DIPLOMAMUNKA

Soltész Dániel

Debrecen

2010

DEBRECENI EGYETEM

INFORMATIKA KAR

INFORMÁCIÓ TECHNOLÓGIA TANSZÉK

Virtualizációs technológiák összehasonlítása

Témavezető: Dr. Krausz Tamás
Egyetemi adjunktus

Készítette: Soltész Dániel
Programtervező matematikus

Debrecen
2010

Tartalomjegyzék

1. Bevezetés

Manapság egyre divatosabb az informatikával foglalkozók körében a *virtualizáció* szó. Ennek oka, hogy az elmúlt évtizedben megváltozott a hardver és a szoftver viszonya. Ugyanis hosszú évtizedekig a fő problémát az jelentette, hogy a készített szoftverek kiszolgálását a rendelkezésre álló hardver eszközök nem tudták maradéktalanul megvalósítani. Az utóbbi néhány évben azonban a hardver eszközök olyan rohamos fejlődésen estek át, hogy jelenleg már azzal a helyzettel állunk szemben, hogy nagy általánosságban a szerverek kihasználtsága nem éri el az 50%-ot.

Dolgozatomban megpróbálom összefoglalni a jelenlegi virtualizációs technológiákat, ahol csak lehet konkrét példákkal alátámasztva. Megkísérlem bemutatni a virtualizáció fajtáit, rétegeit, irányvonalait. A dolgozat második felében kiemelten foglalkozom a piacvezetőnek nevezhető VMware termékével az ESX Serverrel, a Microsoft Hyper-V lehetőségeivel, valamint a Citrix Xen Serverrel. Mindhárom esetben bemutatok egy kisebb virtualizációs környezet létrehozását. A folyamatokat a virtualizációban elterjedt klaszterek egy speciális esetére, a vendégekből létrehozott klaszterek szemszögéből vizsgálom meg, összehasonlítva a három megoldás lehetőségeit, korlátait.

2. A virtualizáció

Maga a szó mára erősen túlterheltté vált, nem konkrét megoldást fejez ki, hanem inkább filozófiai fogalom. Ez annak köszönhető, hogy több különböző technológia jelent meg, amit ezzel a névvel illetünk. Mégis van bennük valami közös, mégpedig az, hogy a hardvereszközök és a programok közé egy új absztrakciós réteget ültetnek be. Ez az új réteg kezeli a fizikai hardvert, és szabályozza, hogy ebből mi látszódjon a hardvert elérni kívánó programok felé.

A virtualizáció nem új keletű dolog, már a 60-as években is létezett olyan külön erre a célra fejlesztett IBM hardver (a neve System/370 volt), amely képes volt rá. A mindennapi életbe is betörő x86 architektúra tervezésekor azonban nem volt szempont, hogy a virtualizációt támogassa, így nagy áttörésnek számított, mikor a VMware mégis megvalósította azt, szoftveres megoldással. Azóta egyre csak szélesedik a virtualizáció felhasználásának köre [33].

2.1. A virtualizáció előnyei

Nem kell messzire mennünk, ha a virtualizáció előnyeire vagyunk kíváncsiak. A szerver konszolidációval egy fizikai szerverre több virtuális gépet telepíthetünk, akár mindegyiknek saját feladatot, szolgáltatást adhatunk, vagy ilyen módon csoportosíthatjuk őket. Ezáltal a régi kis szervereinket új, nagy gépekbe integrálhatjuk, mégis biztosítva az izolációt. A feladatok külön virtuális gépre helyezése segíti a hibakeresést, könnyíti az átláthatóságot, menedzselést. Nem kell minden feladatra külön számítógépet üzemeltetni, ezáltal csökken az energiafelhasználás, a helyigény, a hőtermelés, ugyanakkor növeli a rendelkezésre álló hardver kihasználtságát. Ezzel együtt a virtuális gépek izoláltak, az egyik meghibásodása nincs kihatással a futtató fizikai gépre, és egymáshoz is szigorú keretek között kapcsolódhatnak. A virtuális gépeket egyszerűen mozgathatjuk, törölhetjük, vagy hozhatunk újat létre fél perc alatt is, ezzel a rendszer skálázhatósága is nő.

