Today

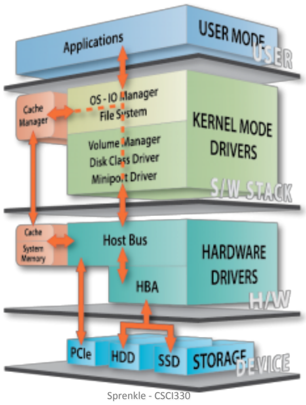
- File Systems intro
- Storage
- Disk scheduling

Review

- What are the synchronization mechanisms we covered?
  - When would you use them?

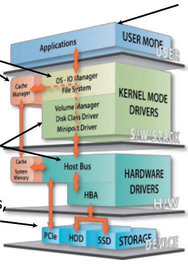
Where We Are …

- We’ve talked about
  - Kernel
  - Processes, process management
  - Synchronization
- Moving toward storage
  - File systems
    - Disk management, storage
  - Memory management



[Calypso]

Storage Stack



Databases, Hadoop, etc.

We care mostly about this stuff.

File system API. Generic, for use over many kinds of storage devices.

Device driver software is a huge part of the kernel, but we mostly ignore it.

Standard block I/O internal interface. Block read/write on numbered blocks on each device/partition. For kernel use only: DMA + interrupts.

Many storage technologies, advancing rapidly with time.

Rotational disk (HDD): cheap, mechanical, high latency.

[Calypso]

Demands on File Systems?

- What do users want from a file system?
  - Do demands differ depending on the machine?

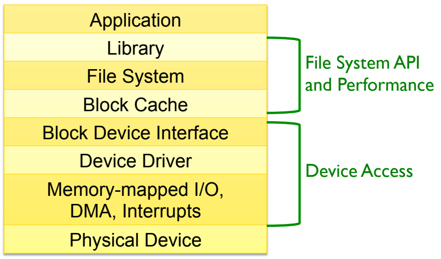
Goals for File Systems

- Reliable
- Large capacity, low cost
- High performance
- Named data
- Controlled sharing

Difference in Perspectives

- User Perspective:
  - A disk is a collection of files and directories that can be manipulated using commands.
- OS Perspective:
  - A disk is a collection of data blocks that can be manipulated via a cylinder:head:sector addresses.
- It is the job of the OS to bridge the gap between these two perspectives.

Layered Abstractions to I/O Systems



Application
Library
File System
Block Cache
Block Device Interface
Device Driver
Memory-mapped I/O, DMA, Interrupts
Physical Device

File System API and Performance

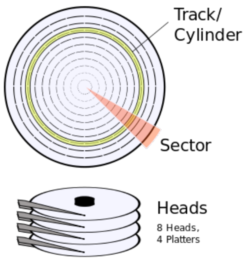
Device Access

Storage Management

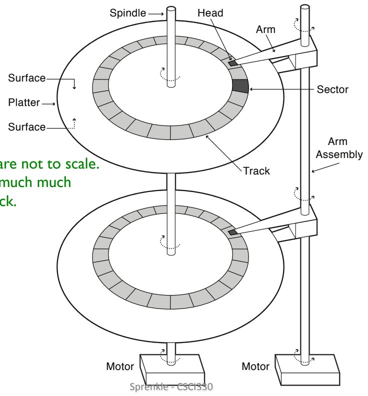
- Storage management is responsible for:
  - Creating / deleting files
  - Creating / deleting directories
  - File / directory manipulation
  - Read / write / change permissions
  - Mapping files and directories onto disk
  - Tracking free / used disk space.

Raw Hardware Assumptions

- Hard Disks:
  - Basic hard disk controller can:
    - Read a sector (or block)
    - Write a sector (or block)
  - Sector to read/write is specified by a cylinder:head:sector (CHS) address



Magnetic Disk



Head and track are not to scale. Head is actually much much bigger than a track.

