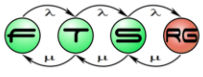
Virtualizációs Technológiák és Alkalmazásaik

Memória és perifériák virtualizációja

Tóth Dániel



Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

Utolsó módosítás:2010. 09. 22.

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

Emlékeztető: virtuális CPU feltételei

- Popek és Goldberg* követelményei egy virtualizált rendszerrel szemben
  - *VMM – virtual machine monitor*:
    olyan komponens, ami a virtuális gépek számára az absztrakciót biztosítja
- **Ekvivalencia**
  - A VMM felett futó program mindig pontosan ugyanazt az eredményt adja futás közben, mint ha fizikai CPU-n közvetlenül futna
- **Erőforrás kezelés**
  - A VMM minden virtualizált erőforrást teljes egészében maga felügyel
- **Hatékonyság**
  - A virtuális gépben futó program utasításainak nagy része változtatás és VMM beavatkozás nélkül fut a fizikai CPU-n

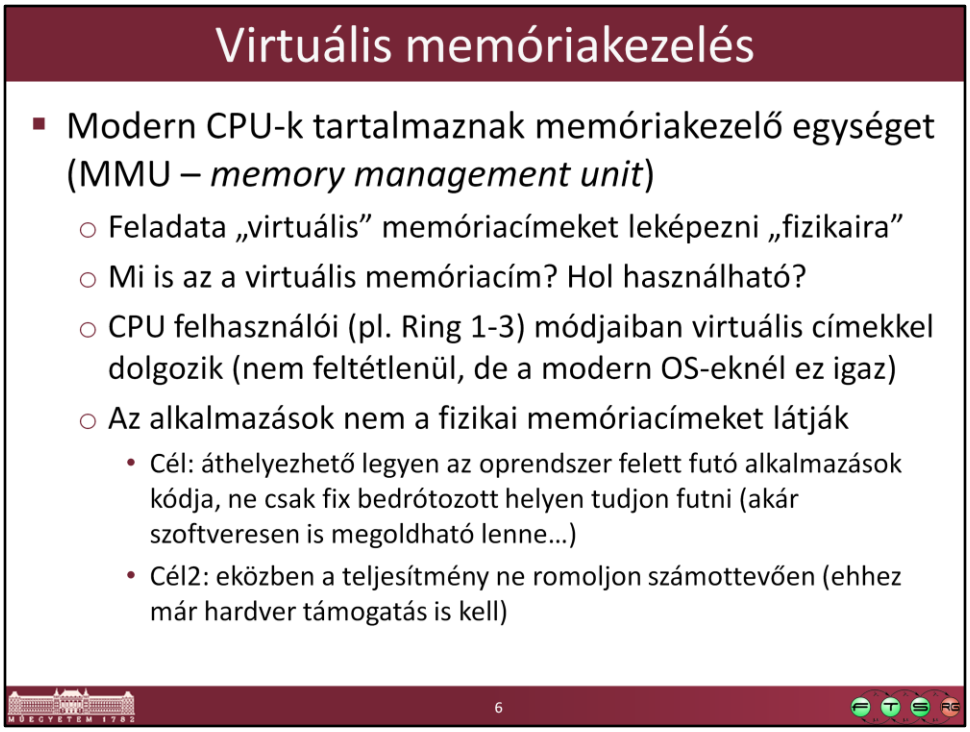
* Gerald J. Popek and Robert P. Goldberg: Formal Requirements for Virtualizable Third Generation Architectures, 1974

Emlékeztető: A három virtualizációs lehetőség

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

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

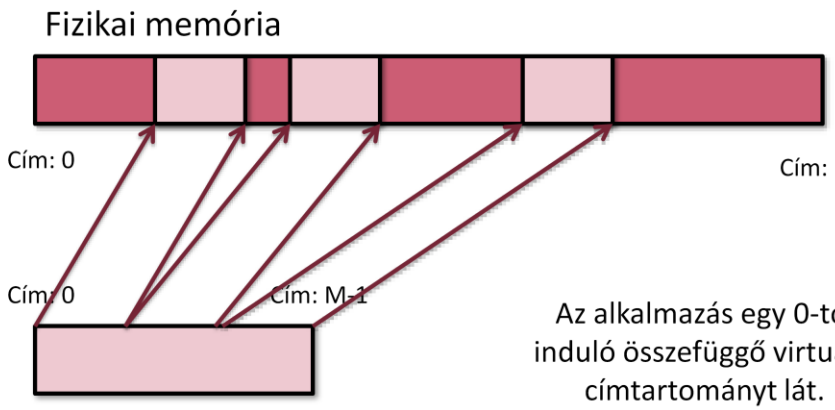


MMU != fizikai memóriaillesztő áramkör. Az utóbbinak a feladata a memória modulok elektromos alacsonyszintű vezérlése, ez sokáig a CPU-n kívül a chipset „északi hídban” foglalt helyet, csak a közelmúltban (Athlon64, Core i7) került be a processzorba. A memóriaillesztő áramkör is végez címfordítást, de ez már teljesen láthatatlan a CPU és a perifériák számára. A fizikai memóriacím alatt a továbbiakban azt értjük amit a CPU és/vagy perifériák kiadnak a buszokon.

