Storage Systems

Main Points

- Survey of physical storage hardware devices
  - SRAM, DRAM, Flash, magnetic disk, tape
- File systems
  - Useful abstraction on top of physical devices
- File system usage patterns
  - Small files and large files are both commonplace

Storage Technologies

- Cost/capacity
- Word vs. block access
- Persistence
- Latency (read/write)
- Throughput
- Power drain (in use or when inactive)
- Weight/volume

Volatile Memory: SRAM

- Static RAM (SRAM)
  - Data stored in a transistor flip/flop
  - Bits degrade on poweroff
  - Access latency range: 1 – 10ns
  - Bit density inversely proportional to clock rate
  - Bit density scales with Moore's Law
  - Typical use: on chip cache, high speed access

Volatile Memory: DRAM

- Dynamic RAM (DRAM)
  - Each bit stored in a capacitor
  - 2D/3D array for dense packing
  - 50-100 ns latency for word-level access
  - Bits degrade even when powered, so must be actively refreshed
  - Power drain proportional to storage capacity
  - Bit density scales with Moore's Law
  - Typical use: off-chip volatile random access

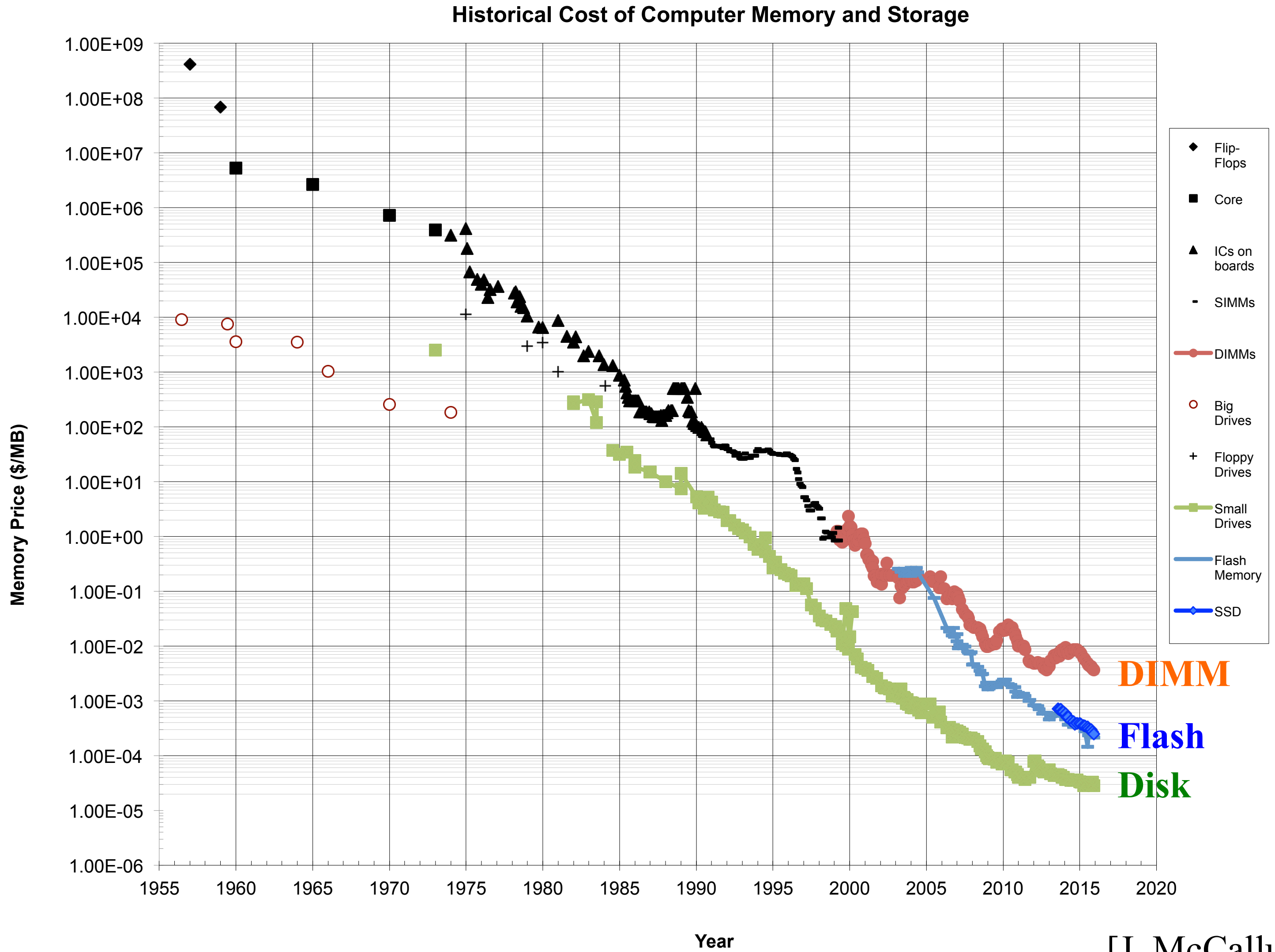
Persistent Memory: Flash

- NAND Flash/Solid State Drive (SSD)
  - Blocks of bits stored persistently in silicon
  - Densely packed in 2-D (soon 3-D) array
  - Blocks remain valid even when unpowered
  - Electrically reprogrammable, for a limited # of times
  - 100 us block level (1KB) random read access
  - Writes must be to a “clean” block, no update in place
  - Erasing only for regions of blocks ~ 256KB
  - Typical use: smartphones, laptops, cloud servers

