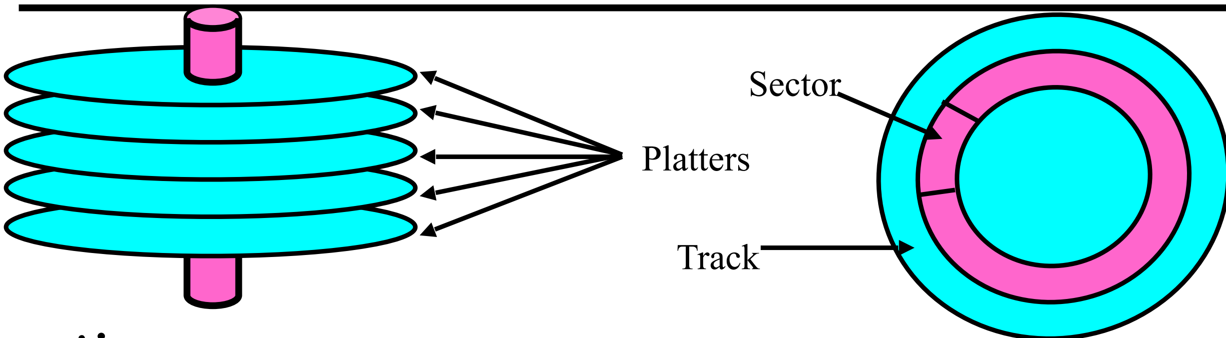
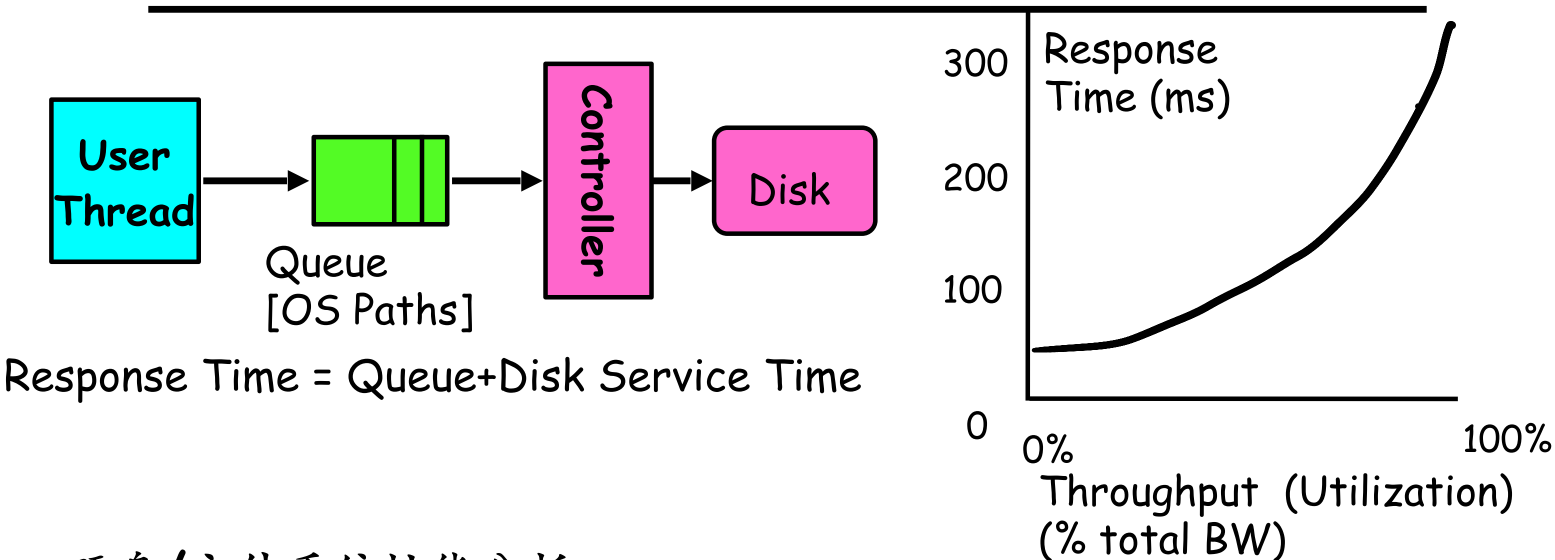
外部存储管理

