W4118: disks

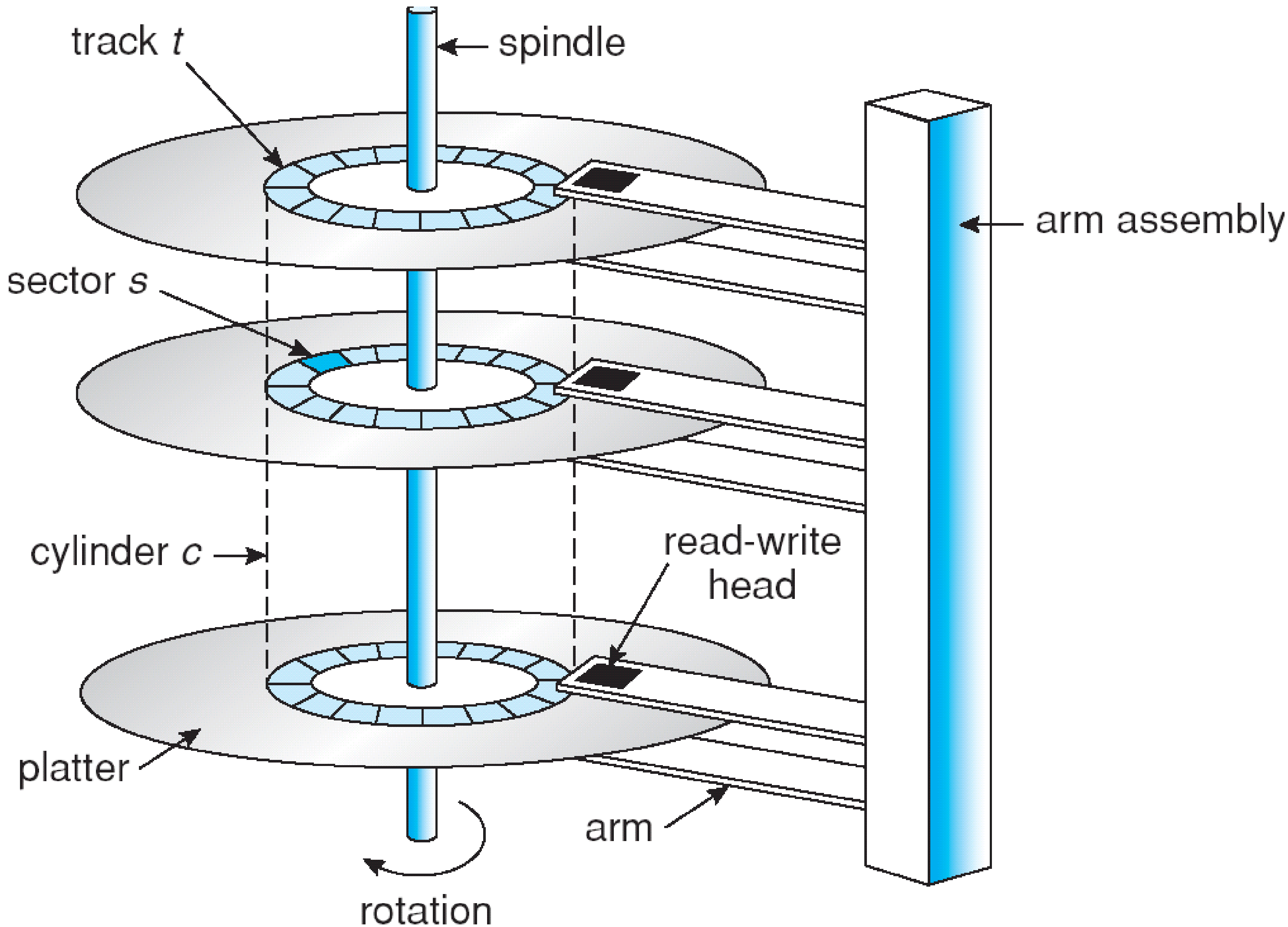
Instructor: Junfeng Yang

References: Modern Operating Systems (3rd edition), Operating Systems Concepts (8th edition), previous W4118, and OS at MIT, Stanford, and UWisc

Outline

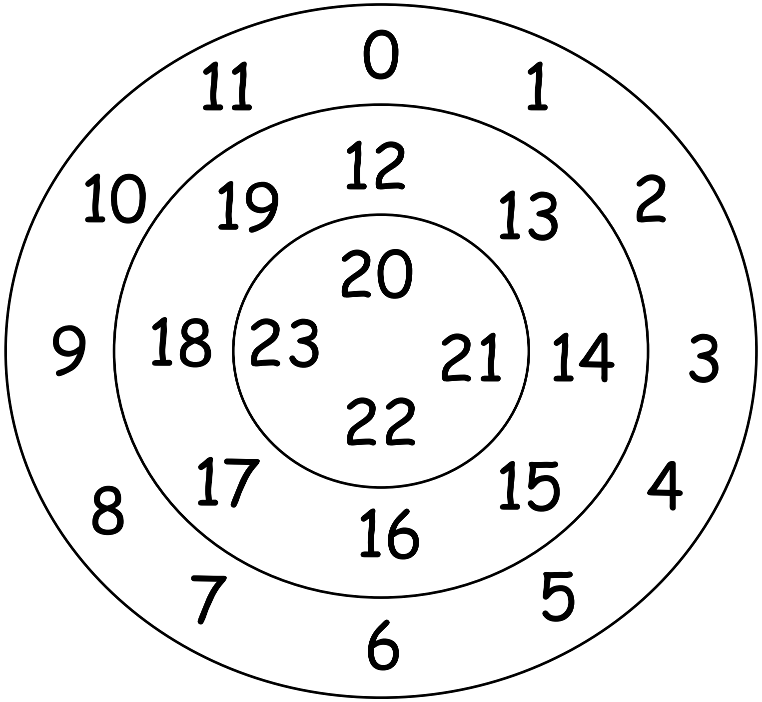
- Disk characteristics
- Disk scheduling
- Flash/SSDs

Disk structure



Disk interface

- From FS perspective: disk is addressed as a one dimension array of logical sectors
- Disk controller maps logical sector to physical sector identified by surface #, track #, and sector #