x86-ra jellemző, hogy kétféle virtuális memóriacím kezelési mechanizmus van: szegmens alapú és lapozott. A szegmens alapú 32 bites üzemmódban _mindig_ be van kapcsolva, de egy szegmens tartalmazhatja a teljes memóriát. 64 bites üzemmódban viszont szegmens alapú memóriaszervezés nincs. A lapozott üzemmód az aktuális ringtől függetlenül lehet ki vagy bekapcsolt állapotban, de átkapcsolni csak ring 0-ban lehet. Manapság tipikusan csak a kernel használ fizikai memóriacímeket (nem feltétlenül az összes funkciójához), az alkalmazások mindig lapozott memóriát használnak.

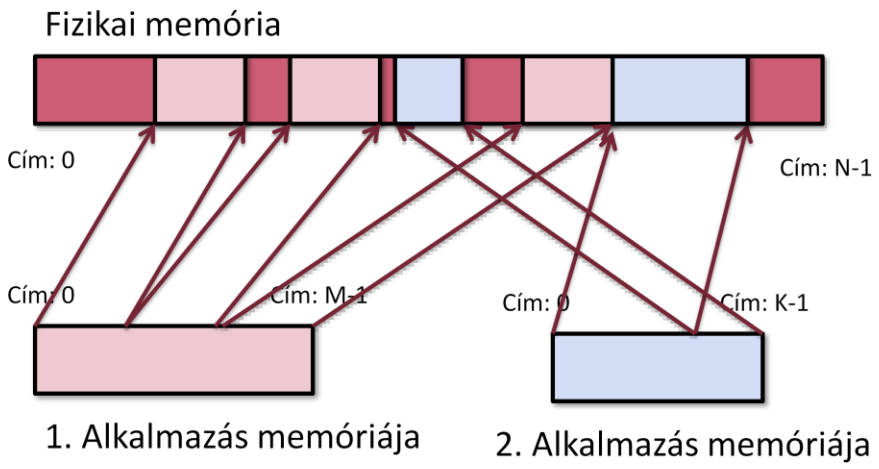
Részletek:
Intel® 64 and IA-32 Architectures Software Developer’s Manual
http://www.intel.com/products/processor/manuals/

Virtuális memóriakezelés



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

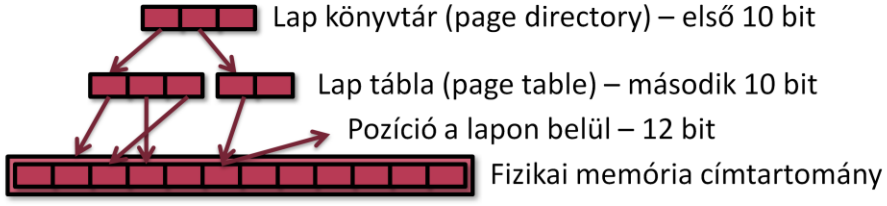
Virtuális memóriakezelés



Ez minden alkalmazásra igaz

Virtuális memória megvalósítása lapokkal

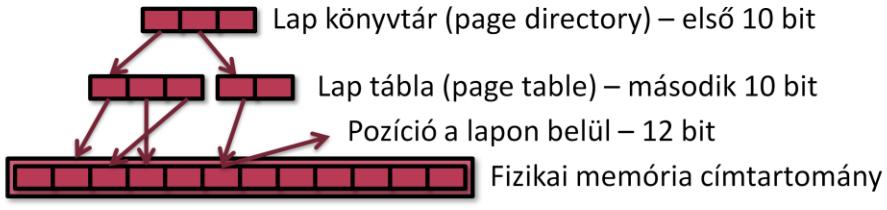
- Memória lapok (*pages*)
  - Virtuális → fizikai memória cím hozzárendelés
  - Tipikusan (x86) 4 kB méretű allokációs egységekben
  - A cím utolsó 12 bitje a lapon belüli cím
  - A cím első 20 bitje kétszintű (10-10 bit) laptábla cím
    - Létezik óriás lap üzemmód is, ilyenkor csak egyszintű laptábla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



Megjegyzés: a felsorolt konkrét számadatok x86 architektúrán érvényesek, más architektúrán előfordul más beosztás is. Az x86-on is léteznek még speciális üzemmódok (PAE – paging area extension, NX – no execute), amik 1-1 felső helyiértékű bitet saját célra használnak.

