

Memória és perifériák virtualizációja

Kovács Ákos

Forrás, BME-VIK

Virtualizációs technológiák

<https://www.vik.bme.hu/kepzes/targyak/VIMIAV8g/>

Emlékeztető: A három virtualizációs lehetőség

- **Virtualizáció** – az utasításokat (egy részüket) változatlanul hagyja végrehajtani, csak a problémás privilegizáltakkal kell valamit kezdeni
 - **Szoftveres virtualizáció** (Trap & emulate + bináris fordítás)
 - **Paravirtualizáció** (módosítjuk a vendég OS forrását)
 - **Hardveres virtualizáció** (Trap & emulate, teljesen hardveres támogatással)

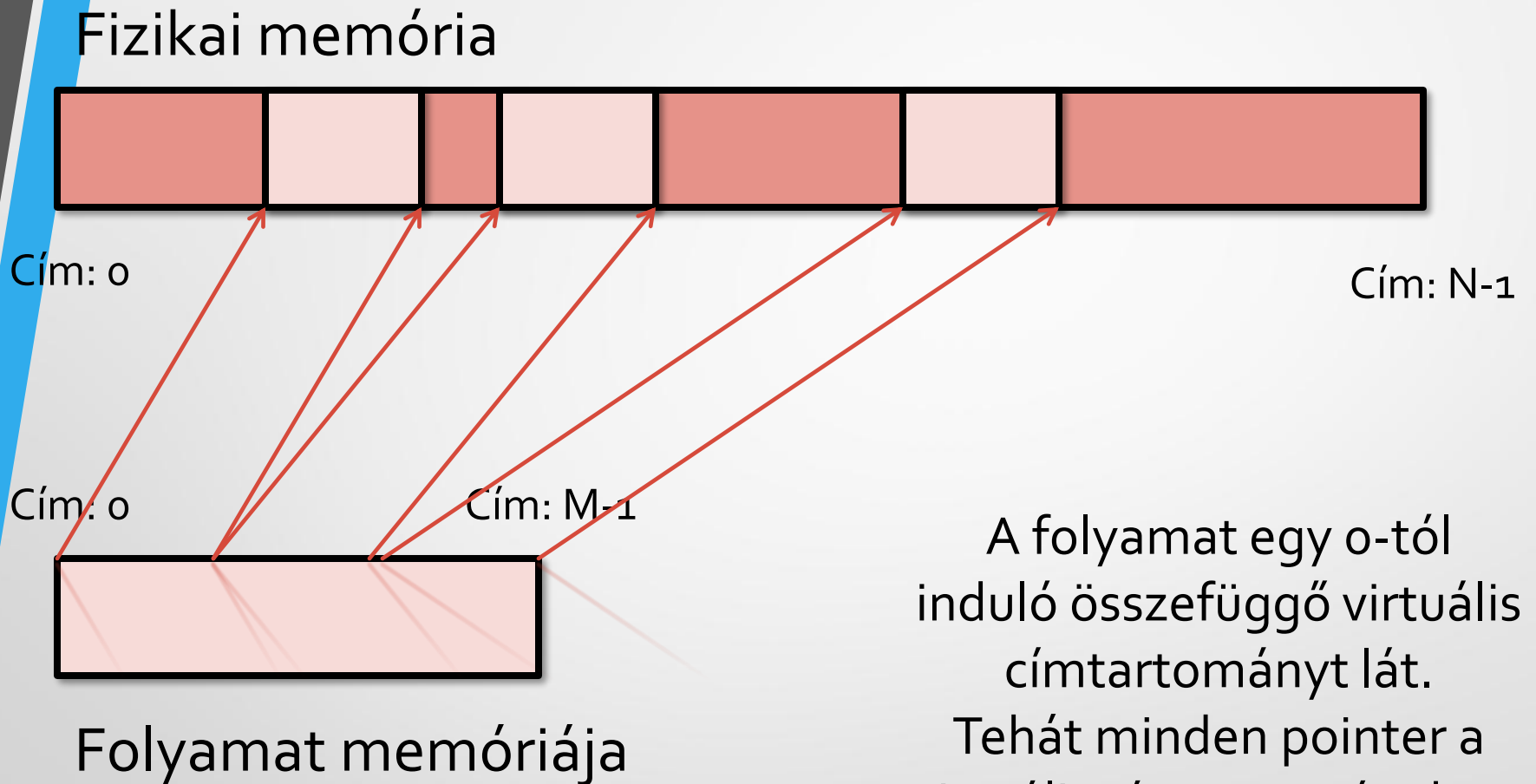
Tartalom

- Előző rész tartalmából:
 - CPU virtualizáció
 - A három alap virtualizációs megközelítés
- Memória virtualizáció
 - **Virtuális memória az operációs rendszerekben**
 - Virtuális memória a platform virtualizációban
 - Virtuális memóriakezelés speciális képességei: megosztás, késleltetett allokáció, memória-ballon
- Perifériák virtualizációja
 - Perifériák programozói felülete általában
 - Periféria virtualizációs architektúrák

