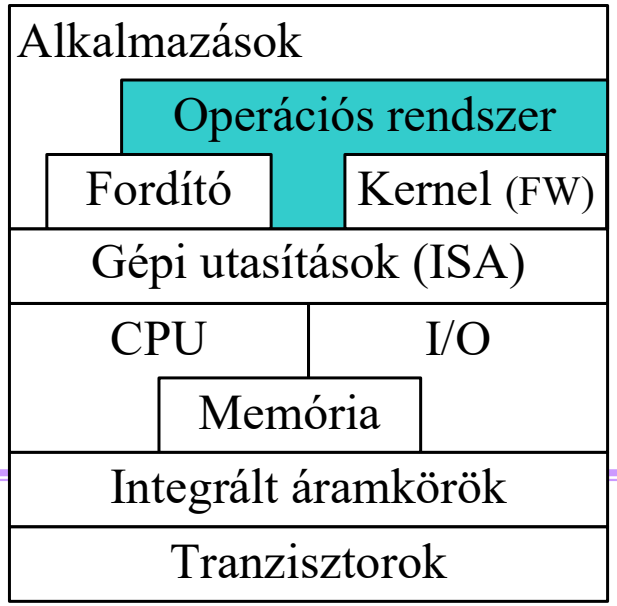
INFORMATIKA I.

