Virtuális gépek kialakítása XEN segítségével

Előbb-utóbb minden rendszergazda szembetalálkozik azzal a helyzettel, amikor újabb kiszolgálók beüzemelésével bízzák meg, csak éppen elfelejtenek hozzá új gépet adni. Paradoxonnak tűnik, ugye? Vagy mégsem? Ilyenkor jöhet jól egy olyan szoftver, amelynek segítségével megoszthatjuk a már rendelkezésünkre álló kiszolgálók erőforrásait.

Xen virtual machine monitor (VMM)

Az ***x86***-os architektúrára jelenleg két elterjedt ***virtualizálási technológia*** létezik. Az egyik a vendég operációs rendszerek módosítás nélküli futtathatóságát helyezi előtérbe (***VMware***), míg a másik a nagyobb teljesítményt (***Xen***, ***Denali***). Az első megközelítés kétségtelen előnye, hogy szinte minden x86 architektúrára készült rendszert képes futtatni, de ennek az ára a relatív kisebb teljesítmény. A második módszer a vendég operációs rendszerből majdnem natív teljesítményt hoz ki, viszont igényli azok módosítását kernel szinten. Ezt a virtualizálási technológiát ***paravirtualizációnak*** hívják. Ezt a megközelítést alkalmazza a ***Xen*** is.

A ***Xen VMM*** a ***XenoServers*** projekt részeként született meg a ***University of Cambridge Computer Laboratory*** berkein belül. A ***Xen***, definíciója szerint: paravirtualizáló virtuális gép monitor x86 architektúrára. A paravirtualizáció a vendég operációs rendszerek számára az alattuk lévő hardver eléréséhez egy szoftver interfészt biztosít. Ez az interfész működésében nagyon hasonlít az eredeti hardverhez, de különbözőségek is akadnak szép számmal. Ezeket az eltéréseket úgy oldotta fel a fejlesztő csapat, hogy a ***Xen***-ből egy külön ***arch***-ot készítettek, mintha a ***Linux*** kernelt egy új platformra kellett volna átültetni. További előnye ennek a rendszernek, hogy a felhasználói programok változtatás nélkül futtathatók. A rendszer struktúráját lásd az *1. ábrán*.

A ***GNU/Linux*** változat mellett jelenleg fejlesztés alatt állnak a ***NetBSD***, ***FreeBSD***, ***Plan 9*** portok is. Mindezek mellett a fejlesztés korai szakaszában elkészült a ***Windows XP*** verzió is, amelyet – sajnos – licencelési problémák miatt nem lehet a nagy közönség számára elérhetővé tenni. Ebben az ügyben előrelépés akkor várható, amikor az ***Intel*** és az ***AMD*** által bejelentett új technológiák végre megjelennek az új processzoraikban (***Intel® Virtualization Technology*** – ➲ ***http://www.intel.com/technology/computing/vptech/*** ; ***AMD Pacifica*** – ➲ ***http://www.amd.com/us-en/Corporate/VirtualPressRoom/0,,51_104_543~98372,00.html***).

A ***Xen x86*** architektúrájú, ***„P6"*** vagy annál újabb processzorokat igényel (***Intel Pentium Pro***-tól ***Xeon***-ig, illetve ***AMD Athlon***, ***Duron***). Támogatja az ***SMP***-t és hajlandó működni ***Hyper-Threading*** képes processzorokon is. A 64 bites verzió fejlesztés alatt áll, habár már most is hajlandó elindulni ilyen gépeken 32 bites módban. Tervezik továbbá a ***PPC*** és ***ARM*** architektúrák támogatását és a memória használat jelenlegi korlátainak bővítését. A ***Xen*** telepíthető notebookra is, de készüljünk fel az akkumulátorok gyors merülésére, mivel sem az ***APM*** sem az ***ACPI*** nem támogatott.