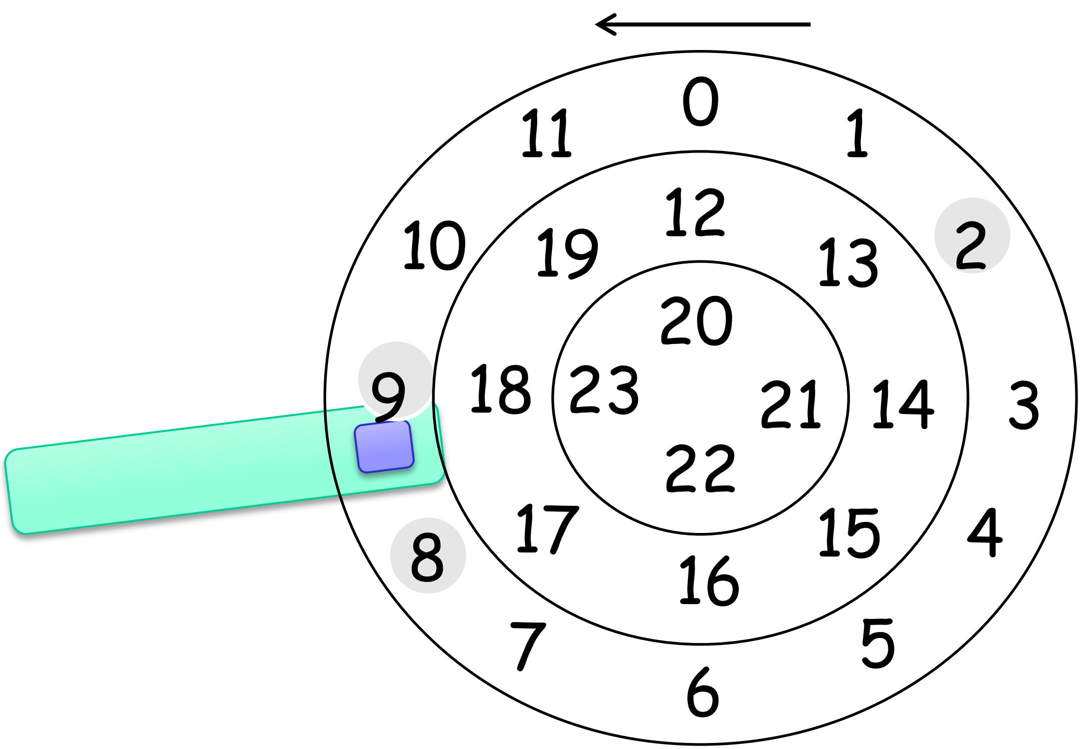


Disk latencies

- Rotational delay: rotate disk to get to the right sector
- Seek time: move disk arm to get to the right track
- Transfer time: get bits off the disk

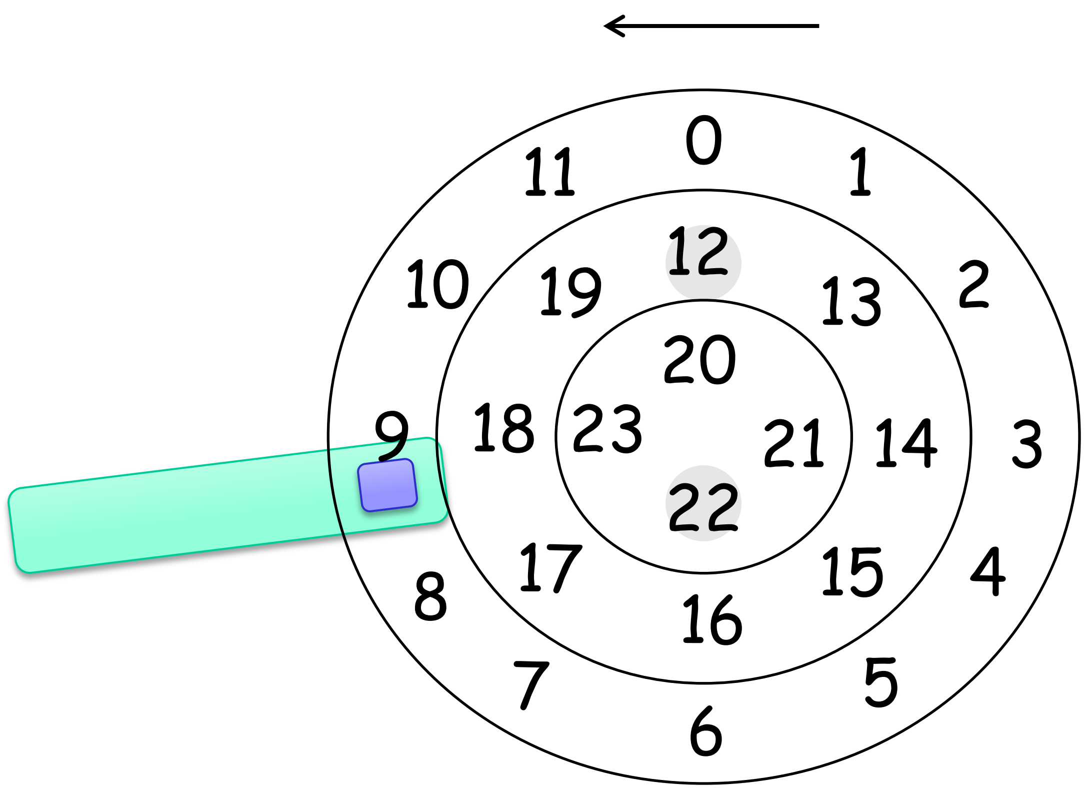
Rotational delay

- Full rotation time: e.g., 4-8ms
- Average rotational delay: half of full rotation time