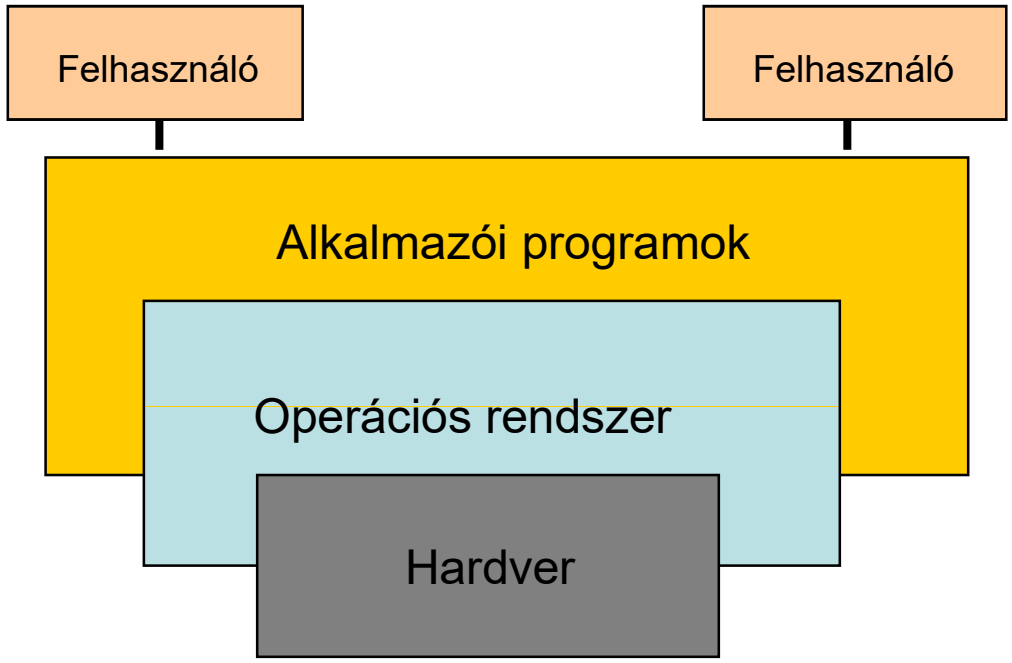
BMEVIIIAB04

Operációs rendszerek



Operációs rendszerek

Számítógép rendszer



Operációs rendszer
kapcsolat a hardver és a felhasználó között

Cél
Hatékony hardver kihasználás
A felhasználó kényelme

Multiprogramozás – Megoldandó feladatok

Melyik JOB fusson

CPU ütemezés

Egyszerre több program lehet a memóriában

Memória gazdálkodás

A periféria használatot koordinálni kell

Periféria kezelés, ütemezés

Együttfutási problémák kezelése

Szinkronizálás

Üközések, patthelyzet

Holtpont kezelés

Védelmi kérdések

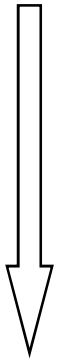
JOB ↔ JOB
JOB → operációs rendszer

Memória gazdálkodás

Eddig: A program futtatható, ha teljesen bent van a memóriában

Egypartíciós rendszer
Többpartíciós rendszer (fix/rugalmas partíciók, lapszervezés, szegmentálás)
Fragmentáció (külső, belső)

Működhetnek a programok, ha nincsenek teljesen a memóriában?



- Vannak ritkán futó programrészek
- A statikus adatszerkezetek túlméretezettek
- A program különböző részei időben elkülönülten működnek

  (Overlay → a program írójának kell kezelnie)
- Az összetartozó utasítások térben és időben is közel esnek egymáshoz (**lokalitás**)

A futtatható program méretét nem korlátozza a fizikai memória mérete

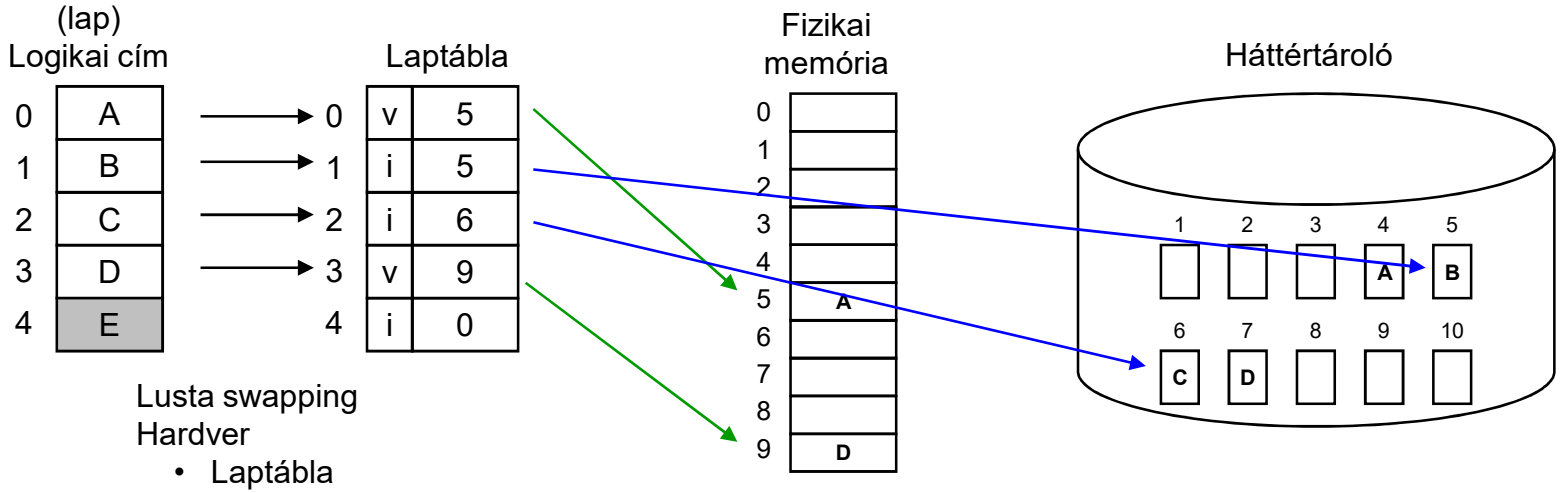
Növelhető a multiprogramozottság foka

A kisebb részek betöltése/mentése kevesebb időt vesz igénybe

Virtuális memória

A program futhat akkor is ha nincs bent teljesen a memóriában
→ laptechnikát alkalmazunk

Laptábla
< v, keret > - fizikai memória keret
< i, I/O cím > - pozíció a háttértáron
< i, 0 > - címhatáron kívül



Lusta swapping
Hardver
- Laptábla
- Háttértár
- Újraindítási lehetőség

Virtuális memória

Mi történik akkor, ha egy utasítás végrehajtásához szükséges lap(ok) nincs(enek) bent a memóriában?