Virtuális memória megvalósítása lapokkal

- A laptáblák is a fizikai memóriában foglalnak helyet
  - Csak az operációs rendszer kernel módosíthatja őket
  - Minden alkalmazáshoz másik táblakészlet tartozik, a kernel ütemezéskor 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 trap-et vált ki a CPU-ban



A trap mechanizmus teszi lehetővé, hogy az éppen nem használt lapokat diszkre lehessen menteni majd vissza behúzni a memóriába akkor, amikor éppen hozzáférés történik. A hozzáférés a nem létező lapra trap-et vált ki, ezt elfogja és ekkor teszi vissza a kernel a lapot a swap területről.

Memória virtualizálása

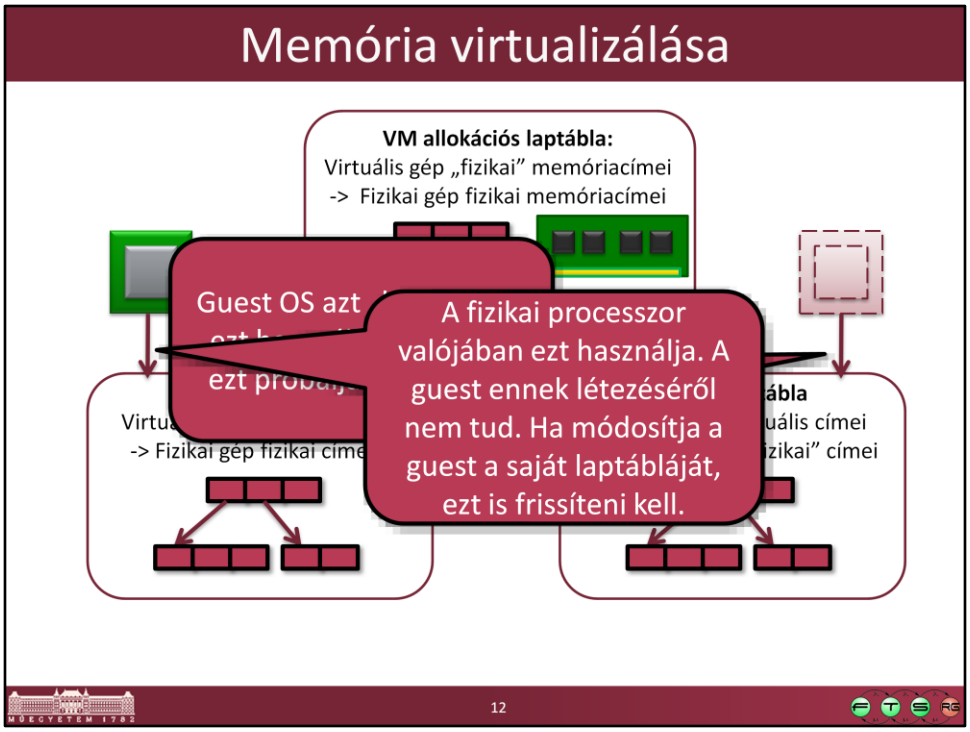
- A Ring 0-tól eltérő szinteken futó folyamatok virtuális memóriát látnak
  - A virtuális -> fizikai cím feloldása hardverben történik laptáblák alapján. Gyors, TLB cache-eli fizikai-virtuális cím hozzárendelést.
- Használhatjuk ezt a vendég gépek memóriájához?
  - Több szint kell: a VM-ben is kell saját laptábla a saját alkalmazásokhoz
  - De a CPU ilyet nem támogat.
  - Lapszervezés kombinálható a szegmens szervezéssel („gányolás” de x86 szoftveres virtualizációnál néhány helyen jól jön)
  - Kell két példány a laptáblából az egyik a vendég kernel számára látható és a vendég VM „fizikai” címeit képezi le a VM-en belül futó alkalmazások „virtuális” címeire.
    A másik a VMM (és a CPU) számára látható ún. *árnyék laptábla (Shadow page table)* és a futtató gép „igazi fizikai” címeit képezi le a VM-en belül futó alkalmazások virtuális címére.