Persistent Memory: Magnetic Storage

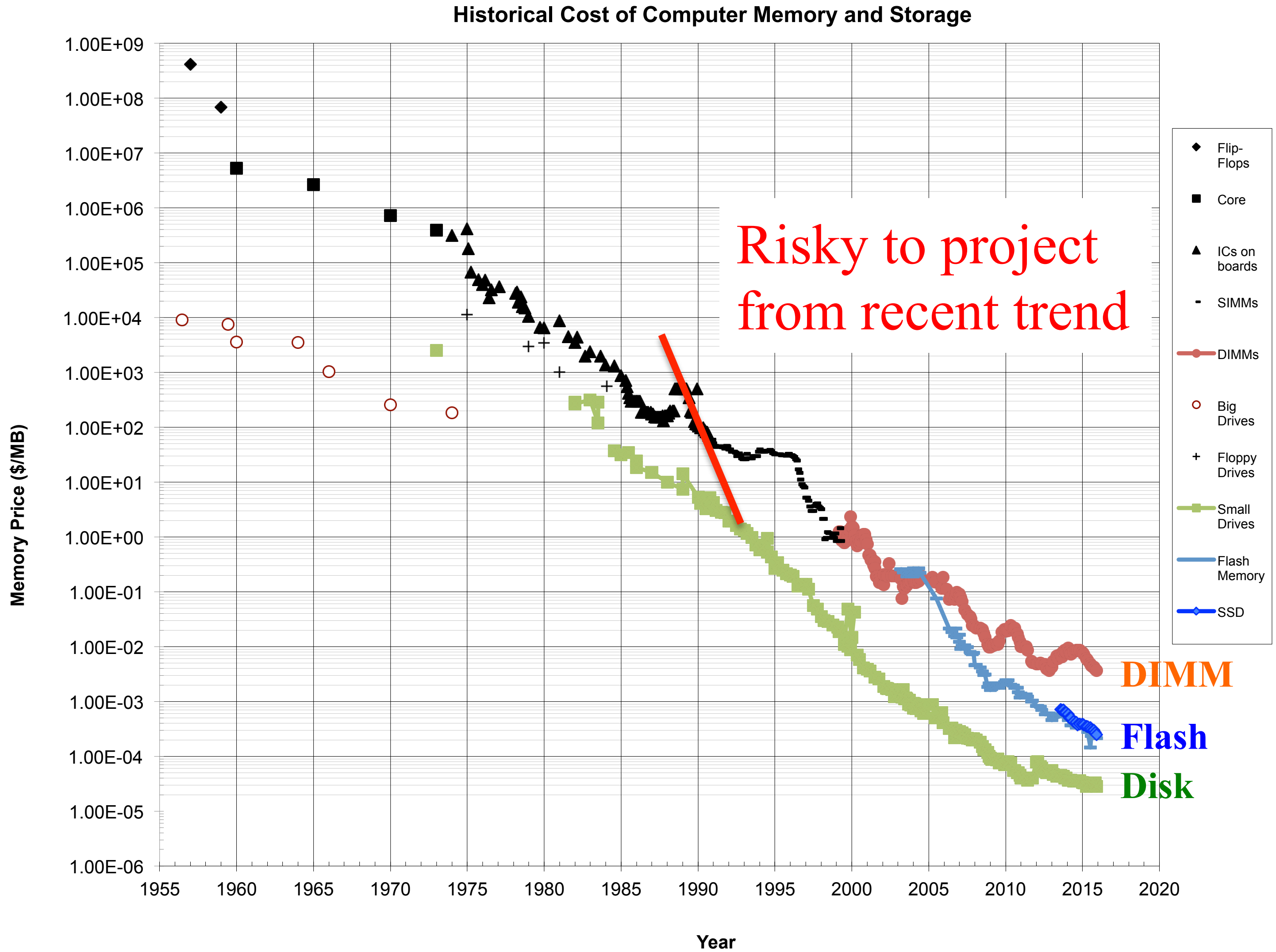
- Bits stored on magnetic surface
  - 1 Tbit per square inch
  - Physical motion needed to read bits off surface
- Magnetic disks
  - Block level random access
  - 10 ms random access latency
  - 150MB/s streaming access
  - Typical use: desktops, data center bulk storage
- Magnetic tapes: archival storage

Memory & storage historical pricing

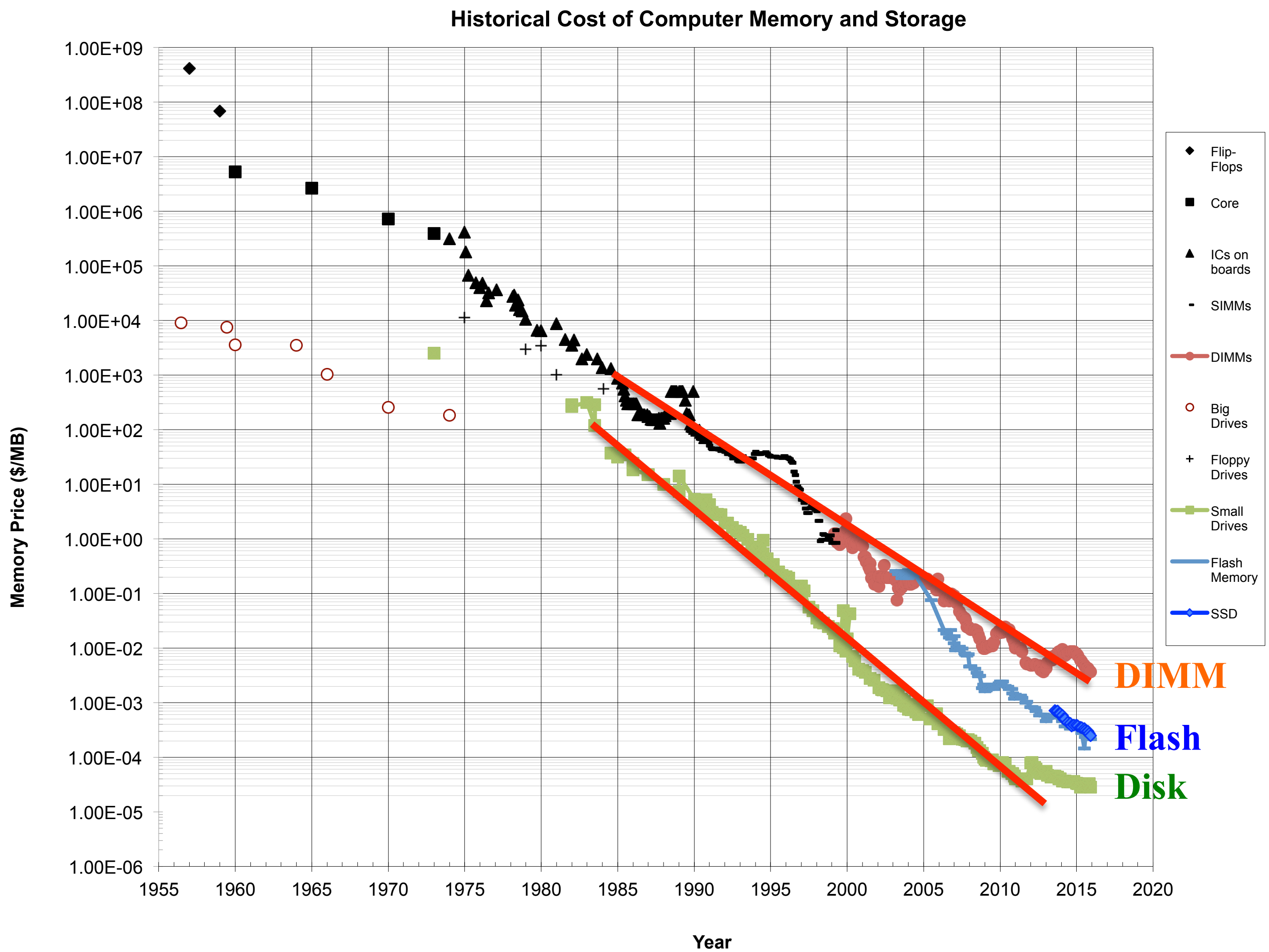


[J. McCallum, jcmit.com]