The First Commercial Disk Drive

1956
IBM RAMDAC computer included the IBM Model 350 disk storage system

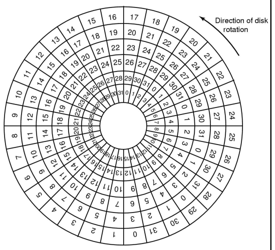
5M (7 bit) characters
50 x 24” platters
Access time = < 1 second

Disk “addressing”

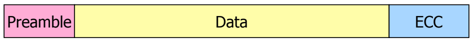
- Millions of sectors on the disk must be labeled
- Two possibilities
  - Cylinder/track/sector
  - Sequential numbering
- Modern drives use sequential numbers
  - Disks map sequential numbers into specific location
  - Mapping may be modified by the disk
    - Remap bad sectors
    - Optimize performance
  - Hide the exact geometry, making life simpler for the OS

Sector layout on disk

- Sectors numbered sequentially on each track
- Numbering starts in different place on each track: *sector skew*
  - Allows time for switching head from track to track
- All done to minimize delay in sequential transfers



Structure of a disk sector



- Preamble contains information about the sector
  - Sector number & location information
- Data is usually 256, 512, or 1024 bytes
- ECC (Error Correcting Code) is used to detect & correct minor errors in the data

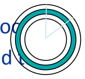
Hard Disk Performance

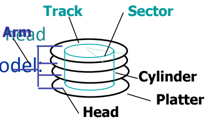
- When working with hard disks three times impact performance:
  - Seek Time
  - Rotational Latency
  - Transfer Time

Access time

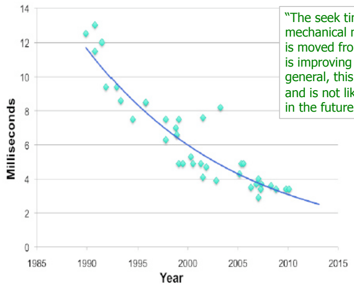
How long to access data on disk?

- 5-15 ms on average for access to random lo
- Includes **seek time** to move head to desired
  - Roughly linear with radial distance
- Includes **rotational delay**
  - Time for sector to rotate under head
- These times depend on drive model:
  - platter width (e.g., 2.5 in vs. 3.5 in)
  - rotation rate (5400 RPM vs. 15K RPM).
  - Enterprise drives use more/smaller platters spinning faster.
- These properties are mechanical and improve slowly as technology advances over time.





Average seek time

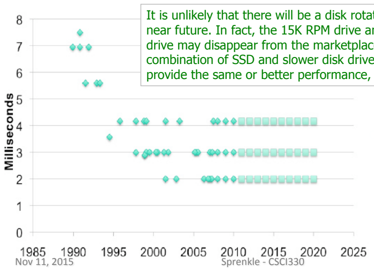


"The seek time is due to the mechanical motion of the head when it is moved from one track to another. It is improving by about 5% CAGR. In general, this is a mature technology and is not likely to change dramatically in the future. "

IBM Research Report 2011
GPFS Scans 10 Billion Files
in 43 Minutes

Rotational latency

The average disk latency is ½ the rotational time of the disk drive. As you can see from its recent history…[it] has settled down to three values 2, 3 and 4.1 milliseconds. These are ½ the inverses of 15,000, 10,000 and 7,200 revolutions per minute (RPM), respectively.



It is unlikely that there will be a disk rotational speed increase in the near future. In fact, the 15K RPM drive and perhaps the 10K RPM drive may disappear from the marketplace…driven by the successful combination of SSD and slower disk drives into storage systems that provide the same or better performance, cost and power.

IBM Research Report 2011
GPFS Scans 10 Billion Files
in 43 Minutes

A few words about SSDs

- Solid State Drives (e.g., Flash memory):
  - No spinning platter, no arm to move, no me
  - Faster than disk (at least for reads), slower than DRAM.
  - **No seek cost**. But writes require slow block erase, and/or limited # of writes to each cell before it fails.
  - Technology is advancing rapidly; costs are dropping.
- How should we use them? Are they just fast/expensive disks? Or can we use them like memory that is persistent? Open research question.
- **Trend**: use them as block storage, and/or combine them with HDDs to make hybrids optimized for particular uses.