Az alkalmazás virtualizációval megszabadulhatunk a minden egyes gépre való telepítés és beállítgatás fáradtságos munkájától, valamint leegyszerűsödik a patch-elés, és az új verzióra való átállás. Szintén növelhető az alkalmazások hordozhatósága [10].

Van azonban a virtualizációnak hátránya is, amiről kevésbé hallani, mégpedig az, hogy a központosítás igaz leegyszerűsíti a karbantartást, viszont ha ebben a központban történik hiba, akkor egyszerre veszíthetünk el mindent, egyszerre szűnhet meg minden futó szolgáltatás és program. Ezzel egy új veszély jelentkezik, egy új kockázati tényező, az üzemeltetőnek végül azt kell eldöntenie, hogy vállalja-e ezt, cserébe a fent említett előnyökért. Nos így már közel sem olyan egyszerű a döntés a virtualizáció mellett, mint azt az előnyök olvasása után gondoltuk volna. A probléma természetesen már a virtualizáció kezdete óta létezik, ezért régóta foglalkoztatja a szakmát, ennek eredményeképpen született a megoldás a fürtözés, avagy klaszterizáció formájában.

2.2. A virtualizáció fajtái

Elöljáróban annyit megemlítenék, hogy a szakirodalomban még nem egységes a terminológia ebben a témakörben. A hypervisor elnevezés okozza a legtöbb zavart, de az alkalmazás virtualizáció is könnyen félrevezeti az embert, ugyanis angolul több különböző technikát is nemes egyszerűséggel „application virtualization”-nek neveznek. De gyakori az is, hogy az elnevezéseket teljesen ellentmondóan használják. Megpróbálom a dolgozatban logikusan követni a szakirodalmat, amennyire ez lehetséges.

2.2.1. Erőforrás virtualizáció

Az operációs rendszerek már egy ideje végeznek bizonyos szintű virtualizációt, elég csak a processzorütemezésre gondolnunk. A programok azt hiszik, hogy övék a processzor, nincsenek tudatában annak, hogy valójában a másodperc törtrésze alatt többször is megállhat a program futása. A virtuális memória a másik szemléletes példa, ami egy hasznos eszköz a fizikai memória kibővítésére. Mivel a RAM drága, ezért általában kevesebb van belőle, mint amennyire szükségünk van. Új RAM vásárlása helyett az épp nem használt lapokat háttértárra mentjük, és ha szükséges, visszalapozzuk. Természetesen ennek ára van, amit a felhasználónak a sebesség csökkenésével kell megfizetnie. De ide tartoznak a virtuális optikai eszközt használó programok (Alkohol 120%), amik egy képfájlt töltenek be a virtuális meghajtóba. Megemlíthetjük még a hálózati eszközökben fellelhető VLAN technológiát,

mellyel teljesen különálló belső hálózatokat hozhatunk létre ugyanazon az eszközön, illetve több eszközön áthidalva. Valamint ide sorolhatóak az aggregált linkek is, mikor kettő vagy több linket egybeszervezünk, és ezután már az ezt kezelni képes hálózati eszköz egy interfészként kezeli, többszörösére növelve az adatforgalom áteresztőképességét. Ezek a módszerek „becsapják” a programokat arról, hogy valójában milyen hardverrel rendelkezünk, és az erőforrás virtualizáció fogalomkörbe tartoznak [2, 5].

2.2.2. Alkalmazás virtualizáció

Az alkalmazás virtualizáció egy gyűjtőfogalom, amely az olyan szoftveres megoldásokat foglalja magában, melyek segítenek a programokat hordozhatóbbá tenni, növelni kompatibilitásukat, és biztonságos környezetben történő futásukat teszi lehetővé. Ez azt is jelenti, hogy egyrészt az alkalmazást elválasztja az operációs rendszertől, másrészt megvédi a rendszert a hibás, vagy ártó szándékú alkalmazásoktól.

