CS162
Operating Systems and Systems Programming
Lecture 17

Performance
Storage Devices, Queueing Theory

March 30th, 2016

Prof. Anthony D. Joseph
http://cs162.eecs.Berkeley.edu

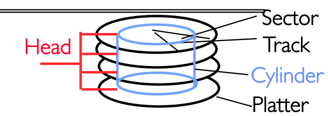
Review: Basic Performance Concepts

- *Response Time or Latency*: Time to perform an operation(s)
- *Bandwidth or Throughput*: Rate at which operations are performed (op/s)
  - Files: mB/s, Networks: mb/s, Arithmetic: GFLOP/s
- *Start up* or "Overhead": time to initiate an operation
- Most I/O operations are roughly linear
  - Latency(n) = Overhead + n/Bandwidth

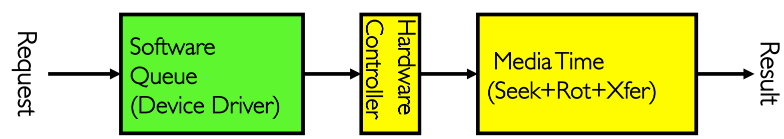
Review: Storage Devices

- Magnetic disks
  - Storage that rarely becomes corrupted
  - Large capacity at low cost
  - Block level random access (except for SMR – later!)
  - Slow performance for random access
  - Better performance for streaming access
- Flash memory
  - Storage that rarely becomes corrupted
  - Capacity at intermediate cost (50x disk ???)
  - Block level random access
  - Good performance for reads; worse for random writes
  - Erasure requirement in large blocks
  - Wear patterns issue

Review: Magnetic Disk Characteristic



- Cylinder: all the tracks under the head at a given point on all surfaces
- Read/write: three-stage process:
  - Seek time: position head/arm over proper track (into proper cylinder)
  - Rotational latency: wait for desired sector to rotate under R/W head
  - Transfer time: transfer a block of bits (sector) under the R/W head
- Disk Latency = Queuing Time + Controller time + Seek Time + Rotation Time + Xfer Time



- Highest Bandwidth:
  - Transfer large group of blocks sequentially from one track

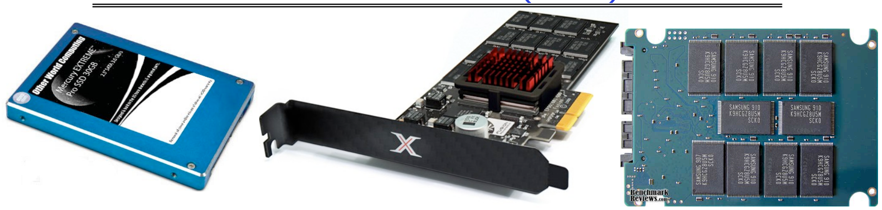
Review: Disk Performance Example

- Assumptions:
  - Ignoring queuing and controller times for now
  - Avg seek time of 5ms,
  - 7200RPM ⇒ Time for rotation: 60000(ms/M)/7200(rev/M) ~= 8ms
  - Transfer rate of 4MByte/s, sector size of 1 Kbyte ⇒ 1024 bytes/4×10⁶ (bytes/s) = 256 × 10⁻⁶ sec ≅ .26 ms
- Read sector from random place on disk:
  - Seek (5ms) + Rot. Delay (4ms) + Transfer (0.26ms)
  - *Approx* 10ms to fetch/put data: 100 KByte/sec
- Read sector from random place in same cylinder:
  - Rot. Delay (4ms) + Transfer (0.26ms)
  - *Approx* 5ms to fetch/put data: 200 KByte/sec
- Read next sector on same track:
  - Transfer (0.26ms): 4 MByte/sec
- Key to using disk effectively (especially for file systems) is to *minimize seek and rotational delays*

