HÁLÓZATI RENDSZEREK
ÉS SZOLGÁLTATÁSOK
TANSZÉK

SZÁMÍTÓGÉP ARCHITEKTÚRÁK

Háttértárak

Horváth Gábor, Belső Zoltán

BME Hálózati Rendszerek és Szolgáltatások Tanszék
ghorvath@hit.bme.hu, belso@hit.bme.hu

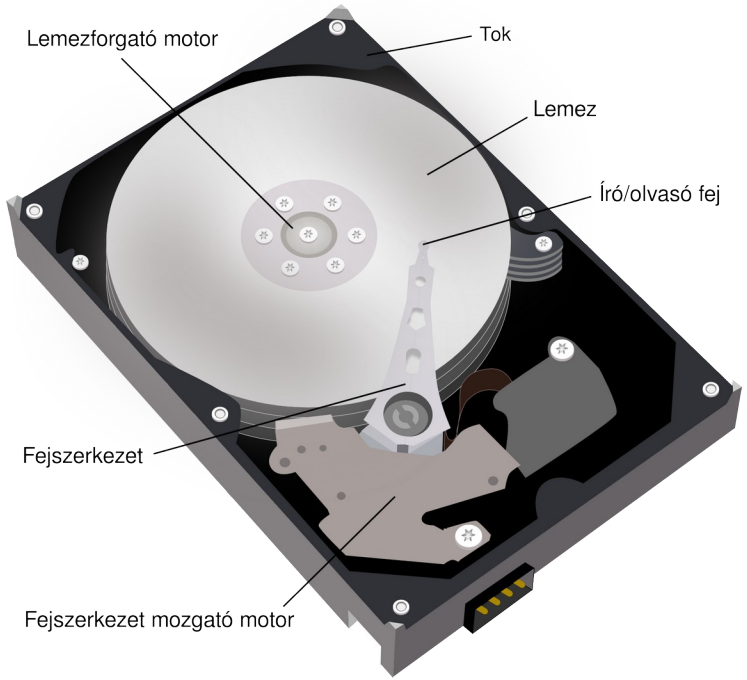
Budapest,
2021.03.10.

MŰEGYETEM 1782

Adattárolás merevlemezen (HDD)

FORGÓLEMEZES ADATTÁROLÓK

- Részei:
  - Lemez(ek)
    - 1 vagy több adathordozó felület
    - Ezen tároljuk az adatot
  - A fej
    - Olvassa/írja az adatokat

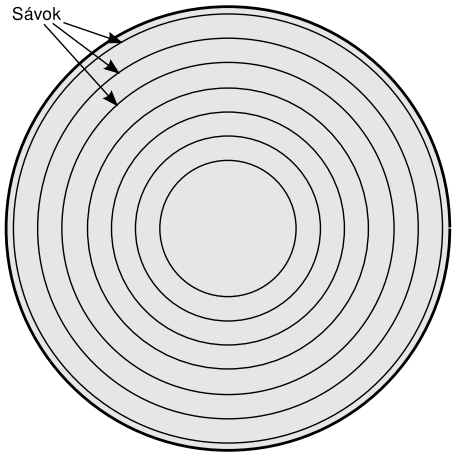


- Adatok elérése:
  1) A fejet a kívánt radiális távolságba toljuk
     - Ez a **seek**
  2) A lemezt a kívánt pozícióba forgatjuk
  3) Megtörténik az írás/olvasás
     - HDD → mágneses elven
     - CD/DVD/BR → optikai elven

AZ ADATOK SZERVEZÉSE

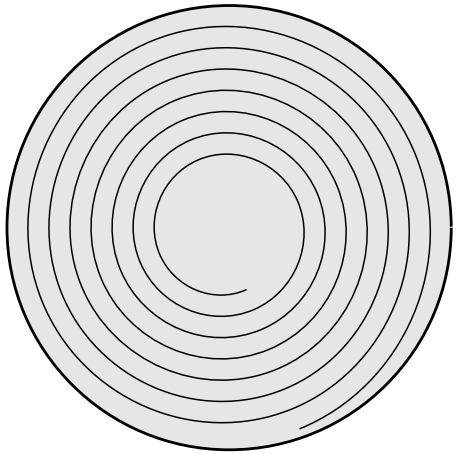
- Domináns megoldás: fix méretű blokkok
  → **szektor** (tipikusan 512 bájt)
- Hivatkozás a szektorokra: lineáris címmel (0-tól)
  - **Logical Block Addressing (LBA)**
  - Leképzés a lemezek geometriájára → a diszk dolga

Merevlemezek:



- Szektorok elhelyezkedése:
  - Merevlemezen:
    - Koncentrikus körökben
      → neve: **sáv**
    - Forgás: állandó szögsebességgel
      CAV: Constant Angular Velocity
  - Optikai lemezeken:
    - Spirálisan
    - Forgás: állandó lineáris sebességgel a spirál mentén
      CLV: Constant Linear Velocity