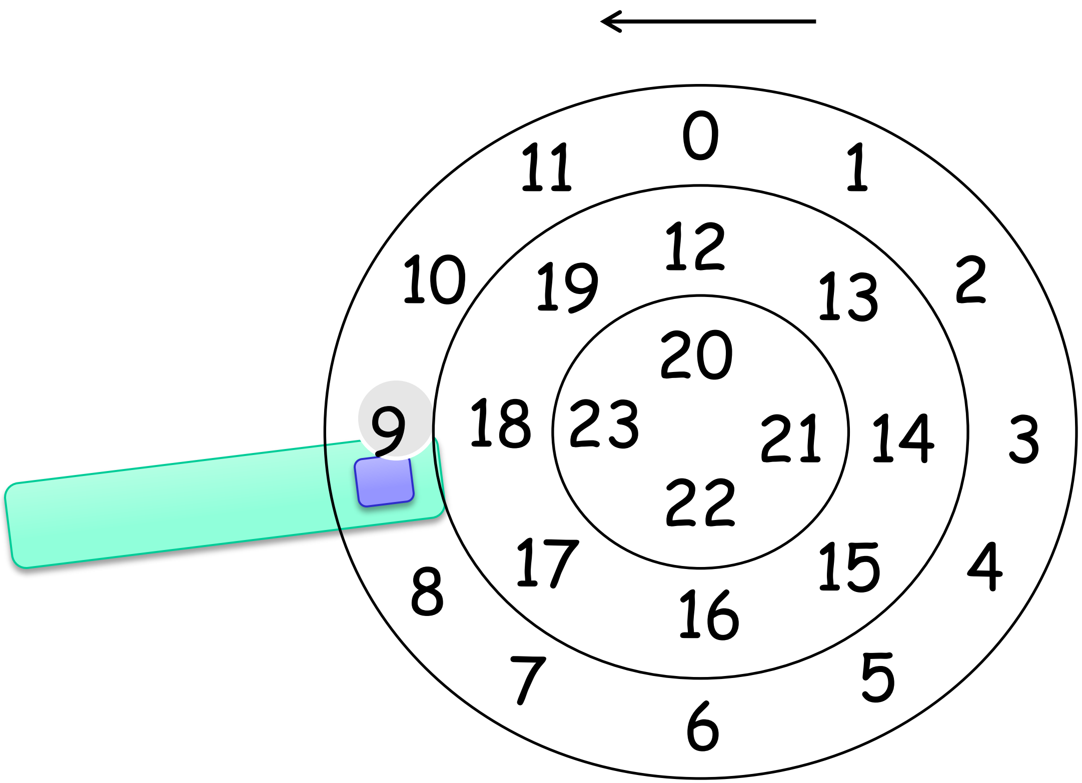
Seek time

- Must move arm to the right track
- Can take a while (e.g., 0.5 - 2ms)



Transfer time

- Transfer bits out of disk
- Actually pretty fast (e.g., 125MB/s)



I/O time (T) and rate (R)

- T = Rotational delay + seek time + txfer time
- R = Size of transfer / T
- Workload 1: large sequential accesses?
- Workload 2: small random accesses?

Example

	Barracuda	**Cheetah 15K.5**
Capacity	1TB	300GB
Rotational speed	7200 RPM	15000 RPM
Rotational latency (ms)	4.2	2.0
Avg seek (ms)	9	4
Max Transfer	105 MB/s	125 MB/s
Platters	4	4
Connects via	SATA	SCSI

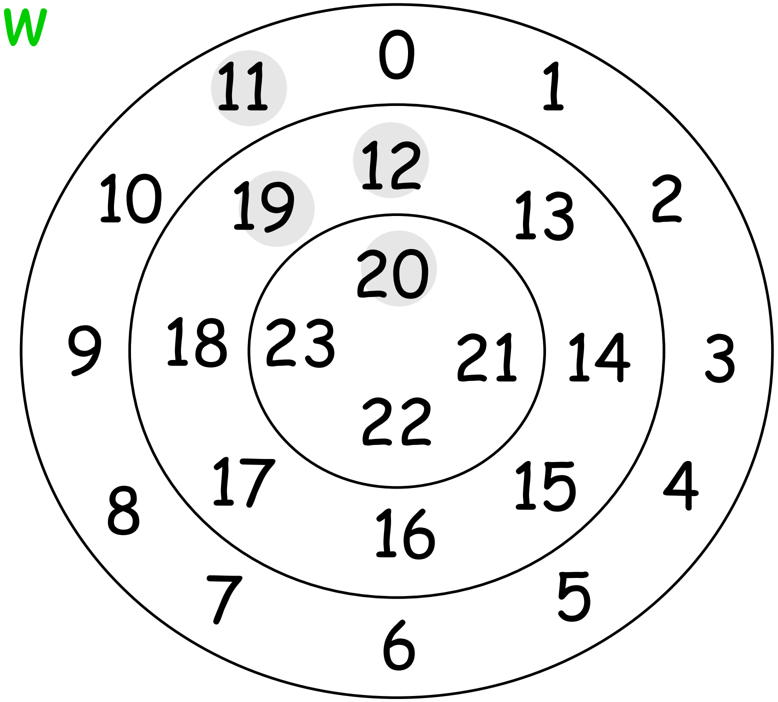
- Random 4KB read
  - Barracuda: T = 13.2ms, R = 0.31MB/s
  - Cheetah: T = 6ms, R = 0.66MB/s
- Sequential 100 MB read
  - Barracuda: T = 950ms, R = 105 MB/s
  - Cheetah: T = 800ms, R = 125 MB/s

Design tip: use disks sequentially

- Disk performance differs by a factor of 200 or 300 for random vs sequential accesses
- When possible, access disks sequentially

Mapping of logical sectors to physical

- Logical sector 0: the first sector of the first (outermost) track of the first surface
- Logical sector address incremented within track, then tracks within cylinder, then across cylinders, from outermost to innermost
- Track skew



Pros and cons of default mapping

- Pros
  - Simple to program
  - Default mapping reduces seek time for sequential access
- Cons
  - FS can't precisely see mapping
  - Reverse-engineer mapping in OS is difficult
    - # of sectors per track changes
    - Disk silently remaps bad sectors

Disk cache

- Internal memory (8MB-32MB) used as cache
- Read-ahead: "track buffer"
  - Read contents of entire track into memory during rotational delay
- Write caching with volatile memory
  - Write back or immediate reporting: claim written to disk when not
    - Faster, but data could be lost on power failure
  - Write through: ack after data written to platter

Disk technology trends

- Data ➔ more dense
  - More bits per square inch
  - Disk head closer to surface
  - Create smaller disk with same capacity
- Disk geometry ➔ smaller
  - Spin faster ➔ Increase b/w, reduce rotational delay
  - Faster seek
  - Lighter weight
- Disk price ➔ cheaper
- Density improving more than speed (mechanical limitations)