Munkánk során előfordul, hogy olyan programokat kellene futtatnunk, amely követelményei ellentmondanak egymásak, például más .NET framework szükséges működésükhöz. Ekkor is segíthet rajtunk, így ugyanis minden programot más és más környezetben helyezhetünk el. Ide tartozik még az az eset is, mikor az alkalmazást valójában egy szerveren futtatjuk, nem a kliensen. Vagy akár megtehetjük azt is, hogy egy programból több verziót is futtatunk a gépünkön, konfliktusok nélkül. Hogy egy konkrét példát is említsek, a Microsoft a Windows Vista operációs rendszertől kezdve bevezette a registry virtualizációt. Erre azért volt szükség, mert a védelem nagyobb szerepet kap a Vistában az elődeinél, a rendszer fontosabb bejegyzéseihez csak adminisztrátori jogokkal lehet hozzáférni. Hogy a korábbi alkalmazásokat emiatt ne legyen feltétlenül szükséges rendszergazdaként futtatni, a registrynek csak egy másolatához férnek hozzá [9, 18].

2.2.3. Alkalmazás szintű virtualizáció

Az alkalmazás szintű virtualizációt sokan keverik össze az alkalmazás virtualizációval. Azonban az alkalmazás virtualizáció nem a platformok közötti átjárhatóságra teszi a hangsúlyt, míg az alkalmazás szintű igen. Egyes források úgy is nevezik ezt, hogy alkalmazás szintű futtatókörnyezetek. A cél az, hogy különböző operációs rendszereken a programunk

gond nélkül futhasson. A Cygwin segítségével például Posix/GNU környezetet készíthetünk Windows alá, a Wine programmal pedig Linuxon tudunk exe állományokat futtatni, ez utóbbi akár működhet anélkül, hogy bármi módosítást kellene végrehajtanunk a kérdéses alkalmazáson. A legkézenfekvőbb és legmodernebb példa pedig a Java, amely a különböző architektúrákon és operációs rendszereken ugyanolyan saját szoftverkörnyezetet emulál, így a programoknak nem kell törődni azzal, hogy valójában milyen körülmények között futnak. A különféle operációs rendszerek egyetlen követelménye, hogy kész legyen a program rá, amely futtatni tudja a Java bájtkódját [2, 3].

2.2.4. Desktop virtualizáció

A desktop virtualizáció arról szól, hogy egy klienssel bejelentkezhetünk a számítógépünkre hálózaton keresztül, ezáltal megkapjuk a felületet, habár a programok nem a mi gépünkön futnak, csak a megjelenítés kerül hozzánk a hálózaton. Ez nem új találmány, és számos program létezik már, amivel ilyen módon átvehetjük az irányítást a gép felett (pl. VNC). Csak nemrégen kezdték el desktop virtualizációnak nevezni, hogy illeszkedjen ebbe a témakörbe. A Microsoftnak például beépített eszköze van erre a célra a Windows XP-től és az NT 4.0 Terminal Server Edition-től kezdve, mai nevén Remote Desktop Protocol (RDP) [5].

2.2.5. Emuláció

Az emuláció során a fizikai hardvertől teljesen különböző virtuális hardverelemeket hozhatunk létre. Ez az eljárás komoly segítség lehet, mikor például alkalmazásunkat más környezetben szeretnénk kipróbálni, más architektúrájú processzoron. A hardver emulálása igen költséges, ezért az emulációról elmondhatjuk, hogy habár nagyon hasznos eszköz, ha nehezen beszerezhető, vagy drága hardvert szeretnénk tesztelni, vagy akár nem létezőt kipróbálni, a teljesítmény gyakran a natív futtatáshoz képest 20%-os veszteséget is eléri. Emulációt alkalmazhatunk még régi alkalmazások futtatására is, például Amiga vagy C64 platformra írtakat, de a Dosboxot is érdemes megemlíteni, amivel DOS-t emulálhatunk. Az emulációs szoftver egy folyamat az operációs rendszerben, úgy viselkedik, mint bármely más program [2, 5, 29].