Virtuális memóriakezelés

- Modern CPU-k tartalmaznak memóriakezelő egységet (MMU – *memory management unit*)
 - Feladata „virtuális” memóriacímeket leképezni „fizikaira”
 - Mi is az a virtuális memóriacím? Hol használható?
 - CPU felhasználói (pl. ring 1-3) módjaiban virtuális címekkel dolgozik (nem feltétlenül, de a modern OS-eknél ez igaz)
 - A folyamatok nem a fizikai memóriacímeket látják
 - Cél: áthelyezhető legyen az oprendszer felett futó alkalmazások kódja, ne csak fix bedrótozott helyen tudjon futni (akár szoftveresen is megoldható lenne...)
 - Cél2: eközben a teljesítmény ne romoljon számottevően (ehhez már hardver támogatás is kell)

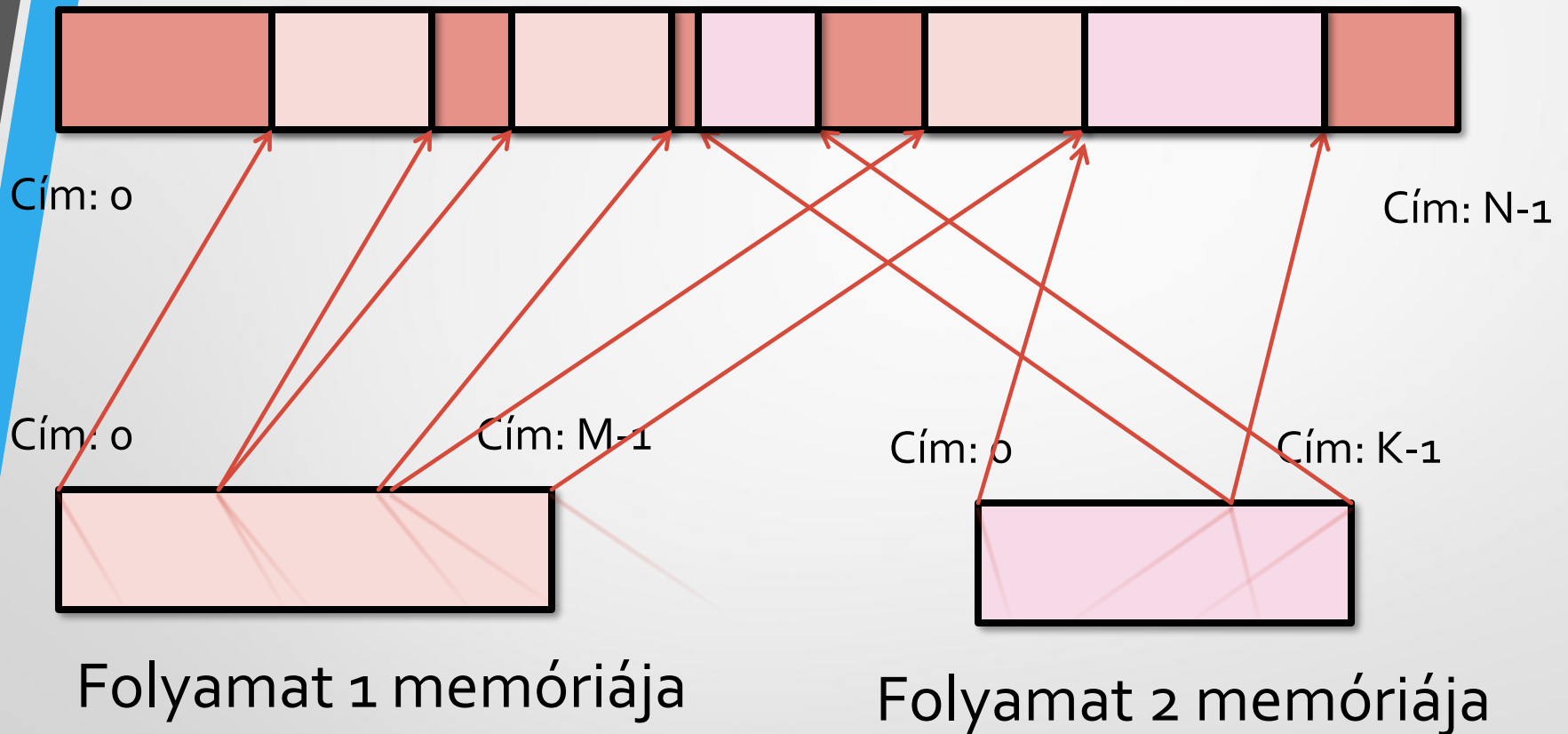
Virtuális memóriakezelés



A folyamat egy 0-tól induló összefüggő virtuális címtartományt lát. Tehát minden pointer a virtuális címtartományban értelmezett.