Goals for Today

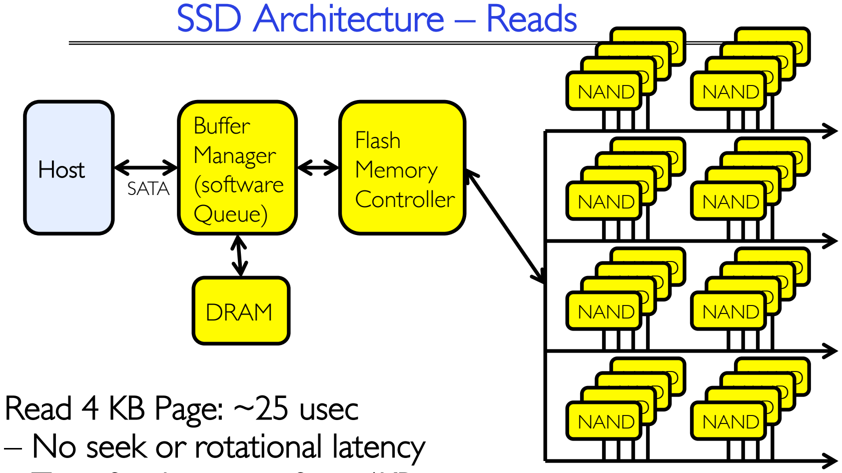
- Solid State Disks
- Discussion of performance
  - Queuing Theory
  - Hard Disk Drive Scheduling
- Start on filesystems

Solid State Disks (SSDs)



- 1995 – Replace rotating magnetic media with non-volatile memory (battery backed DRAM)
- 2009 – Use NAND Multi-Level Cell (2 or 3-bit/cell) flash memory
  - Sector (4 KB page) addressable, but stores 4-64 "pages" per memory block
  - Trapped electrons distinguish between 1 and 0
- No moving parts (no rotate/seek motors)
  - Eliminates seek and rotational delay (0.1-0.2ms access time)
  - Very low power and lightweight
  - Limited "write cycles"
- Rapid advances in capacity and cost ever since!

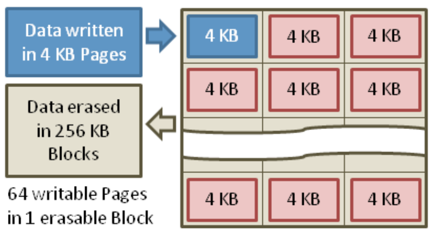
SSD Architecture – Reads



Read 4 KB Page: ~25 usec
- No seek or rotational latency
- Transfer time: transfer a 4KB page
  - SATA: 300-600MB/s => ~4 x10³ b / 400 x 10⁶ bps => 10 us
- Latency = Queuing Time + Controller time + Xfer Time
- Highest Bandwidth: Sequential OR Random reads

SSD Architecture – Writes

- Writing data is complex! (~200μs – 1.7ms)
  - Can only write empty pages in a block
  - Erasing a block takes ~1.5ms
  - Controller maintains pool of empty blocks by coalescing used pages (read, erase, write), also reserves some % of capacity
- Rule of thumb: writes 10x reads, erasure 10x writes



Typical NAND Flash Pages and Blocks

https://en.wikipedia.org/wiki/Solid-state_drive

Amusing calculation: is a full Kindle heavier than an empty one?

- Actually, "Yes", but not by much
- Flash works by trapping electrons:
  - So, erased state lower energy than written state
- Assuming that:
  - Kindle has 4GB flash
  - ½ of all bits in full Kindle are in high-energy state
  - High-energy state about 10^{-15} joules higher
  - Then: Full Kindle is 1 attogram (10^{-18}gram) heavier (Using $E = mc^2$)
- Of course, this is less than most sensitive scale can measure (it can measure 10^{-9} grams)
- Of course, this weight difference overwhelmed by battery discharge, weight from getting warm, ….
- According to John Kubiatowicz (New York Times, Oct 24, 2011)

Storage Performance & Price (Jan 2013)

	Bandwidth (Sequential R/W)	Cost/GB	Size
HDD[2]	50-100 MB/s	$0.03-0.07/GB	2-4 TB
SSD[1,2]	200-550 MB/s (SATA) 6 GB/s (read PCI) 4.4 GB/s (write PCI)	$0.87-1.13/GB	200GB-1TB
DRAM[2]	10-16 GB/s	$4-14*/GB *SK Hynix 9/4/13 fire	64GB-256GB