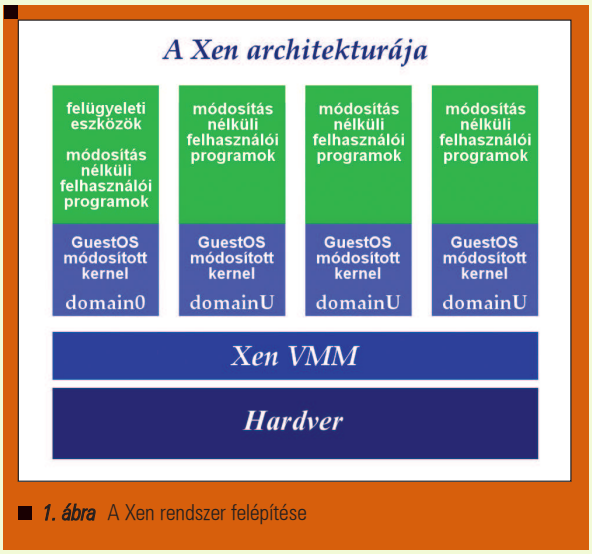
A teljesítményről bővebben a ➲ ***http://www.cl.cam.ac.uk/Research/SRG/netos/xen/performance.html*** weboldalon találunk információt, összehasonlító táblázatokat:

Telepítés

A telepítés menetét ***Debian Sarge***-ra fogom leírni, de más disztribúciókon is hasonló lépéseken kell majd végigmennünk. Lássuk milyen csomagok érhetőek el a projekt web oldalán: először is van a stabil verzió, amely a cikk írásának pillanatában a 2.0.6-os verziónál tart, találhatók továbbá ***testing*** és ***unstable*** verziók is. Természetesen elérhető a program forrása, és a 64 bites verziót is le lehet tölteni. Aki a telepítés nehézségei nélkül szeretné megtekinteni a programot az se keseredjen el, hiszen a ***Xen 2.0.6 Demo CD*** segítségével megteheti. Csak le kell tölteni a ***xendemo-2.0.6.iso***-t, kiírni egy ***CD***-re és egy bootolás után máris láthatjuk a saját gépünkön békésen egymás mellett élni a ***Linux 2.4*** vagy ***2.6*** kerneles verzióit, a ***NetBSD***-t és a ***FreeBSD***-t.

Ha megtetszett amit láttunk és kedvünk lenne telepíteni, akkor töltsük le a ➲ ***http://www.cl.cam.ac.uk/Research/SRG/netos/xen/downloads.html*** oldalról a stabil verziót.

A telepítés történhet előre fordított binárisból vagy forrásból. Szerencsére egyik sem túl bonyolult. Mindenek előtt olvassuk el a csatolt dokumentációkat. Tudom, hogy mindig



1. ábra A Xen rendszer felépítése

mindenki ezt írja, de ez talán nem véletlen. A felhasználói kézikönyv elég részletes, minden szükséges információt tartalmaz közérthető módon tálalva és szerencsére a projekt web oldalán is elérhető (➲ *http://www.cl.cam.ac.uk/Research/SRG/netos/xen/readmes/user/user.html*).
Amennyiben úgy gondoljuk, hogy minden érthető kezdjük a telepítést a binárissal. A letöltött állományt tömörítsük ki, majd az telepítőszkripttel telepítsük:

```
# tar -xvzf xen-2.0.6-install-x86_32.tgz
# cd xen-2.0
# ./install.sh
```

A folyamat végeztével a bináris állományt feltelepítettük. Ha a nagyobb kontroll kedvéért a forrást választanánk a következő lépések várnak ránk:

```
# tar -xvzf xen-2.0.6-src.tgz
# cd xen-2.0
# make world
# make install
```

A telepítő le fogja tölteni a *kernel.org*-ról a neki szükséges kernel forrását, ráteszi a saját foltjait, majd elvégzi a fordítást és a helyükre másolja az állományokat. Előfordulhat, hogy a telepítés megáll valamilyen fájl hiányára panaszkodva. *Debian* rendszeren a következő csomagok megléte szükséges:

```
# apt-get install make gcc libc6-dev bzip2
module-init-tools latex latex2html transfig tgif
bridge-utils zlib1g zlib1-dev python python-dev
python-twisted iproute libcurl3 libcurl3-dev
```