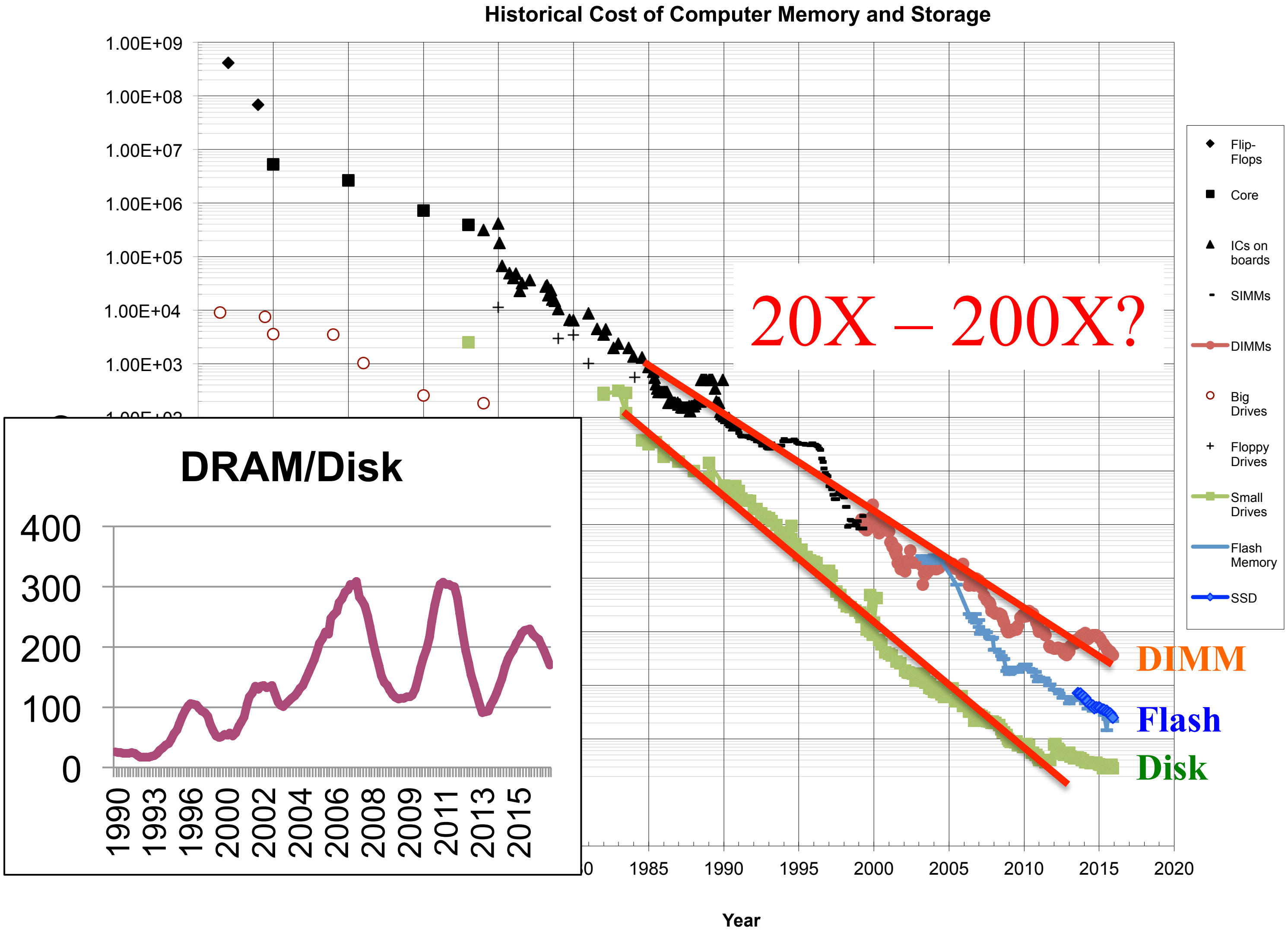
DRAM & disk pricing, 1991 angst



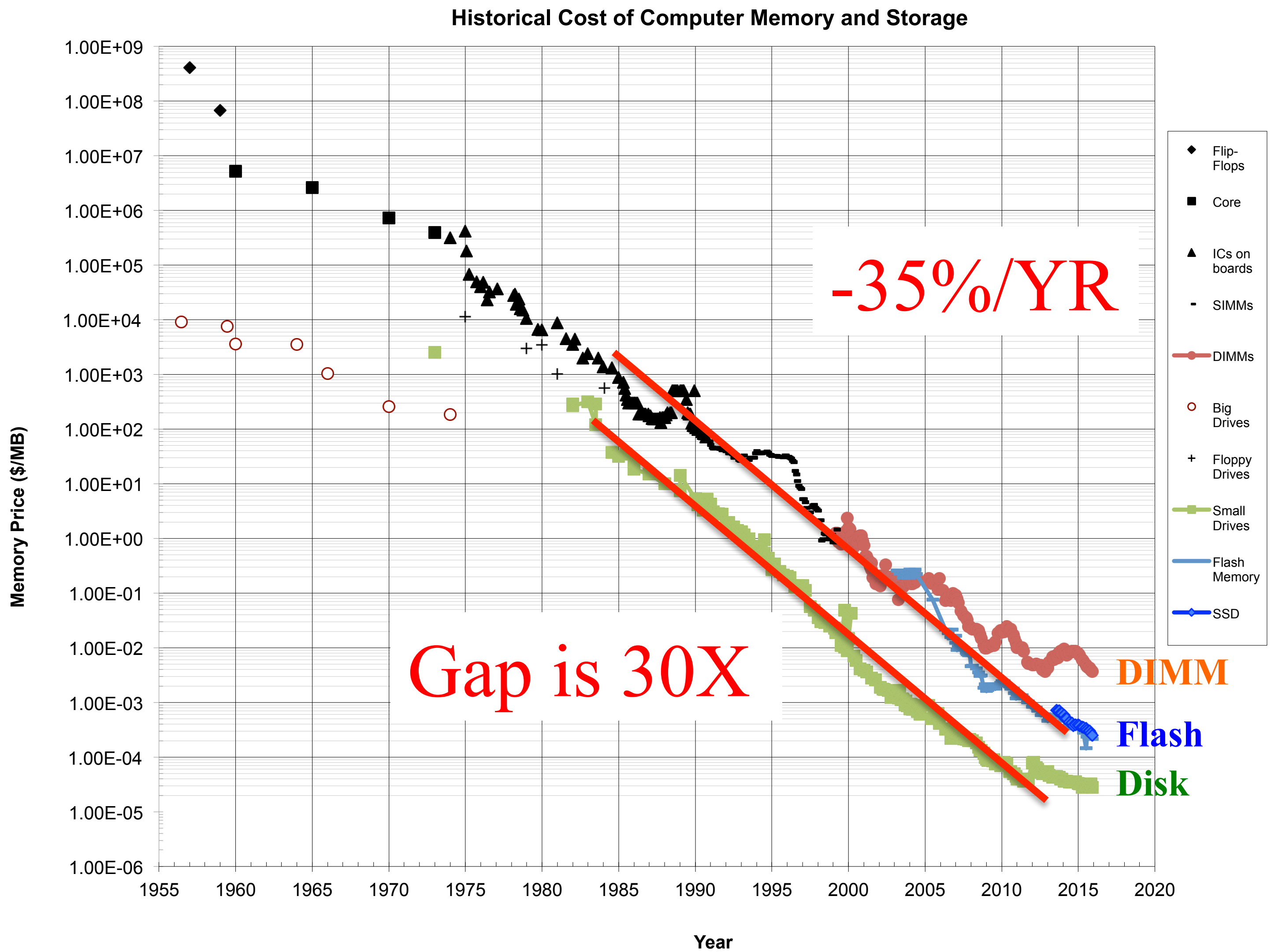
DRAM & disk pricing diverging



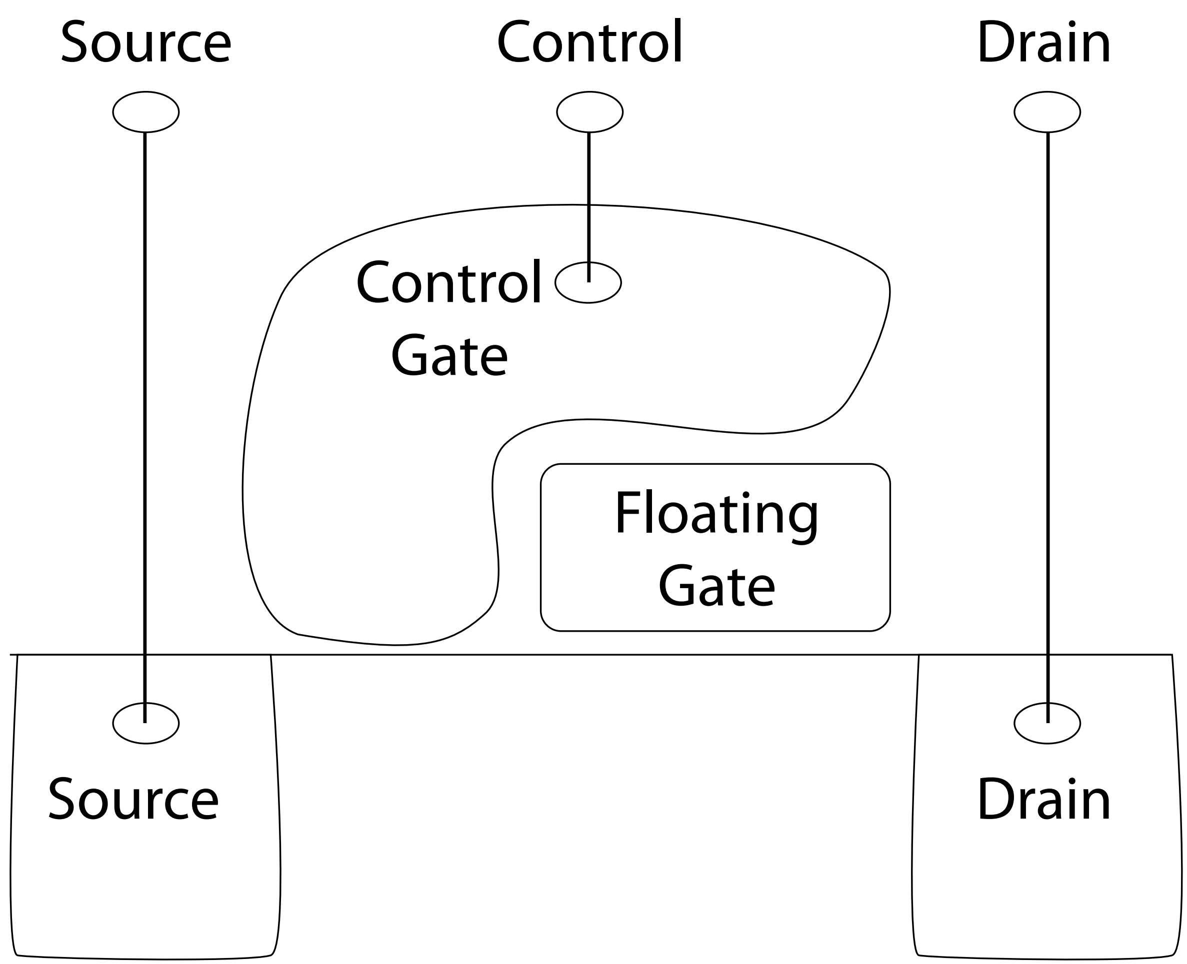
DRAM & disk pricing diverging



Best solid state & disk, Moore's Law?



Flash Memory



Flash Memory

- Basic operation: read/write to 4KB block at a time
  - Latency: 50-100 microseconds
  - Native Command Queueing (NCQ) for concurrent ops
- Blocks arranged in 2-D (soon 3-D) grid
  - Can read/write blocks in different "lanes" concurrently
- Writes must be to "clean" cells
  - Multi-block erasure required before write
  - Erasure block: 128 – 512 KB * # of lanes
  - Erasure time: 1-2 milliseconds
- Limited # of write cycles per block (~ 1000)

Intel SSD DC P3608 (2016)

Capacity	4 TB
Page Size	4 KB
Bandwidth (Sequential Reads)	5 GB/s
Bandwidth (Sequential Writes)	3 GB/s (peak)
Random 4KB Reads/sec	850 K
Random 4KB Writes/sec	50 K
Endurance	5000 erase/write cycles
Idle/Active Power	11W/20-40W
Interface	NVMe

Question

- Why are random writes so slow?
  - Random write/sec: 50K
  - Random read/sec: 850K

- Why are random writes so fast?
  - 1ms/erase => max 1000 writes/sec

Question

- Is persistence a problem?
  - What if OS writes to the same block repeatedly?
  - What if OS writes in a repeated scan?

- 1B blocks, lifetime 5000 writes/block
- 50K writes/sec (random)
- 750K writes/sec (sequential, peak)

Flash Translation Layer (FTL)

- Map logical block # to physical block #
  - Transparent to operating system
  - Translation stored in flash (along with each block)
  - Translation cached in SRAM/DRAM
- On write, put new block anywhere clean
- On read, look up translation to find most recent written location