Virtuális memóriakezelés

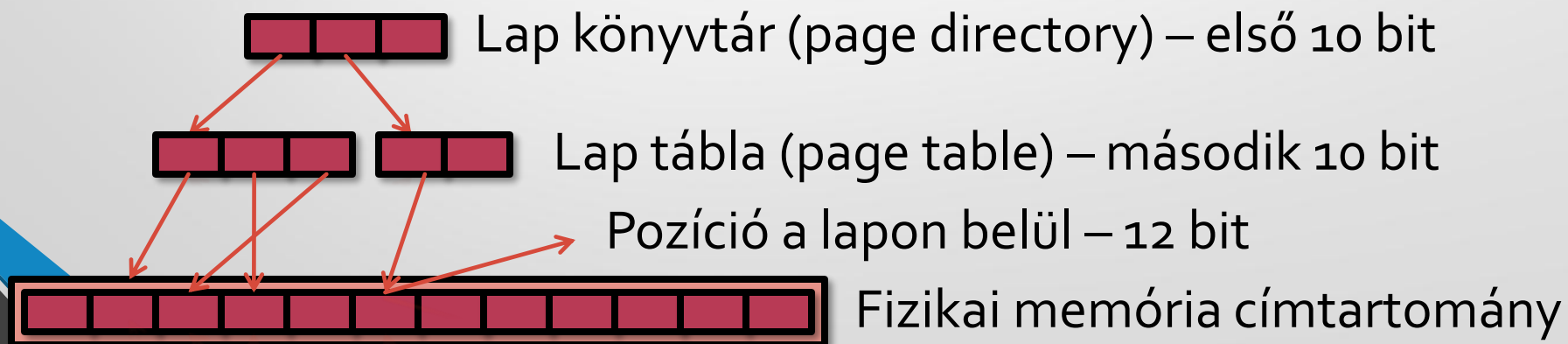
Fizikai memória



Ez minden folyamatra igaz

Virtuális memória megvalósítása lapokkal

- Memória **lapok** (*pages*)
 - Virtuális → fizikai memória cím hozzárendelés
 - Tipikusan (x86) 4 kB méretű allokációs egységekben
 - A cím utolsó 12 bitje a lapon belüli cím
 - A cím első 20 bitje kétszintű (10-10 bit) laptábla cím
 - Létezik óriás lap üzemmód is, ilyenkor csak egyszintű laptábla van, ezen belül 22 bit (4MB) pozíciócím



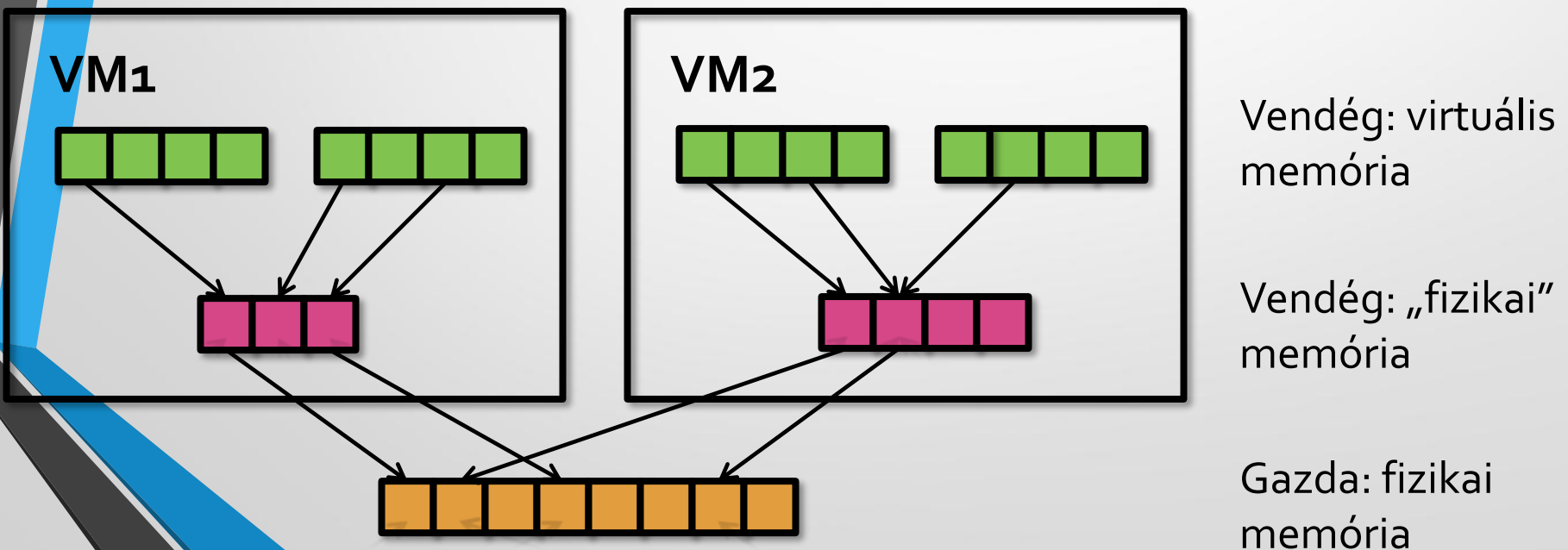
Virtuális memória megvalósítása lapokkal

További jellegzetességek:

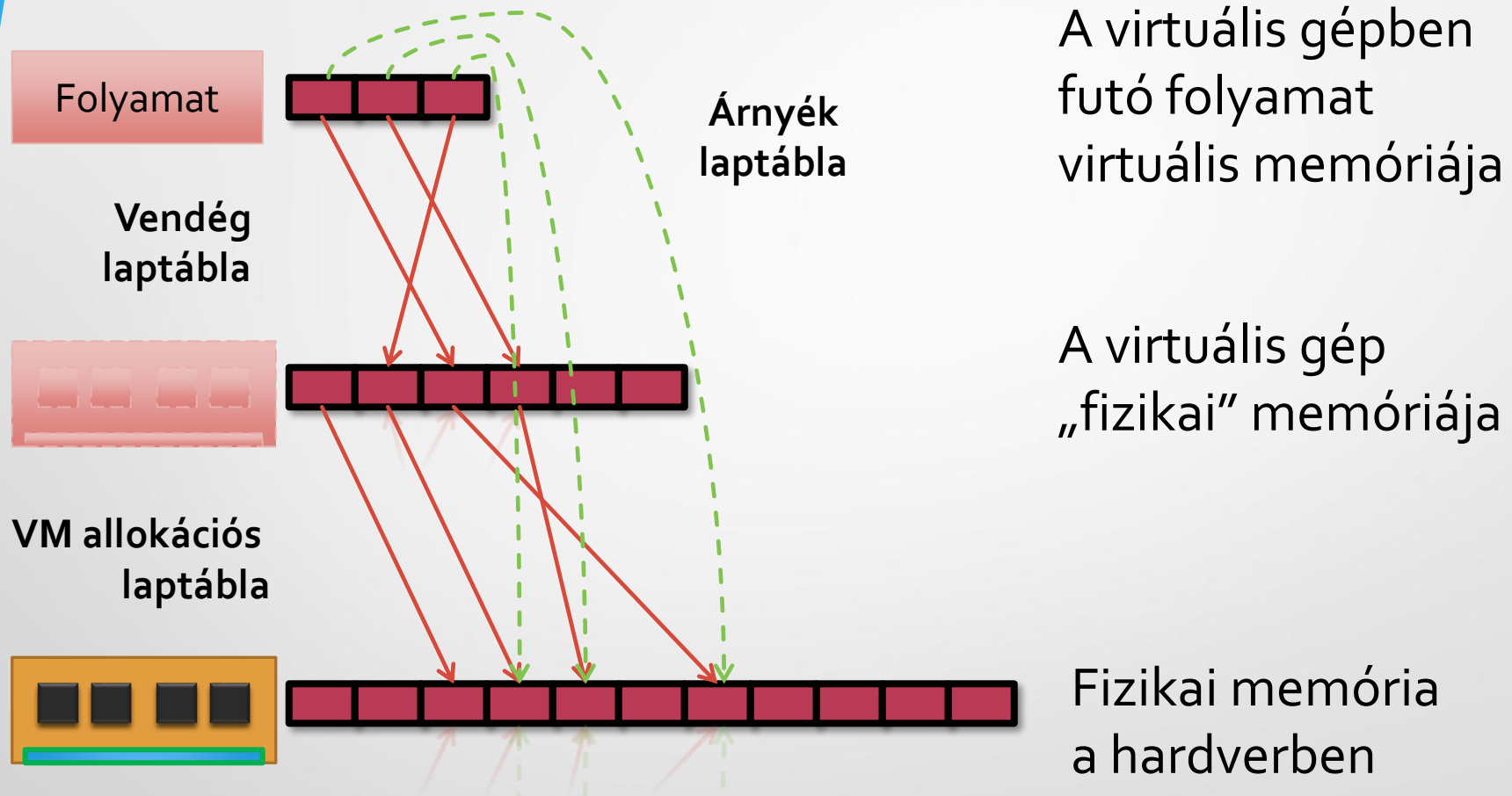
- A laptáblák is a fizikai memóriában foglalnak helyet
- Csak az operációs rendszer kernel módosíthatja őket
- Minden folyamathoz másik táblakészlet tartozik, a kernel kontextus váltáskor cseréli ki mindig a megfelelőre
- Az MMU a CPU laptábla regisztere alapján tudja, hogy hol kell keresni legfelső szintű lap könyvtárat
- Automatikusan feloldja a virtuális címeket fizikaira, a virtuális címeket használó kód módosítás nélkül fut
- A virtuális címtartományból kicímzés vagy read-only bittel jelölt lapra írás hibát (fault) vált ki a CPU-ban

Memória virtualizálása

- A Ring 0-tól eltérő futó folyamatok virtuális memóriát látnak
 - A virtuális -> fizikai cím feloldása hardverben történik laptáblák alapján. Gyors, TLB cache-eli fizikai-virtuális cím hozzárendelést.
- Használhatjuk-e ezt a vendég gépek memóriájához?
 - Több szint kell: a VM-ben is kell saját laptábla a saját alkalmazásokhoz
 - De a CPU ilyet nem támogat