→ Eltérülés (trap) – utasítás közben is felléphet

Mit tehetünk?

- Az utasítás folytatható
  - A processzor belső állapotát is tudni kell menteni
  - A belső állapotot tudni kell visszaállítani
- Az utasítást újrafuttatjuk
  - Mellékhatások

Virtuális memória

Hány lapra van szükség egy utasítás végrehajtásához

Utasítás folytatás

Elvileg egyetlen lap elegendő

Utasítás újrafuttatás

Minden szükséges lapnak rendelkezésre kell állnia

- RISC processzor
  - Műveleten belül → csak az utasítást kell elérni: 1 lap
  - Load/store művelet → ha az adat laphatáron van: 2 lap
- CISC processzor
  - Utasítás – 2 lap
  - Operandus
    - 1 című gép – 2 lap
    - 2 című gép – 4 lap
    - Bonyolult címzési mód – több lap

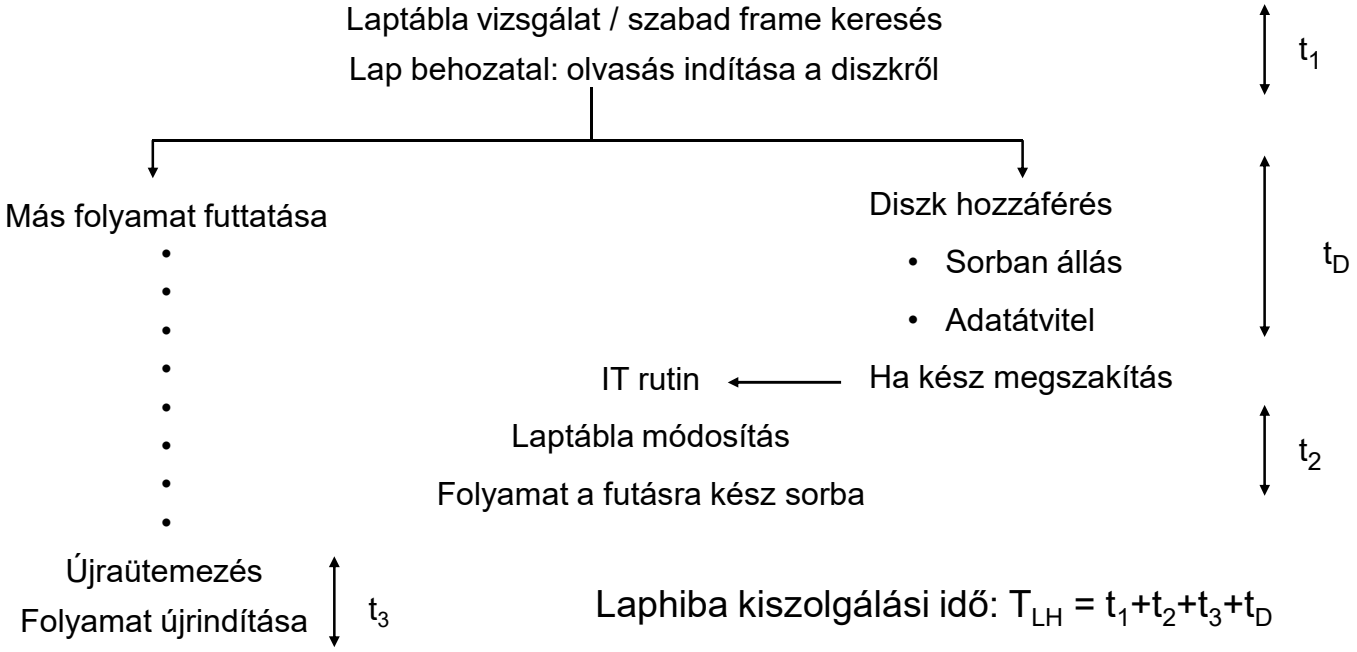
Virtuális memória - Laphiba

Események, tevékenységek

- TRAP
- A regiszterek és a folyamat állapotának elmentése
- Laphiba TRAP felismerése
- Ellenőrzés (érvényesség, a szükséges lap helyének meghatározása a diszken)
- A lap beolvasása egy szabad keretbe
  - Várakozás a diszk sorában
  - Diszk hozzáférés
  - Adatátvitel elkezdése
- Másik folyamat elindítása
- IT (I/O művelet befejeződött)
- A regiszterek és folyamat állapotának elmentése
- I/O IT felismerése
- Laptábla bejegyzés módosítása (érvényes)
- Folyamat áthelyezése a futásra kész sorba, várakozás a futó állapotra
- A regiszterek és a folyamat állapotának visszatöltése és a visszatérés a megszakított utasítás végrehajtásához

Virtuális memória

Laphiba (PAGE-FAULT) történt



Laphiba kiszolgálási idő: $T_{LH} = t_1+t_2+t_3+t_D$

Lapcsere overhead: $t_1 + t_2 + t_3$

Ha két laphiba között eltelt idő: T_{lh}

$$\text{Overhead} = \frac{t_1+t_2+t_3}{T_{lh}} \cdot 100\ [\%]$$

Virtuális memória

Mikor marad a rendszer kiegyensúlyozott?