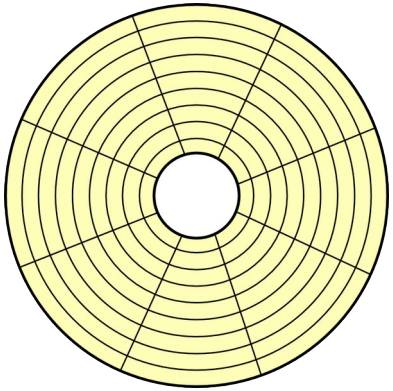
Optikai lemezek:



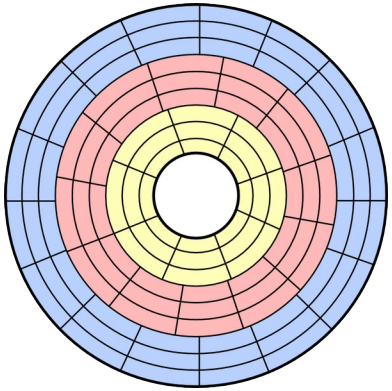
ZÓNA RENDSZERŰ ADATTÁROLÁS MEREVLEMEZEKBEN

- **Ha a sávonkénti szektorok száma fix** az egész diszkre:
  - Minden szektort ugyanannyi ideig tart elolvasni
  - A belső szektorokban nagyobb az adatsűrűség → a külsőkben pazarló
- **Ha az adatsűrűség fix** az egész diszkre:
  - Kifelé haladva minden sávban egyre több szektor lesz
  - Külső sávok felé haladva egyre gyorsabban kell olvasni a szektorokat
  - Sávonként olvasási sebességet váltani → nem megoldható
- Arany középút: nem váltunk sebességet minden sávban, csak zónánként
  → **zóna rendszerű adattárolás (ZBR, Zoned Bit Recording)**
  - Egy zónán belül minden sávban ugyanannyi szektor van
  - Egy zónán belül fix az adatleolvasási sebesség

ZBR nélkül:

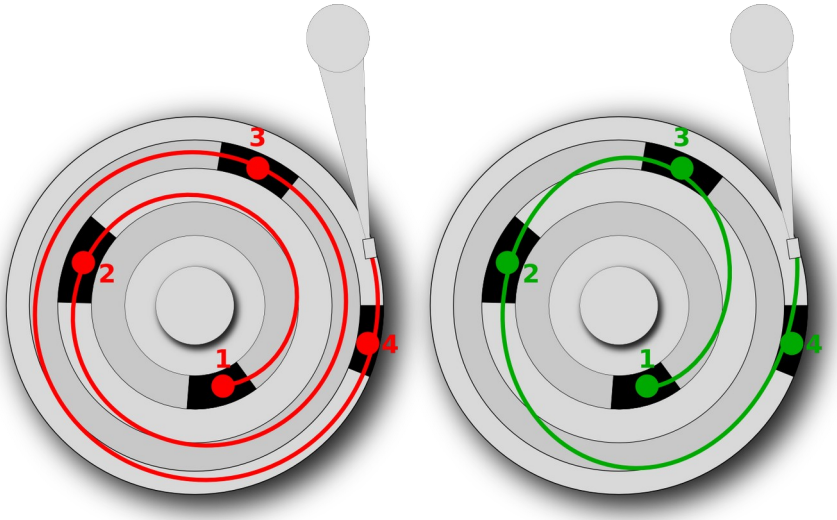


ZBR-rel:



MEREVLEMEZEK TELJESÍTMÉNYE

- Szeretnénk egy szektorsorozatot átvinni
  - A kérés először egy bufferbe kerül
    - Régen: op. rendszer menedzselte
    - Ma: diszk menedzseli → **command queueing**
      - A számára kényelmes (és gyors) sorrendben szolgálja ki őket

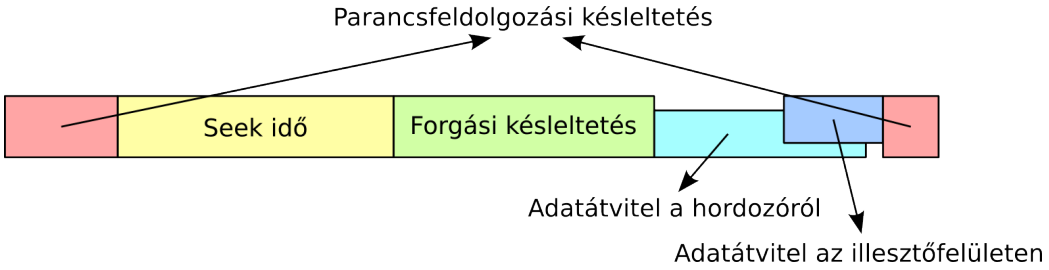


Kiszolgálás érkezési sorrendben Kiszolgálási sorrend optimalizálása

  - A bufferből kivéve a diszk kiszolgálja a kérést

MEREVLEMEZEK TELJESÍTMÉNYE

- Teljesítményjellemzők:
  - **Kiszolgálási idő**
    - 5 komponensből áll:



    - A hordozóról való leolvasás és az interfész átvitel átlapolható!
  - **Átviteli sebesség (throughput)**
    - Kétféle megadás:
      - Kérések számában mérve: IOPS: [kiszolgált kérések száma/sec]
      - Adatsebességben mérve: IOPS x átlag átvitt adatmennyiség [byte/sec]
    - Kis blokkmérettel mérve: **véletlen átviteli sebesség**
      - Seek idő, forgási késleltetés dominál
    - Nagy, folytonos blokkméretekkel mérve: **folytonos átviteli sebesség**
      - Az adatleolvasási idő dominál

1. FELADAT

- A merevlemez paraméterei:
  - 3 db 2 oldalas lemeze van
  - Adathordozó rétegenként 100.000 sáv
  - 2 zóna: 1 – 50.000-ig 2000 szektor/sáv, 50.001 – 100.000-ig 1000 szektor/sáv
  - Forgási sebesség: 10.000 fordulat/perc (RPM)
  - Átlag seek idő: 4,5 ms
  - Parancsfeldolgozási késleltetés: 0,3 ms
  - Illesztőfelület sebessége: $100 \cdot 10^6$ byte/s
  - Szektorok mérete: 500 byte
- Feladatok:
  a) Mekkora a merevlemez kapacitása bájtban mérve?
  b) Mennyi ideig tart a 75.000-es sávban egy **4.000** bájtos adatátvitel (kb. 4 kB)?
  És mennyi ideig tart egy **16.000.000** bájtos adatátvitel (kb. 16 MB)?
  (Feltéve, hogy a szektorok egymás után helyezkednek el a lemezen, adatátvitel közben újabb seek nem szükséges)
  Hasonlítsuk össze a két esetet: mely komponensek dominálnak? Vonjuk le a következtetést!
  c) Számítsuk ki az előbbi 4000 bájtos kérésméretre vonatkozó véletlen- és a 16.000.000 bájtos kérésméretre vonatkozó folytonos átviteli sebességet!

1. FELADAT MEGOLDÁSA

a) A merevlemez kapacitása:
- 3 lemez * 2 oldal * (50.000*2000 szektor + 50.000*1000 szektor) * 500 bájt = $450 \cdot 10^9$ byte

b) Átviteli idők számítása:
- Diszk körülfordulási idő: T = (60.000 ms/perc) / (10.000 fordulat/perc) = 6 ms/fordulat
- Átlagos forgási késleltetés: T/2 = 3 ms
- Egy szektor leolvasása a 75.000-es sávban: T/1000 = 0,006 ms/szektor
- Interfész átviteli idő: (500 bájt/szektor) / ($100 \cdot 10^6$ bájt/s) = $5 \cdot 10^{-6}$ s/szektor = 0,005 ms/szektor
- Seek idő: adott, 4,5 ms, parancsfeldolgozási idő: adott, 0,3 ms
- A 4.000 bájtos kérés kiszolgálási ideje (=8 szektor)
  - 0,3 + 4,5 + 3 + 8*0,006 + 0,005 = **7,853 ms**
  - Figyelembe vettük az átlapolást, az interfész átviteli időnél csak az utolsó szektorral kellett számolni!
- A 16.000.000 bájtos kérés kiszolgálási ideje (=32.000 szektor)
  - 0,3 + 4,5 + 3 + 32.000*0,006 + 0,005 = **199,805 ms**
- Következtetés: kis méretnél a mechanikai késleltetés (seek + forgás) dominál, nagynál az adatleolvasási idő

c) Átviteli sebességek kiszámítása:
- A 4.000 bájtos kérésre:
  - IOPS = (1.000 ms/sec) / (7,853 ms/kérés) = 127,34 kérés/sec
  - Bájtsebességben: (127,34 kérés/sec) * (4.000 bájt/kérés) = **509360 bájt/sec** ≈ 510 kB/sec !!!
- A 16.000.000 bájtos kérésre:
  - IOPS = (1.000 ms/sec) / (199,805 ms/kérés) = 5,005 kérés/sec
  - Bájtsebességben: (5,005 kérés/sec) * (16.000.000 bájt/kérés) = **80.080.000 bájt/sec** ≈ 80 MB/sec !!!