Az eddig leírtak első olvasásra talán jól elkülönülő módszereknek tűnnek, azonban nem ritkaság, hogy egy technológia egy másikra épül, abból vesz ötleteket, vagy egymással karöltve léteznek. A következő rész a platform virtualizációról fog szólni, de még előtte említést kell tenni a Kernel-based Virtual Machine-ről (KVM), ami a Linux kernel része a 2.6.20 verziója óta, és gyakorlatilag az emulációra épül. A QEMU ugyanis egy processzor illetve architektúra emulátor, és ezt használja a KVM, amit viszont tekinthetünk platform virtualizációs eszköznek is. Ez egy jó példa arra, hogy néhol mennyire elmosódnak a határok a virtualizáció témakörében [34].

2.3. Platform virtualizáció

Átfutottunk eddig pár lényeges technológiát, de az ok, amiért a virtualizáció olyan felkapott lett, az a platform virtualizálásban keresendő. Az eddig említett technológiák valamilyen speciális célt szolgáltak, például egy program futását tették lehetővé, vagy éppen könnyebbé. Amikor azonban egy platformot virtualizálunk, akkor nem elégszünk meg ezzel. Inkább azt várjuk el, hogy a virtuális gépünkben a virtualizált operációs rendszer ugyanúgy fusson, mintha a szokásos módon telepítettük volna, és kizárólagos hozzáférése lenne a hardver erőforrásokhoz.

Miért hódít a virtualizáció ezen fajtája? Miért vett az informatika új irányt a virtualizáció irányába? Az évek során rengeteg szerverpark, adattárház jött létre, és a felmérések azt mutatják, hogy ezeket nem használjuk ki. Ez annyit jelent, hogy a processzorok, memóriák, videokártyák, háttértárak idejük nagyobb részét „idle” állapotban töltik, kihasználtságuk mértéke alacsony, vagyis feleslegesen fogyasztják az áramot, termelik a hőt, sőt felesleges fenntartani miattuk karbantartó személyzetet, rendszergazdákat. Továbbá megfigyelhető, hogyha új szolgáltatást akar bevezetni a cég, akkor új szervereket szerez be, mert nem akarja a jól bejáratott infrastruktúrát megbolygatni. Rövidtávon ez az egyszerűbb megoldás. Vagy mégsem? Lehet ez ellen tenni?

A virtualizációval az ilyen típusú problémák kiküszöbölhetőek. Ha elég erős hardver áll rendelkezésünkre, radikálisan csökkenthetjük a szerverek számát, ezáltal költséghatékony infrastruktúrát hozhatunk létre. Ezen kívül pedig hozzájárulhatunk az egyre többet hangoztatott zöld informatika kialakításához is [17].

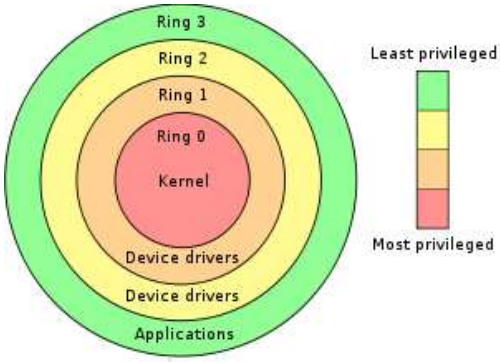
2.3.1. Operációs rendszer szintű virtualizáció