Innentől a fizikai memória fogalmánál meg kell különböztetni a hoszt gép szempontjából fizikai memóriacímeket és guest operációs rendszer által „fizikai” memóriának látott címeket, amik valójában a hoszt gép szempontjából már virtuálisak. Tehát az idézőjelnek itt most jelentésmegkülönböztető szerepe van. :)

Érdekesség, hogy x86-on a szegmens szervezés teszi lehetővé, hogy a guest-tel azonos memóriatartományban, de mégis a guest beavatkozásától védett módon legyenek elhelyezhetőek a VMM kezelésében lévő memóriaterületek. Például a JIT fordított kódterületek ilyen elhelyezésével lehet gyors (trap és kontextusváltás nélküli) áthívás a kétféle memóriaterület között. Viszont 64 bites üzemmódban a szegmens alapú védelem nem működik, ez a legfőbb oka annak, hogy x86-64 bites üzemmódot bináris fordítással a legtöbb virtualizációs szoftverben nem építettek ki.

Az AMD egyes processzoraiban éppen emiatt viszont korlátozottan mégis van szegmens alapú védelem 64 bites üzemmódban is, de ez nem túl széles körben támogatott.

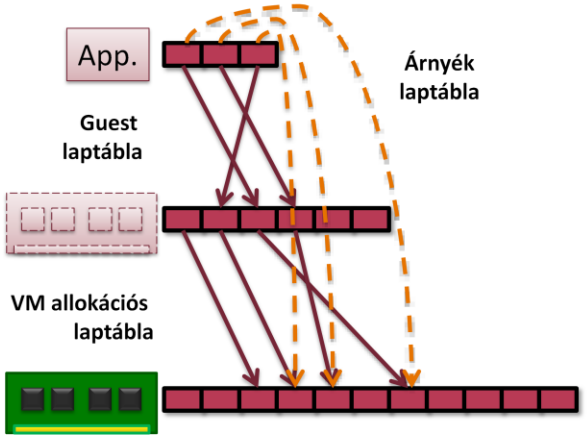
Részletek:
Keith Adams, Ole Agesen: A Comparison of Software and Hardware Techniques for x86. Virtualization.



Az árnyék laptábla szerepe, hogy az eredetileg kétszintű indirekcióból egyszintű indirekciót csináljon.

Memória virtualizálása

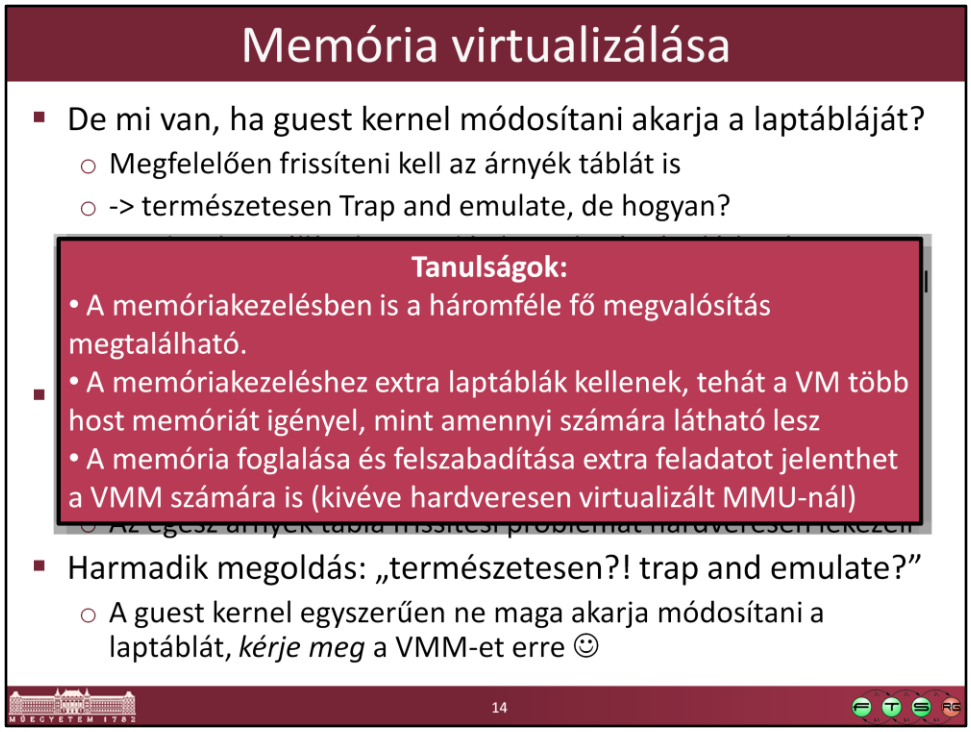
- Másik nézetben:



A virtuális gépben futó alkalmazás virtuális memóriája

A virtuális gép „fizikai” memóriája

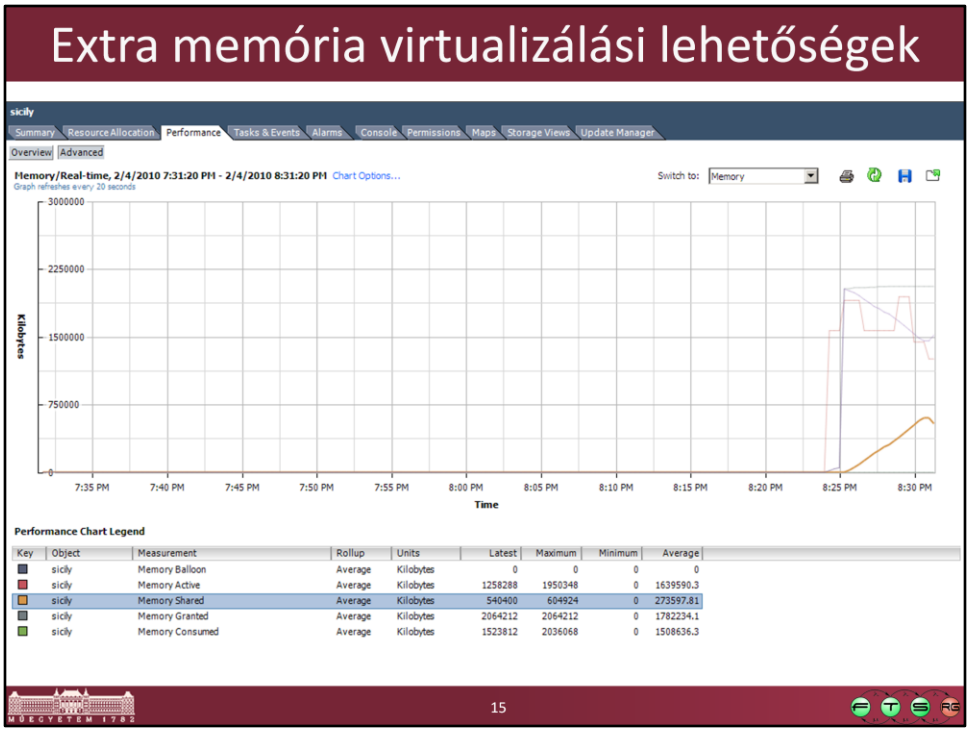
Fizikai memória a hardverben



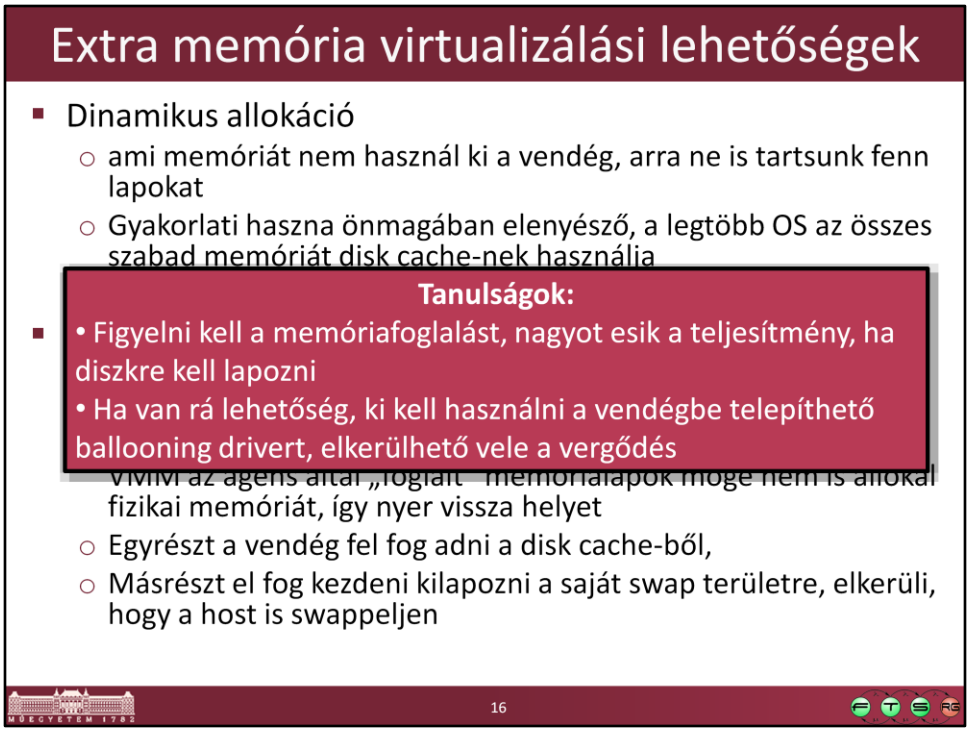
A hardveres laptábla kezelés elég sok előnnyel jár, ezzel már a hardveres virtualizáció egyértelműen gyorsabb lehet a bináris fordításnál. Elkerülhetőek vele a kontextusváltásnál a TLB ürítések, a VMM-ben tett körök a memória allokáció változtatásánál stb.