Mindig kell futásra kész folyamat

$\rightarrow T_{lh} \geq t_D$, különben a CPU várakozik

Laphiba gyakoriság: p = laphibát okozó memória hozzáférés / összes memória hozzáférés

$0 \leq p \leq 1$

Effektív hozzáférési idő $T_{eff} = (1-p) * t_m + p * T_{LH}$

$t_m = 100$ ns $= 10^2$ ns

$T_{LH} = 10$ ms $= 10^7$ ns

$p = 10^{-5}$ (1/100000)

$1-p \approx 1 \rightarrow T_{eff} \approx t_m + p*T_{LH}$

p:	10^{-5}	10^{-6}	10^{-7}	($p=10^{-3}$)
T_{eff}:	200 ns	110 ns	101 ns	(10,1µs)

Virtuális memória

Mi történjen akkor, ha be kell hozni egy lapot, de nincs szabad keret?

- Abortáljuk a folyamatot
- Egy folyamatot felfüggesztünk
  Csökkentjük a multiprogramozottság fokát
- Egy lapot kisöprünk és a helyére hozzuk a szükségeset
  Lapcsere kérdései
  1) Melyik lapot hozzuk be
     FETCH staregy
  2) Hogyan válasszunk áldozatot
     Replacement strategy
  3) Melyik folyamat számára hány lapot biztosítsunk
     Allocation strategy

Virtuális memória - FETCH stratégiák

- **Igény szerinti lapozás (Demand paging)**
  - A választás egyszerű
  - Csak a biztosan szükséges lapok vannak a memóriában
  - Egy új lap biztos laphibát okoz
- **Előre tekintő lapozás (Anticipatory paging)**
  - Megpróbáljuk kitalálni melyik lapra lesz szükség, és üres időben előre behozzuk
    - Ha jól jósolunk: csökken a laphibák száma
    - Ha rosszul jósolunk: tele leszünk felesleges lapokkal
- **Clusterezés (Clustering)**
  - Ha egy lap kell, a következő lapo(ka)t is behozzuk

Ha a fizikai memória mérete nagy → előre tekintő lapozás

Virtuális memória - Replacement stratégiák (lapcsere algoritmusok)

Ha nincs szabad keret → áldozat keresése

- Az új lappal felülírjuk az áldozatot
- Lehet, hogy felülírás előtt ki kell írni a diszkre

Milyen a jó áldozat?

