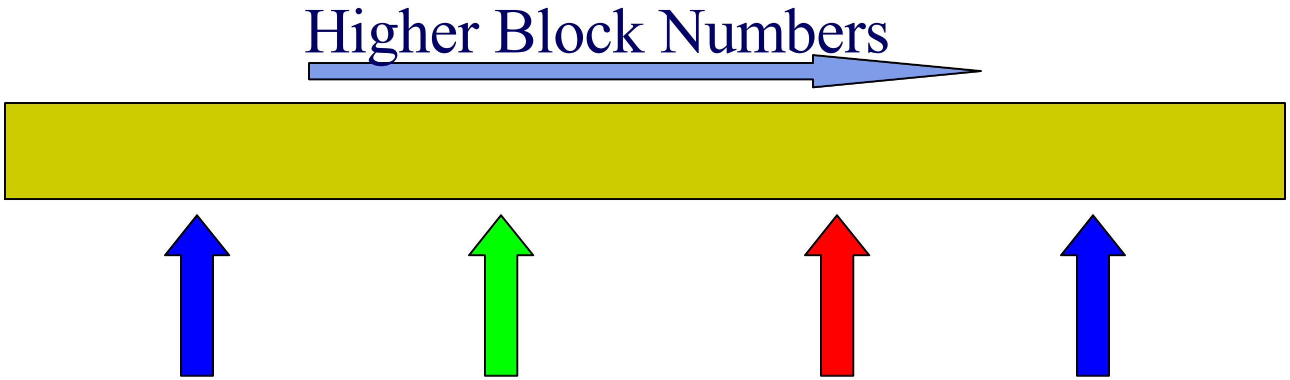
C-LOOK



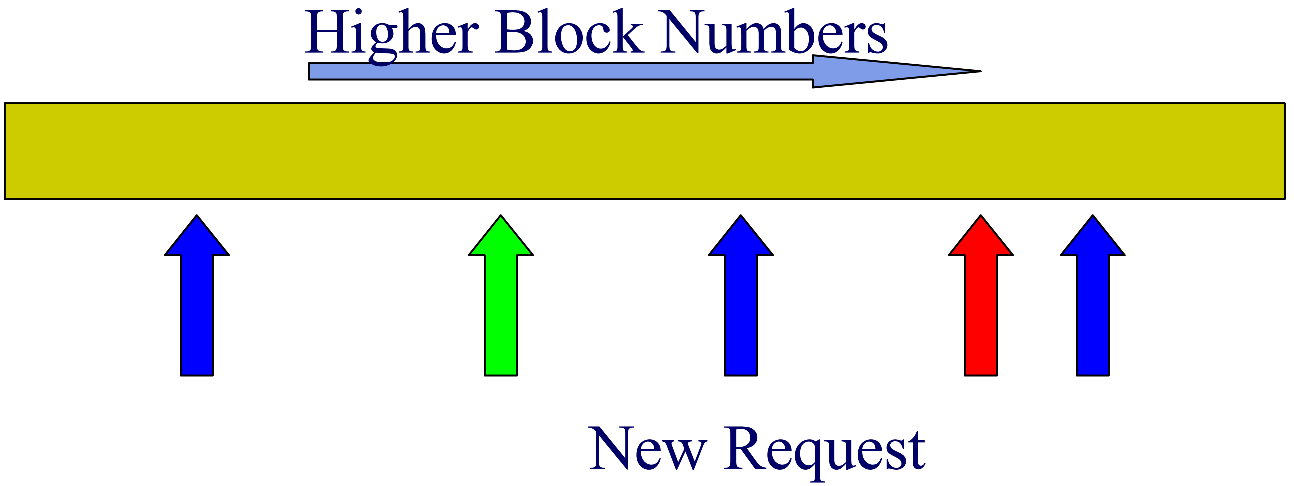
C-LOOK



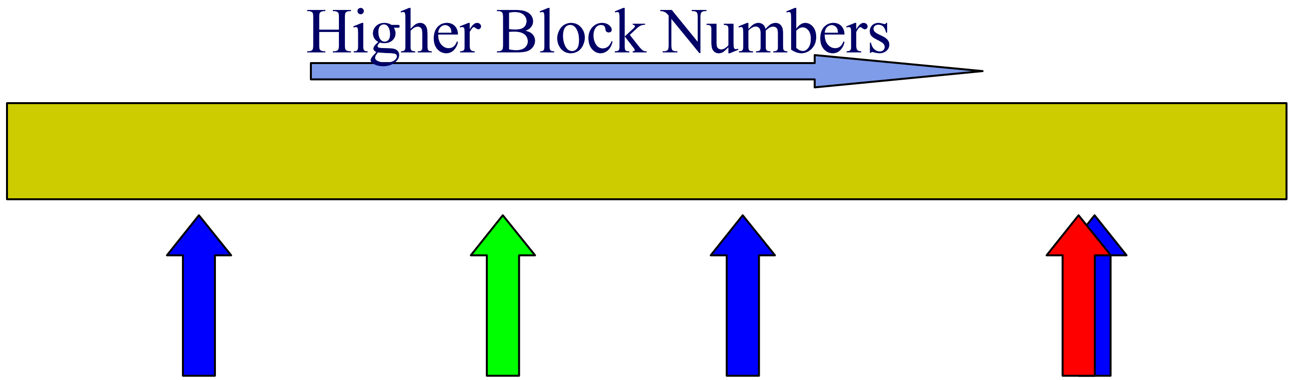
