Storage Devices

◆ Magnetic disks
◆ Disk arrays
◆ Flash memory

◆ The devices provide
  ● Storage that (usually) survives across machine crashes
  ● Block level (random) access
  ● Large capacity at low cost
  ● Relatively slow performance
    • Magnetic disk read takes 10-20M processor instructions

◆ Users typically access via file system, which provides a very different interface and translates to blocks

Where Are We?

◆ Covered:
  ● Management of CPU & concurrency
  ● Management of main memory & virtual memory

◆ Currently --- “Management of I/O devices”
  ● Last lecture: Interacting with I/O devices, device drivers
  ● This lecture: storage devices

◆ Then, file systems
  • File system structure
  • Naming and directories
  • Efficiency and performance
  • Reliability and protection

Storage devices

◆ Magnetic disks
  ● Storage that rarely becomes corrupted
  ● Large capacity at low cost
  ● Block level random access
  ● 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
A Typical Magnetic Disk Controller

- External interfaces
  - IDE/ATA, SATA(1.0, 2.0, 3.0)
  - SCSI, SCSI-2, Ultra-(160, 320, 640) SCSI
  - Fibre channel
- Cache
  - Buffer data between disk and interface
- Control logic
  - Read/write operations (incl. disk head positioning, etc.)
  - Cache replacement
  - Failure detection and recovery

Caching in a Disk Controller

- Method
  - Disk controller has DRAM to cache recently accessed blocks
    - e.g. Hitachi disk has 16MB
    - Some of the RAM space stores "firmware" (an embedded OS)
  - Blocks are replaced usually in an LRU order + "tracks"
  - Disk and Flash devices have CPU in them
- Pros
  - Good for reads if accesses have locality
- Cons
  - Expensive
  - Doesn't really help with writes since they need to be reliable

Disks Were Large

First Disk: IBM 305 RAMAC (1956)
- 5MB capacity
- 50 platters, each 24" diam

Storage Form Factors Are Changing

- Form factor: 24mm · 32mm · 2.1mm
  - Storage: 1-2TB
- Form factor: .5-1" · 4" · 5.7"
  - Storage: 0.5-6TB
- Form factor: .4-7" · 2.7" · 3.9"
  - Storage: 0.5-2TB
- Form factor: PCI card
  - Storage: 0.5-10TB
Areal Density vs. Moore’s Law

(Fontana, Decad, Hetzler, 2012)

50 Years (Mark Kryder at SNW 2006)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity</td>
<td>5MB</td>
<td>160GB</td>
<td>32,000</td>
</tr>
<tr>
<td>Areal Density</td>
<td>2K bits/in²</td>
<td>130 Gbits/in²</td>
<td>65,000,000</td>
</tr>
<tr>
<td>Disks</td>
<td>50 @ 24” diameter</td>
<td>2 @ 2.5” diameter</td>
<td>1 / 2,300</td>
</tr>
<tr>
<td>Price/MB</td>
<td>$1,000</td>
<td>$0.01</td>
<td>1 / 100,000</td>
</tr>
<tr>
<td>Spindle Speed</td>
<td>1,200 RPM</td>
<td>5,400 RPM</td>
<td>5</td>
</tr>
<tr>
<td>Seek Time</td>
<td>600 ms</td>
<td>10 ms</td>
<td>1 / 60</td>
</tr>
<tr>
<td>Data Rate</td>
<td>10 KB/s</td>
<td>44 MB/s</td>
<td>4,400</td>
</tr>
<tr>
<td>Power</td>
<td>5000 W</td>
<td>2 W</td>
<td>1 / 2,500</td>
</tr>
<tr>
<td>Weight</td>
<td>~ 1 ton</td>
<td>4 oz</td>
<td>1 / 9,000</td>
</tr>
</tbody>
</table>

Magnetic disk

Moving-head Disk Mechanism
Tracks, Cylinders, Sectors

- **Tracks**
  - Concentric rings around disk surface, bits laid out serially along each track
- **Cylinder**
  - A track of the platter, 1000-5000 cylinders per zone, 1 spare per zone
- **Sector**
  - Arc of track holding some min # of bytes, variable # sectors/track

Disk Tracks

- ~1 micron wide
  - Wavelength of light is ~0.5 micron
  - Resolution of human eye is 50 microns
  - 100K tracks on a typical 2.5” disk
- Tracks separated by unused guard regions
  - Reduces likelihood of corrupting nearby tracks during write
- Track length varies across disk
  - Outer tracks have more sectors per track, higher bandwidth
  - Disk organized into “zones” of tracks, each with same no. of sectors per track
  - Only outer half of disk radius is typically used