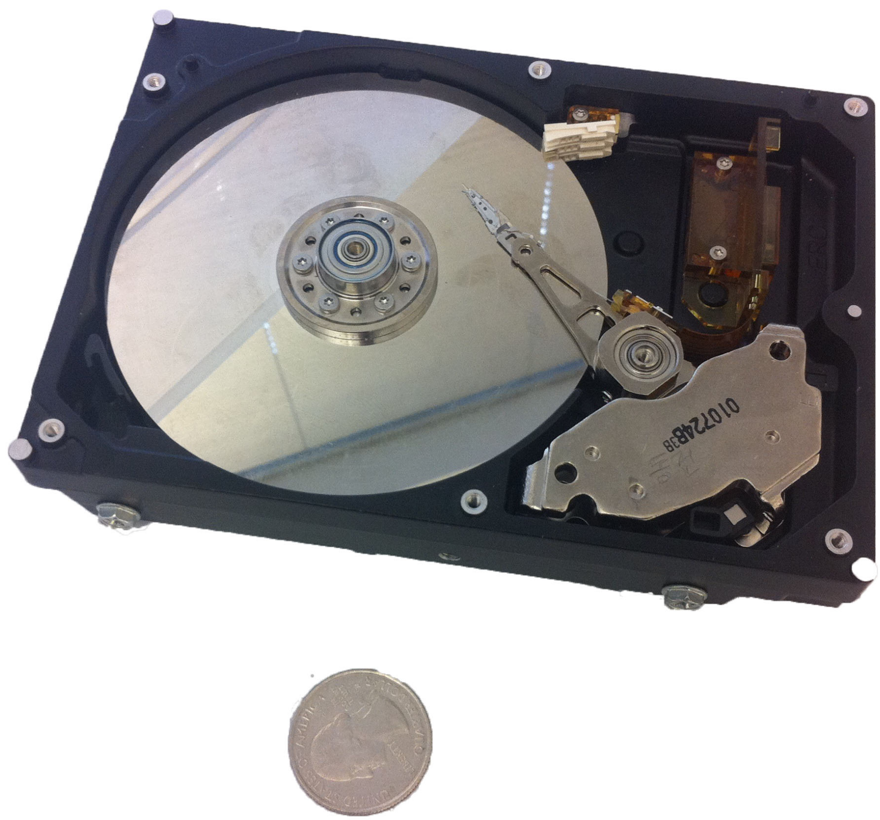
FTL Garbage Collection

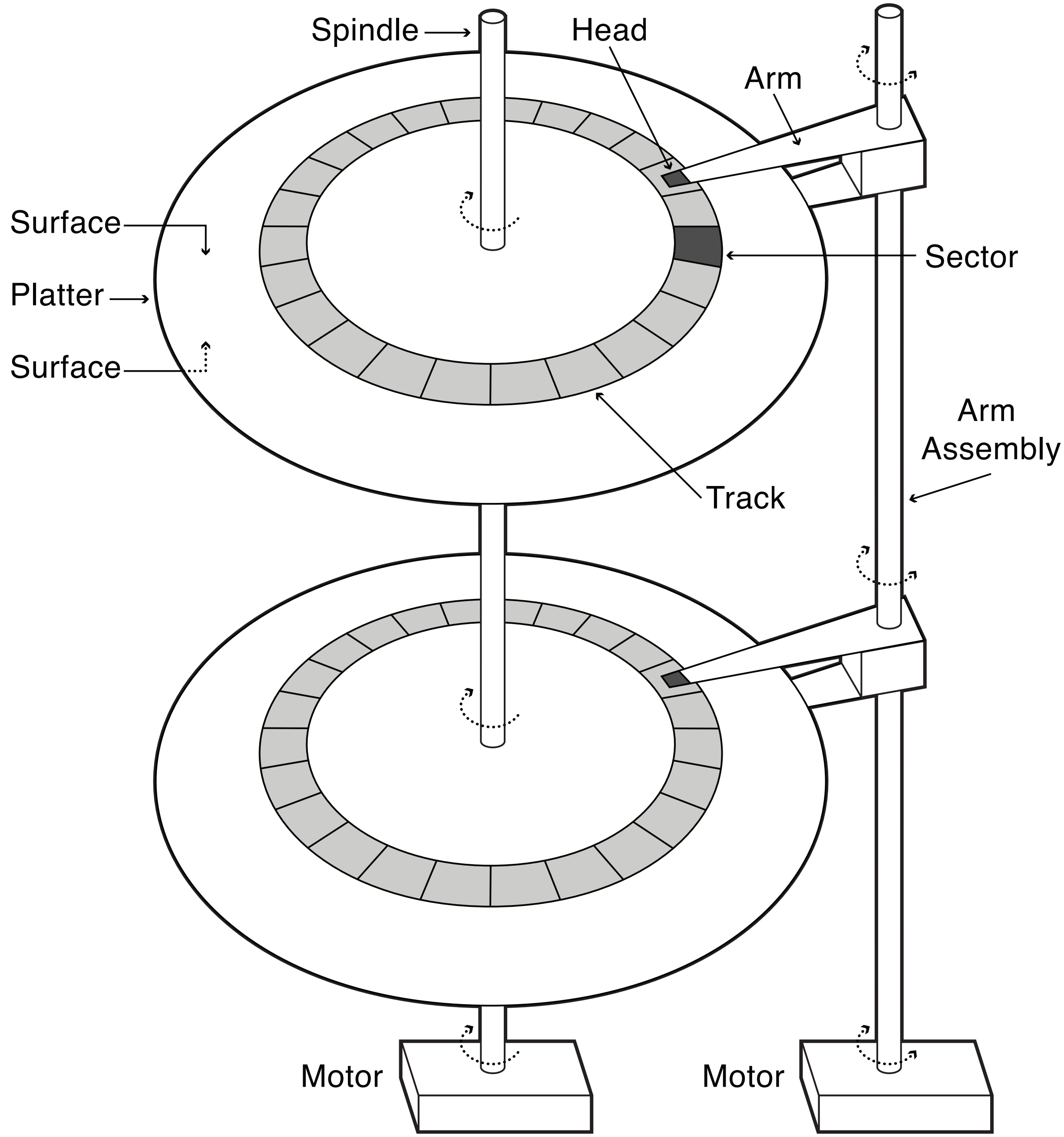
- Keep regions of recently erased blocks
- # of physical blocks > # of logical blocks (20-30% extra)
- Every block write creates an empty spot
  - OS can also declare blocks dead (TRIM command)
- Empty spot must be erased before reused
  - Erasure only of multi-block regions (can be multi-MB)
- Empty region by copying live pages to clean region
  - More efficient if blocks stored together are deleted together

Wear Levelling

- Each block can only be written a maximum number of times
  - FTL tracks # of erase/write cycles for each block
  - Unmaps blocks that have worn out
- Preferentially write new blocks into regions with fewer update cycles

Magnetic Disk




Spindle
Head
Arm
Surface
Platter
Surface
Sector
Arm
Assembly
Track
Motor
Motor

Disk Tracks

- ~ 1 micron wide
  - Wavelength of light is ~ 0.5 micron
  - Resolution of human eye: 50 microns
  - 100K tracks on a typical 2.5" disk
- Separated by unused guard regions
  - Reduces likelihood neighboring tracks are corrupted during writes (still a small non-zero chance)
- Track length varies across disk
  - Outside: More sectors per track, higher bandwidth
  - Disk is organized into regions of tracks with same # of sectors/track
  - Only outer half of radius is used
    - Most of the disk area in the outer regions of the disk

Sectors

Sectors contain sophisticated error correcting codes

- Disk head magnet has a field wider than track
- Hide corruptions due to neighboring track writes

- Sector sparing
  - Remap bad sectors transparently to spare sectors on the same surface
- Slip sparing
  - Remap all sectors (when there is a bad sector) to preserve sequential behavior
- Track skewing
  - Sector numbers offset from one track to the next, to allow for disk head movement for sequential ops

Disk Performance

Disk Latency =

- Seek Time + Rotation Time + Transfer Time
- Seek Time: time to move disk arm over track (1-20ms)
  - Fine-grained position adjustment necessary for head to "settle"
  - Head switch time ~ track switch time (on modern disks)
- Rotation Time: time to wait for disk to rotate under disk head
  - Disk rotation: 4 – 15ms (depending on price of disk)
  - On average, only need to wait half a rotation
- Transfer Time: time to transfer data onto/off of disk
  - Disk head transfer rate: 50-100MB/s (5-10 usec/sector)
  - Host transfer rate dependent on I/O connector (USB, SATA, …)

Toshiba Disk (2008)

Size	
Platters/Heads	2/4
Capacity	320 GB
Performance	
Spindle speed	7200 RPM
Average seek time read/write	10.5 ms/ 12.0 ms
Maximum seek time	19 ms
Track-to-track seek time	1 ms
Transfer rate (surface to buffer)	54–128 MB/s
Transfer rate (buffer to host)	375 MB/s
Buffer memory	16 MB
Power	
Typical	16.35 W
Idle	11.68 W

HGST Ultrastar He10 (2016)

Capacity	10 TB, 7 platters
Spin Speed	7200 RPM
Sustained Transfer Rate	249 MB/s (read), 225 MB/s (write)
Interface Transfer Rate	1200 MB/s
Seek time (avg)	8 ms (read), 8.6 ms (write)
Rotational latency (avg)	4.16 ms
Cache	256 MB
Idle/Operating Power	6W/9.5W
Bit Error Rate (read)	10^-15

Question

- How long to complete 100 random 4KB disk reads, in FIFO order?