C-LOOK



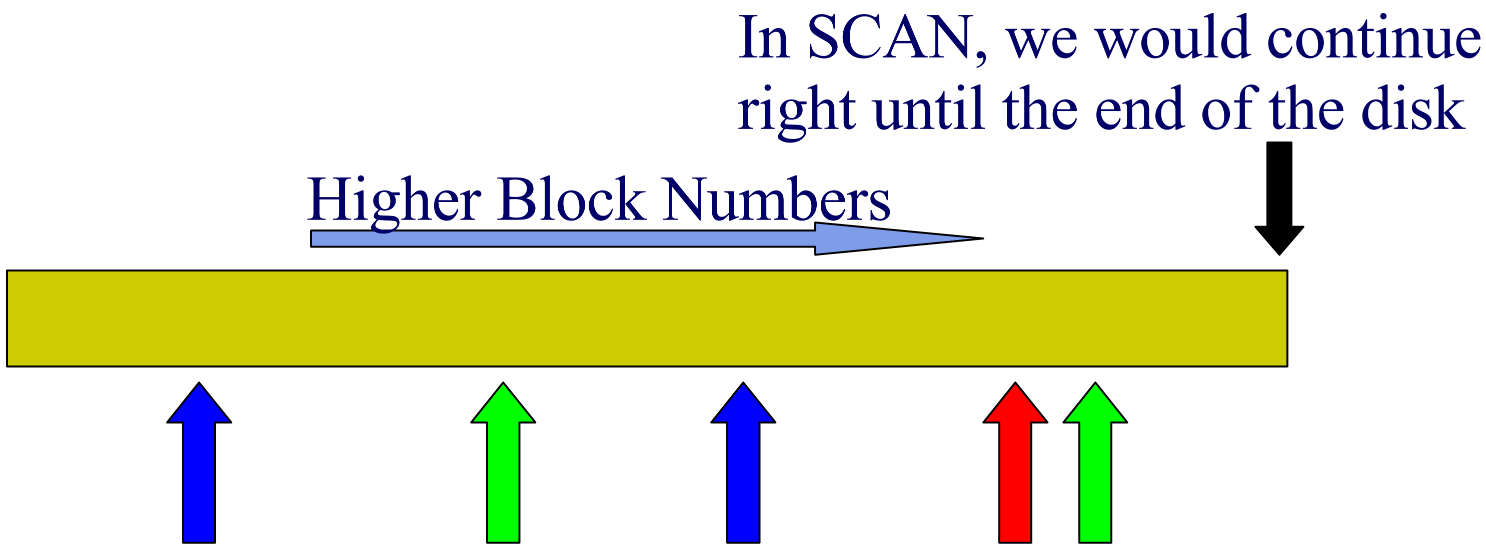
C-LOOK



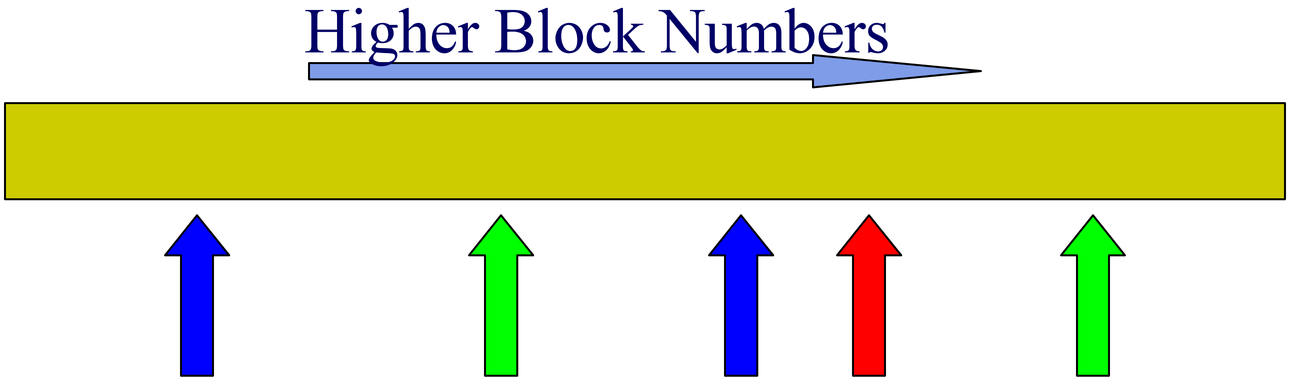
C-LOOK