A későbbiek során szükségünk lehet a kernel újrafordítására. Ehhez lépjünk be a *Xen* forrásában található *linux-2.6.XX-xen0* könyvtárba, majd konfiguráljuk a *Xen arch*-ra a kernelt:

```
# cd linux-2.6.XX-xen0
# make ARCH=xen menuconfig
# cd ..
# make dist
# make install
```

Mind a bináris mind a forrásból telepített *Xen* állományokat helyez el a következő könyvtárakba:
/boot – a Xen; *domain0* és *domainU* kernelek;
/etc/xen – konfigurációs állományok
/etc/init.d – indító szkriptek
/lib/modules – kernel modulok
/usr – felügyeleti eszközök, dokumentációk, program könyvtárak, *Python* állományok

Amennyiben a használat során a rendszer megmagyarázhatatlannak tűnő lassulását észlelnénk, tiltsuk le a *TLS (Thread Local Storage)* program könyvtárat, mert jelenlegi verziójában nem kompatibilis a *Xen*-el.

```
# mv /lib/tls /lib/tls.disable
```

Amennyiben mégis hiányát éreznénk bármikor helyre tudjuk állítani az eredeti állapotot a könyvtár vissza nevezésével. Már csak egyetlen lépés maradt hátra, módosítanunk kell a *GRUB* konfigurációs állományát (a *Xen* a *GRUB* rendszerbetöltő meglétét igényli), hogy a frissen telepített *XenLinux* kernelről induljon a rendszerünk. Adjuk hozzá a `/boot/grub/menu.lst` állományhoz a következő sorokat:

```
title Xen 2.0 / XenLinux 2.6
  kernel /xen.gz dom0_mem=131072
  module /vmlinuz-2.6-xen0 root=/dev/hdXY ro
  ➥console=tty0
```

ahol `/dev/hdXY` helyére írjuk be a rendszerünk root partíciójának nevét. Ha mindezzel készen vagyunk indítsuk újra a gépet és győződjünk meg róla, hogy a megfelelő bejegyzést válasszuk a *GRUB* indító menüjéből.

Felügyeleti eszközök

A bootfolyamat során előbb a *Xen* néhány soros üzenetei láthatóak, aztán elindul a *XenLinux* kernel a *domain0* virtuális gépen. A *Xen* terminológiában a vendég operációs rendszereket *domain*-eknek hívjuk. A különböző domain-ek között kiemelt szerepet kap a domain0, amely felügyeleti jogokkal rendelkezik. Ez a szerkezeti felépítés látható az *1. ábrán*.
A *domain0*-án keresztül szabályozhatjuk a *Xen* működését a rendelkezésünkre álló felügyeleti eszközökkel. A felügyeleti tevékenységek nagy részét az *xm* nevű eszközzel fogjuk végezni. Az `xm` parancs megadási formája a következő:

```
# xm utasítás [kapcsolók] [argumentumok] [változók]
```

Lássuk az `xm` legfontosabb utasításai:
`xm list` – kilistázza a futó domain-eket;
`xm consoles` – információt nyújt a domain konzolokról;
`xm console` – konzolt nyit a paraméterként megadott azonosítóval (ID) rendelkező domain-hez;
`xm save` – elmenthetünk fájlba egy futó domain-t a domain0 fájlrendszerébe (a Debian Sarge nem tartalmazza a `libcurl.so.2` fájlt, amire a `save` utasításnak szüksége van, ezért készítsünk ilyen néven egy symlinket a `/usr/lib/libcurl.so.3.0.0` fájlról);
`xm restore` – az `xm save`-vel elmentett domain-t tudjuk ezzel a paranccsal elindítani;
`xm migrate` – ezzel a paranccsal lehetőségünk van egy xen-t futtató szerverről átmásolni egy domain-t egy másik

xen-t futtató gépre, akár úgy, hogy az adott domain-en futó szolgáltatások felhasználói ebből gyakorlatilag ne vegyenek észre semmit;
`xm create` – létrehoz egy új domain-t;
`xm shutdown` – leállít egy domain-t.