Question

- How long to complete 100 random 4KB disk reads, in FIFO order?
  – Seek: average 8 msec
  – Rotation: average 4.16 msec
  – Transfer: 4KB / 249 MB/s = 16 usec
- 100 * (8 + 4.16 + 0.016) = 1.2 seconds

Question

- How long to complete 100 sequential 4KB disk reads?

Question

- How long to complete 100 sequential 4KB disk reads?
  - Seek Time: 8 ms (to reach first sector)
  - Rotation Time: 4.16 ms (to reach first sector)
  - Transfer Time: 400KB / 249MB/sec = 1.6 ms

Total: 8 + 4.16 + 1.6 = 13.8 ms

  - Might need an extra head or track switch (+1ms)
  - Track buffer may allow some sectors to be read out of order (-2ms)

Question

- How large a transfer is needed to achieve 80% of the max disk transfer rate?

Question

- How large a transfer is needed to achieve 80% of the max disk transfer rate?

  Assume 12.16 ms to reach first sector

  Assume x rotations are needed, 8.5ms/rotation

  Then solve for x:

  $0.8\,(12.16\text{ms} + 8.5\text{ms}\ x) = 8.5\text{ms}\ \ x$

Total: x = 5.7 rotations, 12.1 MB

Disk Scheduling

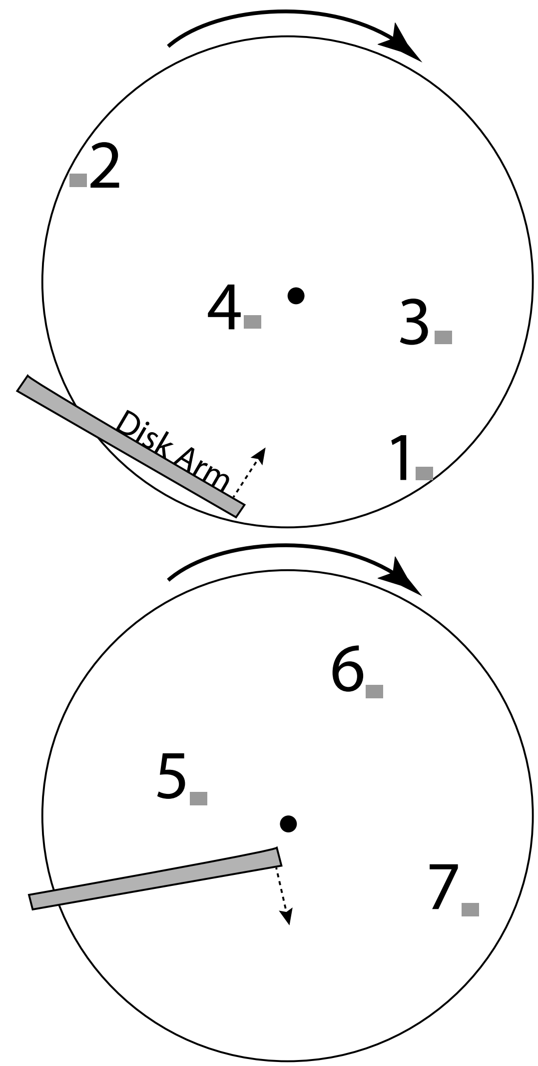
- FIFO
  - Schedule disk operations in order they arrive
  - Downsides?

Disk Scheduling

- Shortest seek time first
  - Not optimal!
    - Suppose cluster of requests at far end of disk
  - Downsides?

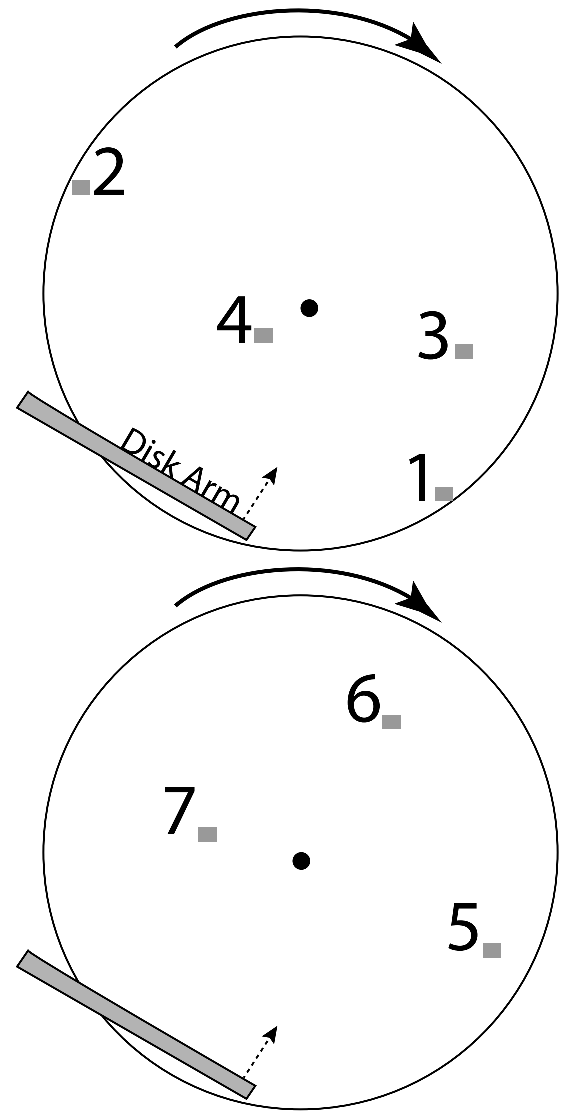
Disk Scheduling

- SCAN: move disk arm in one direction, until all requests satisfied, then reverse direction
- Also called “elevator scheduling”



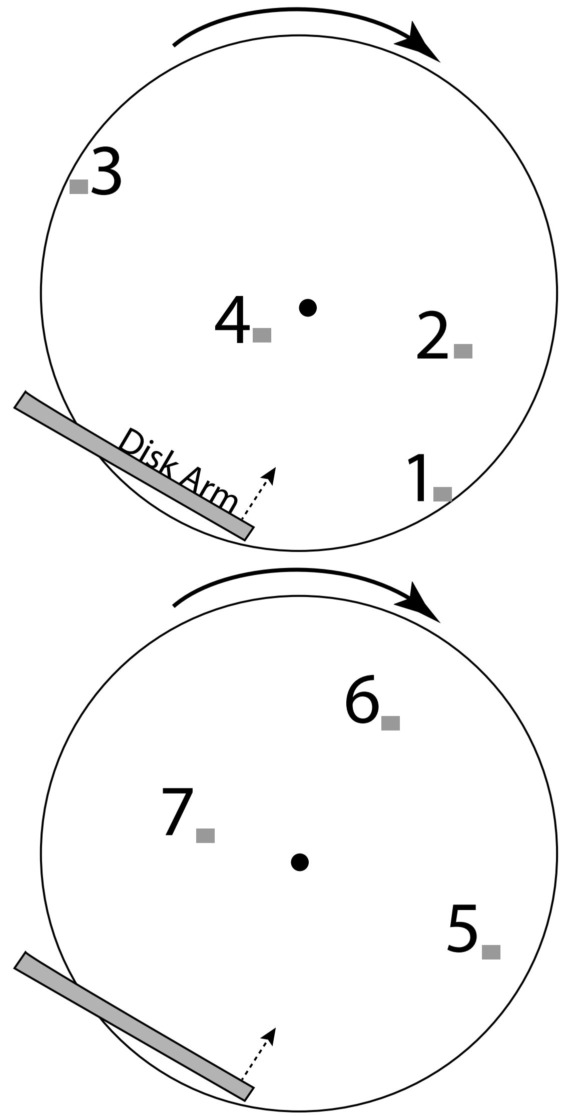
Disk Scheduling

- CSCAN: move disk arm in one direction, until all requests satisfied, then start again from farthest request



Disk Scheduling

- R-CSCAN: CSCAN but take into account that short track switch is < rotational delay



Question

- How long to complete 100 random disk reads, in any order?