- Nem használják
  - → hiánya nem okoz laphibát
- Nem módosult
  - → a diszken lévő példánya megegyezik a memóriaképpel
  - → nincs szükség a kiírásra

Hardver támogatás szükséges

Virtuális memória - Replacement stratégiák (lapcsere algoritmusok)

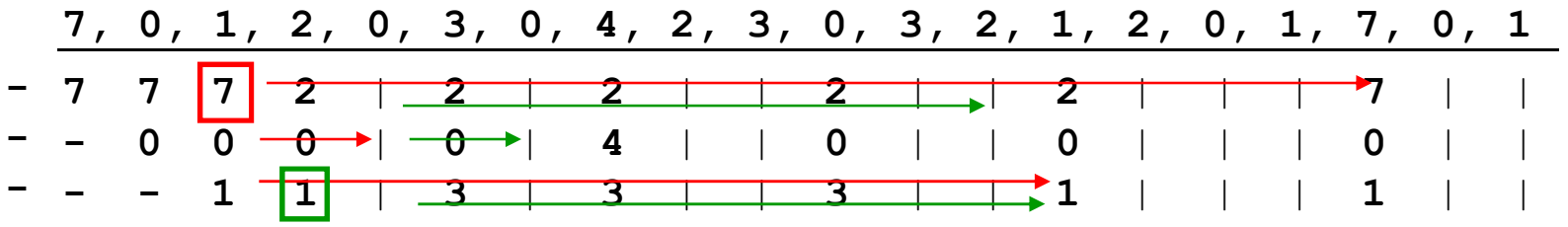
Cél: a laphiba gyakoriság minimalizálása

Lapcsere algoritmusok összehasonlítása

Egy adott hivatkozás-sorozatra vizsgáljuk hány laphiba jutott

Keretszám: 3
Sorozat: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

- **Optimális megoldás**



Laphiba: 9

Virtuális memória - Replacement stratégiák (lapcsere algoritmusok)

❑ FIFO algoritmus

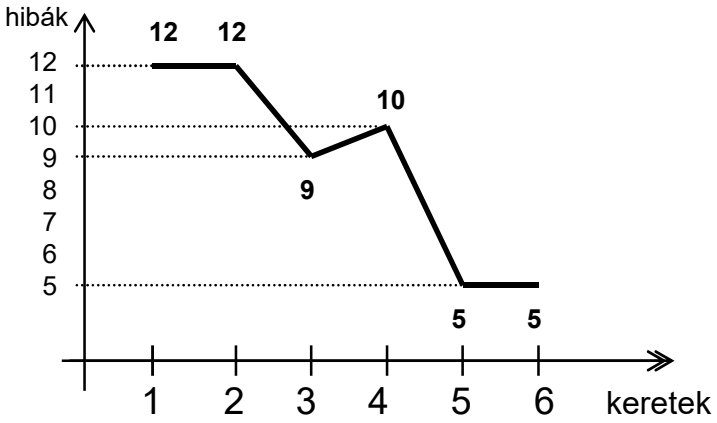
7,	0,	1,	2,	0,	3,	0,	4,	2,	3,	0,	3,	2,	1,	2,	0,	1,	7,	0,	1
-	7	7	7	2	2	2	4	4	4	0	0	0	7	7	7				
-	-	0	0	0	3	3	3	2	2	2	1	1	1	0	0				
-	-	-	1	1	1	0	0	0	3	3	3	2	2	2	1				

Laphiba: 15

Ha több keretet adunk csökken a laphibák száma?