[1] http://www.fastestssd.com/featured/ssd-rankings-the-fastest-solid-state-drives/

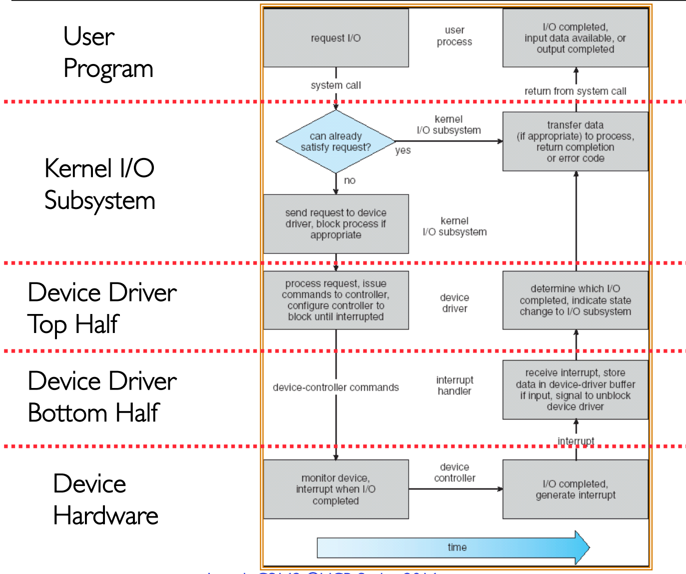
[2] http://www.extremetech.com/computing/164677-storage-pricewatch-hard-drive-and-ssd-prices-drop-making-for-a-good-time-to-buy

BW: SSD up to x10 than HDD, DRAM > x10 than SSD
Price: HDD x20 less than SSD, SSD x5 less than DRAM

SSD Summary

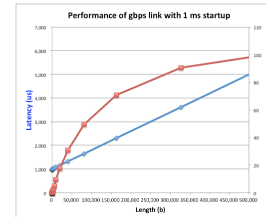
- Pros (vs. hard disk drives):
  - Low latency, high throughput (eliminate seek/rotational delay)
  - No moving parts:
    - » Very light weight, low power, silent, very shock insensitive
  - Read at memory speeds (limited by controller and I/O bus)
- Cons
  - Small storage (0.1-0.5x disk), expensive (20x disk ???)
    - » Hybrid alternative: combine small SSD with large HDD
  - Asymmetric block write performance: read pg/erase/write pg
    - » Controller garbage collection (GC) algorithms have major effect on performance
  - Limited drive lifetime
    - » 1-10K writes/page for MLC NAND
    - » Avg failure rate is 6 years, life expectancy is 9–11 years
- These are changing rapidly!

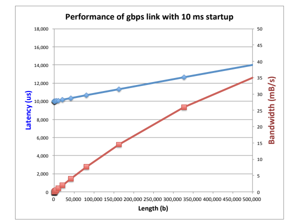
Review: Life Cycle of An I/O Request



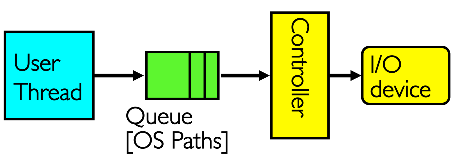
What Goes into Startup Cost for I/O?

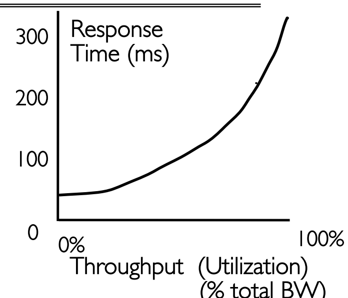
- Syscall overhead
- Operating system processing
- Controller Overhead
- Device Startup
  - Mechanical latency for a disk
  - Media Access + Speed of light + Routing for network
- Queuing (next topic)