Memória virtualizálása



Memória virtualizálása

- Mi van, ha vendég kernel módosítani akarja a laptábláját?
 - Megfelelően frissíteni kell az árnyék táblát is
- Három megoldás:
- **1.** Természetesen Trap and emulate, de hogyan?
 - Read-only-ra állítjuk a vendég kernel számára látható laptáblákat, ha azt módosítani akarja, akkor jön a kivétel, átkerül a vezérlés a VMM-hez ami biztonságosan elvégzi a módosítást az árnyék táblán is
- **2.** „természetesen?! trap and emulate?”
 - A vendég kernel egyszerűen ne maga akarja módosítani a laptáblát, *kérje meg* a VMM-et erre 😊
 - Paravirtualizációs megoldás
- **3.** Hardveres kiegészítés több szintű laptáblák kezelésére
 - Core i7 és Phenom processzoroktól kezdve van (EPT / RVI)
 - Az egész árnyék tábla frissítési problémát hardveresen lekezeleli
 - Ezzel együtt hardveres virt. gyorsabb lehet a szofteres virtualizálásnál

Tartalom

- Előző rész tartalmából:
 - CPU virtualizáció
 - A három alap virtualizációs megközelítés
- Memória virtualizáció
 - Virtuális memória az operációs rendszerekben
 - Virtuális memória a platform virtualizációban
 - **Virtuális memóriakezelés speciális képességei: megosztás, késleltetett allokáció, memória-ballon**
- Perifériák virtualizációja
 - Perifériák programozói felülete általában
 - Periféria virtualizációs architektúrák

Extra memória virtualizálási lehetőségek

- Memórialap deduplikáció
 - azonos tartalmú memórialapok megosztása több vendég VM között
 - hasonlóképpen azonos lapok megosztása egy vendégen belül is
 - gyakorlati haszna főleg speciális alkalmazásokban (Virtual Desktop Infrastructure), tipikusan több példány fut azonos OS-ből
 - Megvalósítása
 - gyors hash számítás, ez alapján egyezés keresés
 - közösített lapok megbontása beleíráskor, copy-on-write elv
- Hasonló: memória tömörítés
 - Egészen új lehetőség VMM-ekben (az ötlet persze régi)
 - CPU költsége nagyon nagy lenne, ezért:
 - az inaktív, amúgy háttértárra kilapozásra ítélt lapokat szokás tömöríteni -> a ki/be tömörítés még így is gyorsabb a merevlemezénél
 - Nem csodaszer... kompromisszumot kell kötni a tömörítetlen és tömörített lapoknak fenntartott memória mérete között, csak korlátozott méretben előnyös

Extra memória virtualizálási lehetőségek

- Dinamikus allokáció:
 - Ami memóriát nem használ a vendég, azt ne is kapja meg
 - Gyakorlati haszna önmagában elenyésző, a legtöbb OS az összes szabad memóriát disk cache-nek használja
 - Háttértárra swappelhetők a lapok a vendég OS tudta nélkül
- Memória felfújás (memory ballooning)
 - Ha kifogy a host memóriája, akkor „elvesz” a vendégtől
 - Egy ágens vagy driver a vendég kernelben (paravirtualizációs szemléletmód) elkezd memóriát foglalni a VMM utasítására. A VMM az ágens által „foglalt” memórialapok mögé nem is allokal fizikai memóriát, így nyer vissza helyet
 - Egyrészt a vendég fel fog adni a disk cache-ből,
 - Másrészt el fog kezdeni kilapozni a saját swap területre, elkerüli, hogy a host is swappeljen