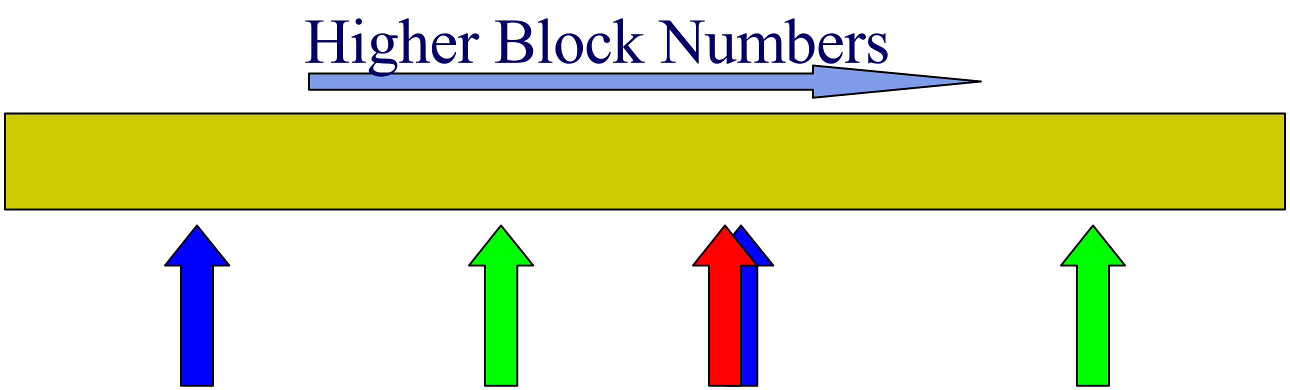
C-LOOK


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

C-LOOK

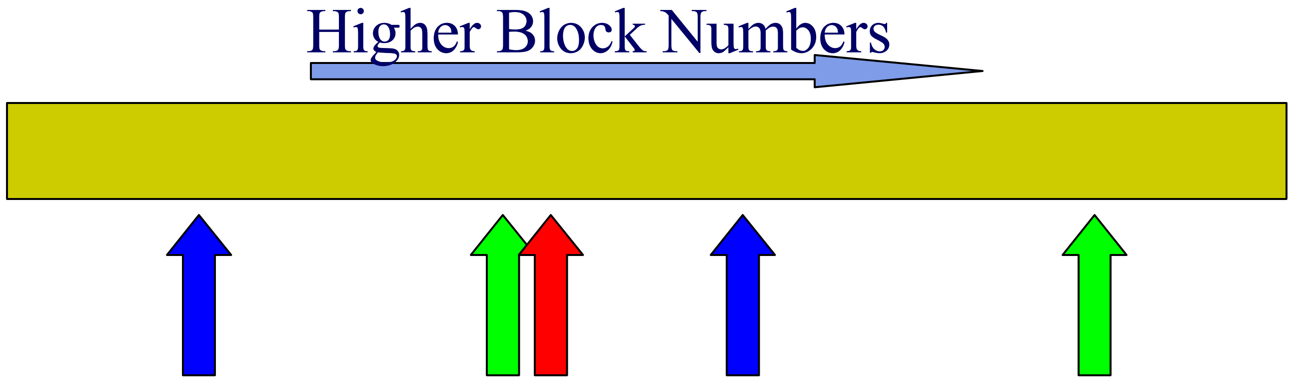