I/O Performance





Response Time = Queue + I/O device service time

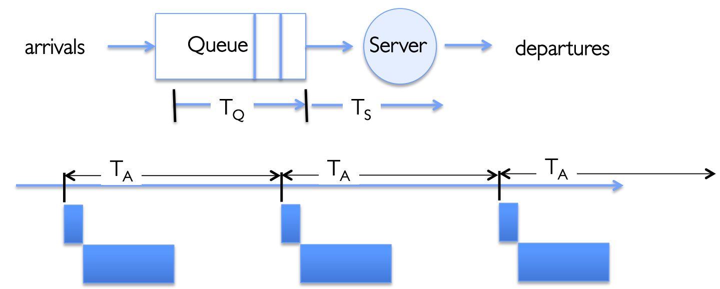
- Performance of I/O subsystem
  - Metrics: Response Time, Throughput
  - Effective BW per op = transfer size / response time
    - » EffBW(n) = n / (S + n/B) = B / (1 + SB/n)
  - Contributing factors to latency:
    - » Software paths (can be loosely modeled by a queue)
    - » Hardware controller
    - » I/O device service time
- Queuing behavior:
  - Can lead to big increases of latency as utilization increases
  - Solutions?

Administrivia

- Project 2 design reviews this week

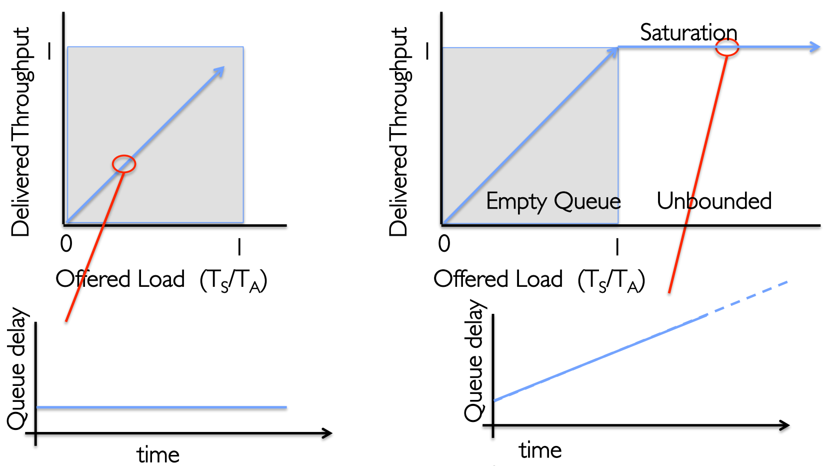
BREAK

A Simple Deterministic World



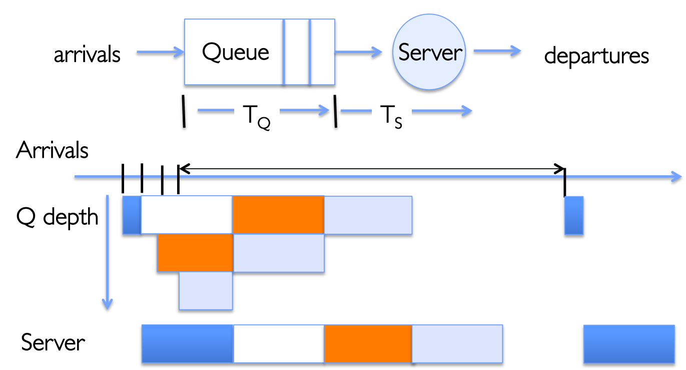
- Assume requests arrive at regular intervals, take a fixed time to process, with plenty of time between …
- Service rate ($\mu = 1/T_S$) - operations per sec
- Arrival rate: ($\lambda = 1/T_A$) - requests per second
- Utilization: U = λ / μ , where $\lambda < \mu$
- Average rate is the complete story

A Ideal Linear World



- What does the queue wait time look like?
  – Grows unbounded at a rate ~ (T_S/T_A) till request rate subsides

A Bursty World

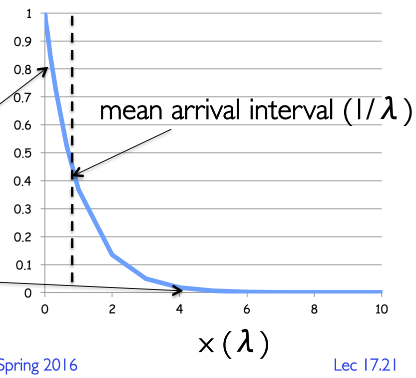


- Requests arrive in a burst, must queue up till served
- Same average arrival time, but almost all of the requests experience large queue delays
- Even though average utilization is low

So how do we model the burstiness of arrival?