Ha egy meglévő operációs rendszert szeretnénk több példányban futtatni, akkor használatos ez a technika. A megvalósításhoz szükség van az operációs rendszerben egy speciális programra, amellyel új környezeteket hozhatunk létre, elszigetelve őket egymástól. Ekkor minden egyes környezet a közös kernelt használja, viszont saját partíciókat hozunk létre nekik. Egy ilyen partíció része egy saját hálózati interfész, lemezterület, felhasználói csoportok, memóriaterület, eszközök, stb. Így egy egész operációs rendszert kapunk, ennyiben több ez, mint az alkalmazás szintű virtualizáció. Ez a módszer nyílt forráskódú operációs rendszerek esetén alkalmazható, mint amilyen a Linux. A megvalósításból fakadóan csak az eredeti operációs rendszerünket tudjuk ilyen módon sokszorosítani, ám egy ilyen virtuális rendszer esetén a teljesítmény nagyon közelít a virtualizáció nélkülihez. Olyan esetekben szokták alkalmazni, mikor a felhasználóknak külön virtuális gépeket szeretnének adni. Jól beállítva a rendszert maximálisan kihasználhatjuk ezzel a technikával, de a rugalmatlansága miatt kevésbé elterjedt módszer. Így működik az OpenVZ és a VServer, de a FreeBSD Jail és a Solaris Containers is. Egy félmegoldásnak nevezhetjük a platform virtualizációban, itt még nem jelenik meg a hypervisor, egyes források úgy fogalmaznak, hogy a hypervisor maga az elsőként telepített operációs rendszer [1, 7].

2.3.2. A három módszer

Az x86 architektúrát úgy tervezték, hogy a rajta futó szoftverek azt higgyék, hogy kizárólagosan birtokolják az erőforrásokat, a programok pedig négy privilégiumszinten futhatnak. Ezek a szintek rendre a ring 0, ring 1, ring 2, ring 3 nevet viselik. A ring 0-ban futók bármilyen utasítást végrehajthatnak, míg a ring 3-ban futók nem, sőt tőlük elvárjuk, hogy bármilyen hiba esetén az ne legyen hatással a rendszer létfontosságú részeire. A népszerű operációs rendszerek (Windows, Linux) a középső két szintet szokás szerint nem használják, viszont szükség esetén lehet kényszeríteni használatukat, konfigurálni őket. De valójában igazán csak két futási módot különböztetnek meg: a supervisor módot, és a user módot. A supervisor mód tulajdonképpen egy flag értéke, ami meghatározza, hogy a program, ami éppen birtokolja a processzort ring 0-ban van-e vagy sem. Ez a flag dönti el, hogy engedélyezett-e a folyamatnak megváltoztatni bizonyos regiszterek tartalmát, módosítani a

leíró táblákat, vagy letiltani a megszakításokat. A supervisor mód egy olyan végrehajtási mód, melyben minden utasítás engedélyezve van, éppen ezért a rendszerfolyamatok ebben a módban futnak. Ugyanakkor különösképpen kell törekedni az ilyen módban futó programok biztonságára és megbízhatóságára, mert hiba esetén a rendszer lefagyhat, vagy fizikai kár is eshet a számítógépben.



1. ábra. Az x86 architektúra jogosultságkezelésének gyűrűi [32]

Az operációs rendszer szintűn kívül a platform virtualizálásnak három fő fajtáját ismerjük, ezeket szeretném most körbejárni. Mindhárom módszerben szerepet játszik a virtuális gépeket kezelő szoftver, a Virtual Machine Manager (továbbiakban VMM), amely felügyeli a virtuális gépeket, biztosítja számukra az erőforrásokat, és menedzseli a környezetüket.

Az első mód az úgynevezett szoftveres virtualizáció, ami úgy működik, hogy a VMM tulajdonképpen egy program az operációs rendszerben. Ez az emuláció egyik fajtájának is nevezhető. Ide tartozik a Java VM, és a .NET Common Language Runtime (CLR). A Java-t már említettem, de akkor az alkalmazások szempontjából, most pedig a virtuális gépe szempontjából, amelyben a Java programok futnak. A szoftveres virtualizáció a legrugalmasabb a platform virtualizációs módszerek között, mert a virtuális gépnek bármilyen utasításkészletet képes biztosítani, az sem baj, ha a tényleges fizikai processzorétól eltér. A futtatás segíthető futás idejű fordítással (JIT, röpfordító).