Properties of a Hard Magnetic Disk



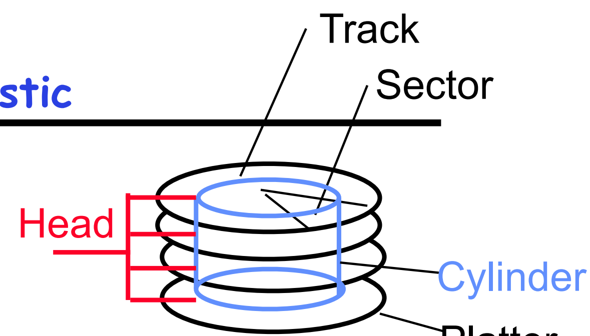
- Properties
  - Independently addressable element: sector
    - » OS always transfers groups of sectors together—"blocks"
  - A disk can access directly any given block of information it contains (random access). Can access any file either sequentially or randomly.
  - A disk can be rewritten in place: it is possible to read/modify/write a block from the disk
- Typical numbers (depending on the disk size):
  - 500 to more than 20,000 tracks per surface
  - 32 to 800 sectors per track
    - » A sector is the smallest unit that can be read or written
- Zoned bit recording
  - Constant bit density: more sectors on outer tracks
  - Speed varies with track location

Disk I/O Performance

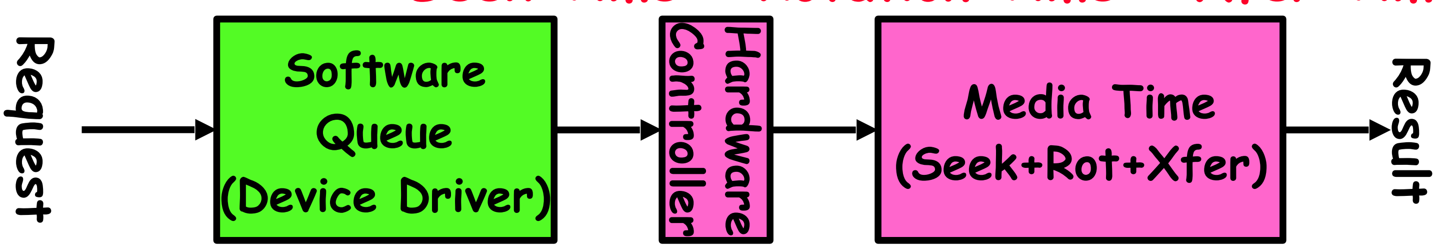


- 硬盘/文件系统性能分析
  - **Metrics:** 响应时间，吞吐(**throughput**)
  - **Contributing factors to latency:**
    - » Software paths (can be loosely modeled by a queue)
    - » Hardware controller
    - » Physical disk media
  - **Queuing behavior:**
    - » Can lead to big increases of latency as utilization approaches 100%

Magnetic Disk Characteristic



- Cylinder: all the tracks under the head at a given point on all surface
- Read/write data is a three-stage process:
  - 寻道时间/Seek time: position the head/arm over the proper track (into proper cylinder)
  - 旋转延迟/Rotational latency: wait for the desired sector to rotate under the read/write head
  - 传输时间/Transfer time: transfer a block of bits (sector) under the read-write head
- Disk Latency = Queueing Time + Controller time + Seek Time + Rotation Time + Xfer Time



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

Typical Numbers of a Magnetic Disk

- Average seek time as reported by the industry:
  - Typically in the range of 8 ms to 12 ms
  - Due to locality of disk reference may only be 25% to 33% of the advertised number
- Rotational Latency:
  - Most disks rotate at 3,600 to 7200 RPM (Up to 15,000RPM or more)
  - Approximately 16 ms to 8 ms per revolution, respectively
  - An average latency to the desired information is halfway around the disk: 8 ms at 3600 RPM, 4 ms at 7200 RPM
- Transfer Time is a function of:
  - Transfer size (usually a sector): 512B – 1KB per sector
  - Rotation speed: 3600 RPM to 15000 RPM
  - Recording density: bits per inch on a track
  - Diameter: ranges from 1 in to 5.25 in
  - Typical values: 2 to 50 MB per second
- Controller time depends on controller hardware
- Cost drops by factor of two per year (since 1991)

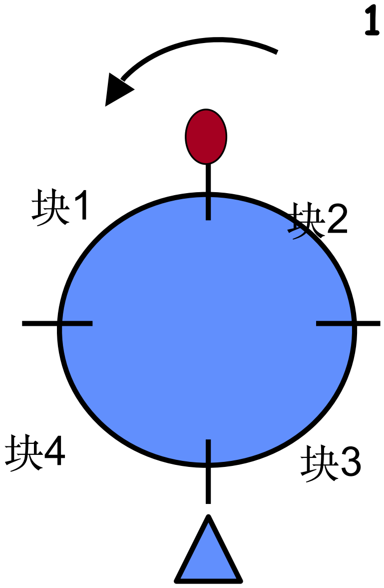
Disk Performance

- Assumptions:
  - Ignoring queuing and controller times for now
  - Avg seek time of 5ms, avg rotational delay of 4ms
  - Transfer rate of 4MByte/s, sector size of 1 KByte
- Random place on disk:
  - Seek (5ms) + Rot. Delay (4ms) + Transfer (0.25ms)
  - Roughly 10ms to fetch/put data: 100 KByte/sec
- Random place in same cylinder:
  - Rot. Delay (4ms) + Transfer (0.25ms)
  - Roughly 5ms to fetch/put data: 200 KByte/sec
- Next sector on same track:
  - Transfer (0.25ms): 4 MByte/sec
- Key to using disk effectively (esp. for filesystems) is to minimize seek and rotational delays

