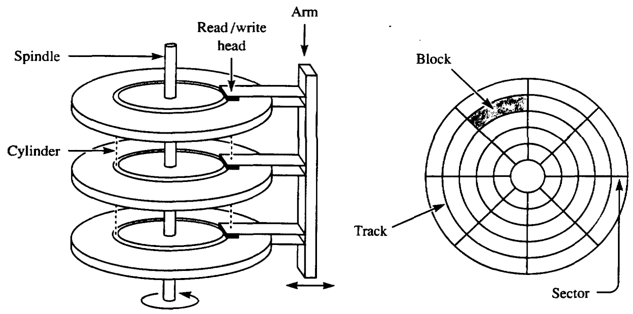
Disk Hardware



(a) A hard disk drive. (b) A single disk.

Diagram from *Computer Science*, Volume 2, J. Stanley Warford, Heath, 1991.

- *Seek time* — time required to position heads over the track / cylinder
  - Typically 10 ms to cross entire disk
- *Rotational delay* — time required for sector to rotate underneath the head
  - 120 rotations / second = 8 ms / rotation

Disk Access Times

- From last lecture (typical numbers):
  - 32-64 sectors per track
  - 1K bytes per sector
- *Data transfer rate* is number of bytes rotating under the head per second
  - 1 KB / sector * 32 sectors / rotation * 120 rotations / second = 4 MB / s
- Disk I/O time = seek + rotational delay + transfer
  - If head is at a random place on the disk
    - Avg. seek time is 5 ms
    - Avg. rotational delay is 4 ms
    - Data transfer rate for a 1KB is 0.25 ms
    - I/O time = 9.25 ms for 1KB
    - ↪Real transfer rate is roughly 100 KB / s
  - In contrast, memory access may be 20 MB / s (200 times faster)

Selecting the Sector Size

- The read / write head needs to synchronize with the head as it rotates
  - Need 100-1000 bits between each sector to measure how fast disk is spinning
- If sector size is 1 byte
  - Only 1% of disk holds useful data
  - 1/1000 transfer rate as before = 100 B / s
- If sector size is 1 KB
  - 90% of disk holds useful data
  - Transfer rate is 100 KB / s
- If sector size is 1 MB
  - Almost all of disk holds useful data
  - Transfer rate is 4 MB / s (full disk transfer rate — seek and rotational latency usually won't matter anymore)

Evolution of UNIX Disk Management

- In traditional UNIX, and Berkeley BSD 3.0 UNIX
  - Disk lock size was 512 bytes
- In Berkeley BSD 4.0 UNIX:
  - Block size was changed to 1024 bytes
  - More or less doubled performance
    - Each block access fetched twice as much data, so there was less disk seek overhead
    - More files could use only the direct block of the inode, which saved further space
  - When file system was first created
    - Free list was ordered, and they got transfer rates up to 175 KB / s
    - After a few weeks, data and free blocks got so randomized that the transfer rate went down to 30 KB / s
    - This was less than 4% of the maximum transfer rate!

Evolution of UNIX Disk Management (cont.)

- What about making the blocks bigger?
  - Causes internal fragmentation
  - Most files are small, maybe one block
- Some measurements from a file system at UC Berkeley:

Organization	Space used	Waste
Data only	775.2	0%
+inodes, 512B block	828.7	6.9%
+inodes, 1KB block	866.5	11.8%
+inodes, 2KB block	948.5	22.4%
+inodes, 4KB block	1128.3	45.6%

- The presence of small files kills the performance for large files!
  - Want big blocks to reduce the seek overhead for big files
  - But… big blocks increase fragmentation for small files

Evolution of UNIX Disk Management (cont.)

- In Berkeley BSD 4.2 UNIX:
  - See "A Fast File System for UNIX" on class home page for details
  - Introduced concept of a *cylinder group*
    - A *cylinder* is the set of corresponding tracks on all the disk surfaces
      - For a given head position, it's just as easy to access one track in the cylinder as it is to access any other
      - A cylinder group is a set of adjacent cylinders
    - Each cylinder group has a copy of super block, bit map of free blocks, ilist, and blocks for storing directories and files
    - The OS tries to put related information together into the same cylinder group
      - Try to put all inodes in a directory in the same cylinder group
      - Try to put blocks for one file contiguously in the same cylinder group
        - » Bitmap of free blocks makes this easy
      - For long files, redirect each megabyte to a new cylinder group

Evolution of UNIX Disk Management (cont.)

- In Berkeley BSD 4.2 UNIX: (cont.)
  - Block size was changed to 4096 bytes
    - Reduced fragmentation as follows:
      - Each disk block can be used in its entirety, or can be broken up into 2, 4, or 8 *fragments*
      - For most of the blocks in the file, use the full block
      - For the last block in the file, use as small a fragment as possible
      - Can get as many as 8 very small files in one disk block
    - This change resulted in
      - Only as much fragmentation as a 1KB block size (w/ 4 fragments)
      - Data transfer rates that were 47% of the maximum rate
  - Other improvements:
    - Bit map instead of unordered free list (easier to keep files contiguous)
    - Variable length file names, symbolic links
    - File locking, disk quotas

Improving Performance with Good Block Management

- OS usually keeps track of free blocks on the disk using a *bit map*
  - A bit map is just an array of bits
    - 1 means the block is free,
    - 0 means the block is allocated to a file
  - For a 1.2 GB drive, there are about 307,000 4KB blocks, so a bit map takes up 38.4 KB (usually kept in memory)
- Try to allocate the next block of the file close to the previous block
  - Works well if disk isn't full
  - If disk is full, this is doesn't work well
    - Solution — keep some space (about 10% of the disk) in reserve, and don't tell users; never let disk get more than 90% full
    - With multiple platters / surfaces, there are many possibilities (one surface is as good as another), so the block can usually be allocated close to the previous one