- Elegant mathematical framework if you start with *exponential distribution*
  - Probability density function of a continuous random variable with a mean of $1/\lambda$
  - $f(x) = \lambda e^{-\lambda x}$
  - *"Memoryless"*

Likelihood of an event occurring is independent of how long we've been waiting

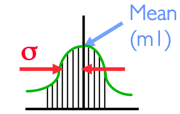


Lots of short arrival intervals (i.e., high instantaneous rate)

Few long gaps (i.e., low instantaneous rate)

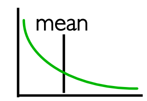
Background: General Use of random distributions

- Server spends variable time with customers
  - Mean (Average) $m1 = \Sigma p(T)\times T$
  - Variance $\sigma^2 = \Sigma p(T)\times(T-m1)^2 = \Sigma p(T)\times T^2-m1^2$
  - Squared coefficient of variance: $C = \sigma^2/m1^2$ Aggregate description of the distribution.



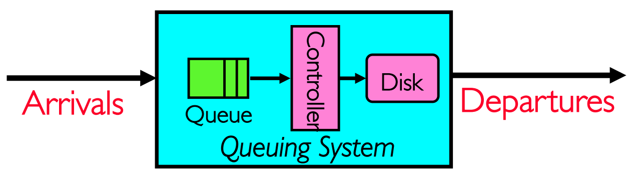
Distribution of service times

- Important values of C:
  - No variance or deterministic ⇒ C=0
  - "memoryless" or exponential ⇒ C=1
    - » Past tells nothing about future
    - » Many complex systems (or aggregates) well described as memoryless
  - Disk response times C ≈ 1.5 (majority seeks < avg)



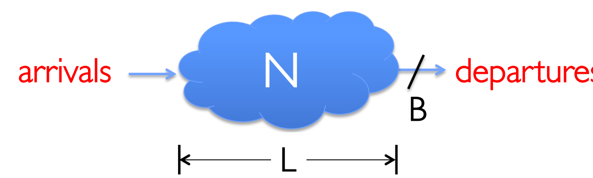
Memoryless

Introduction to Queuing Theory



- What about queuing time??
  - Let's apply some queuing theory
  - Queuing Theory applies to long term, steady state behavior ⇒ Arrival rate = Departure rate
- Arrivals characterized by some probabilistic distribution
- Departures characterized by some probabilistic distribution

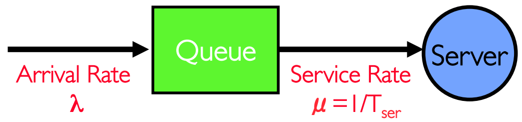
Little's Law



- In any *stable* system
  - Average arrival rate = Average departure rate
- The average number of tasks in the system (N) is equal to the throughput (B) times the response time (L)
- N (ops) = B (ops/s) × L (s)
- Regardless of structure, bursts of requests, variation in service
  - Instantaneous variations, but it washes out in the average
  - Overall requests match departures

A Little Queuing Theory: Some Results

- Assumptions:
  - System in equilibrium; No limit to the queue
  - Time between successive arrivals is random and memoryless