C-LOOK

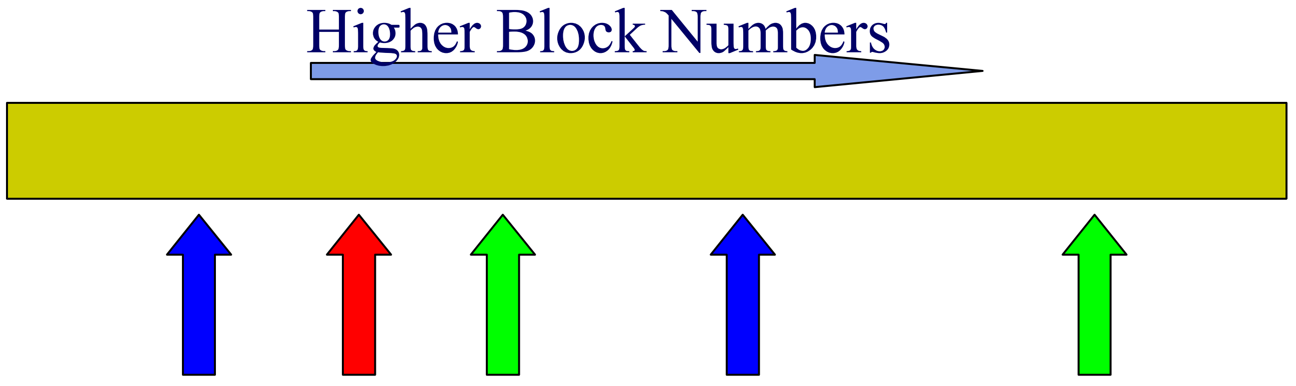


In LOOK, we would have read this request
(unfair extra service—so we'll skip it)