Béládi anomália

Sorozat: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5



Keret: 3

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	4	4	4	5	|	|	5	5	|
-	2	2	2	1	1	1	|	|	3	3	|
-	-	3	3	3	2	2	|	|	2	4	|

Keret: 4

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1	|	|	5	5	5	5	4	4
-	2	2	2	|	|	2	1	1	1	1	5
-	-	3	3	|	|	3	3	2	2	2	2
-	-	-	4	|	|	4	4	4	3	3	3

Virtuális memória - Replacement stratégiák (lapcsere algoritmusok)

❑ Least Recently Used (LRU) algoritmus

	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
-	7	7	7	2	2	2	2	4	4	4	0	0	0	1	1	1	1	1	1	1
-	-	0	0	0	0	0	0	0	0	3	3	3	3	3	3	0	0	0	0	0
-	-	-	1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	7	7	7

Laphiba: 12

Implementációk

- Számláló
- Stack

LRU közelítés

Referencia bit: hozzáféréskor a hardver billenti

Virtuális memória - Replacement stratégiák (lapcsere algoritmusok)

❑ Second Chance (Második esély) algoritmus

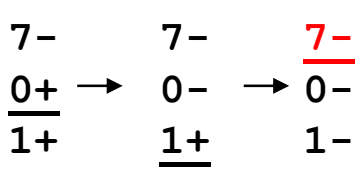
FIFO + referencia bit

Ha a referencia bit 1, töröljük és visszatesszük a FIFO végére (második esély)

Implementálás: FIFO helyett körpuffer (clock algoritmus)

7,	0,	1,	2,	0,	3,	0,	4,	2,	3,	0,	3,	2,	1,	2,	0,	1,	7,	0,	1
-	7+	7+	7+	2+	2+	2+	2+	4+	4+	4+	4-	3+	3+	3-	3-	0+	0+	0-	0+	0+
-	-	0+	0+	0-	0+	0-	0+	0-	2+	2+	2-	2-	2+	1+	1+	1+	1+	7+	7+	7+
-	-	-	1+	1-	1-	3+	3+	3-	3-	3+	0+	0+	0+	0-	2+	2+	2+	2-	2-	1+

Laphiba: 14



Virtuális memória - Replacement stratégiák (lapcsere algoritmusok)

- **Least Frequently Used algoritmus**

  Számoljuk a hivatkozásokat

  Az algoritmus nem felejt

  → Ha sokat használtuk - bent ragad

  → A számlálót periodikusan jobbra léptetjük

  → Ha most töltöttük be - áldozat lehet

  → Első használatig befagyasztjuk (page locking)

- **Most Frequently Used algoritmus**

Virtuális memória - Replacement stratégiák (lapcsere algoritmusok)

- **Lap osztályozás**

R: hivatkozott
M: módosított

Prioritás	R	M	
0	0	0	nem használt, nem módosított
1	0	1	nem használt, módosított
2	1	0	használt, nem módosított
3	1	1	használt, módosított