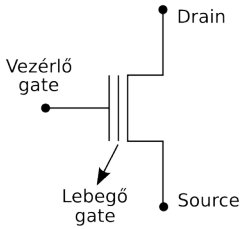
2. FELADAT

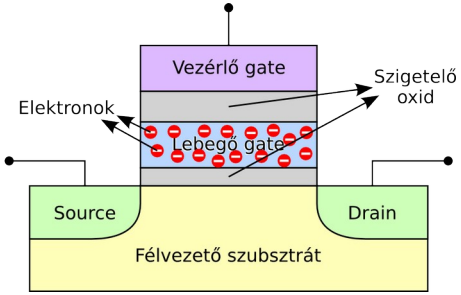
- A merevlemez paraméterei:
  - 3 db 2 oldalas lemez
  - Adathordozó rétegenként 20.000 sáv
  - Minden sávban 1000 szektor (ZBR nincs)
  - Szektorok mérete: 500 bájt
  - Átlagos seek idő: 4 ms
- Feladatok:
  a) Mekkora a merevlemez kapacitása bájtban mérve?
  b) Mérésekkel megállapítottuk, hogy az 1 véletlen szektorra vonatkozó olvasási kérések átlagos kiszolgálási ideje 10 ms.
  Ha a parancsfeldolgozási késleltetéstől és az interfészen való átviteli időtől eltekintünk, milyen gyorsan forog a lemez? (fordulat/perc-ben)
  c) Meddig tart egy szektor leolvasása az adathordozóról, ha már ott van a fej?
  d) Ha a parancsfeldolgozási idő 0,1 ms, az adatátviteli interfész sebessége pedig $50 \cdot 10^6$ bájt/s, akkor mennyi a 2.000 bájtos blokkokra vonatkozó (véletlen) adatátviteli sebesség?
  e) Mennyi az $50 \cdot 10^6$ bájtos blokkokra vonatkozó (folytonos) adatátviteli sebesség?

Félvezető alapú háttértárak (SSD)

EGY BIT TÁROLÁSA

- Alapja a lebegő gate-es tranzisztor
  - A lebegő gate-be elektronok zárhatók
    - Benne is maradnak (táp nélkül)
    - Megváltoztatják a tranzisztor viselkedését
- Adatok reprezentálása:
  - Egy tranzisztor 1 bitet tárol (egyelőre)
  - Vannak elektronok: **0**, nincsenek: **1**
- Váltás 1-ből 0-ba: **programozás**
  - Elektronokat tömünk a lebegő gate-be
- Váltás 0-ból 1-be: **törlés**
  - Kiszivattyúzzuk az elektronokat a lebegő gate-ből
- Mindkét művelet nagy stresszt jelent a tranzisztornak!
- **A programozás és törlés mellékhatása:**
  - A lebegő gate-be és onnan ki mászkáló elektronok beragadhatnak a szigetelőbe (örökre)
  - A szigetelőréteg egyre kevésbé fog szigetelni
  - **Öregedés!**



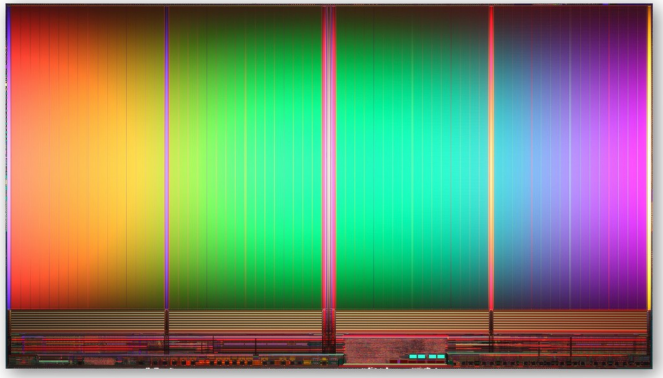


NÉHÁNY BIT TÁROLÁSA

- Eddig 2 töltöttségi állapot volt: nincs töltés → 1, van → 0
- Miért ne használnánk több töltöttségi szintet?
  - ***N* bit tárolásához 2^N töltöttségi szintet kell megkülönböztetni**
    - SLC flash: 1 tranzisztor → 1 bit
    - MLC flash: 1 tranzisztor → több (tipikusan 2) bit
  - Újabban a gyártók máshogy értelmezik az MLC-t:
    - SLC: N=1 – leggyorsabb, legdrágább
    - MLC: N=2
    - TLC: N=3
    - QLC: N=4 – leglassabb, legolcsóbb
- Minél nagyobb N, annál gyorsabban öregszik! Elviselt törlési ciklusok száma:
  - SLC: kb. 100.000
  - MLC: kb. 10.000
  - TLC: kb. 1.000 – 3.000
  - QLC: kb. 100 – 750 (!)