Intel megvalósításának részletes leírása:
http://www.intel.com/products/processor/manuals/
Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 3B: System Programming Guide, chapter 25.2

Aki még nem jött volna rá, a 3. megoldást természetesen paravirtualizációs megközelítésnek nevezik.



A nem túl jól látható (éles gépen éppen elcsípett) képernyőképen az látható, hogy az indítás után a guest gép memóriáját fokozatosan pásztázza végig a VMM, megosztható memórialapokat keresve, így fokozatosan emelkedik a shared memória mennyisége.



Miért olyan rossz az, ha a hoszt swappel? Mert a guest nem tudja, hogy a memóriatarományának egy része diszken van tehát lassú, ezt a területet diszk cache-nek vagy alkalmazásoknak fogja használni, amivel még többet árt. Esetleg ha a host és a guest is swappel egyszerre, akkor lehet, hogy a guest a saját swap területéről a (memóriának képzelt) host swap területére fog mozgatni adatot és viszont ->mega trashelés. Ha csak a guest swappel, akkor ez a helyzet nem fordulhat elő.

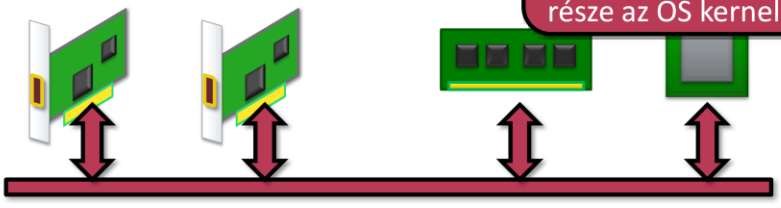
Tartalom

- 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

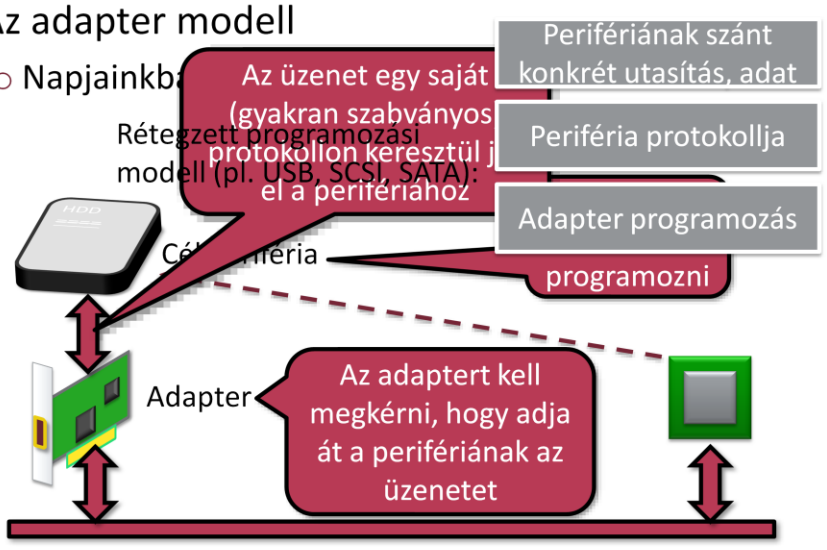
- A perifériák kezelése jellegzetesen
  - CPU felprogramozza a perifériát, regiszterek átírása
  - Periféria eseményt jelez a CPU felé, megszakítás
    - Ilyenkor valamilyen módon le kell kezelni az eseményt, valamit reagálni kell rá
  - Periféria maga elvégzi a feladatát, közvet… …mória hozzáférés
    - Kiolvas elküldendő adatot, vagy berak beér…
    - Külön lefoglalt fizikai memóriaterület kell e…