Disk Sectors

- **What is a sector?**
  - Header (ID, defect flag, …)
  - Real space (e.g. 512 bytes)
  - Trailer (ECC code)
- Skewed from one track to next
  - Accommodate head movement for sequential operations
- Logically addressed (usually)
- Have sophisticated ECC
  - If not recoverable, replace with a spare
- **Sector sparing**
  - When bad sector, remap it to spare sectors on same surface
  - Skip bad sectors in the future
- **Slip sparing**
  - When bad sector, remap all sectors to preserve sequential behavior

How Data are Read/Written

- **Disk surface**
  - Coated with magnetic material
- **Disk arm**
  - A disk arm carries disk heads
- **Disk head**
  - Mounted on an actuator
  - Reads/writes on disk surface
- **Read/write operation**
  - Disk controller gets read/write with (track, sector)
  - Seek the right cylinder (tracks)
  - Wait until the sector comes under the disk head
  - Perform read/write
Disk Performance

- Disk latency = seek + rotation + transfer (time)
- Seek time
  - Position heads over cylinder, typically 1-20 ms
- Rotation time
  - Wait for a sector to rotate underneath the heads
  - Disk rotation time is typically 4-15 ms
  - On average, need to wait half a rotation
- Transfer time
  - Transfer bandwidth is typically 70 -250 Mbytes/sec
- Example:
  - Performance of transfer 1 Kbytes of Desktop HDD, assuming BW = 100MB/sec, seek = 5ms, rotation = 4ms
  - Total time = 5ms + 4ms + 0.01ms = 9.01ms
  - What is the effective bandwidth?

Sample Disk Specs (from Seagate)

<table>
<thead>
<tr>
<th></th>
<th>Enterprise Performance</th>
<th>Desktop HDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity (GB)</td>
<td>600</td>
<td>4096</td>
</tr>
<tr>
<td>Discs / heads</td>
<td>3 / 6</td>
<td>4 / 8</td>
</tr>
<tr>
<td>Sector size (bytes)</td>
<td>512</td>
<td>512</td>
</tr>
<tr>
<td>Performance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>External interface</td>
<td>STA</td>
<td>SATA</td>
</tr>
<tr>
<td>Spindle speed (RPM)</td>
<td>15,000</td>
<td>7,200</td>
</tr>
<tr>
<td>Average latency (msec)</td>
<td>2.0</td>
<td>4.16</td>
</tr>
<tr>
<td>Seek time, read/write (ms)</td>
<td>3.5/3.8</td>
<td>8.5/9.5</td>
</tr>
<tr>
<td>Track-to-track read/write (ms)</td>
<td>0.2/0.4</td>
<td>0.8/1.0</td>
</tr>
<tr>
<td>Transfer rate (MB/sec)</td>
<td>138-258</td>
<td>146</td>
</tr>
<tr>
<td>Cache size (MB)</td>
<td>128</td>
<td>64</td>
</tr>
<tr>
<td>Power</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average / Idle / Sleep</td>
<td>8.5 / 6 / NA</td>
<td>7.5 / 5 / 0.75</td>
</tr>
<tr>
<td>Reliability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recoverable read errors</td>
<td>1 per 10^12 bits read</td>
<td>1 per 10^11 bits read</td>
</tr>
<tr>
<td>Non-recoverable read errors</td>
<td>1 per 10^13 bits read</td>
<td>1 per 10^12 bits read</td>
</tr>
</tbody>
</table>

Question

- How long to complete 500 random disk reads, in FIFO order?

  - Seek: average 10.5 msec
  - Rotation: average 4.15 msec
  - Transfer: 5-10 usec

  \[ 500 \times \frac{(10.5 + 4.15 + 0.01)}{1000} = 7.3 \text{ seconds} \]
Question

- How long to complete 500 sequential disk reads?
  - Seek Time: 10.5 ms (to reach first sector)
  - Rotation Time: 4.15 ms (to reach first sector)
  - Transfer Time: (outer track)
    500 sectors * 512 bytes / 128MB/sec = 2ms
  Total: 10.5 + 4.15 + 2 = 16.7 ms

Disk Performance

- Seek and rotational times dominate the cost of small accesses
  - Disk transfer bandwidth is wasted
  - Need algorithms to reduce seek time
  - Let’s look at some disk scheduling algorithms

FIFO (FCFS) order

- Method
  - First come first serve
- Pros
  - Fairness among requests
  - In the order applications expect
- Cons
  - Arrival may be on random spots on the disk (long seeks)
  - Wild swings can happen
  - Low throughput, esp with small transfers

SSTF (Shortest Seek Time First)

- Method
  - Pick the one closest on disk
  - Can include rotational delay in calculation
- Pros
  - Try to minimize seek (and rotation) time
- Cons
  - Starvation
- Question
  - Is SSTF optimal?
  - Can we avoid the starvation?