Tartalom

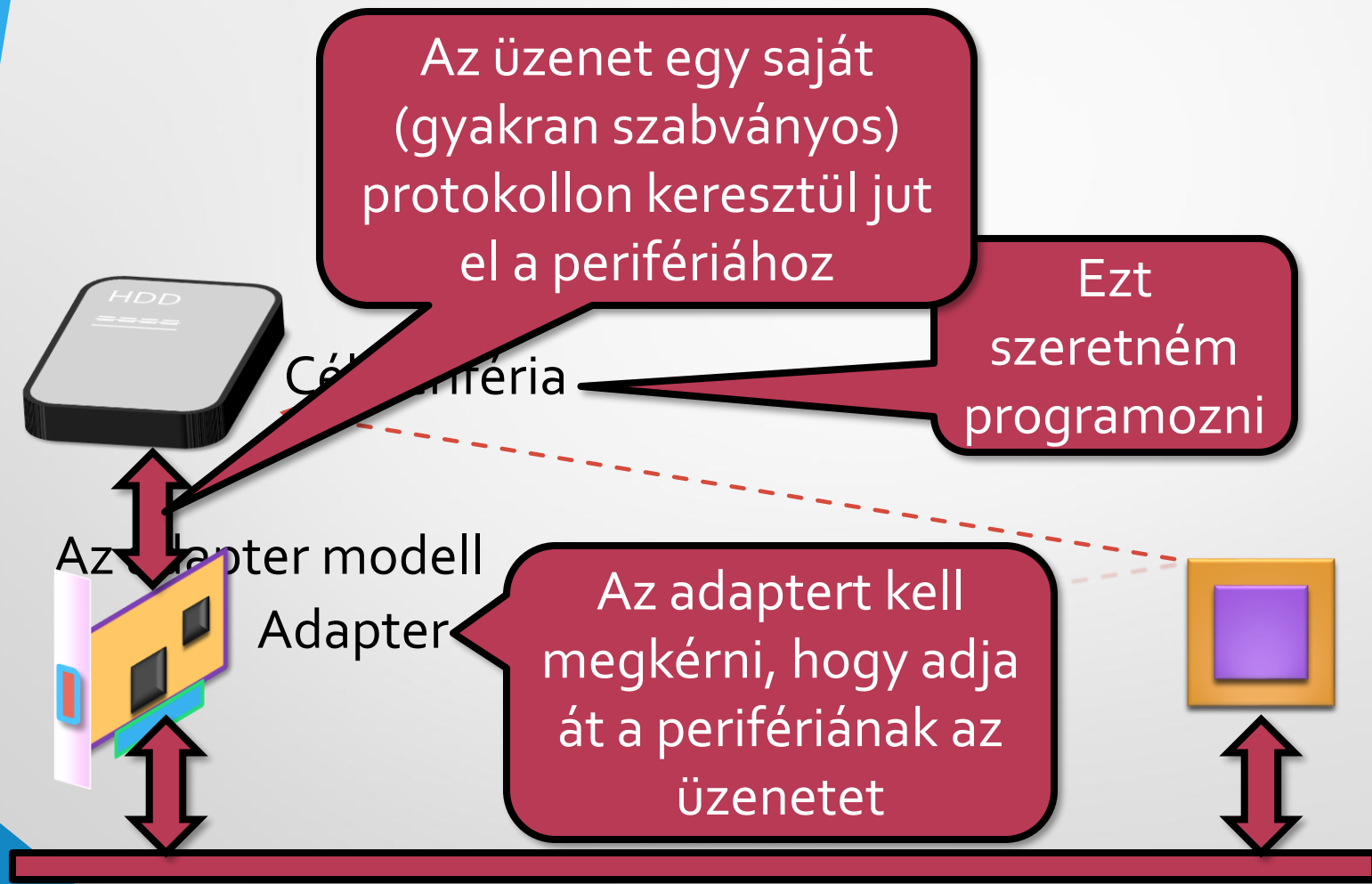
- Előző rész tartalmából:
 - CPU virtualizáció
 - A három alap virtualizációs megközelítés
- Memória virtualizáció
 - Virtuális memória az operációs rendszerekben
 - Virtuális memória a platform virtualizációban
 - Virtuális memóriakezelés speciális képességei: megosztás, késleltetett allokáció, memória-ballon
- **Perifériák virtualizációja**
 - **Perifériák programozói felülete általában**
 - Periféria virtualizációs architektúrák

Perifériákról általában



- A perifériák kezelése jellegzetesen
 - CPU felprogramozza a perifériát, regiszterek átírása
 - Periféria eseményt jelez a CPU felé, megszakítás
 - Ilyenkor valamilyen módon le kell kezelni az eseményt, valamit reagálni kell rá (driver felelőssége)
 - Periféria maga elvégzi a feladatát, közvetlen memória hozzáférés
 - Kiolvas elküldendő adatot, vagy berak beérkező adatot a memóriába
 - Külön lefoglalt fizikai memóriaterület kell erre a célra