Outline

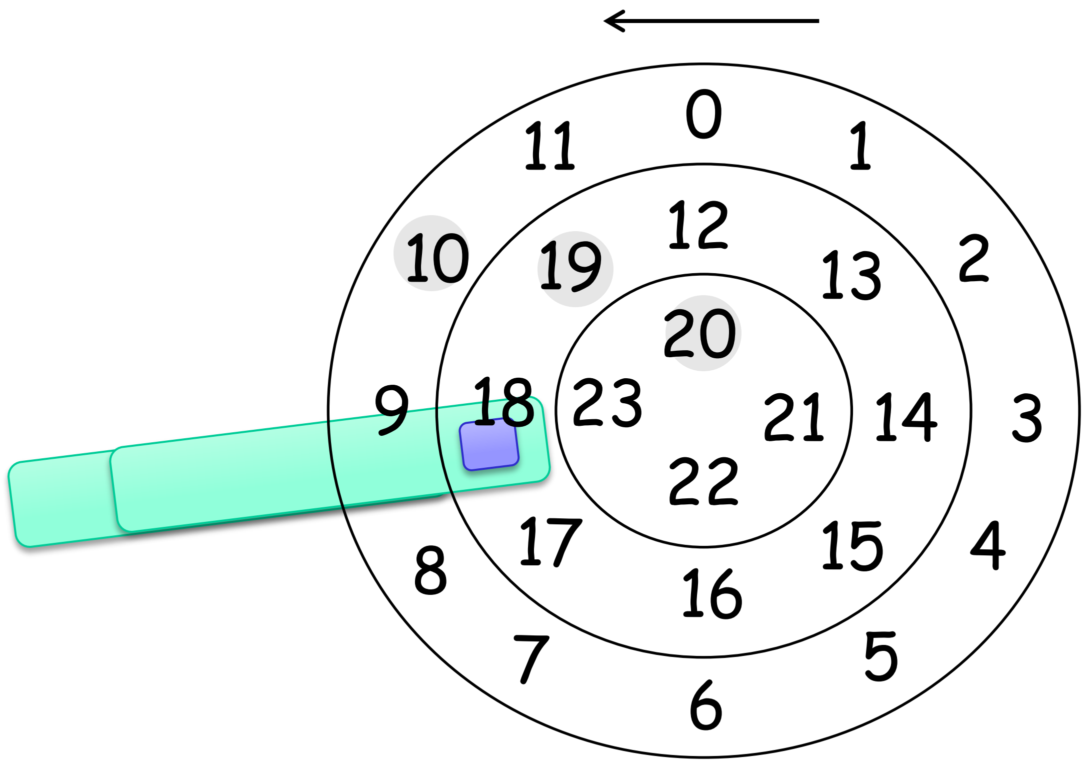
- Disk characteristics
- Disk scheduling
- Flash/SSDs

Disk scheduling

- Goal: minimize positioning time
  - Performed by both OS and disk itself
  - Why?
- Schedule requests in order received (FCFS)
  - Advantage: fair
  - Disadvantage: high seek cost and rotation
- Shortest seek time first (SSTF):
  - Handle nearest cylinder next
  - Advantage: reduces arm movement (seek time)
  - Disadvantage: unfair, can starve some requests

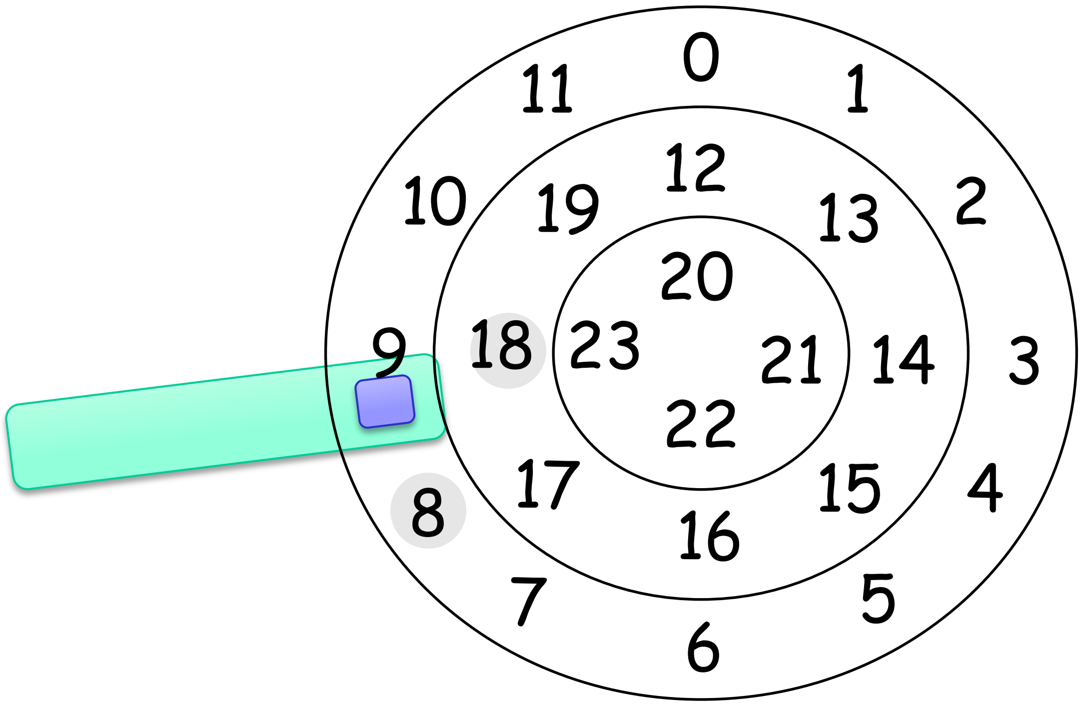
Elevator (aka SCAN or C-SCAN)

- Disk arm sweeps across disk
- If request comes for a block already serviced in this sweep, queue it for next sweep



Modern disk scheduling issues

- Elevator (or SSTF) ignores rotation!
- Shortest positioning time first (SPTF)
- OS + disk work together to implement



Outline

- Disk characteristics
- Disk scheduling
- Flash/SSDs

Flash and SSDs

- Solid state storage
  - Use silicon transistors to store data rather than spinning magnetic platters
  - Fundamentally different characteristics than disks
  - Increasing popularity in mobile devices, large server farms
- Pros
  - No moving parts - robust to mechanical failure
  - No mechanical limitations: high throughput, random access
  - Less energy use, less heat
  - High density
- Cons
  - Expensive
  - Unfavorable reliability characteristics over time (bit rot)
  - Limitations on read-modify-write cycles
  - Complex to use

Basic Idea

- Use silicon devices based on MOSFETs
  - Metal Oxide Field Effect Transistor
  - Also used in DRAM
  - Each cell contains a single MOSFET with an additional "floating gate"