Disk Tradeoffs

- How do manufacturers choose disk sector sizes?
  - Need 100-1000 bits between each sector to allow system to measure how fast disk is spinning and to tolerate small (thermal) changes in track length
- What if sector was 1 byte?
  - Space efficiency – only 1% of disk has useful space
  - Time efficiency – each seek takes 10 ms, transfer rate of 50 – 100 Bytes/sec
- What if sector was 1 KByte?
  - Space efficiency – only 90% of disk has useful space
  - Time efficiency – transfer rate of 100 KByte/sec
- What if sector was 1 MByte?
  - Space efficiency – almost all of disk has useful space
  - Time efficiency – transfer rate of 4 MByte/sec

– 循环排序
  - » 按照数据的分布对输入/输出请求进行排序，提高处理的效率
  - » 举例
    - 假设每个磁道上保存**4**个记录（块），磁盘旋转速度是**20ms/**转，如果收到如下请求序列：读记录**4**、读记录**3**、读记录**2**、读记录**1**，则如何安排输入/输出顺序，到达理想的处理性能。
      - – 按请求次序读取上述记录，总的处理时间：**(1/2+1/4+3*3/4) * 20 = 60 (ms)**
      - – 按读取记录**1，2，3，4**的顺序，则总的处理时间为：**(3/4 + ¼ + 3* ¼) * 20 = 35 (ms)**
      - – 如果知道当前读位置为记录**3**，则按读取记录**4,1,2,3**顺序，则总的处理时间为：**(4 * ¼) * 20 = 20 (ms)**



磁盘访问优化

- 优化分布
  - 按照数据处理的规律，合理安排其磁盘上的分布，以提高处理的效率
  - 举例
    - 假设每个磁道上划分为**10**个块，分别存放**A~J**十个逻辑记录，磁盘旋转速度是**20ms/**转。如果处理程序读出每个记录后花**4ms**进行处理，则如何安排逻辑记录的存放位置，以达到理想的处理性能？

1	A
2	H
3	E
4	B

2*2 = 4ms

Disks vs. Memory

- Smallest write: sector
- Atomic write = sector
- Random access: 5ms
  - not on a good curve
- Sequential access: 200MB/s
- Cost $.002MB
- Crash: doesn't matter ("non-volatile")

- (usually) bytes
- byte, word
- 50 ns
  - faster all the time
- 200-1000MB/s
- $.10MB
- contents gone ("volatile")