Az utasítások pontos paraméterezését megtalálhatjuk a dokumentációban vagy az `xm help utasítás` parancs kiadásával. Miután birtokában vagyunk az alapvető kezelési ismereteknek nézzük mit kell tudnunk a vendég operációs rendszerekről.

Megérkezik az első vendég

Példaként telepíteni fogunk egy ***Debian Sarge*** rendszert `debootstrap` segítségével, ezért telepítsük is fel:

```
# apt-get install debootstrap
```

Ha még nem tettük volna meg, indítsuk el a ***Xen*** démont:

```
# /etc/init.d/xend start
```

Az `xm list` paranccsal ellenőrizzük a futó domain-eket.

```
# xm list
Name      d  Mem(MB)  CPU  State  Time(s)  Console
Domain-0 0       123    0  r----      9.0
```

Látható, hogy a domain0 már fut, és csak annyi területet foglal a rendszermemóriából, amennyit részére a ***GRUB*** konfigurációs állományában beállítottunk. Vendég operációs rendszerünk viszont jól láthatóan még nincs. Hozzunk akkor létre egyet. Amennyiben van üres terület a merevlemezünkön egyszerű a helyzetünk. Alakítsunk ki két partíciót, egyet a root és egyet a swap számára, majd formázzuk meg őket. A cikk további részén helyettesítsük a példákba a létrehozott két partíció útvonalát ott, ahol az szükséges. Mit tegyünk viszont akkor, ha nincs elég szabad területünk? Természetesen használhatjuk valamelyik linuxos programot, hogy átméretezzük a meglévő partícióinkat, vagy kihasználhatjuk a xen azon lehetőségét, hogy a fájlokban létrehozott blokk egységeket is képes kezelni. Készítsünk egy-egy fájlt a root és a swap partíciók részére:

```
# dd if=/dev/zero of=/opt/domain1/root bs=1024k
➥count=2048
# dd if=/dev/zero of=/opt/domain1/swap bs=1024k
➥count=256
```

Az első paranccsal létrehoztunk egy 2048 megabyte-os fájlt „root” néven a root partíció részére az */opt/domain1* könyvtárban, míg a második paranccsal egy 256 megabyte-os fájlt ”swap” néven a swap partíciónak. Készítsük el a fájlrendszert ezekben az állományokban majd csatoljuk fel a root partíciót a loop eszközön keresztül:

```
# mkfs.reiserfs /opt/domain1/root
# mkswap /opt/domain1/swap
# mount -o loop /opt/domain1/root /mnt
```

Kezdjük el a rendszer telepítését a `debootstrap` segítségével:

```
# debootstrap --arch i386 sarge /mnt
➥http://ftp.hu.debian.org/debian
```

Amikor befejeződött a telepítés lépjünk be az új rendszerünkbe:

```
# chroot /mnt /bin/bash
```

Módosítsuk az új rendszer igényeinek megfelelően a következő konfigurációs állományokat:

```
# /etc/resolv.conf
# /etc/hostname
# /etc/hosts
# /etc/network/interfaces
# /etc/networks
# /etc/apt/sources.list
```