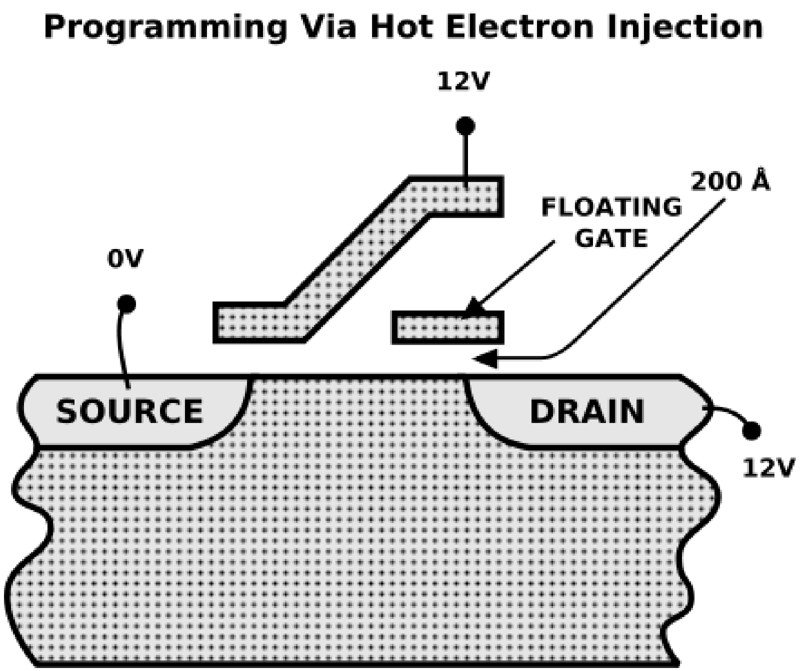
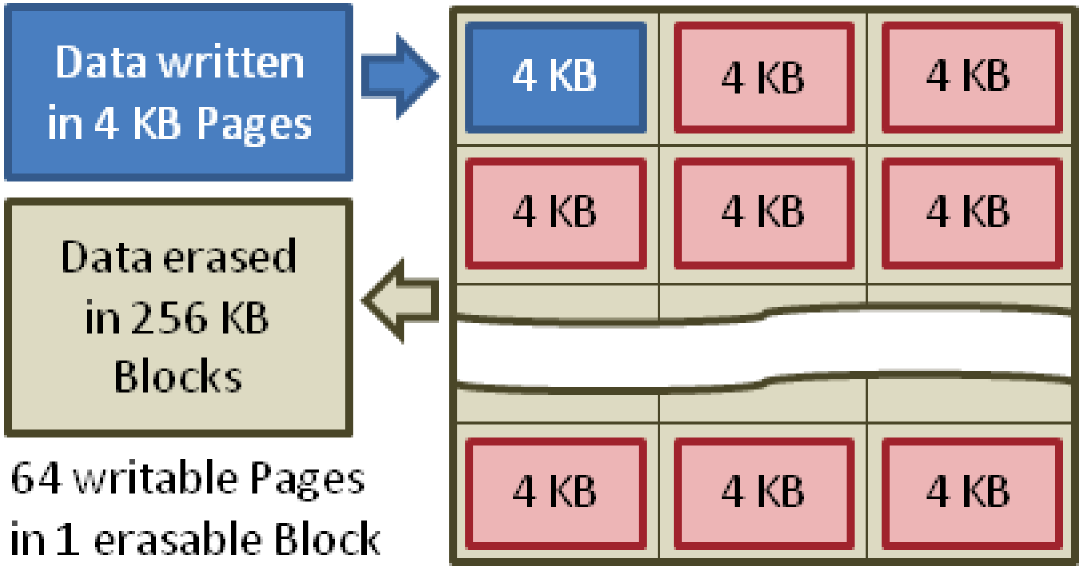


Image source: wikipedia

NAND Flash Programming Model



Typical NAND Flash Pages and Blocks

- Can read data in page level units. Fast: 10 us.
- Can program data in page level units. Fast: 10-100 us
- Can only erase entire block. Slow 1-10 msec

Reliability Characteristics

- The process of reading/writing from a cell impacts it ability to retain data
- P/E Cycles
  - High voltage, charge moves into/out of floating gate
  - Some charge gets stuck in oxide layer
  - Over time, cell gets "stuck" and cant be programmed
- Read/write disturb
  - Occurs because multiple cells are connected in series
  - Read/program voltages on a cell can cause leakage in other cells, causing their values to "flip"
  - Can result in "bitrot"