SOK BIT TÁROLÁSA

- A tranzisztorokat 2D rácsba rendezzük
- De hogyan?
- **NOR flash:**
  - Byte szinten címezhető
  - Utasításmemóriaként használható (BIOS, firmware, stb.)
- **NAND flash:**
  - Kezdetektől fogva háttértárnak szánták
  - Olvasás és írás egységei: **lapok**
    - Lap: a tárolómező egy sorának bitjei
  - Az egész tárolómező neve: **blokk**
  - **Törölni csak *teljes blokkot* lehet, lapokat egyesével *nem***
  - Tárolási hierarchia:
    - 1 lebegő gate-es tranzisztor: 1 – 3 bitet tárol
    - 1 lap: 512 byte – 8 kB
    - 1 blokk: 128 – 256 lap
    - 1 tárolósík: 1024 blokk
    - 1 szilíciumlapka: 1 – 4 tárolósík
    - 1 tok: 1 – 4 szilíciumlapka



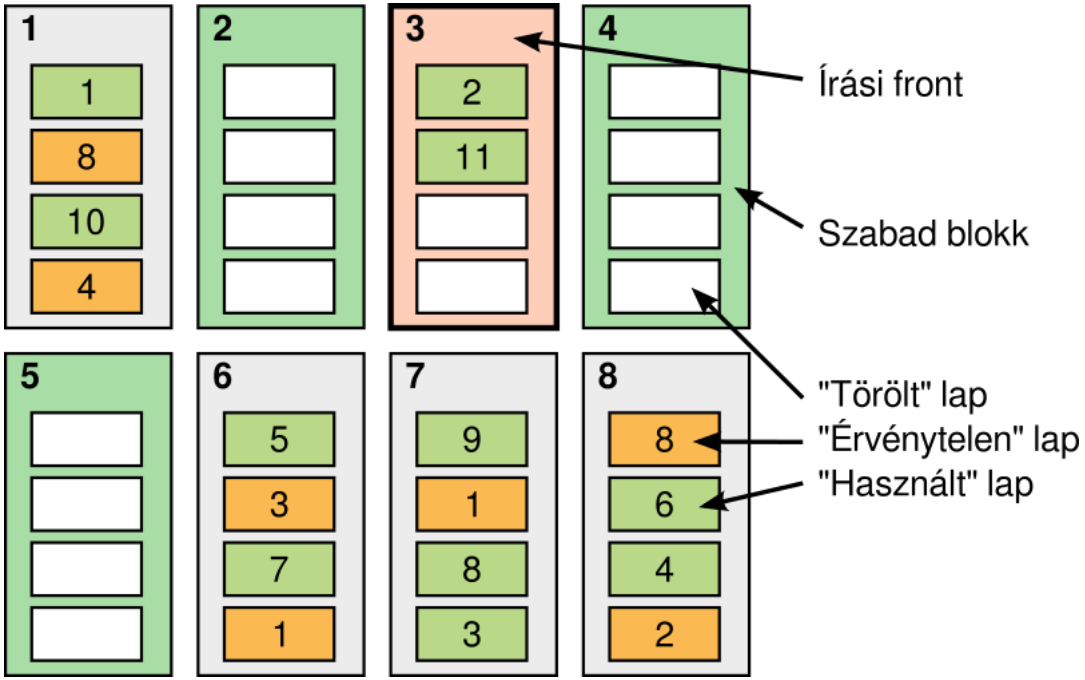
Az Intel 8 GB-os 2-bites MLC NAND flash lapkája

SZEKTORSZINTŰ MŰVELETEK

- **Olvasás**
  - LBA címéből → fizikai elhelyezkedés, kiolvasás, kész.
- **Írás**
  - HDD-nél: Ha egy fájlt többször mentünk, HDD felülírja a régit
  - SSD-nél: Nem tud lapot felülírni!!!
  - Minden új változatot új, szűz helyre tesz, a régit érvénytelennek jelöli
  - Új szűz lapok generálása: ha egy teljes blokkon már nincs hasznos adat, törli (akár 2 MB!)
  - Minden laphoz állapotot tárol:
    - „Használt”
    - „Érvénytelen”
    - „Törölt”

