Mass Storage Structure

- These days, mass storage is pretty much all about disks, in one form or another
- Physically speaking, disks consist of circular *platters* over which *read-write heads* can retrieve or encode bits
- Each platter breaks down into concentric *tracks*, and each track consists of multiple *sectors*
- When a disk has more than one platter, the platters are arranged vertically, with the tracks in alignment; these aligned sets of tracks are called *cylinders*

- While the presentation of a disk as a linear array of blocks is a logical (albeit low-level) abstraction, this structure does have a physical basis: logically adjacent blocks generally correspond to physically adjacent sectors, with a physical address corresponding to a (cylinder, track, sector) designation
- Note how this physical layout bears out the relative durations of disk seek and transfer times: physically, a disk seek consists of moving the head to the desired cylinder, then rotating the platters to the right sector — a distinct action that results in *rotational latency*
- Transfer is a matter of "flying over" the spinning track; thus, we see why seek tends to weigh much more heavily on performance than transfer

Disk Attachment

- *Host-attached storage* refers to disks that connect directly to some dedicated port on the computer, using technologies such as [E]IDE, [S]ATA, USB, SCSI, fiber channel, and/or FireWire (IEEE 1394)
- *Network-attached storage* (NAS) refers to disks that are visible through a network, with communication done purely through a network protocol
- *Storage-area networks* (SAN) move the networking "beneath" a single access point, allowing for flexibility on how storage units are allocated and managed

Disk Scheduling

- The physical aspects of a disk pose non-trivial scheduling issues: given a sequence of block requests, what is the best way to service them?
- This problem is analogous to process or I/O scheduling, with of course constraints and optimizations that are specific to disk structure
- Current algorithms include: the always-intuitive-but-seldom-optimal FCFS, SSTF (analogous to SJF), [C-]SCAN, and [C-]LOOK — the latter two specifically use a disk's circular physical form factor

Disk Management Issues

- At this point it should be clear that a file system on a disk is a multi-layered, persistent data structure, and as with any data structure, initialization is necessary before it can be used: thus, we *format* or *partition* disks
- Additionally, computers need to start *somewhere* when loading an operating system; this is encoded in a standardized *boot block* or *boot partition* on the disk
- Finally, disk media do fail, resulting in *bad blocks* — we need error-correcting code (ECC) or other methods for tracking or dealing with these

Swap-Space Management

- Recall that disks aren't just used for files — the operating system uses disk storage for other tasks as well, not the least of which is virtual memory
- Disk storage used by virtual memory is called *swap space*, and for performance reasons it may be managed outside the normal umbrella of the file system; in the outside-the-file-system case, a special *raw partition* may be set aside for swap-space use
- Ideally, an OS should allow multiple swap storage options, depending on how a computer system is used

RAID

- "RAID" once stood for *redundant array of inexpensive disks*; however, as virtually all disks are inexpensive now, RAID technology is used more for reliability and speed rather than cost, and so *independent* has replaced *inexpensive*, and the acronym is preserved :)
- RAID adds a layer of abstraction that allows multiple physical disk units to be seen as a single disk
- The main forces that drive RAID design decisions are *redundancy* and *parallelism* — a *RAID level* is some configuration that achieves these in a particular way

- *Redundancy* may be achieved by *mirroring* (copying) or by ECC (*parity* mainly, but may be something else)
- *Parallelism* is achieved by *striping* — distributing simultaneously-accessed data across the disk array; this may occur at the *bit-*, *block-*, or some other level
- Since RAID involves a new layer of abstraction, the question of *where* that layer is implemented arises: options include software-only (OS-level), hardware (host adapters, RAID controllers), and network (SAN)
- Other implementation issues include the availability of a *hot spare* (one or more reserved disks that come online immediately after a disk failure) and *replication* (like mirroring, but across different disk arrays, ideally in different sites to protect against local disasters)

Stable Storage

- Another category of storage is required for journaling file systems — a key assumption of journaling (or *write-ahead logging*) is that the journal or log is *stable*: errors in the log should be recoverable or must not lead to inconsistencies in the disk state
- The main trick to stable storage is to write logical blocks to two or more distinct physical blocks; recovery consists of using ECC on the blocks and comparing their contents — the more block copies, the better, but two is usually good enough

Tertiary Storage

- Certain storage media are viewed as *tertiary storage* — a level beyond the storage provided by fixed disks, distinguished primarily because they are *removable*
- *Removable disks* represent one category: CDs, DVDs
- *Tapes* represent another category: great for sequential access, but impossible for direct access, so we almost never put a file system on them
- *Non-volatile RAM (NV-RAM) "drives"* consisting of *flash memory* are the latest technology to go mainstream

Mass Storage and Cost

- Once upon a time, a storage infrastructure had a distinct hierarchical component — the idea was that we have succeeding layers of progressively slower but cheaper (and therefore larger-capacity) storage media
- In this *hierarchical storage management* model, devices such as a *jukebox* stored theoretically more data than disks at lower cost, but at significantly greater latency
- This paradigm has since faded, since today's disks are so large and inexpensive that the actual *need* for a storage layer beyond disks has largely disappeared