Flash Reliability

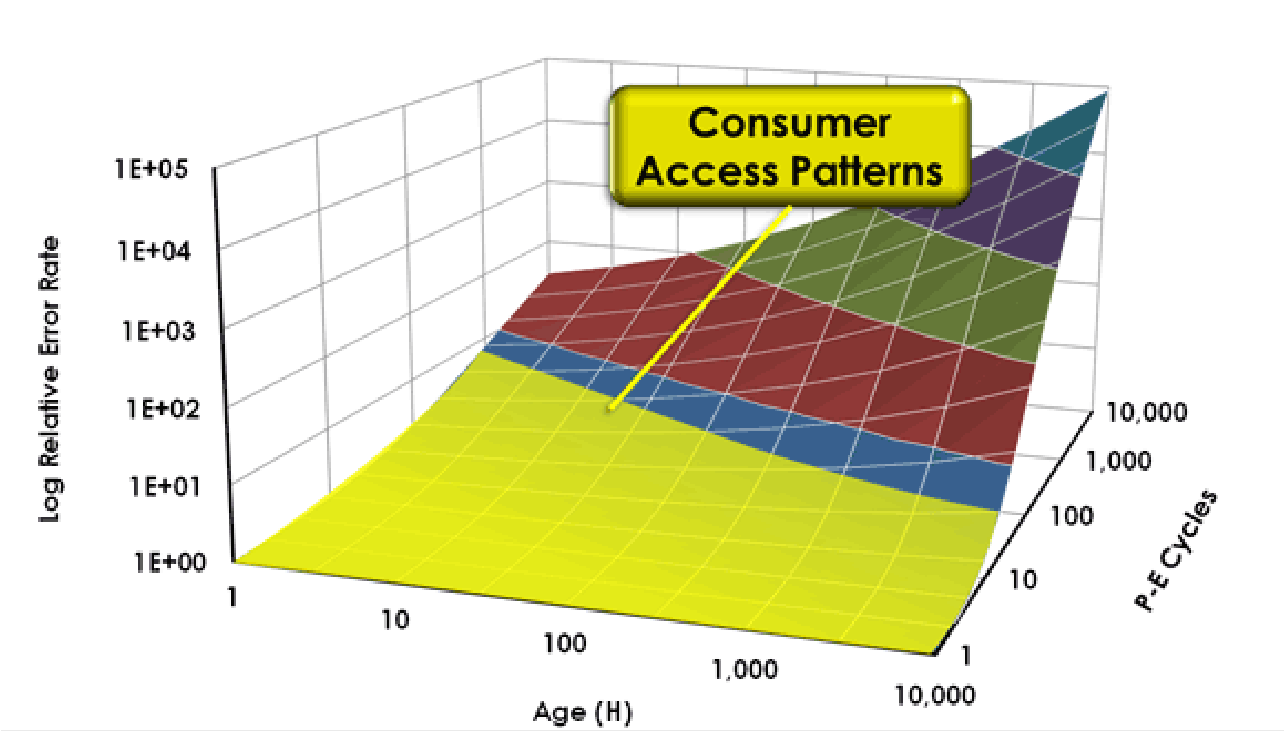


Figure is for illustrative purposes only

- BER: bit error rate
- RBER: raw bit error rate (can be reduced through error checking codes)
- UBER: uncorrected bit error rate
- P/E cycles: number of program/erase cycles a cell is subjected to
- Typical SLC 100k P/E cycles, MLC < 10k P/E cycles for HDD-like error rates

Implications of Flash Storage

- Block level erase
  - Erasing takes more time than reading/writing
  - Can only do block at a time
- Wear leveling
  - Cell reliability degrades with P/E cycles
  - Distribute P/E cycles equally between cells
- Random access
  - No concept of seeks
  - No need for scheduling

Who deals with Flash quirks?

- OS Filesystem
  - Log structured handles block level erase
  - Implement wear leveling through log cleaning
  - E.g., Linux JFFS/JFFS2, YAFFS (2002) for NAND flash, Android YAFFS2, Samsung F2FS (2012)

- On disk controller
  - Block level erase handled through FTL (flash translation layer)
  - FTL maps logical block (LBA) to physical block
  - Modify cycle allocates new phy block and changes FTL mapping
  - Garbage collection pass erases partially used blocks
  - More common for high end SSD drives
  - Normal block device interface exported to OS

TRIM

- To garbage-collect a block, must read live pages and write somewhere else
- What if the "live" pages are actually not used by FS?
  - FS creates then deletes a large file
  - Disk controller does not the blocks of the file are not used
  - Eventually, SSD controller thinks whole disk is full, and every write needs a corresponding cleaning operation
  - Excessive overhead
- TRIM command
  - OS informs SSD that a particular block not being used
  - Relatively recent (e.g., OS X supports since 2011)
  - Still fairly expensive (hundreds of msec)
  - Active debate on how OS should use

Write Amplification

- Write amplification = Data written to flash/Data written by OS
- Factors that impact write amplification
  - Garbage collection (increases WA during cleaning)
  - Over-provisioning (less cleaning, decrease WA)
  - TRIM (less cleaning, decrease WA)
  - Free user space (less cleaning, decrease WA)
  - Wear leveling (more rewrites, increase WA)
  - Separating static and dynamic data (decrease WA)
  - Sequential writes (low WA)
  - Random writes (more cleaning, more WA)

Backup Slides

New mass storage technologies

- New memory-based mass storage technologies avoid seek time and rotational delay
  - NAND Flash
  - Battery-backed DRAM (NVRAM)
- Disadvantages
  - Price: more expensive than same capacity disk
  - Reliability: more likely to lose data
- Open research question: how to effectively use flash in commercial storage systems

Basic Idea

- Use silicon devices based on MOSFETs
  - Metal Oxide Field Effect Transistor
  - Also used in DRAM
  - Each cell contains a single MOSFET with an additional "floating gate"

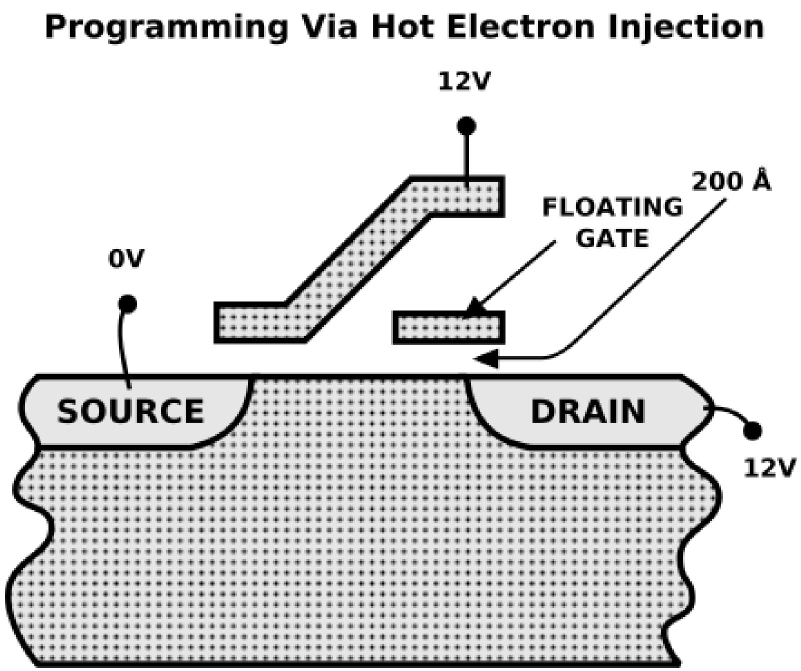
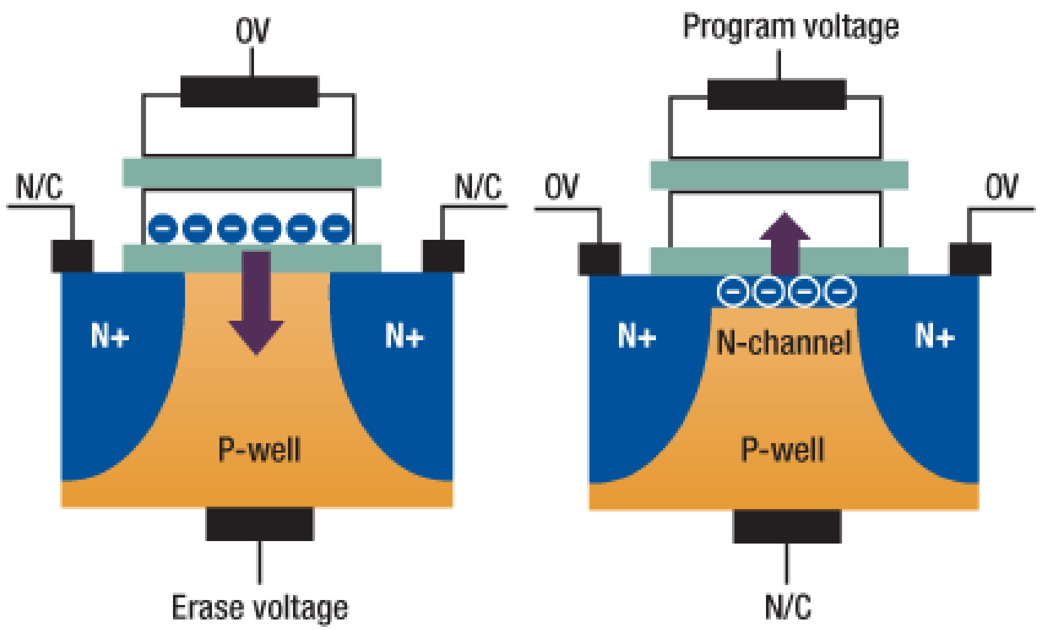