Az első három fájl kitöltésével nem lehet problémánk, de időzzünk el egy kicsit a hálózati beállításoknál. Mielőtt hozzá kezdenénk el kell döntenünk néhány fontos kérdést. Először is: szükségünk van-e egyáltalán hálózatra? Első hallásra talán butaságnak tűnhet a kérdés felvetése, de nem az. Habár a cikk címe szerint szervereket fogunk majd üzemeltetni a virtuális gépeken, de kifejezetten ajánlják például kernel fejlesztéshez is. Ebben az esetben valószínűleg tényleg nem lesz szükségünk hálózatra, hiszen a virtuális gépet elérhetjük az `xm` parancs `console` funkcióján keresztül. Ha maradunk a cikk eredeti irányvonala mentén és szervert készülünk telepíteni virtuális gépünkre, akkor viszont minden bizonnyal szükségünk lesz hálózatra. Ezzel el is érkeztünk a második megválaszolandó kérdéshez: bridge-elt vagy bridge-elt/routeolt hálózatot fogunk használni? A xen mindkét esetben virtuális hálózati interfészt hoz létre a virtuális gépekben, a különbség ezek működésében keresendő. Az első esetben az adott virtuális gép virtuális hálókártyáján ugyanazok a csomagok fognak megjelenni, mint a hálózathoz fizikailag kapcsolt kártyán. Minden virtuális kártya saját külső IP és MAC címmel fog rendelkezni, mintha az adott hálózathoz fizikailag is több gép lenne csatlakoztatva. A második esetet képzeljük el úgy, mintha az összes virtuális gépet egy router-en keresztül csatlakoztatnánk a hálózathoz. Válasszuk tehát az első lehetőséget akkor, ha minden virtuális gépünk saját külső IP címmel fog rendelkezni, a másodikat pedig akkor, ha a külvilág számára az összes gép egy IP címen lesz elérhető.
A bridge-elés a xen telepítésénél az alapbeállítás. Tételezzük fel, hogy a ***domain0***-ás gépünk külső IP címe 1.2.3.4, az 1.2.3.0/24-es hálózathoz tartozik és átjárója 1.2.3.254. Ebben az esetben a ***domain1*** hálózati kártyája részére osszunk ki egy szabad IP címet a 1.2.3.0/24-es tartományból és ezzel készen is vagyunk. Legyen az `/etc/network/interfaces` fájl tartalma például ez:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
  address 1.2.3.100
  network 1.2.3.0
  netmask 255.255.255.0
  broadcast 1.2.3.255
  gateway 1.2.3.254
```

LVM létrehozása RAID tömbön

```
# pvcreate /dev/md1                          # inicializáljuk a partíciót
# vgcreate storage /dev/md1                  # létrehozzuk a "storage" nevu volume group-ot a /dev/md1 raid tömbön
# lvcreate -L4096M -n dom1root storage       # létrehozunk egy 4GB-os partíciót  "dom1root" néven
# mkfs.reiserfs /dev/storage/dom1root        # leformázzuk az új partíciót
```

Partíció méretének csökkentése

```
# resize_reiserfs -s -1G /dev/storage/dom1root     # csökkentjük a fájlrendszer méretét 1GB-al
# lvreduce -L -1G storage/dom1root                 # majd csökkentjük a partíció méretét is 1GB-al
```

Partíció méretének növelése

```
# resize_reiserfs -s +1G /dev/storage/dom1root     # növeljük a fájlrendszer méretét 1GB-al
# lvextend -L +1G storage/dom1root                 # majd növeljükjük a partíció méretét is 1GB-al
```


Kicsit bonyolódik a helyzet, amennyiben a második lehetőséget választjuk. Jelöljünk ki egy hálózati tartományt a virtuális gépek részére, legyen mondjuk a 192.168.1.0/24-es. Végezzük el a szükséges módosításokat a ***domain0*** konfigurációjában. Adjunk hozzá az `/etc/network/interfaces` fájlhoz még egy interfészt:

```
auto xen-intbr
iface xen-intbr inet static
   pre-up brctl addbr xen-intbr
   post-down brctl delbr xen-intbr
   address 192.168.1.254
   network 192.168.1.0
   netmask 255.255.255.0
   broadcast 192.168.1.255
   bridge_fd 0
   bridge_hello 0
   bridge_stp off
```

Módosítsuk az `/etc/xen/xend-config.sxp` állományt a következő módon:

```
## Use the following if VIF traffic is routed.
# The script used to start/stop networking for xend.
(network-script network-route)
# The default script used to control virtual
➥interfaces.
#(vif-script vif-route)