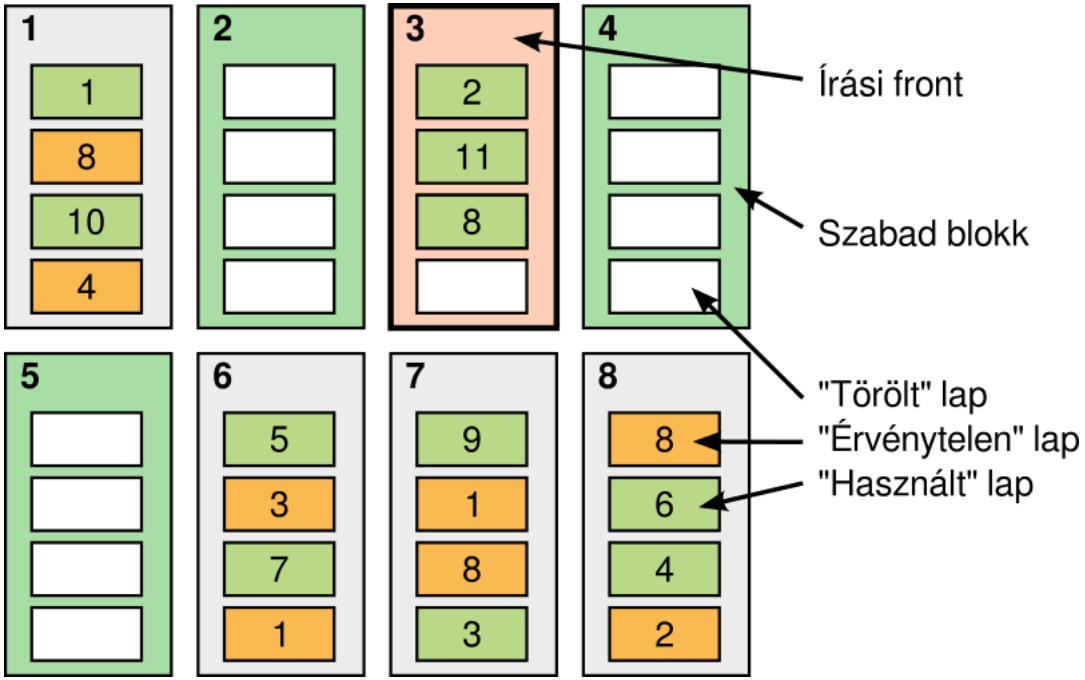
LAPÍRÁS MENEDZSMENT

- Minden írási művelet az **írási front**ra vonatkozik
- Ha az írási front megtelik, újat választ a törölt blokkok közül
- Példa: írási kérések a 8, 12, 1 LBA címekre



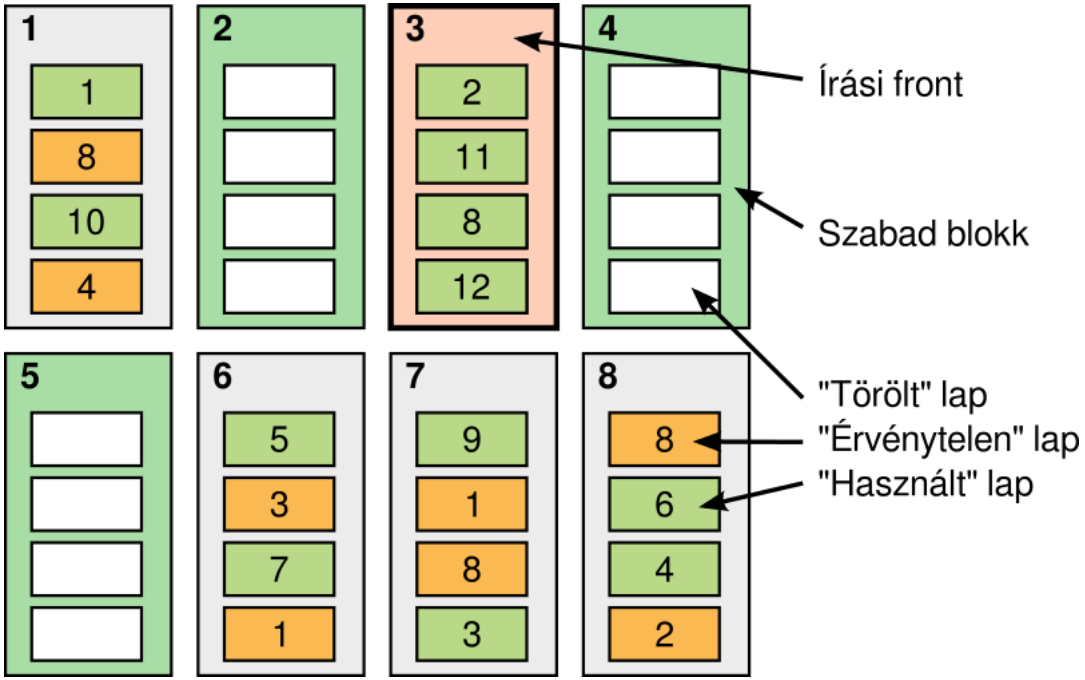
LAPÍRÁS MENEDZSMENT

- Minden írási művelet az **írási front**ra vonatkozik
- Ha az írási front megtelik, újat választ a törölt blokkok közül
- Példa: írási kérések a 8, 12, 1 LBA címekre