Image source: wikipedia

Programming a Flash Cell

- Two basic operations: erase (reset) and program
  - Erase clears charge on floating gate. Allows channel to conduct, setting bit to "1"
  - Program forces charge onto floating gate (via tunneling/hot electron injection), blocking the channel, and setting bit to "0"



http://www.electroiq.com/articles/sst/2011/05/solid-state-drives.html

NAND Flash

- Two basic types
  - Differ in how cells are connected and accessed
- NOR: bit level addressability, lower density, expensive
- NAND: "block" level addressability, higher density, cheap

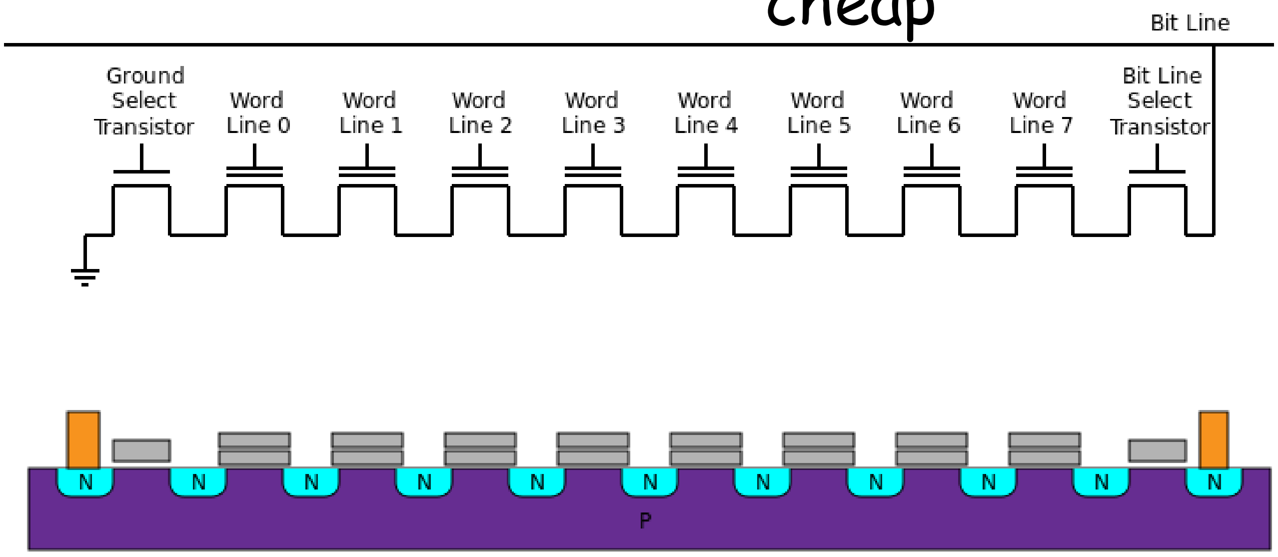
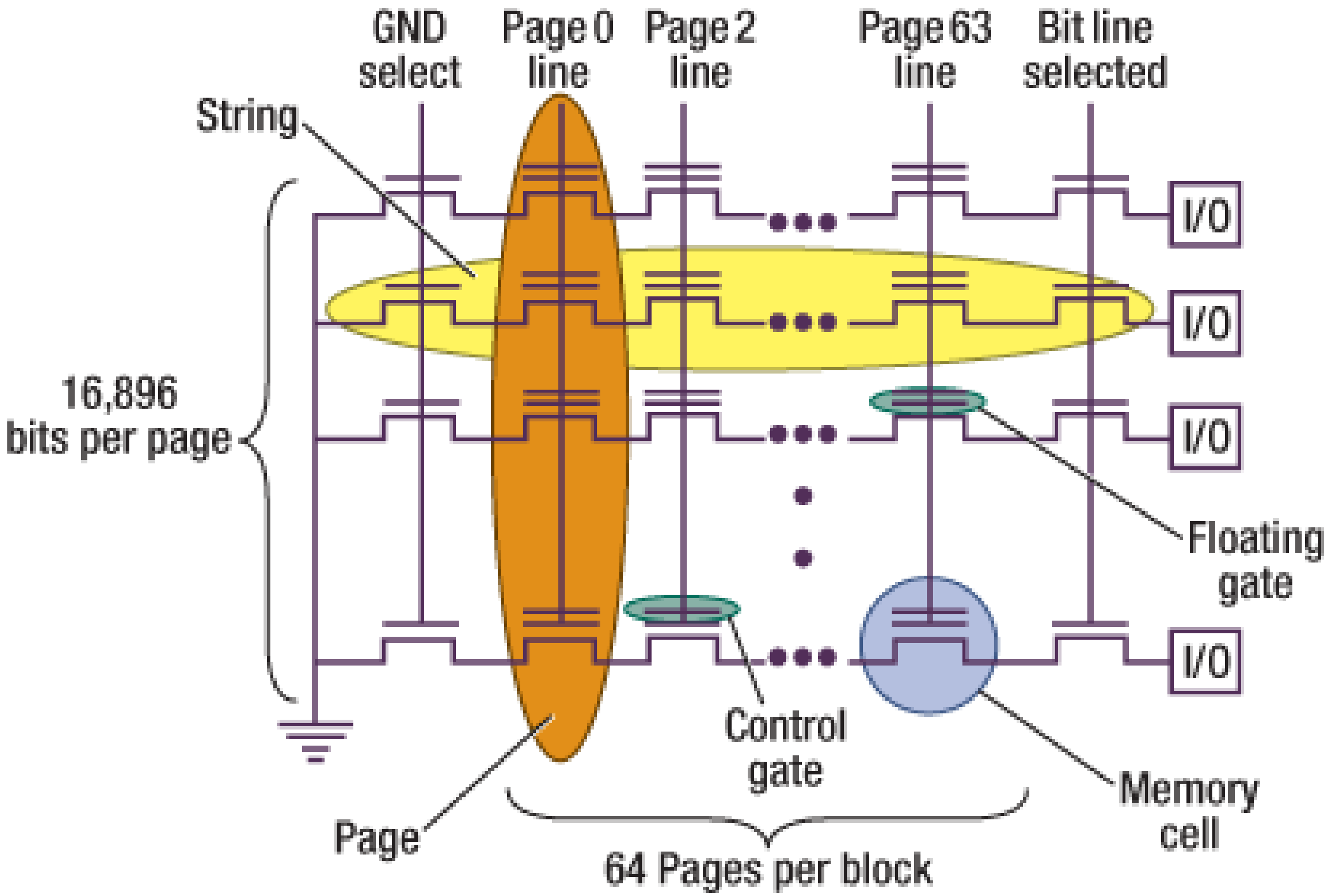