A legkisebb prioritásúak közül választunk

- FIFO
- véletlenszerűen

Az R bitet időnként töröljük

Virtuális memória - Allokációs stratégiák

Gazdálkodás a fizikai tárral

- Lokális lapcsere
- Globális lapcsere
- Paging daemon
- Szabad POOL

Minimum hány lapra van szükség

A hardver határozza meg

Virtuális memória - Allokációs stratégiák

Allokációs algoritmusok

- Egyenlő eloszlás
- Arányos eloszlás

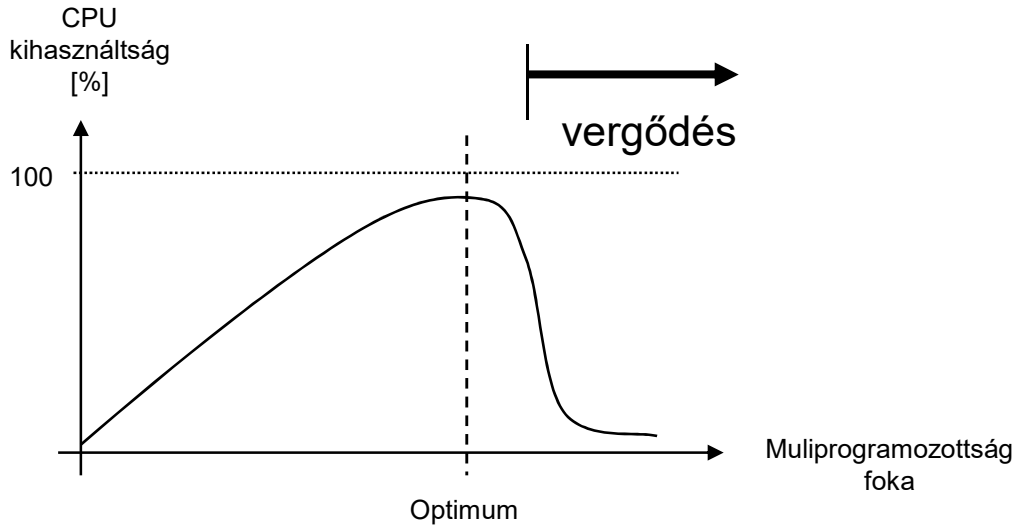
Sok laphiba

A folyamatok az I/O sorban várakoznak

CPU tétlenség

A laphibák által okozott teljesítmény csökkenés: TRASHING (vergődés)

Virtuális memória - Vergődés



t_{LH} = laphiba kiszolgálási idő

t_{lh} = két laphiba kötött eltelt idő

A vergődés határa: $t_{LH} = t_{lh}$

t_m = memória hozzáférési idő

p = laphiba gyakoriság (t_m / t_{lh})

$T_{eff} = (1 - p) * t_m + p * t_{LH} \approx 2 * t_m$

Virtuális memória - Allokációs stratégiák

Folyamatonként mérhető:

Két laphiba közötti futási idő (T_f)

Átlagos laphiba kiszolgálási idő (T_k)

Ha minden folyamatra teljesül $T_f > T_k$ → a rendszer egyensúlyban van

→ Minden folyamat annyi keretet kapjon, ahányra hivatkozik a laphiba kiszolgálás átlagos ideje alatt

Program lokalitás

Gond, ha az adott szakaszban a lokalitáshoz nincs elég keret

WORKING SET (munkahalmaz): ws

Az utolsó Δ számú hivatkozásban előforduló lapok száma

Mekkora legyen Δ?

Kicsi: nem fedi le a lokalitást

Nagy: több lokalitást is lefed

∞ : a teljes programot lefedi

Virtuális memória - Allokációs stratégiák

D = Σ ws : a teljes igényelt lapszám

K: a keretek maximális száma (rendelkezésre álló fizikai memória mérete/keret mérete)

Ha **D** > **K** → vergődés

Csökkenteni kell a multiprogramozottság fokát (folyamat felfüggesztés)