A második megközelítés, amikor az operációs rendszer maga a VMM, vagyis beépített módon rendelkezik a VMM komponenssel, egy kernel szintű modullal. Ezt egyes források nevezik hibrid vagy hosted megoldásnak, a wikipedia pedig ezt nevezi Type-2 hypervisornak. Az ide tartozó konkrét példák főleg kliens megoldások, úgymint a VMware Workstation, VMware Player, Microsoft Virtual PC és Sun VirtualBox.

Végül a harmadik eset, mikor a VMM az operációs rendszer és a hardver közé kerül. Ezt hívjuk bare-metal virtualizációnak, vagy Type-1 megoldásnak. A dolgozatomban csak ebben az esetben hívom majd a VMM-et hypervisornak. Úgy kell elképzelnünk ezt a módszert, hogy mikor telepítjük az operációs rendszert, akkor azzal feltelepül a hypervisor, és ez a kis program fogja vezérelni az eredetileg a fizikai hardverre telepített operációs rendszert is. Vagyis a hypervisor nem osztozik senkivel, ami a legerősebb privilégiumszintet illeti. Az ide tartozó példák a VMware ESX, Citrix XenServer és a Microsoft Hyper-V [7, 35].

Akkor már látjuk azt, hogy miként kapcsolódhat a VMM az operációs rendszerhez. De tulajdonképpen mit csinál a VMM? Hogyan hiteti el magáról egy programmal, hogy ő egy operációs rendszer, aki ráadásul valós fizikai hardveren fut, és hogyan biztosítja számára az erőforrásokat? Mi is kell tulajdonképpen egy platform virtualizáláshoz? Azt mondhatjuk, hogy alapvetően három részre kell bontanunk ezt a kérdéskört: a processzor, a memória, és a perifériák virtualizálására.

2.4. A processzor virtualizálása

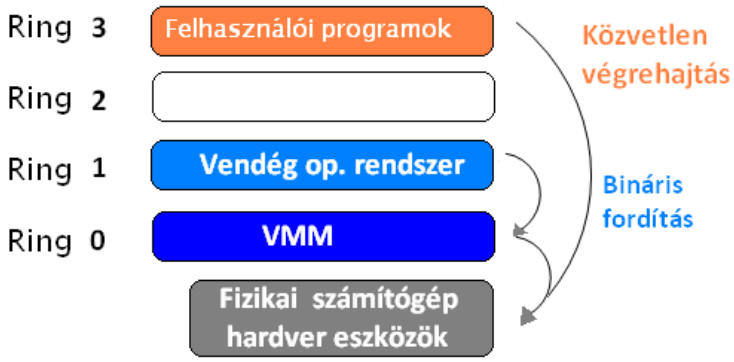
Az x86 architektúrában vannak utasítások (számszerint 17, pl. POPF, PUSHF/PUSHFD) melyeket csak supervisor módú folyamat adhat ki, illetve ha más adja ki, akkor az eredmény más lesz, vagy nem ugyanúgy hajtódik végre, mintha a folyamat Ring 0 privilégiumszinttel rendelkezne. A három virtualizációs technika különbsége pontosan abban rejlik, hogy miként oldja meg ezt a problémát, miként valósítja meg az operációs rendszerben keletkező privilegizált parancsok végrehajtását. Az mindig igaz, hogy az operációs rendszer és a tényleges hardver közé egy új réteget kell beszúrni [27].

2.4.1. Teljes CPU virtualizáció – bináris újrafordítással

A teljes CPU virtualizációt a VMware 1998-ban mutatta be, ez volt az első megoldás az x86 processzor virtualizálására. Kifejlesztették a bináris újrafordítás technológiáját, ami a direktben történő futtatással kombinálva jó sebességet tud elérni a natív futtatáshoz képest is.