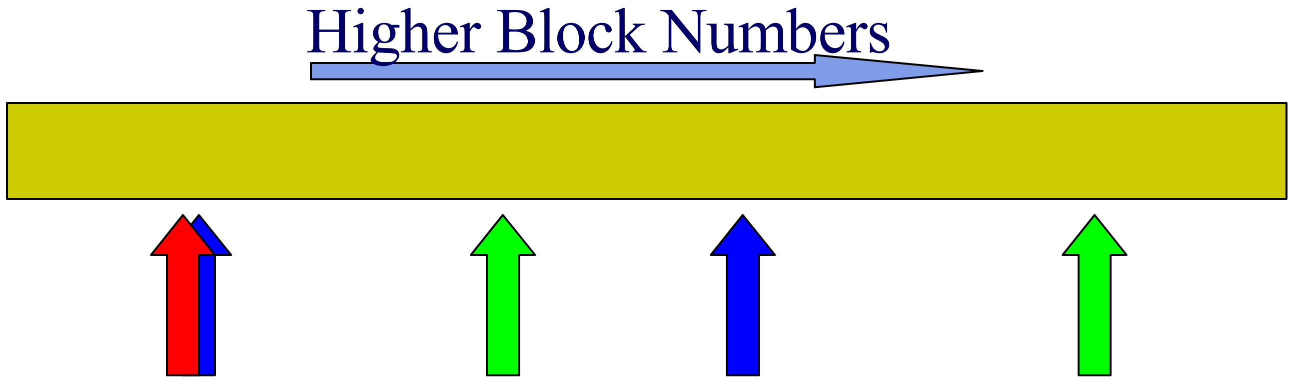
C-LOOK



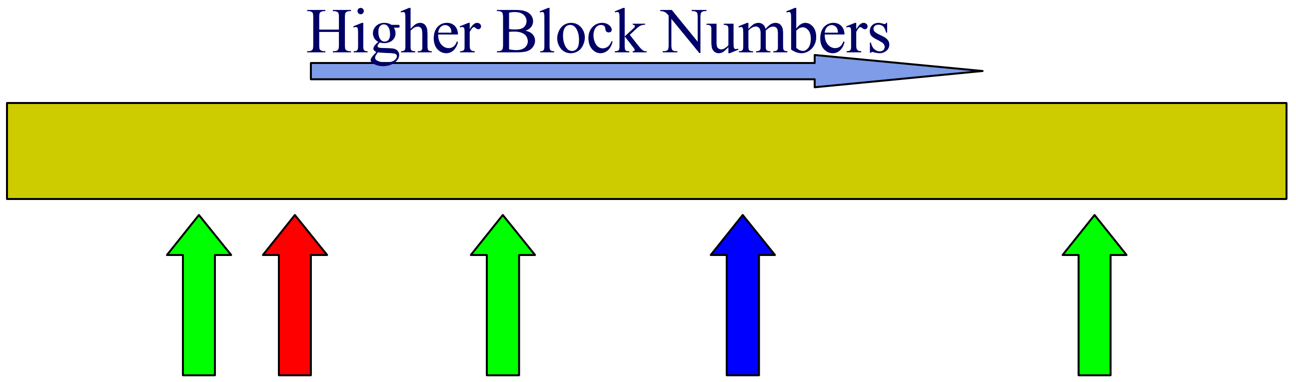
C-LOOK



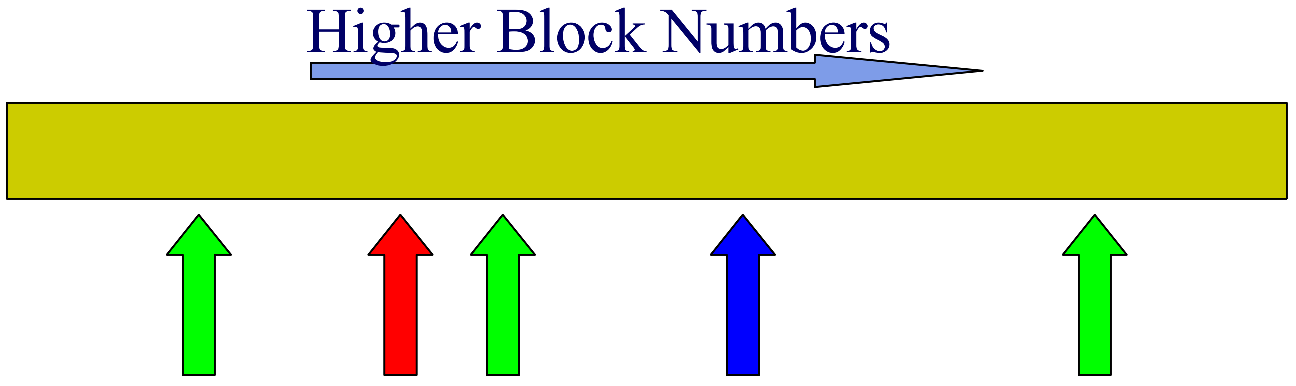
C-LOOK



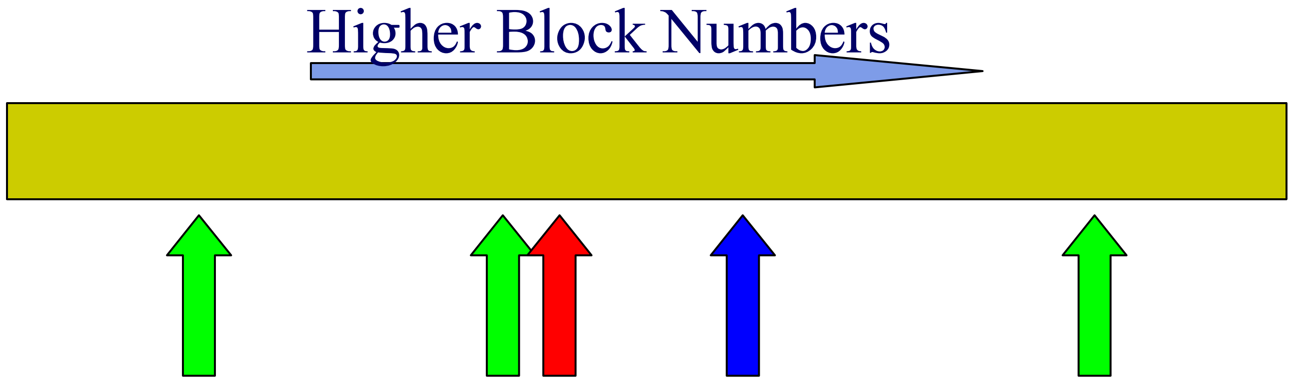
C-LOOK



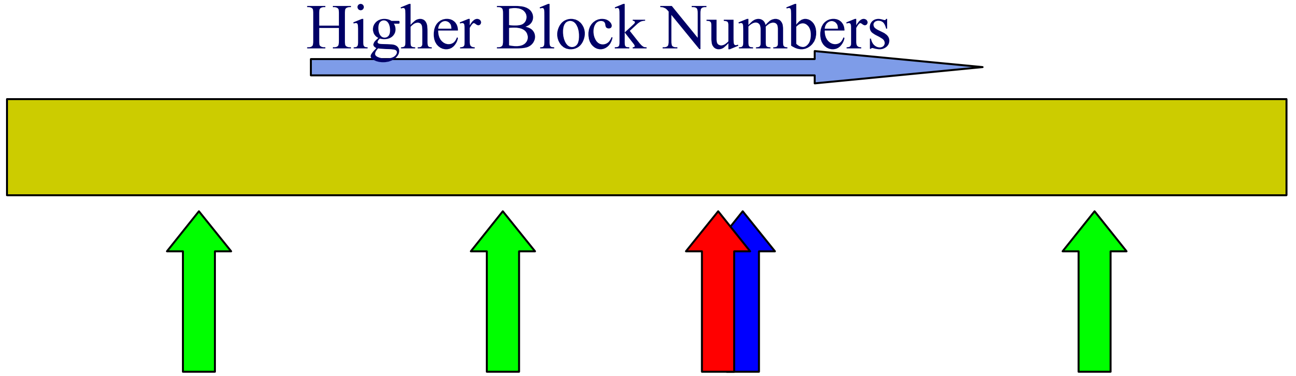
C-LOOK



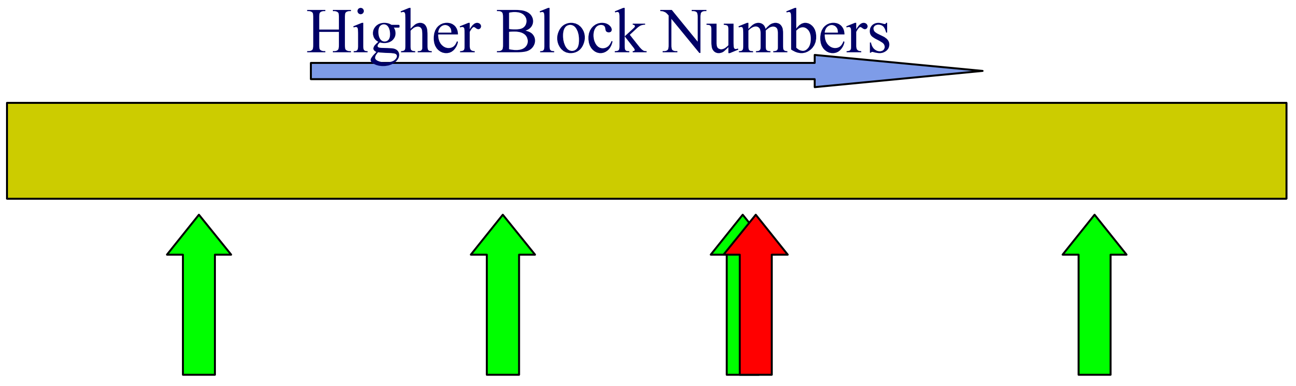
C-LOOK



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 (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

Lies Disks Tell

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

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

- 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

Forcing truth (when necessary)

- **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”**

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**

Disks lie

- **Some lies are good for performance, but be careful!**

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