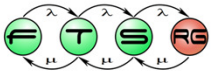
Virtualizációs technológiák és alkalmazásaik

Operációs rendszer szintű virtualizáció

Tóth Dániel, Szatmári Zoltán

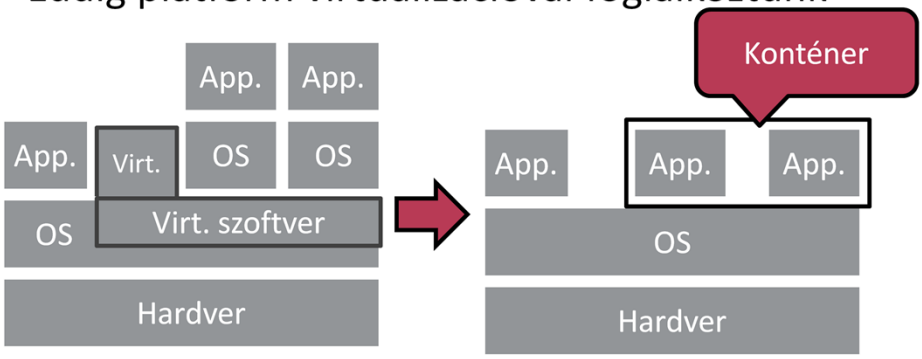


Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

Utolsó módosítás: 2012.11. 08.

Operációs rendszer szintű virtualizáció

- Eddig platform virtualizációval foglalkoztunk



-Ha megfelel, hogy azonos az OS kernel a vendég gépek között
-Ha csak izoláció, erőforrás-korlátozás kell
-> nem kell VMM, elég az OS alkalmazás szétválasztása

Operációs rendszer szintű virtualizáció

- Operációs rendszer szintű virtualizáció – más néven konténer alapú virtualizáció
  - BSD-ken: Jail
  - Solaris: Containers, Zones
  - Linux alatt: OpenVZ (Parallels Virtuozzoból a Linux kernel módú részei), Linux VServer
  - AIX: WPAR (workload partitions)
  - Windows: Parallels Virtuozzo

Operációs rendszer szintű virtualizáció

- nagyon kis költségű
  - elvileg nem 0 – csak éppen nem lehet kimérni olyan kicsi ☺
  - konténerenként nincsenek további vendég kernelek
- erőforrás virtualizációs és
- erőforrás gazdálkodási szempontból problémamentes
  - nincs fixen lefoglalt memória, nem kell trükközés ballonozással stb.
  - nincs fixen lefoglalt háttértár – a hoszt fájlrendszere fájl szinten elérhető
- biztonsági szempontból kevésbé jó izoláció
- közös kernellel kell élni (azonos verzió, fordítási paraméterek)

OS szintű virtualizáció vs. Hypervisor

- Hypervisor
  - Egy fizikai, számos virtuális hardver
  - Vegyes operációs rendszerek
  - Kisebb sűrűség
  - Kisebb teljesítmény
- OS szintű virtualizáció
  - Egy fizikai hardver, nincs virtuális
  - Egy kernel, userspace példányok
  - Nagy sűrűség
  - Közel natív teljesítmény

Megvalósítás

- Kezdetben volt a *chroot*…
  - A fájlrendszer gyökerét átirányíthatjuk egy alkönyvtárra (egy processzre vonatkozik!)
  - Cél:
    - Általában: az abszolút fájl elérési útvonalak módosítása nélkül tudjunk több példányt fenntartani egy-egy fájlból
    - Kicsit „antipattern” alkalmazás: biztonság növelése
  - Ez nem teljes körű izoláció, de sok esetben működik
    - Kernel minden adatszerkezete közös (processz lista, hálózati interfész, IP, routing, sysctl beállítások…)
    - A chrootból ráadásul ki is lehet navigálni a VFS adatszerkezeten keresztül…
  - Hogy is néz ki:
    - egy teljes alap OS installációt készítünk egy alkönyvtárba, ami kicsit eltérő is lehet az eredetitől
    - Általában véve a programkönyvtáraknak, konfig fájloknak meg kell lenniük a chroot-on belül is
  - Problémás globális könyvtárak: /proc, /sys, /dev, /tmp, /var, …
    - Lehet mount binddal trükközni, de nem lesz tökéletes…

Megvalósítás

- További probléma:
  - nincs elkülönítés a folyamatok között
  - nincs külön hozzáférés szabályozás
  - nincs erőforrás gazdálkodás
- Megoldás:
  - Ne látszódjanak ki a kernel singleton erőforrásai…
  - Ehhez módosítani kell a kernelt
    - Bevezetni a konténer fogalmát
    - Minden rendszerhívást ellátni a konténer kontextus szerinti válogatással
    - Singleton erőforrásokat dinamikusan példányosíthatóvá alakítani
    - A konténerből kifele mutató referenciák mostantól biztonsági réseknek számítanak!
  - A módosítások ára: <1% teljesítményveszteség
    - Nem átlagban, a rendszerhívásokra nézve ennyi!
    - -> különösen I/O igényes feladatoknál lényegesen jobb teljesítmény a platform-virtualizációhoz képest

Alkotóelemek

- Kernel
  - Névterek: virtualizáció és izoláció
  - Cgroups: erőforrás menedzsment
  - Checkpoint/Restart: live migration
- Userspace eszközök
  - Konténer menedzsment parancssori eszközök
- OS sablonok

Névterek

- Minden konténer rendelekezik:
  - Fájlrendszer: chroot()
  - Folyamat fa (PID namespace)
  - Hálózat (NET namespace)
  - IPC objektumok (IPC namespace)
  - …