A várható munkahalmazt célszerű előre behozni → prepaging

Honnan vegyük a szükséges keretet

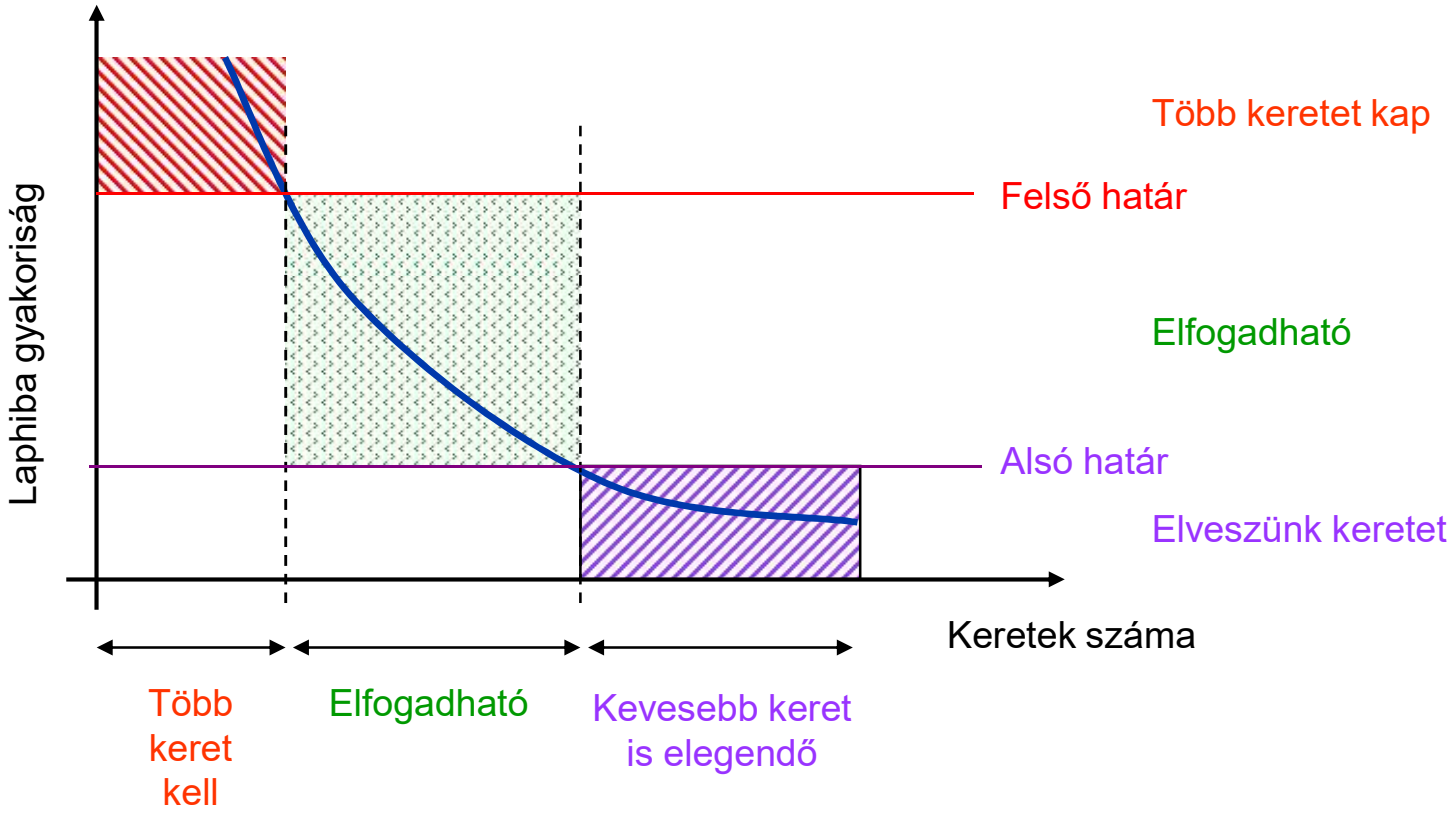
- Globális lapcsere
- Lokális lapcsere

Dinamikus lokális tárgazdálkodás

- A munkahalmazt mérjük – nem egyszerű
- Laphiba gyakoriság (laphibák között eltelt idő) mérése
  - Ha a gyakoriság nő: keretet kap
  - Ha a gyakoriság csökken: elveszünk tőle keretet

Virtuális memória - Allokációs stratégiák

Rendszeregyensúly biztosítása laphiba gyakoriság alapján



Virtuális memória

I/O kölcsönhatás

- I/O kérés folyamatban, a folyamatot felfüggesztik
- Az I/O puffer nem cserélhető
  - A puffer a az operációs rendszer területén
  - A keret zárolható

Keret (lap) mérete

- Hardver kérdés
- Nagyobb méret
  - kevesebb lap, kisebb laptábla
  - az adatátviteli idő csökken
  - a belső fragmentáció nő
  - kevésbé igazodik a lokalitáshoz
- A háttértár blokk méretéhez igazodás