A periféria kezelése a „driver” felelőssége, ami többé kevésbé része az OS kernelnek



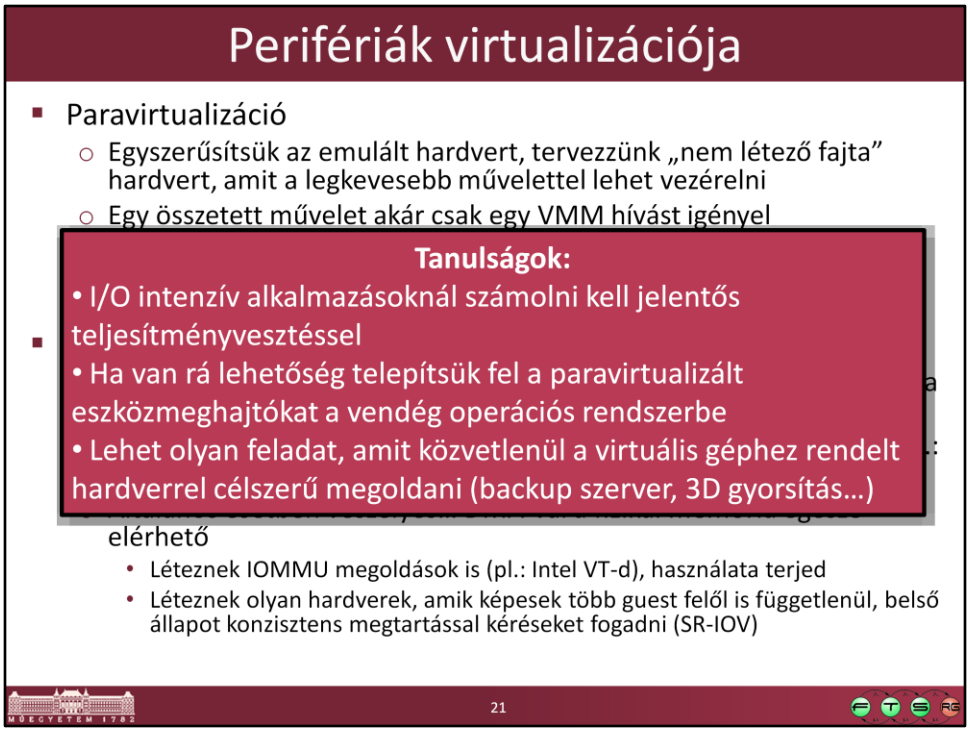
Perifériákról általában

- Az adapter modell
  - Napjainkb
    Rétegzett programozási modell (pl. USB, SCSI, SATA):