Perifériákról általában



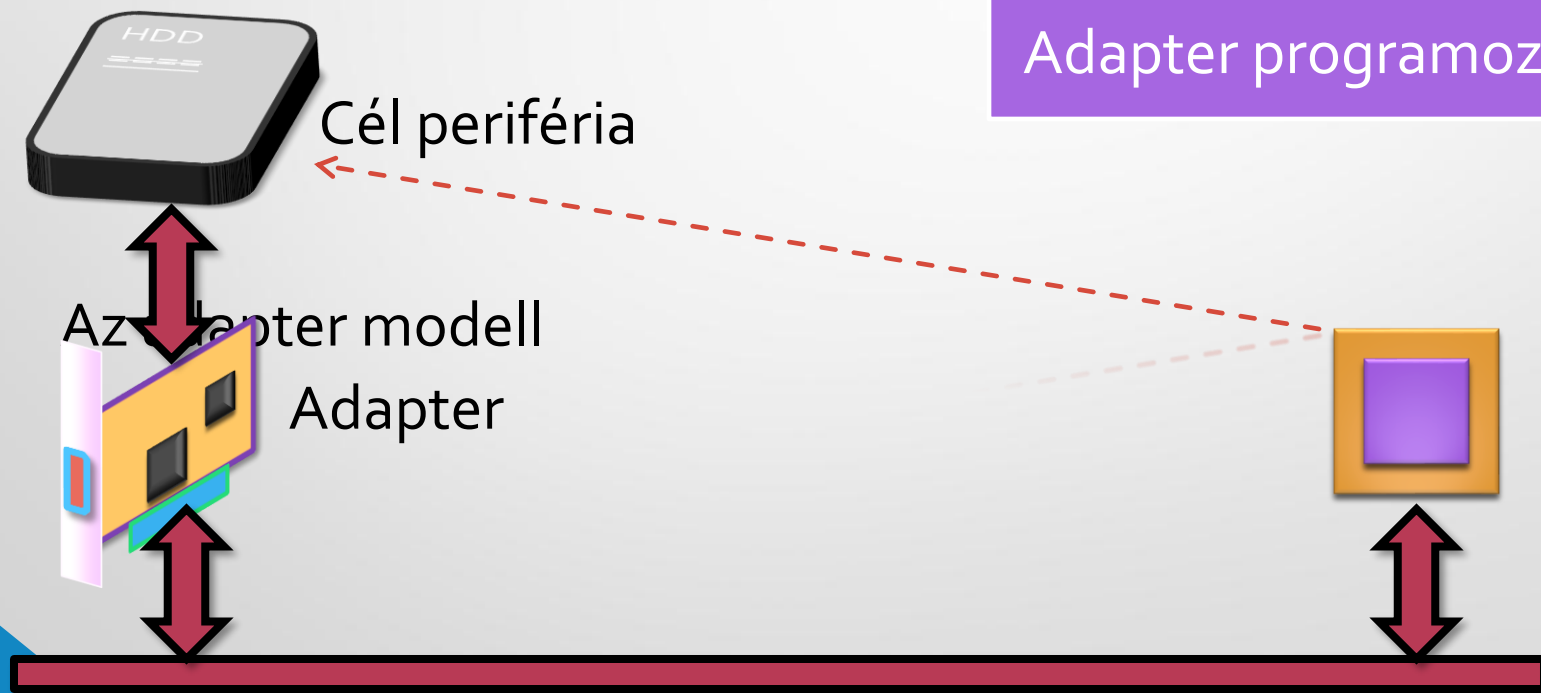
Perifériákról általában

Perifériának szánt
konkrét utasítás, adat

Periféria protokollja

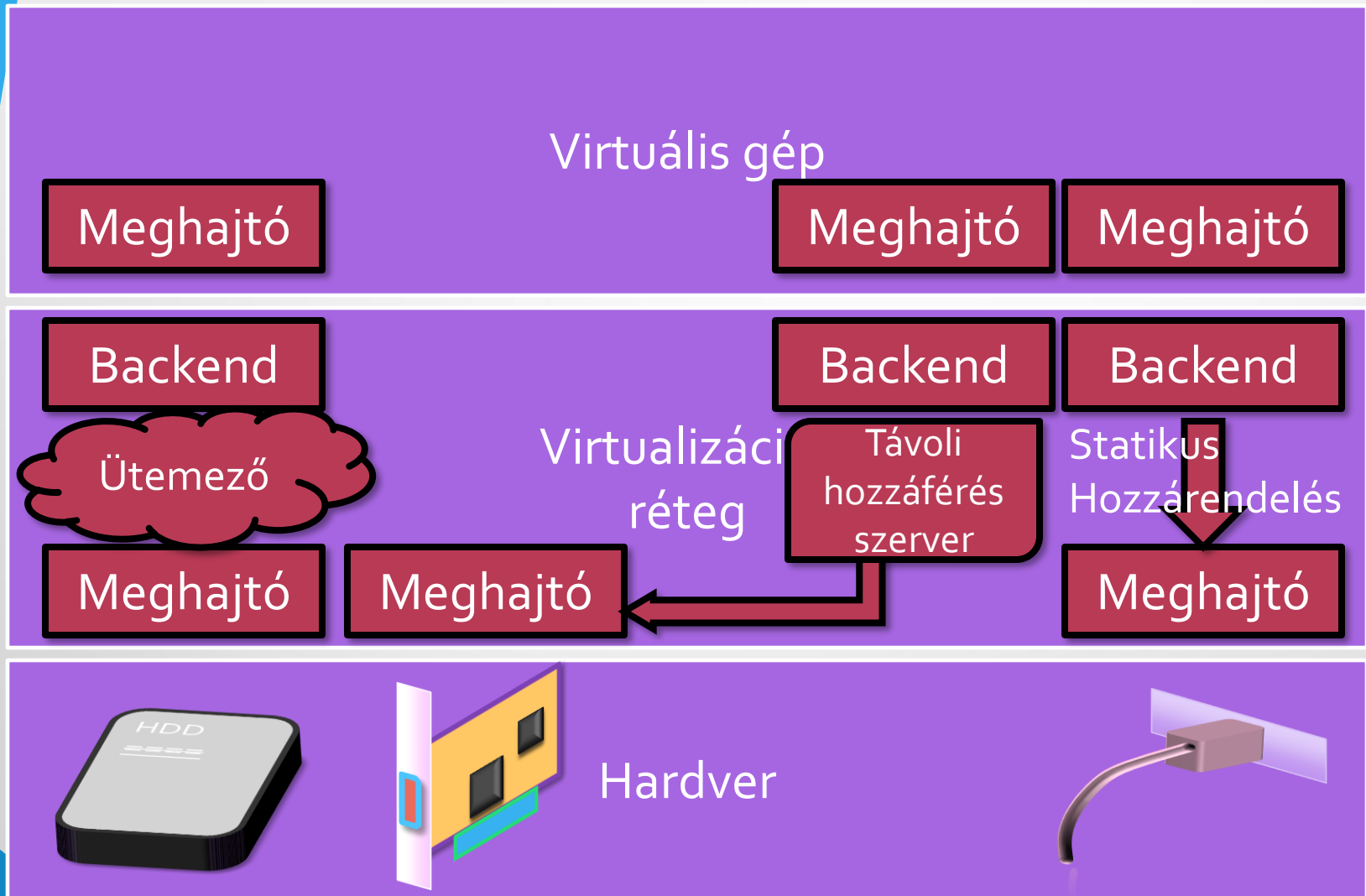
Adapter programozás

Rétegzett programozási
modell (pl. USB, SCSI, SATA):