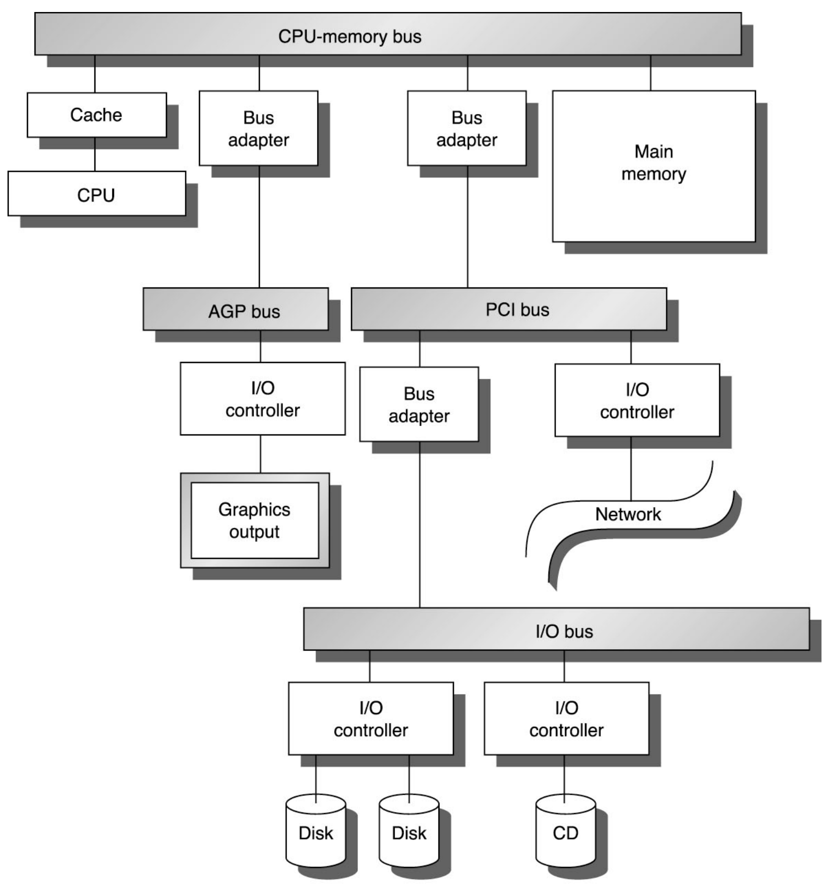
Using RAM for Storage

- Disks are about 100 times cheaper ($/MB)
- DRAM is about 100,000 faster (latency)
- Solid-State Disks
  - Actually, a DRAM and a battery
    - Much faster than disk, more reliable
    - Expensive (not very good for archives and such)
- Flash memory
  - Much faster than disks, but slower than DRAM
  - Very low power consumption
  - Can be sold in small sizes (few GB, but tiny)

Busses for I/O

- Traditionally, two kinds of busses
  - CPU-Memory bus (fast, short)
  - I/O bus (can be slower and longer)
- Now: PCI
  - Pretty fast and relatively short
  - Can connect fast devices directly
  - Can connect to longer, slower I/O busses
- Data transfers over a bus: transactions

Buses in a System



Bus Design Decisions

- **Split transactions**
  - Traditionally, bus stays occupied between request and response on a read
  - Now, get bus, send request, free bus (when response ready, get bus, send response, free us)
- **Bus mastering**
  - Which devices can initiate transfers on the bus
  - CPU can always be the master
  - But we can also allow other devices to be masters
  - With multiple masters, need arbitration

CPU-Device Interface

- Devices typically accessible to CPU through control and data registers
- These registers can be either
  - Memory mapped
    - Some physical memory addresses actually map to I/O device registers
    - Read/write through LS/ST
    - Most RISC processors support only this kind of I/O mapping
  - Be in a separate I/O address space
    - Read/write through special IN/OUT instrs
    - Used in x86, but even in x86 PCs some I/O is memory mapped

CPU-Device Interface

- Devices can be very slow
  - When given some data, a device may take a long time to become ready to receive more
  - Usually we have a Done bit in status register
- Checking the Done bit
  - Polling: test the Done bit in a loop
  - Interrupt: interrupt CPU when Done bit becomes 1
  - Interrupts if I/O events infrequent or if device is slow
    - Each interrupt has some OS and HW overhead
  - Polling better for devices that are done quickly
    - Even then, buffering data in the device lets us use interrupts
  - Interrupt-driven I/O used today in most systems

Dependability

- Quality of delivered service that justifies us relying on the system to provide that service
  - Delivered service is the *actual behavior*
  - Each module has an ideal *specified behavior*
- Faults, Errors, Failures
  - Failure: actual deviates from specified behavior
  - Error: defect that results in failure
  - Fault: cause of error

Reliability and Availability

- Reliability
  - Measure of continuous service accomplishment
  - Typically, Mean Time To Failure (MTTF)

- Availability
  - Service accomplishment as a fraction of overall time
  - Also looks at Mean Time To Repair (MTTR)
    - MTTR is the average duration of service interruption
  - Availability=MTTF/(MTTF+MTTR)

Faults Classified by Cause

- Hardware Faults
  - Hardware devices fail to perform as designed
- Design Faults
  - Faults in software and some faults in HW
  - E.g. the Pentium FDIV bug was a design fault
- Operation Faults
  - Operator and user mistakes
- Environmental Faults
  - Fire, power failure, sabotage, etc.

Faults Classified by Duration

- **Transient Faults**
  - Last for a limited time and are not recurring
  - An alpha particle can flip a bit in memory but usually does not damage the memory HW
- **Intermittent Faults**
  - Last for a limited time but are recurring
  - E.g. overclocked system works fine for a while, but then crashes… then we reboot it and it does it again
- **Permanent Faults**
  - Do not get corrected when time passes
  - E.g. the processor has a large round hole in it because we wanted to see what's inside…

Improving Reliability

- **Fault Avoidance**
  - Prevent occurrence of faults by construction
- **Fault Tolerance**
  - Prevent faults from becoming failures
  - Typically done through redundancy
- **Error Removal**
  - Removing latent errors by verification
- **Error Forecasting**
  - Estimate presence, creation, and consequences of errors

Disk Fault Tolerance with RAID

- Redundant Array of Inexpensive Disks
  - Several smaller disks play a role of one big disk
- Can improve performance
  - Data spread among multiple disks
  - Accesses to different disks go in parallel
- Can improve reliability
  - Data can be kept with some redundancy