Question

- How long to complete 100 random disk reads, in any order?
  - Disk seek: 1ms (most will be short)
  - Rotation: 4.16ms
  - Transfer: 16usec
- Total: 100 * (1 + 4.16 + 0.016) = 0.52 seconds
  - Would be a bit shorter with R-CSCAN
  - vs. 1.2 seconds if FIFO order

Question

- How long to read all of the bytes off of a disk?

Question

- How long to read all of the bytes off of a disk?
  - Disk capacity: 10TB
  - Disk bandwidth: 249MB/s (average)
- Transfer time = 40K seconds (12 hours)

Question

- If you read all the data off the disk, how likely will some of the data be corrupted?

Question

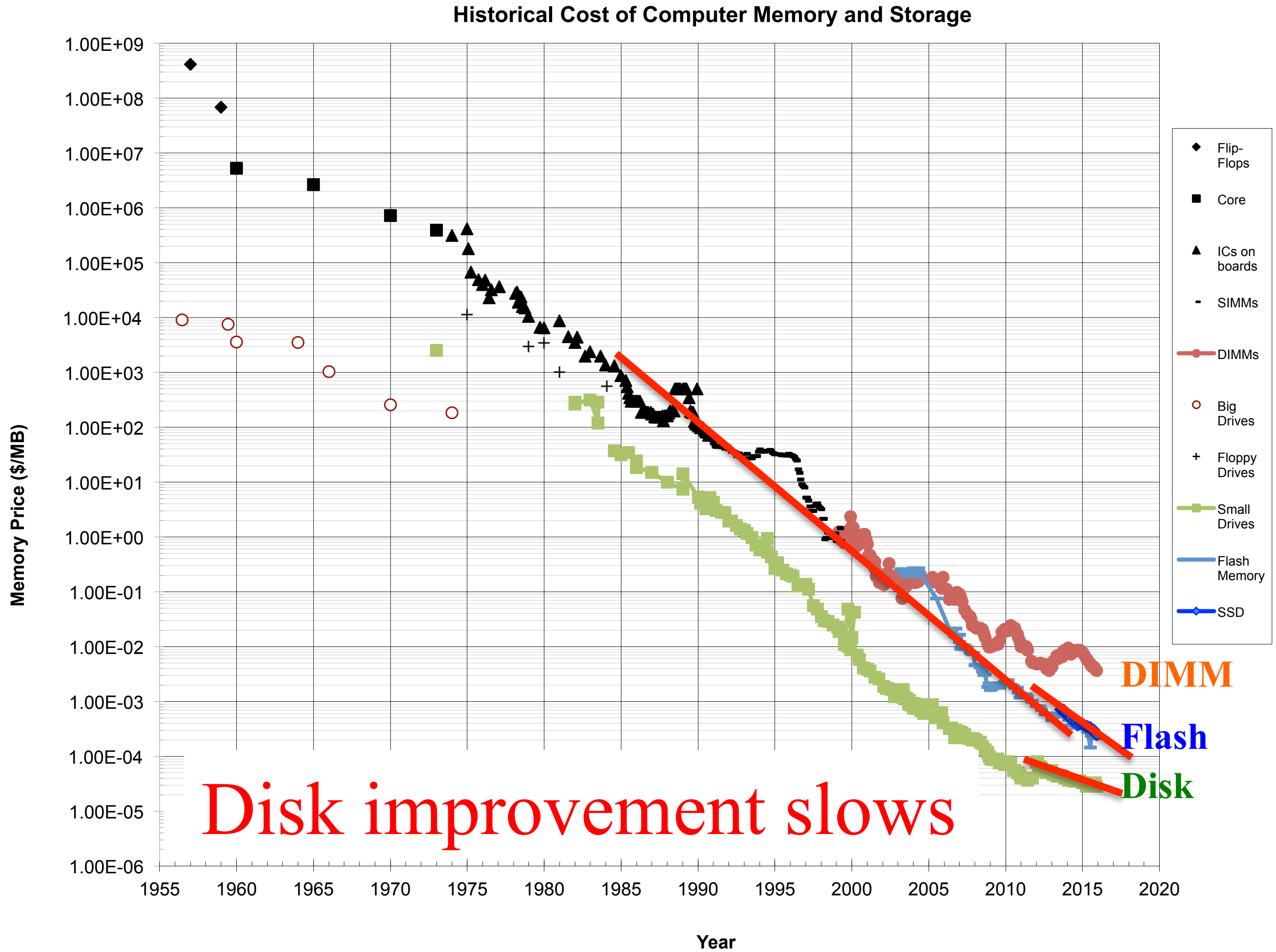
- If you read all the data off the disk, how likely will some of the data be corrupted?

Bit error rate = 10^-15

Bits per disk = 10TB = 10^14

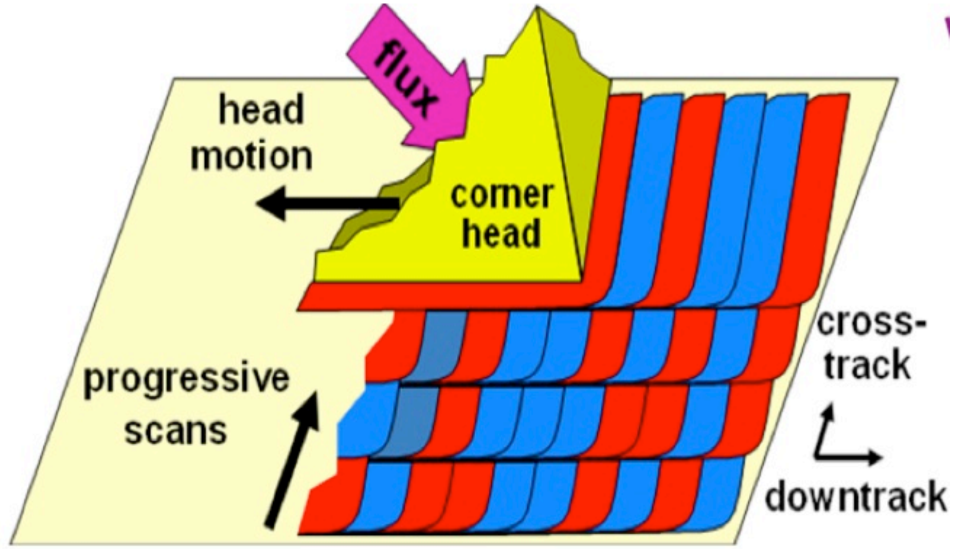
=> 10% !!