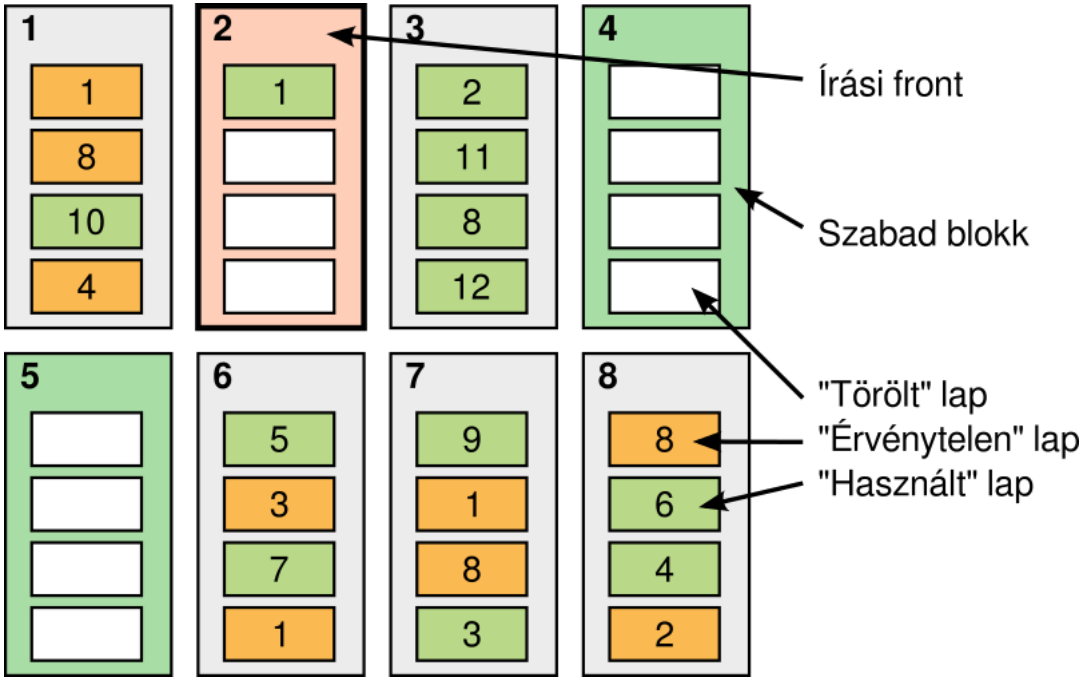
LAPÍRÁS MENEDZSMENT

- Minden írási művelet az **írási front**ra vonatkozik
- Ha az írási front megtelik, újat választ a törölt blokkok közül
- Példa: írási kérések a 8, 12, 1 LBA címekre



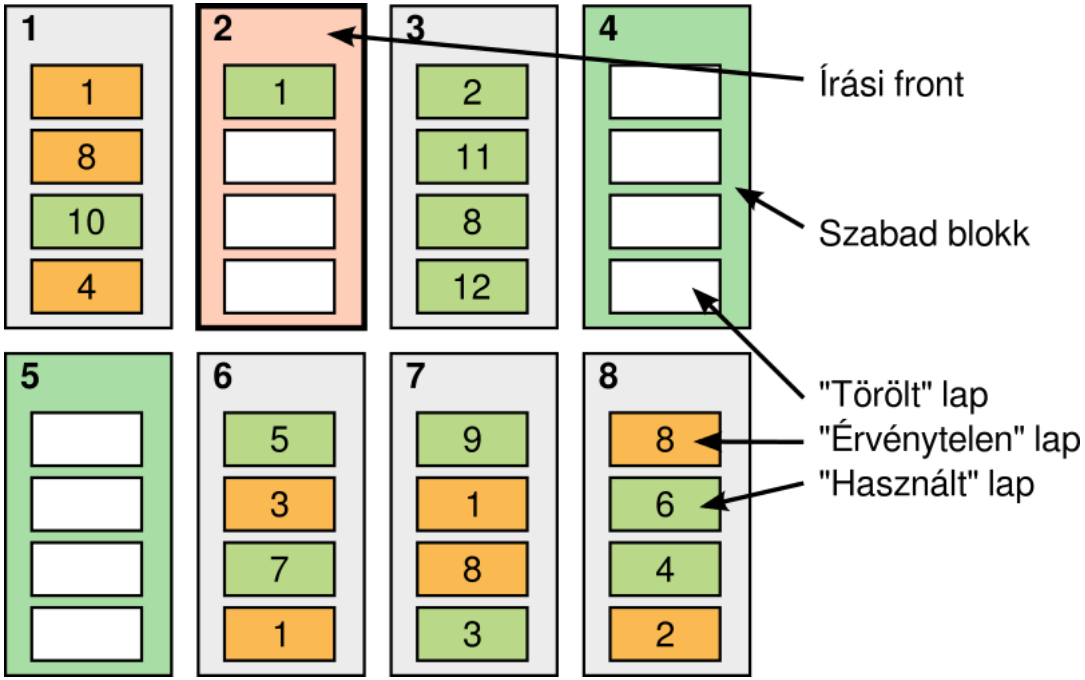
LAPÍRÁS MENEDZSMENT

- Minden írási művelet az **írási front**ra vonatkozik
- Ha az írási front megtelik, újat választ a törölt blokkok közül
- Példa: írási kérések a 8, 12, 1 LBA címekre