Erőforrás-gazdálkodás

- CPU – a kernel beépített ütemezője, prioritáskezelője
- Memória – a kernel beépített memóriakezelője, *rlimit*-tel megadható maximális foglalások)
- Háttértár – a fájlrendszer egy alkönyvtára, *quota* rendszerrel korlátozható foglalás
- Hálózat – a kernel beépített Ethernet hídja vagy routing táblája, pl. IPtables QoS paraméterekkel korlátozható
- Egyéb perifériák – a kernelben lévő meghajtón keresztül

Részletesebb összefoglaló: http://mirrors.unbornmedia.com/openvz/doc/openvz-intro.pdf

Erőforrás-gazdálkodás

- CPU – a kernel beépített ütemezője, prioritáskezelője
- Memória – a kernel beépített memóriakezelője, *rlimit*-tel megadható maximális foglalások)
- Hátté *ota* rends
- Hálóz gy routin kel korlátozható
- Egyéb perifériák – a kernelben lévő meghajtón keresztül

Mi hiányzik?

Részletesebb összefoglaló: http://mirrors.unbornmedia.com/openvz/doc/openvz-intro.pdf

Erőforrás-gazdálkodás

- Azt szeretnénk, ha a konténerre vonatkozna a korlátozás, nem pedig az egyes folyamatokra
- Hierarchikus erőforrás foglalás:
  - A folyamatok fa hierarchiába szervezettek (nemcsak Unix, Windows alatt is)
  - Ún. Beancounter rendszer, ami csoportok szerint összesíti a foglalást
- A memóriafoglalás elég sokrétű probléma:
  - Alkalmazás virtuális/fizikai memórialapok
  - Cache lapok
  - Pufferek (socketek, hálózati kapcsolatok)
  - Megosztott lapok „igazságos” költségelszámolása
  - Néhány alkalmazás elég „zűrös” memóriafoglaló (Java VM) – kézi beállítást igényelhet

Beancounters részletes leírása:
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.125.3856&rep=rep1&type=pdf

Checkpointing/Migration

- Teljes konténer állapot
  - Menthető
    - Futó folyamatok
    - Megnyitott fájlok
    - Hálózati kapcsolatok
    - Memória szegmensek
  - Visszaállítható
  - Visszaállítható másik szerveren

Hozzáférés a vendég gépekhez

- Szöveges konzol elérés van
- Grafikus felület
  - általában nehéz, mert globális erőforrásokhoz (Unix alatt X Server, socket) igényel hozzáférést
  - Gyakorlatban használt megoldás: távoli elérés szerver indítása a konténerben, tipikusan VNC szerver
- Fájlrendszer
  - Konténeren kívülről belátunk a konténer fájlrendszerébe
  - Konténeren belülről nem látunk kifelé
- Folyamatok
  - Mint a fájlrendszernél…

DEMO OpenVZ

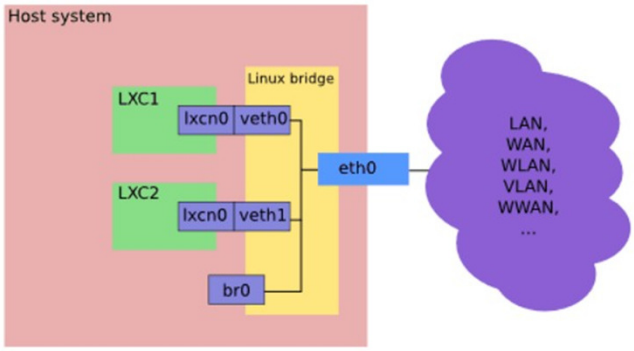
- Linux alatti teljesen nyílt forrású megoldás
  - Container neve: VE – Virtual Environment
  - Gyors példányosítás sablonokból (template) (egy teljes alap OS fájlrendszer kép tömörítve)
  - Plan rendszer – memória és I/O korlátozási beállítások külön tárolása
  - Grafikus felhasználói felület: Vtonf (webes konzol)
- Parallels Virtuozzo – Windows
  - Teljes kereskedelmi megoldás OS szintű kiegészítésekkel és integrált felhasználói felülettel
  - OpenVZ-vel azonos elveken épül fel

OpenVZ és LXC

- OpenVZ
  - Történelmileg korábbi (2000)
  - Kernel patch-et igényel
  - Live migration-t támogat
- LXC
  - Újabb kezdeményezés (2006)
  - Linux-VServer és OpenVZ utóda
  - Ma már a mainline Linux kernel része
  - Jelenleg is folyamatos fejlesztés alatt áll

Hálózati topológia

- Pont-pont kapcsolat a konténer és a hoszt között
- A hoszt hálózati interfészei között lehetőség van NAT vagy Bridged jellegű beállításra



Összefoglalás

- Konténer alapú virtualizáció - eltérő koncepció a platform virtualizációtól
  - Közös kernel
  - Kisebb erőforrás igény
  - Operációs rendszer szintű erőforrásokat virtualizál
  - Jellemzően a host felől nézve a konténer átlátszó, belülről nézve átlátszatlan
- OpenVZ, Parallels Virtuozzo, LXC
  - Linux illetve Windows környezetben nagyon hasonló megoldások
  - Konténerek példányosítása sablonokból
  - Erőforrás beállítások kezelése plan-ek alapján

További információ

- OpenVZ: http://wiki.openvz.org/Main_Page
- Parallels Virtuozzo: http://www.parallels.com/products/pvc45/
- LXC: http://lxc.sourceforge.net/
- Publikációk:
  - http://mirrors.unbornmedia.com/openvz/doc/openvz-intro.pdf
  - http://www.kernel.org/doc/ols/2008/ols2008v2-pages-85-90.pdf
  - http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.125.3856&rep=rep1&type=pdf