Flash SSD & disk pricing, recently



Shingled magnetic recording (SMR)

- Uses ~current tech
- Overlap adjacent tracks (no gap)
- More tracks/inch
- No sector overwrite



Wood, Trans. Magnetics., 2009

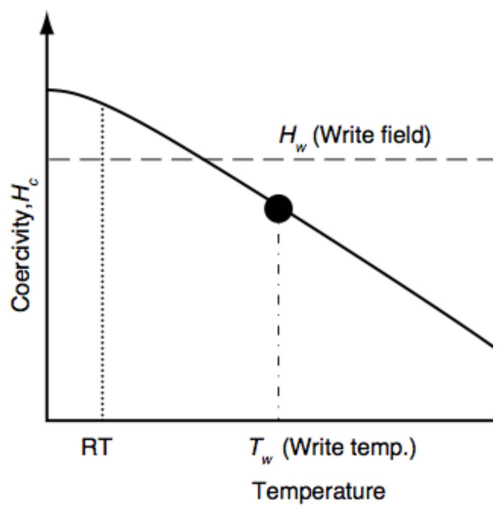
- Two-dimensional magnetic recording (TDMR)
  - Inter-track interference ever worse, data dependent
  - Give up on flying head path staying “in track”
  - Include 2 (then 3) read sensors per head
    - Read multiple “sub-tracks”, signal process to data

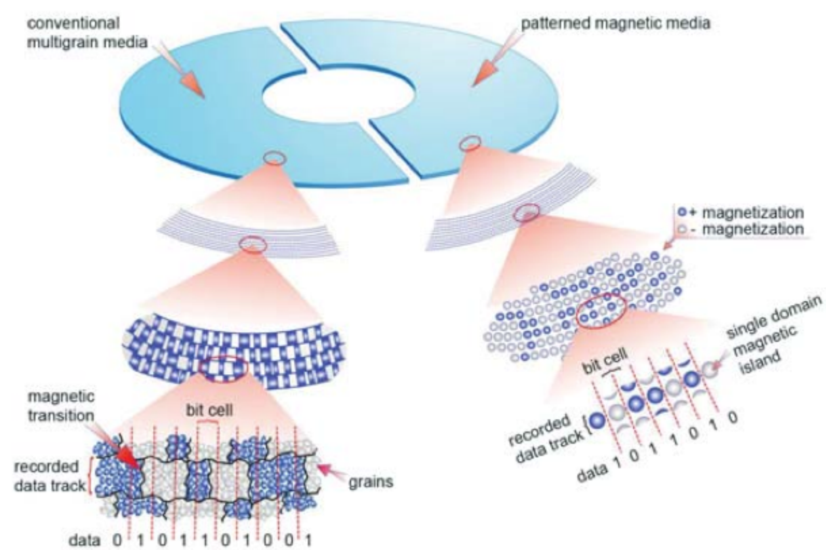
SMR today/TDMR soon

- Hidden behind “Shingle Translation Layer (STL)”
  - Embedded layer that re-writes entire region
  - New blocks go to empty spill region
  - Re-write/coalesce existing regions when mostly empty
- Adding 10% - 30% areal density (not 2X soon)
- Interesting parallel/convergence
  - FTL sequentially writes flash pages in erase block
  - Flash erase block analogous to shingled band

More Changes In Store for Disks

- Heat-Assisted (HAMR)
  - Small bits need high coercivity media to retain orientation
  - High coercivity media is not changed by normal writing
  - Heated media lowers coercivity
  - Include lasers on Rd/Wr head?
- Bit-Patterned (BPM)
  - Small bits retain orientation more easily if bits kept apart
  - Pattern media so only write a single dot per bit
  - Tera-dots per sq. inch?





Still, not looking good for disk

- Driven from margin-rich enterprise apps
- Driven from volume rich mobile
- Big changes in fabrication & materials
- Small number of companies playing
  - Natural disasters can change everything

- How much will cloud storage growth pay?

- Watch for HAMR roll out in next few years

Non-flash solid state

- 3D Xpoint, PCM, Memristor, ReRAM
  - Non-volatile is about lower operating power (TCO)
  - Chasing DRAM market share
    - Pressure on SSD market likely to be incidental
  - A layer behind DIMMs (or Hybrid Memory Cube)
    - Or a program managed second memory type
- Orders of magnitude better endurance
  - But latency benefiting as much or more
  - Direct access w/o wear leveling expires cell in mins
- For big data, these are memory, not storage

File System as Illusionist: Hide Limitations of Physical Storage

- Persistence of data stored in file system:
  - Even if crash happens during an update
  - Even if disk block becomes corrupted
  - Even if flash memory wears out
- Naming:
  - Named data instead of disk block numbers
  - Directories instead of flat storage
  - Byte addressable data even though devices are block-oriented
- Performance:
  - Cached data
  - Data placement and data structure organization
- Controlled access to shared data

File System Abstraction

- File system
  - Persistent, named data
  - Hierarchical organization (directories, subdirectories)
  - Access control on data
- File: named collection of data
  - Linear sequence of bytes (or a set of sequences)
  - Read/write or memory mapped
- Crash and storage error tolerance
  - Operating system crashes (and disk errors) leave file system in a valid state
- Performance
  - Achieve close to the hardware limit in the average case

File System Workload

- File sizes
  - Are most files small or large?
  - Which accounts for more total storage: small or large files?

File System Workload

- File sizes
  - Are most files small or large?
    - SMALL
  - Which accounts for more total storage: small or large files?
    - LARGE

File System Workload

- File access
  - Are most accesses to small or large files?
  - Which accounts for more total I/O bytes: small or large files?

File System Workload

- File access
  - Are most accesses to small or large files?
    - SMALL
  - Which accounts for more total I/O bytes: small or large files?
    - LARGE

File System Workload

- How are files used?
  - Most files are read/written sequentially
  - Some files are read/written randomly
    - Ex: database files, swap files
  - Some files have a pre-defined size at creation
  - Some files start small and grow over time
    - Ex: program stdout, system logs

File System Design

- For small files:
  - Small blocks for storage efficiency
  - Concurrent ops more efficient than sequential
  - Files used together should be stored together
- For large files:
  - Storage efficient (large blocks)
  - Contiguous allocation for sequential access
  - Efficient lookup for random access
- May not know at file creation
  - Whether file will become small or large
  - Whether file is persistent or temporary
  - Whether file will be used sequentially or randomly

File System Abstraction

- Directory
  - Group of named files or subdirectories
  - Mapping from file name to file metadata location
- Path
  - String that uniquely identifies file or directory
  - Ex: /cse/www/education/courses/cse451/12au
- Links
  - Hard link: link from name to metadata location
  - Soft link: link from name to alternate name
- Mount
  - Mapping from name in one file system to root of another

UNIX File System API

- create, link, unlink, createdir, rmdir
  - Create file, link to file, remove link
  - Create directory, remove directory
- open, close, read, write, seek
  - Open/close a file for reading/writing
  - Seek resets current position
- fsync
  - File modifications can be cached
  - fsync forces modifications to disk (like a memory barrier)

File System Interface

- UNIX file open is a Swiss Army knife:
  - Open the file, return file descriptor
  - Options:
    - if file doesn't exist, return an error
    - If file doesn't exist, create file and open it
    - If file does exist, return an error
    - If file does exist, open file
    - If file exists but isn't empty, nix it then open
    - If file exists but isn't empty, return an error
    - …

Interface Design Question

- Why not separate syscalls for open/create/exists?
  - Would be more modular!

```
if (!exists(name))
    create(name);   // can create fail?
fd = open(name);   // does the file exist?
```