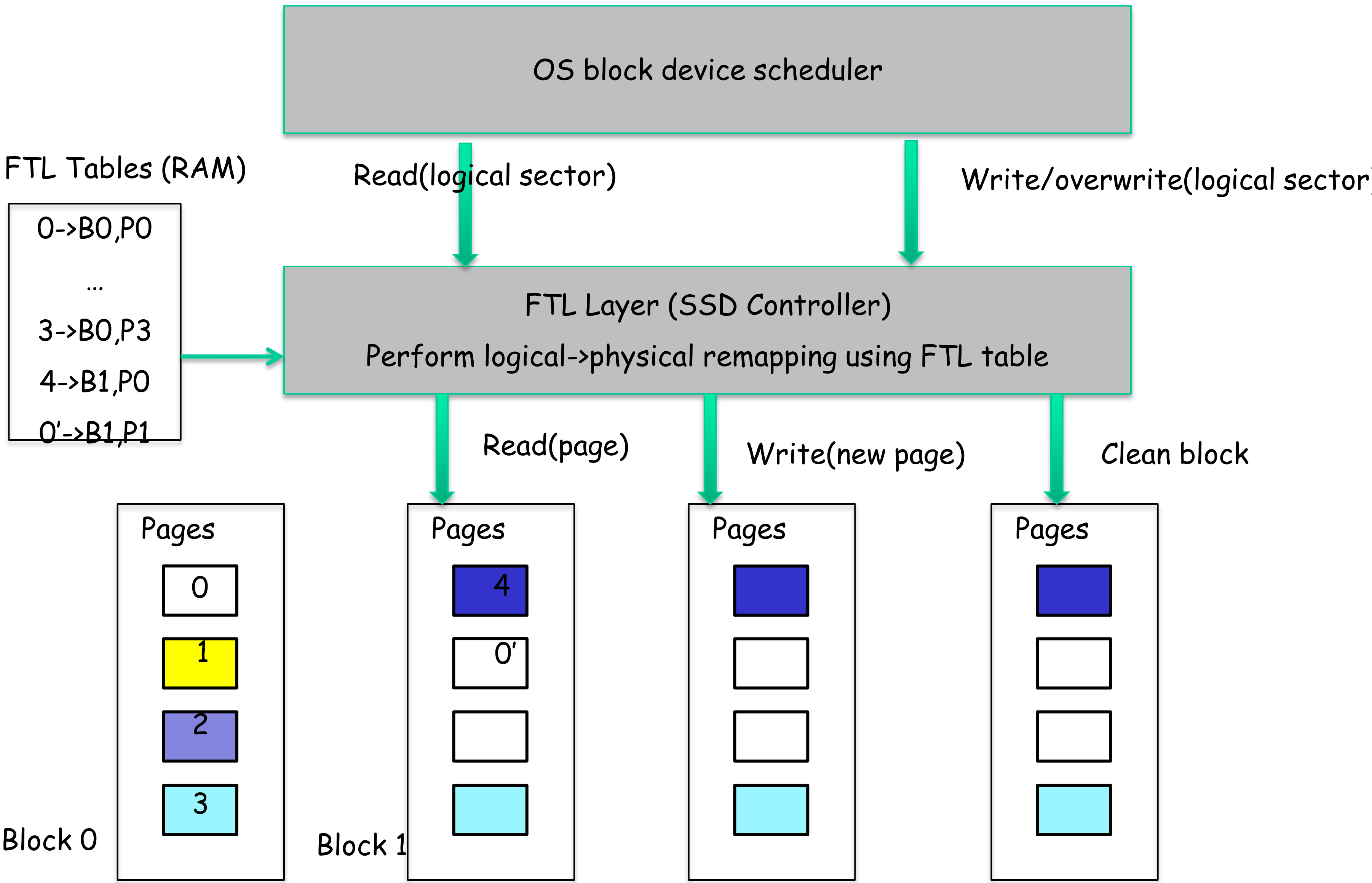
Image source: wikipedia

NAND Flash Structure



Source: http://www.electroiq.com/articles/sst/2011/05/solid-state-drives.html

FTL Layer


OS block device scheduler
Read(logical sector)
Write/overwrite(logical sector)
FTL Tables (RAM)
0->B0,P0
...
3->B0,P3
4->B1,P0
0'->B1,P1
FTL Layer (SSD Controller)
Perform logical->physical remapping using FTL table
Read(page)
Write(new page)
Clean block
Pages
0
1
2
3
Block 0
Pages
4
0'
Block 1
Pages
Pages

Wear Leveling

- No wear leveling
- Dynamic wear leveling
  - Always write to new page
  - Garbage collect old blocks (compare to LFS)
  - Infrequently changing blocks left untouched
- Static wear leveling
  - Similar to dynamic wear leveling, but
  - Also periodically move unmodified blocks
  - More overhead, but better leveling