- Parameters that describe our system:
  - λ: mean number of arriving customers/second
  - T_{ser}: mean time to service a customer ("m1")
  - C: squared coefficient of variance = $\sigma^2/m1^2$
  - μ: service rate = $1/T_{ser}$
  - u: server utilization ($0 \le u \le 1$): $u = \lambda/\mu = \lambda \times T_{ser}$
- Parameters we wish to compute:
  - T_q: Time spent in queue
  - L_q: Length of queue = $\lambda \times T_q$ (by Little's law)
- Results:
  - Memoryless service distribution (C = 1):
    - » Called M/M/1 queue: $T_q = T_{ser} \times u/(1 - u)$
  - General service distribution (no restrictions), 1 server:
    - » Called M/G/1 queue: $T_q = T_{ser} \times ½(1+C) \times u/(1 - u))$

A Little Queuing Theory: An Example

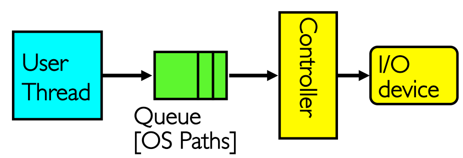
- Example Usage Statistics:
  - User requests 10 x 8KB disk I/Os per second
  - Requests & service exponentially distributed (C=1.0)
  - Avg. service = 20 ms (From controller + seek + rotation + transfer)
- Questions:
  - How utilized is the disk (server utilization)? Ans:, $u = \lambda T_{ser}$
  - What is the average time spent in the queue? Ans: T_q
  - What is the number of requests in the queue? Ans: L_q
  - What is the avg response time for disk request? Ans: $T_{sys} = T_q + T_{ser}$
- Computation:

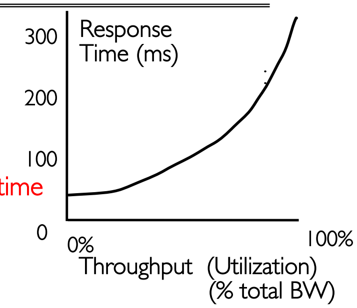
λ (avg # arriving customers/s) = 10/s
T_{ser} (avg time to service customer) = 20 ms (0.02s)
u (server utilization) = $\lambda \times T_{ser}$= 10/s x .02s = 0.2
T_q (avg time/customer in queue) = $T_{ser} \times u/(1 - u)$
= 20 x 0.2/(1-0.2) = 20 x 0.25 = 5 ms (0 .005s)
L_q (avg length of queue) = $\lambda \times T_q$=10/s x .005s = 0.05
T_{sys} (avg time/customer in system) =$T_q + T_{ser}$= 25 ms

Queuing Theory Resources

- Handouts page contains Queueing Theory Resources:
  - Scanned pages from Patterson and Hennessy book that gives further discussion and simple proof for general eq.
  - A complete website full of resources
- Some previous midterms with queueing theory questions
- Assume that Queueing theory is fair game for Midterm II and/or the final!

Optimize I/O Performance





Response Time = Queue + I/O device service time

- How to improve performance?
  - Make everything faster ☺
  - More Decoupled (Parallelism) systems
  - Do other useful work while waiting
    - » Multiple independent buses or controllers
  - Optimize the bottleneck to increase service rate
    - » Use the queue to optimize the service
- Queues absorb bursts and smooth the flow
- Admissions control (finite queues)
  - Limits delays, but may introduce unfairness and livelock

When is Disk Performance Highest?

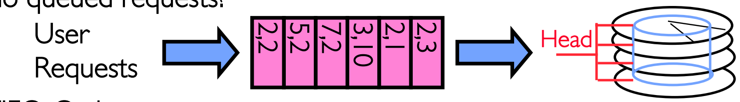
- When there are big sequential reads, or
- When there is so much work to do that they can be piggy backed (reordering queues—one moment)

- OK, to be inefficient when things are mostly idle
- Bursts are both a threat and an opportunity
- <your idea for optimization goes here>
  - Waste space for speed?

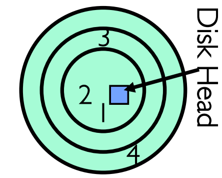
- Other techniques:
  - Reduce overhead through user level drivers
  - Reduce the impact of I/O delays by doing other useful work in the meantime

Disk Scheduling

- Disk can do only one request at a time; What order do you choose to do queued requests?