## Use the following if VIF traffic is bridged.
# The script used to start/stop networking for xend.
#(network-script network)
# The default bridge that virtual interfaces should
➥be connected to.
(vif-bridge xen-intbr)
# The default script used to control virtual
➥interfaces.
(vif-script vif-bridge)
```

A ***domain1*** hálózati beállításait végezzük el a következőképpen:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
   address 192.168.1.1
   network 192.168.1.0
   netmask 255.255.255.0
   broadcast 192.168.1.255
   gateway 192.168.1.254
```

Ahhoz, hogy működjön a NAT-olás már csak egy teendőnk maradt, engedélyeznünk kell a csomagok továbbítását a ***domain0***-án.

```
iptables -A POSTROUTING -o eth0 -s 192.168.1.0/24
➥-j SNAT --to-source 1.2.3.4
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Ha a későbbiek során elérhetővé szeretnénk tenni valamely szolgáltatást a virtuális gépen, a tűzfalat kell módosítanunk. Példa: a ***domain1*** ssh szerverének elérése a külső IP címen és a 20022-es porton:

```
iptables -A PREROUTING -p tcp -m tcp -d 1.2.3.4 -
➥-dport 20022 -j DNAT --to-destination 192.168.1.1
```

A kis hálózati kitérő után folytassuk az első virtuális gép konfigurálását. Hozzuk létre az */etc/fstab* állományt a következő tartalommal:

```
/dev/sda1  /       reiserfs         errors=remount-ro  0  1
/dev/sda2  none    swap  sw         0      0
proc       /proc   proc  defaults   0      0
```

Ne lepődjünk meg az `sda1` és `sda2` partíciókon, a virtuális gépek így fogják látni a számukra kiajánlott partíciókat. Lépjünk ki a ***chroot***-ból, majd csatoljuk le az */opt/domain1/root* eszközt.

```
# exit
# umount /mnt
```

Elkészült az első vendég operációs rendszerünk. Hozzunk létre számára az `/etc/xen/domain1.conf` konfigurációs állományt. Használjuk bátran a minta állományokat, legyen a tartalma például valami hasonló:

```
name="Domain-1"
memory=128
kernel="/boot/vmlinuz-2.6-xenU"
nics=1
disk = ['file:/opt/domain1/root,sda1,w',
```

```
➥'file:/opt/domain1/swap,sda2,w' ]
root="/dev/sda1 ro"
vif = ['mac=00:06:AA:B5:C1:76']
```

Figyeljük meg a `disk` sort, itt állítjuk be azt, hogy a domain0-ban létrehozott eszközök a virtuális gép számára milyen partícióként látszódjanak. Amennyiben különálló partíciókat, vagy a későbbiekben ajánlott ***LVM***-ben létrehozott partíciókat használunk, a sort módosítsuk ilyen formán:

```
disk=['phy:/dev/storage/dom1root,sda1,w','phy:/
➥dev/storage/dom1swap,sda2,w']
```

Említést érdemelnek még a `memory` és a `vif` kezdetű sorok. Az előző az új virtuális gép által használható memória mennyiségét határozza meg, míg a második a virtuális gép virtuális hálókártyájának állít be ***MAC*** címet. Ez kifejezetten előnyös, az egyébként használatos automatikusan generált címekkel szemben. Elkerülhetőek így például a változó címekből adódó ***ARP*** feloldási problémák.
Elérkeztünk a nagy pillanathoz, indítsuk el első virtuális gépünket.

```
# xm create -f /etc/xen/domain1.conf
```

Amennyiben az előzőekben elfelejtettük volna lecsatolni a root partíciót, most hibajelzést kapnánk. A ***Xen*** figyel arra, hogy ugyanaz az eszköz ne legyen két rendszerben is írásra felcsatolva, mert az súlyos adatvesztést okozhat. Ilyen esetben figyelmeztetést kapunk és nem indítja el a domain-t. A log fájlokban egyébként bőségesen kapunk visszajelzést az esetlegesen felmerült hibákról. Ha mindent jól csináltunk nem kapunk hibaüzenetet és elindul virtuális rendszerünk.

