Disk Management

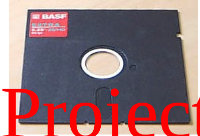
(w

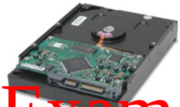
axgopala@imperial.ac.uk

Imperial College London

Disk Evolution

Capacity increases exponentially, but access speeds not so much





		Seagate
cylinders		
Tracks/cylinder	2	
Sectors/track	9	
Bytes/sector	512	
Sectors/disk	720	781,422,768
Disk capacity	360 KB	400 GB

`http://disctech.com/Seagate-ST3400832AS-SATA-Hard-Drive`

Surface divided into 20 or more **zones**

- Outer zones have more sectors per track $\rightarrow$ ensures that sectors have same physical length
- Zones hidden using virtual geometry

Physical hardware address: (cylinder, surface, sector)

- But actual geometry complicated → hide from OS

Mod
addr

-
- Makes disk management much easier
- Helps work around BIOS limitations
  - Original IBM PC BIOS → 8 GB m
  - 6 bits for sector, 4 bits for head, 14 bits for cylinder

Disk capacity statements can be confusing ☺

1 KB $= 2^{10}$ bytes $= 1024$ bytes vs 1 KB $= 10^3$ bytes $= 1000$ bytes

1 MB $=$ ... 00^2
byte

1 GB $= 2^{30}$ bytes $= 1024^3$ bytes vs 1 GB $= 1$ ... ^{9} ... ^{3}
bytes

If necessary, just make it consistent on the exam

Before a disk can be used, it must be formatted

- **Low level format**

    - Disk sector layout

- **High level format**

    - Boot block
    - Free block list
    - Root directory
    - Empty file system

Amount of cylinder skew depends on the drive geometry

Example Problem

Consider a 10,000 rpm drive with each track having 300 sectors and tr

Tim

300 se

300

$10^{-5} = 20\ \mu s$

Track to track seek time is $800\ \mu s \Rightarrow$

in one seek $= \frac{800}{20} = 40$

Hence, cylinder skew $= 40$

Typical disk

Sector size	512 bytes
Seek time (adjacent cylinder)	$<$ 1 ms
Seek time (average)	8 ms
Rot	
Tr	up to 100 MB/s

Disk Scheduling

- Minimise seek and/or latency times
- Order pending disk requests with respect to head position

Seek time $\approx 2-3$ times larger than latency time $\rightarrow$ more important to optimise

Given

b – number of bytes to be transferred

N – number of bytes per track

r – rotation speed in revolutions per second

See		
Latency time (rotational delay)	t_{late}	
Transfer time	$t_{transfer}$	$\frac{}{N \times r}$
Total access time (t_{access})		$t_{seek} + t_{latency} + t_{transfer}$

Disk Performance

Average seek time: 10ms

Rotation speed: 10,000 rpm

512 byte sectors

320 se

File s

Calc

1. read file stored as compactly as possible on di

   occupies all sectors on 8 adjacent tracks
   sectors/track = 2560 sectors)

2. read file with all sectors randomly distributed across disk

Example Problem

Answer: Disk Performance

1. Average seek $= 10$ ms

   Latency time $= 3$ ms $= \frac{1}{2 \times (\frac{10000}{60})}$

   b

Total time $= 19$ ms $+ 7 \times 9$ ms $= 82$ ms $= 0.082$ secon

2. Average seek and latency time same as abov

   Read 1 sector $= 0.01875$ ms $= \frac{}{512 \times 320 \times (\frac{10000}{60})}$

   Total $= 13.01875$ ms

Total time $= 2560 \times 13.01875$ ms $= 33.328$ seconds

First Come First Served (FCFS)

No ordering of requests → random seek patterns

- OK for lightly-loaded disks
- But poor performance for heavy loads
- Fair scheduling

Que

Shortest Seek Time First (SSTF)

Order requests according to shortest seek distance from current head position

- Discriminates against innermost/outermost tracks
- Unpredictable and unfair performance

Queue: 98, 183, 37, 122, 14, 130, 60, 67 (head starts at 53)

If, when handling request at 14, new requests arrive for 50, 70, 100 → long delay before 183 serviced

SCAN Scheduling

Choose requests which result in shortest seek time in preferred direction

- Only change direction when reaching outermost/innermost cylinder (or no further requests in preferred direction)
- Most common scheduling algorithm (also called elevator scheduling)
- Long delays for requests at extreme locations

Queu
is tow

Services requests in one direction only

- When head reaches innermost request, jump to outermost request
- Lower variance of requests on extreme tracks
- May delay requests indefinitely (though less likely)

Que

N-Step SCAN

As for SCAN, but services only requests waiting when sweep began

- Requests arriving during sweep serviced during return sweep
- Doesn't delay requests indefinitely

Queue: 98, 183, 37, 122, 14, 130, 60, 67 (head starts at 53; direction $\rightarrow$ 0); requests 80, 140 arrive when head moving outwards

I/O requests placed in request list

- One request list for each device in system
- **bio** structure: associates memory pages with requests

Bloc

-
- Driver must perform all operations in list
- Device drivers do not define read/write ope

Some devices drivers (e.g. RAID) order their own r

- Bypass kernel for request list ordering

Default: variation of SCAN algorithm

- Kernel attempts to merge requests to adjacent blocks
- But synchronous read requests may starve during large writes

Dea

-

Anticipatory scheduler: delay after read request completes

- Idea: process will issue another synchrono
  before its quantum expires

- Reduces excessive seeking behaviour
- Can lead to reduced throughput if process does not issue another read request to nearby location
  - Anticipate process behaviour from past behaviour

Problem

- CPU performance doubling every 18 months
- Disk performance has increased only 10 times since 1970

Solution

-

RAI

- Array of physical drives appearing as single v
- Stores data distributed over array of physic parallel operation (called striping)

Use redundant disk capacity to respond to disk failure

- More disks $\rightarrow$ lower mean-time-to-failure (MTTF)

Use multiple disks and spread out data

Disks can seek/transfer data concurrently

- May also balance load across disks

No re

- Mirror data across disks
- Reads can be serviced by either disk (fast)
- Writes update both disks in parallel (slower)
- Failu
- High s

Parallel access by striping at bit-level

- Use Hamming error-correcting code (ECC)
- Corrects single-bit errors (and detect double-bit errors)

Very high throughput for reads/writes

-
-

Only

- ECC disks become bottleneck
- High storage overhead

Only single parity strip used

- Parity $=$ data1 $\oplus$ data2 $\oplus$ data3 . . .
- Reconstruct missing data from parity and remaining data

Lower storage overhead than RAID Level 2

-

RAID

Level 4 (Block-level XOR)

Parity strip handled on block basis

- Each disk operates independently

Potential to service multiple reads concurrently

Pari

-

Like RAID Level 4, but distribute parity

- Most commonly used

Some potential for write concurrency

Good storage efficiency/redundancy trade-off

-

Category	Level	Description	I/O Data Transfer (R/W)	I/O Request rate (R/W)
Striping	0	Non-redundant	+/+	+/+
M				+/0
P ac				0/0
	3	Bit interleaved parity		
Independent access	4	Block interleaved parity		
	5	Block interleaved distributed parity	+/-	+/- or 0

better than single disk (+) / same (0) / worse (-)

- Use Piazza for Q & A
- Mar
- Prov

- Feedback also possible through Mentimeter (9
- If time permits, possible guest lecture on Security