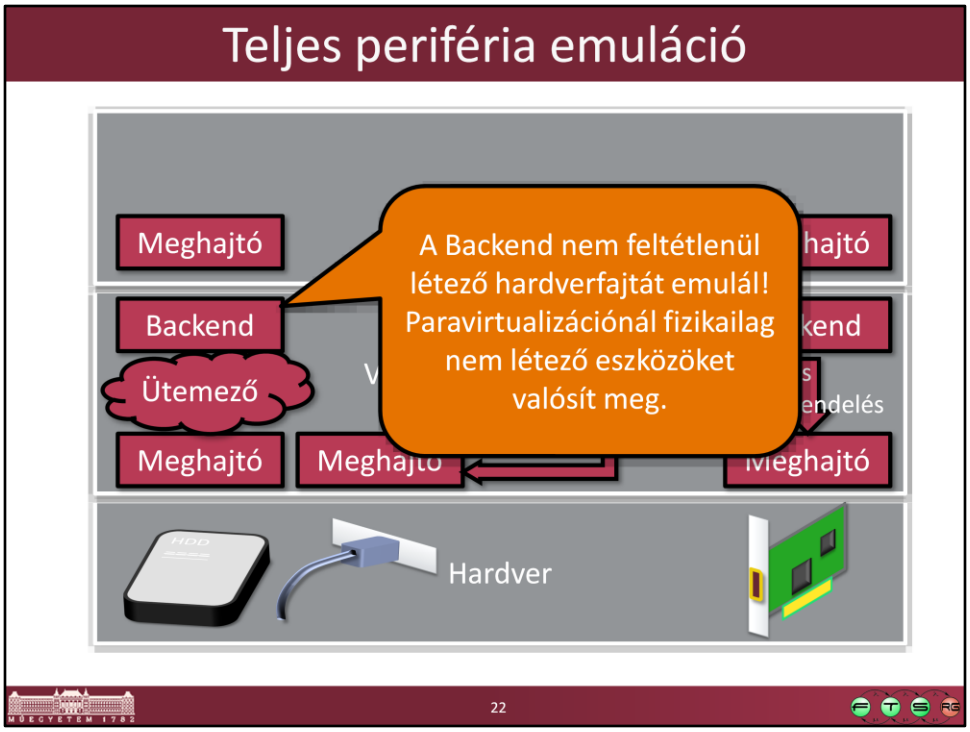
Perifériák virtualizációja

- Emuláció
  - Trap and emulate -> az I/O műveleteket kell elfogni
    - Ez adódik a CPU futási szintjéből, Ring1-3-ban az I/O műveleteket elfogja
    - Ha a periféria memóriatartományba illesztett, akkor read-only memórialappal fogható el a felprogramozása
  - Valamilyen létező fajta hardver működését (regiszterek, megszakítás, DMA) emulálni kell egy szoftveres komponenssel
    - Egyfajta „anti-driver”, általános elnevezése *Backend*
    - A vendégben futó meghajtóprogram valójában ezzel beszélget.
  - Minden I/O művelet egy kör VMM-ben -> lassú



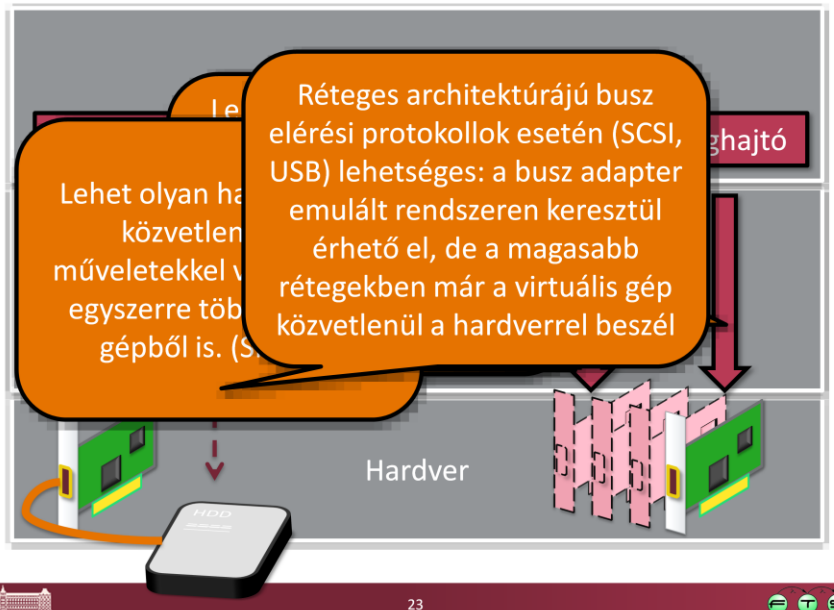
Fontos tanulság megint: amikor a paravirtualizációnál magasszintű erőforrásokat adunk át a guest OS-nek, akkor látható, hogy itt is - mint mindenhol máshol is – folytonos átmenet kezd kialakulni az operációs rendszerek és virtualizációs környezetek között. Tehát a szigorú taxonomizálás valójában megint csak egy „modell”, ami a megértést segíti, de a valóságot kicsit absztrahálja.

SR-IOV, manapság főleg hálózati adapterekhez készítették el
Részletek:
http://www.xen.org/files/xensummitboston08/Xen-SR-IOV.pdf
http://blog.scottlowe.org/2009/12/02/what-is-sr-iov/



Teljes periféria emulációnál a virtualizációs környezetben vannak a fizikai eszköz meghajtó programjai (driver), minden perifériát ilyenek kezelnek.
A virtuális gép számára egy szoftveres „backend” (vagy „anti-driver”, driver ellendarab) emulálja a hardvert.
3 jellegzetes eset lehetséges:
-Valamilyen többszörös hozzáférést biztosító erőforrásmegosztó vagy ütemező komponens osztja szét a backendek felé a fizikai hardver erőforrást (Háttértár, hálózat, stb.)
-Statikusan valamelyik virtuális géphez hozzárendelünk egy-egy hardvert (Soros port, USB eszközök, esetleg háttértár, CD meghajtó)
-Távoli elérés szerver komponens hálózaton keresztül tesz elérhetővé valamilyen elemet a virtuális hardverből (Grafikus konzol, CD meghajtó)

Hardveres periféria virtualizáció



Összefoglaló

- A virtualizáció alapjai I-II
  - CPU virtualizáció
    - CPU utasításkészlet architektúra és szerepe
    - 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