```
Using config file "/etc/xen/domain1.conf".
Started domain Domain-1, console on port 9601
```

Ellenőrizzük újra a futó domain-ek listáját:

```
# xm list
Name      Id  Mem(MB)  CPU  State  Time(s)  Console
Domain-0  0      123    0   r----     15.9
Domain-1  1      127    1   r----    127.7    9601
```

Remekül látható, hogy fut az első vendég "Domain-1" néven 1-es ID-val. Csatlakozzunk az új rendszerhez:

```
# xm console 1
```

Kezdjük el friss ***Debianunk*** beállítását, erre használjuk a rendszer nyújtotta `base-config` segédprogramot. Ezután adjunk jelszót a root felhasználónak.

```
# /usr/sbin/base-config
# passwd
```

Ha még nem tettük volna meg, állítsuk be a hálózatot. Módosítsuk az `/etc/network/interfaces` fájlt a hálózatunk igényeinek megfelelően a korábbiakban leírtak alapján.

Frissítsük a rendszert az internetről és telepítsük fel az ***ssh*** szervert a könnyebb felügyelhetőség érdekében:

```
# apt-get update
# apt-get upgrade
# apt-get install ssh
```

Amikor készen vagyunk az alapvető beállításokkal lépjünk ki a virtuális gépből egyszerre megnyomva a ***Ctrl*** és ***]*** gombokat. Már csak egyetlen dolgunk maradt a végére, biztosítsuk, hogy a ***Xen*** démon elinduljon minden újraindulás alkalmával és elindítsa a vendég gépeket is:

```
ln -s /etc/init.d/xend /etc/rc2.d/S19xend
ln -s /etc/init.d/xend /etc/rc0.d/K90xend
ln -s /etc/init.d/xend /etc/rc6.d/K90xend
ln -s /etc/init.d/xendomains
/etc/rc4.d/S99xendomains
ln -s /etc/init.d/xendomains
/etc/rc0.d/K10xendomains
ln -s /etc/init.d/xendomains
/etc/rc6.d/K10xendomains
ln -s /etc/xen/domain1.conf
/etc/xen/auto/domain1.conf
```

Konklúzió

Térjünk vissza egy gondolat erejéig a cikk elején idézett definícióhoz. Ahogy látható a ***Xen*** valódi paravirtualizálást végez, nagy számú párhuzamosan futtatható virtuális szervert bírhatunk működésre ugyanazon a hardver eszközön és azok működését teljes körűen felügyelhetjük (monitorozhatjuk) a felügyeleti eszközök segítségével. Amennyiben szervereket szeretnénk üzemelni a létrehozott tartományokban, akkor valószínűleg fontos az adatbiztonság és a rugalmasság. Ha lehetőségünk van rá, akkor használjunk több merevlemezt, hozzunk létre ***RAID 5*** tömböt a vendég operációs rendszereknek, majd azon ***LVM*** segítségével alakítsuk ki a szükséges partíciókat (a szükséges lépéseket lásd az keretes listában). Ezzel a módszerrel egy könnyen konfigurálható és biztonságos fájlrendszert tudunk a ***Xen*** alá rakni, ami megfelel a fent említett igényeknek.
Saját tapasztalataim alapján azt mondhatom, hogy kiváló eszközt kapnak a rendszergazdák a kezükbe akár tesztelésre, akár éles üzemben való használatra. A rendszerek elmenthetőségének és visszaállításának, valamint a migrálásnak a lehetősége kiemelt rugalmasságot biztosít. Értékét tovább emeli a stabil, megbízható működés, és a – számunkra oly kedves – nyílt forrás, hiszen a szoftver teljes forrása ***GNU GPL*** védelme alatt érhető el.

Hábit József (Habit@osi.hu)
Ha mások kérdeznék, imádok sportolni, túrázni és főzni. Gyakorlatilag viszont zseniálisan kezelem a távirányítót és felnőtt fejjel sem vetem meg a számítógépes játékokat. Szeretem a focit és a sört!