98, 183, 37, 122, 14, 124, 65, 67
Elevator (SCAN)
- Method
  - Take the closest request in the direction of travel
  - Real implementations do not go to the end (called LOOK)
- Pros
  - Bounded time for each request
- Cons
  - Request at the other end will take a while

C-SCAN (Circular SCAN)
- Method
  - Like SCAN
  - But, wrap around
  - Real implementation doesn’t go to the end (C-LOOK)
- Pros
  - Uniform service time bound regardless of where on disk
- Cons
  - Do nothing on the return, so the bound can be larger than in Elevator

Discussions
- Which is your favorite?
  - FIFO
  - SSTF
  - SCAN
  - C-SCAN
- Disk I/O request buffering
  - Where would you buffer requests?
  - How long would you buffer requests?
- More advanced issues
  - Can the scheduling algorithm minimize both seek and rotational delays?

RAID (Redundant Array of Independent Disks)
- Main idea
  - Compute XORs and store parity on disk P
  - Upon any failure, one can recover the block from using P and other disks
- Pros
  - Reliability
  - High bandwidth?
- Cons
  - Cost
  - The controller is complex
Synopsis of RAID Levels

- **RAID Level 0**: Non redundant
- **RAID Level 1**: Mirroring
- **RAID Level 2**: Byte-interleaved, ECC
- **RAID Level 3**: Byte-interleaved, parity
- **RAID Level 4**: Block-interleaved, parity
- **RAID Level 5**: Block-interleaved, distributed parity

**RAID Level 6 and Beyond**

- **Goals**
  - Less computation and fewer updates per random write
  - Small amount of extra disk space
- **Extended Hamming code**
- **Specialized Eraser Codes**
  - IBM Even-Odd, NetApp RAID-DP, ...
- **Beyond RAID-6**
  - Reed-Solomon codes, using MOD 4 equations
  - Can be generalized to deal with k (>2) disk failures

**NAND Flash Memory**

- **High capacity**
  - Single cell (more expensive, durable) vs. multiple cell
- **Small block**
  - Each page 512 + 16 Bytes (data + ECC etc)
  - 32 pages in each block
- **Large block**
  - Each page is 2048 + 64 Bytes
  - 64 pages in each block
NAND Flash Memory Operations

- **Speed**
  - Read page: ~10-20 us
  - Write page: 20-200 us
  - Erase block: ~1-2 ms

- **Limited performance**
  - Can only write 0's, so erase (set all 1) then write
  - Erasure blocks of 128-512KB are written into

- **Solution: Flash Translation Layer (FTL)**
  - Map virtual page to physical page address in flash controller
  - Keep erasing unused blocks
  - Garbage collect by copying live pages to new locations, and erasing large blocks
  - Remap to currently erased block to reduce latency

NAND Flash Lifetime

- **Wear out limitations**
  - ~50k to 100k writes / page (SLC – single level cell)
  - ~15k to 60k writes / page (MLC – multi-level cell)

- **Wear Leveling:**
  - Spread erases evenly across blocks, rather than using same block repeatedly
  - Remap pages that no longer work (like sector sparing on magnetic disks)
  - Question: Suppose write to cells evenly and 200,000 writes/sec, how long does it take to wear out 1,000M pages on SLC flash (50k/page)?

- **Who does “wear leveling?”**
  - Flash translation layer
  - File system design (later)

Flash Translation Layer

- Flash Translation Layer (FTL) in device controller
  - Remapping
  - Wear-leveling
  - Write buffering
  - Log-structured file system (later)

Example: Fusion I/O Flash Memory

- **Flash Translation Layer (FTL) in device controller**
  - Remapping
  - Wear-leveling
  - Write buffering
  - Log-structured file system (later)

- **Performance**
  - Fusion-IO Octal
  - 10TB
  - 6.7GB/s read
  - 3.9GB/s write
  - 45µs latency
Summary

- Disk is complex
- Disk real density has been on Moore’s law curve
- Need large disk blocks to achieve good throughput
- System needs to perform disk scheduling
- RAID improves reliability and high throughput at a cost
- Flash memory has emerged at low and high ends