- FIFO Order
  - Fair among requesters, but order of arrival may be to random spots on the disk ⇒ Very long seeks
- SSTF: Shortest seek time first
  - Pick the request that's closest on the disk
  - Although called SSTF, today must include rotational delay in calculation, since rotation can be as long as seek
  - Con: SSTF good at reducing seeks, but may lead to starvation



- SCAN: Implements an Elevator Algorithm: take the closest request in the direction of travel
  - No starvation, but retains flavor of SSTF
- C-SCAN: Circular-Scan: only goes in one direction
  - Skips any requests on the way back
  - Fairer than SCAN, not biased towards pages in middle

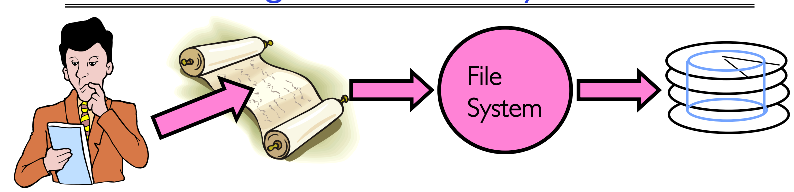
Building a File System

- File System: Layer of OS that transforms block interface of disks (or other block devices) into Files, Directories, etc.

- File System Components
  - Disk Management: collecting disk blocks into files
  - Naming: Interface to find files by name, not by blocks
  - Protection: Layers to keep data secure
  - Reliability/Durability: Keeping of files durable despite crashes, media failures, attacks, etc.

- User vs. System View of a File
  - User's view:
    - » Durable Data Structures
  - System's view (system call interface):
    - » Collection of Bytes (UNIX)
    - » Doesn't matter to system what kind of data structures you want to store on disk!
  - System's view (inside OS):
    - » Collection of blocks (a block is a logical transfer unit, while a sector is the physical transfer unit)
    - » Block size ≥ sector size; in UNIX, block size is 4KB

Translating from User to System View



- What happens if user says: give me bytes 2—12?
  - Fetch block corresponding to those bytes
  - Return just the correct portion of the block
- What about: write bytes 2—12?
  - Fetch block
  - Modify portion
  - Write out Block
- Everything inside File System is in whole size blocks
  - For example, `getc()`, `putc()` ⇒ buffers something like 4096 bytes, even if interface is one byte at a time
- From now on, file is a collection of blocks

Disk Management Policies

- Basic entities on a disk:
  - File: user-visible group of blocks arranged sequentially in logical space
  - Directory: user-visible index mapping names to files (next lecture)
- Access disk as linear array of sectors. Two Options:
  - Identify sectors as vectors [cylinder, surface, sector], sort in cylinder-major order, not used anymore
  - Logical Block Addressing (LBA): Every sector has integer address from zero up to max number of sectors
  - Controller translates from address ⇒ physical position
    - » First case: OS/BIOS must deal with bad sectors
    - » Second case: hardware shields OS from structure of disk
- Need way to track free disk blocks
  - Link free blocks together ⇒ too slow today
  - Use bitmap to represent free space on disk
- Need way to structure files: File Header
  - Track which blocks belong at which offsets within the logical file structure
  - Optimize placement of files' disk blocks to match access and usage patterns

Summary

- Devices have complex protocols for interaction and performance characteristics
  - Response time (Latency) = Queue + Overhead + Transfer
    - » Effective BW = BW * T/(S+T)
  - HDD: controller + seek + rotation + transfer
  - SDD: controller + transfer (erasure & wear)
- Bursts & High Utilization introduce queuing delays
- Systems (e.g., file system) designed to optimize performance and reliability
  - Relative to performance characteristics of underlying device
- Disk Performance:
  - Queuing time + Controller + Seek + Rotational + Transfer
  - Rotational latency: on average ½ rotation
  - Transfer time: spec of disk depends on rotation speed and bit storage density
- Queuing Latency:
  - M/M/1 and M/G/1 queues: simplest to analyze
  - As utilization approaches 100%, latency → ∞

$$T_q = T_{ser} \times ½(1+C) \times u/(1 – u))$$