Disk Scheduling

- The operating system is responsible for using hardware efficiently
  - For the disk drives: having a fast access time and disk bandwidth
- Minimize seek time
- Seek time ≈ seek distance
- **Disk bandwidth** is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer

Disk Scheduling

- Many sources of disk I/O request
  - OS
  - System processes
  - Users processes
- I/O request includes input or output mode, disk address, memory address, number of sectors to transfer
- OS maintains queue of requests, per disk or device
  - Idle disk can immediately work on I/O request
  - Busy disk means work must queue

Optimizing Disk Scheduling

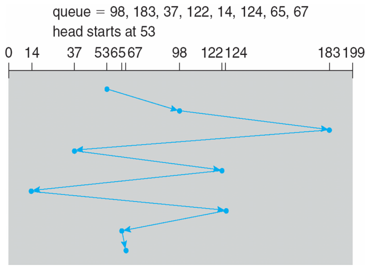
- Goal: optimize performance
  - First: disk bandwidth
  - Any other concerns?
- How can we optimize disk scheduling?
- What are possible algorithms?
  - What are their tradeoffs?
- What concerns/questions do we have in picking an algorithm?

Disk Scheduling

- Several algorithms exist to schedule the servicing of disk I/O requests
- The analysis is true for one or many platters
- Consider a request queue
  - 98, 183, 37, 122, 14, 124, 65, 67
  - Head pointer 53

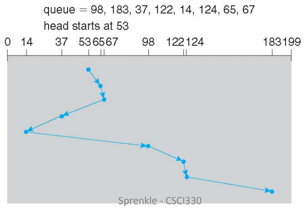
FCFS: First Come First Serve

Illustration shows total head movement of 640 cylinders



Shortest Seek Time First (SSTF)

- SSTF selects request with the minimum seek time from the current head position
- SSTF scheduling is a form of SJF scheduling
  - may cause starvation of some requests
- Illustration shows total head movement of 236 cylinders



SCAN Algorithm

- The disk arm starts at one end of the disk, and moves toward the other end
  - services requests until it gets to the other end of the disk
  - head movement is reversed and servicing continues
- SCAN algorithm - sometimes called the elevator algorithm

SCAN Algorithm

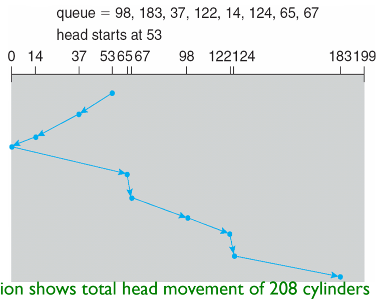
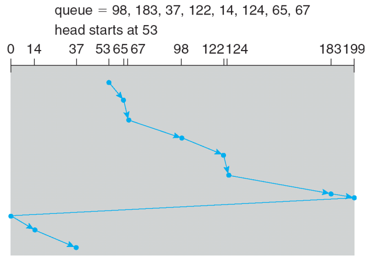


Illustration shows total head movement of 208 cylinders
Note: if requests are uniformly dense,
largest density at other end of disk and those wait the longest

C-SCAN

- Provides a more uniform wait time than SCAN
- Head moves from one end of the disk to the other, servicing requests as it goes
  - ➢ When it reaches the other end, it immediately returns to the beginning of the disk, without servicing any requests on the return trip
- Treats the cylinders as a circular list that wraps around from the last cylinder to the first one

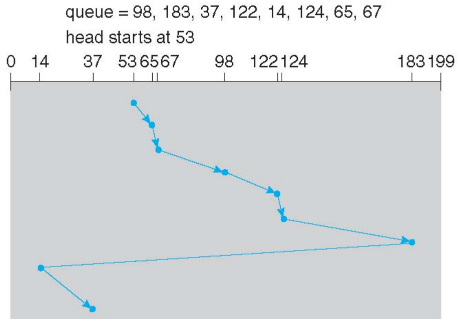
C-SCAN



C-LOOK

- LOOK a version of SCAN, C-LOOK a version of C-SCAN
- Arm only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk

C-LOOK



Selecting a Disk-Scheduling Algorithm

- SSTF is common and has a natural appeal
- SCAN and C-SCAN perform better for systems that place a heavy load on the disk
  - ➢ Less starvation
- Performance depends on the number and types of requests
- Requests for disk service can be influenced by the file-allocation method and metadata layout
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary
- Either SSTF or LOOK is a reasonable choice for the default algorithm

Looking Ahead

- Project 4 due Sunday, Nov 29
  - ➢ Shorter in words but not in difficulty
  - ➢ Friday: work period