Tartalom

- Előző rész tartalmából:
 - CPU virtualizáció
 - A három alap virtualizációs megközelítés
- Memória virtualizáció
 - Virtuális memória az operációs rendszerekben
 - Virtuális memória a platform virtualizációban
 - Virtuális memóriakezelés speciális képességei: megosztás, késleltetett allokáció, memória-ballon
- **Perifériák virtualizációja**
 - Perifériák programozói felülete általában
 - **Periféria virtualizációs architektúrák**



Lehetőségek perifériák virtualizációjára I.

Emuláció

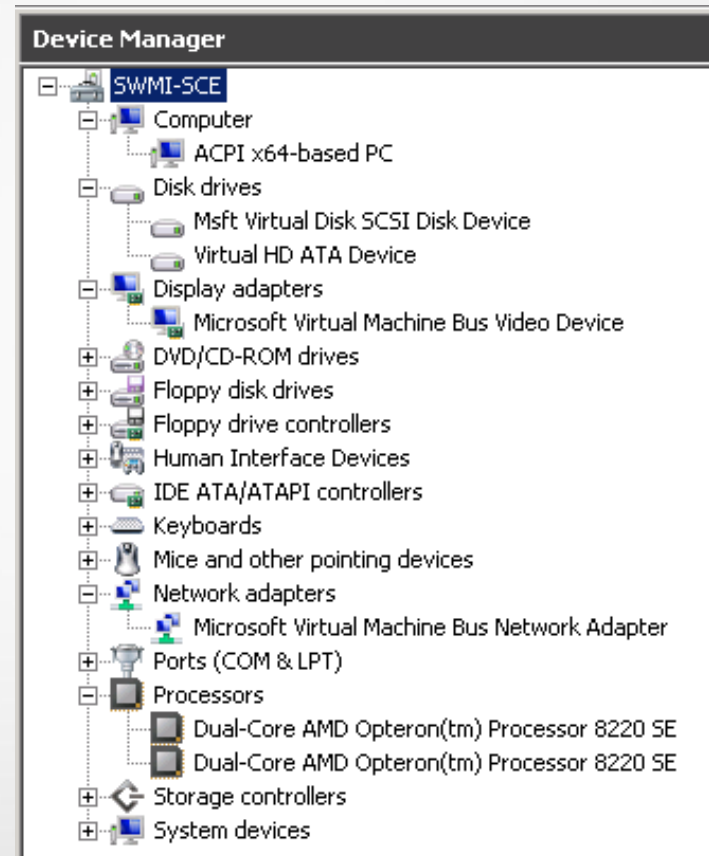
- Trap and emulate -> az I/O műveleteket kell elfogni
 - Adódik: ring 1-3-ban az I/O műveleteket elfogja a CPU
 - Memóriatartományba illesztett periféria: read-only memóriával fogható el
- Valamilyen létező hardver működését emuláljuk
 - Szoftveres komponens segítségével: *backend*
 - Hardver pontos emulálása (regiszterek, megszakítás, DMA)
 - Vendégben használható a klasszikus meghajtó program
- Minden I/O művelet egy kör a VMM-ben -> lassú

Lehetőségek perifériák virtualizációjára II.

Paravirtualizáció

- Egyszerűsítsük az emulált hardvert, tervezzünk „nem létező fajta” hardvert, amit a legkevesebb művelettel lehet vezérelni
- Egy összetett művelet akár csak egy VMM hívás
- Saját „hardverek”, amik magas szintű műveleteket végeznek
 - Pl. hoszt fájlrendszerhez hozzáférés
 - Itt kezd keveredni a virtualizáció és az OS...
- Speciális meghajtót kell telepíteni a VM-ben!

Paravirtualizált I/O eszközök



(VMware Tools, Hyper-V Integration Components kell)

További információ

- Carl Waldspurger and Mendel Rosenblum. [I/O virtualization](#). *Commun. ACM* 55, 1 (January 2012), 66-73. DOI=10.1145/2063176.2063194
- Abramson, D. *et al.* "[Intel® Virtualization Technology for Directed I/O](#)." Intel Technology Journal. (August 2006).
- Darvas Dániel, Horányi Gergő. „[Intel és AMD technológiák a hardveres virtualizáció megvalósítására](#)”, virttech házi feladat, 2010.
- Garaczi Tamás. [Intel VT-d \(IOMMU\) technológia részleteinek megismerése](#), virttech HF, 2010.