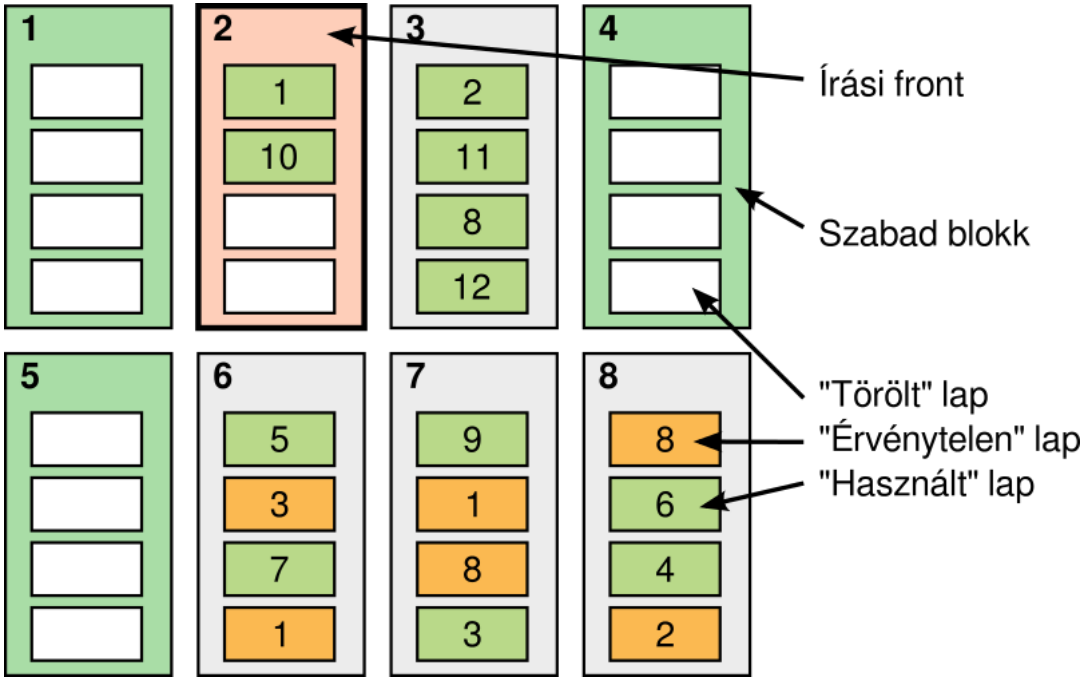
LAPÍRÁS MENEDZSMENT

- Ha fogynak törölt blokkok → **szemétgyűjtés**
  - Kinéz egy blokkot
  - „Használt” lapjait → írási frontra
  - Törli az egész blokkot
- Példa:



LAPÍRÁS MENEDZSMENT

- Ha fogynak törölt blokkok → **szemétgyűjtés**
  - Kinéz egy blokkot
  - „Használt” lapjait → írási frontra
  - Törli az egész blokkot
- Példa:



LAPÍRÁS MENEDZSMENT

- A működés következménye:
  - **Az SSD idővel lassul!**
  - Mert amikor új, nem kell szemetet gyűjtenie
- Korszerű SSD-k megoldása:
  - *Ráérő idejükben szemetet gyűjtögetnek*
    - Csínján kell bánni a lapok rendezgetésével – koptatja az SSD-t!
    - Csak akkor megy, ha van ráérő idő. Ha nincs (pl. terhelt szerver), drasztikus a lassulás.
  - *Kopás-kiegyenlítés*
    - Minden blokk egyformán kopjon (statikus és változékony egyaránt)
  - *Adattömörítés*
    - Ha az adatokat tömörítjük, kevesebbet kell írni (lassul az öregedés)
  - *Több írási kérés bevárása*
    - Hátha jön 1-2 kérés, ami ugyanarra a lapra vonatkozik
    - Ezek egyetlen írási művelettel végrehajthatók
  - Stb.

3. FELADAT

- 8 blokkból álló SSD pillanatnyi állapota:

1 #18	
T	
T	
T	
T	

2 #2	
H	1
É	11
É	9
H	3

3 #17	
T	
T	
T	
T	

4 #8	
H	13
H	7
É	8
É	3

5 #7	
É	5
É	6
H	6
H	9

6 #18	
H	11
H	8
H	4
T	

7 #3	
T	
T	
T	
T	

8 #9	
É	1
H	5
H	10
É	3

a) Hogyan változik az SSD állapota, ha sorban egymás után az 5-ös, a 13-mas és a 2-es LBA címekre érkezik írási kérés? Ha új írási frontra van szükség, az SSD válassza azt, amelyik a lehető legegyenletesebb kopáshoz vezet!

b) Hogyan változik az SSD állapota, ha a kiindulási állapotban **bekapcsol a szemétgyűjtő algoritmus**, és meg sem áll, amíg a törölt blokkok listája eggyel nem nő (vagyis négy törölt blokk nem lesz)? (A szemétgyűjtő válassza mindig a legtöbb érvénytelen lapot tartalmazó blokkot, ha több ilyen is van, akkor azon közül a legkevésbé kopottat.)

HÁLÓZATI RENDSZEREK
ÉS SZOLGÁLTATÁSOK
TANSZÉK