Szemléletesen úgy képzelhetjük el, hogy a VMM kerül a Ring 0-ba, míg az operációs rendszer más szintre szorul, az alkalmazások Ring 3 szintje és Ring 0 közé. A user módú

utasítások továbbra is natív módon futhatnak a hardveren, így az olyan gyorsasággal történik, mintha nem is történt volna virtualizáció. A syscall vagy int rendszerhívások ugyanúgy a Ring 0-ba váltást eredményezik, viszont ott már a VMM helyezkedik el, akinek kezelnie kell ezeket. Meg is teszi, mégpedig úgy, hogy a kernelnek továbbítja a hívásokat, valamint ha magától a kerneltől érkezik privilegizált utasítás (kisebb privilégiumszintről), akkor a VMM elkapja, esetleg átalakítja más utasításokká (trap and emulate), de oly módon, hogy a virtuális gépben ugyanaz legyen az eredménye, mintha közvetlenül fért volna hozzá a hardverhez, vagyis mintha Ring 0-ban futna.



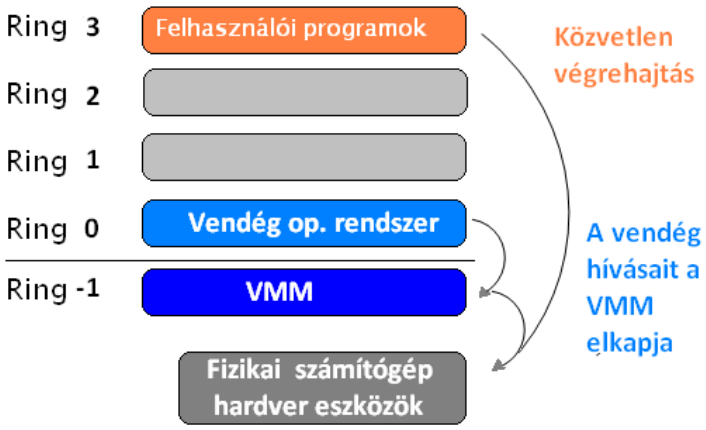
2. ábra. Teljes CPU virtualizáció [20]

Minden operációs rendszert lehet ilyen módon virtualizálni, ezért nagyon közkedvelt technika. Főleg azért, mert sem hardveres támogatást nem igényel, sem az operációs rendszerben nem kell módosításokat végezni, így a Microsoft Windows-t is használhatjuk, aminek mint tudjuk nem nyílt a forráskódja.

Ez a módszer magas szintű izolációt és biztonságot nyújt a virtuális gépeknek, és megkönnyíti a migrációt, valamint a hordozhatóság is nagymértékben jelen van. A VMware módszere tekinthető de facto-nak teljes CPU virtualizáció terén, de az egész iparágat tekintve nem született megegyezés arról, hogy szabványosítsák a virtualizáció megvalósítását, és menedzselését, így minden cégnek szabadon kell megválasztania, hogy miként kezeli az ilyenkor felmerülő problémákat [20].

2.4.2. Natív CPU virtualizáció

A natív vagy más néven hardveresen támogatott CPU virtualizáció fő jellemzője, hogy csak olyan processzorokkal működik, amelyek rendelkeznek olyan utasításkészlettel, amik segítik a virtualizációt. Az Intel és az AMD oldalán is igaz, hogy a 2006 után gyártott processzorok rendelkeznek ezzel a képességgel, a technológia neve pedig az Intelnél Intel VT-x (illetve Intel VT-i), az AMD-nél pedig AMD-V. Ez tehát egy új követelmény a teljes CPU virtualizáláshoz képest, viszont megmarad az a jó tulajdonság, hogy az operációs rendszert nem kell külön felkészíteni a virtualizálásra.



3. ábra. Hardveresen támogatott CPU virtualizáció [20]

A két új processzortechnológia új utasítások bevezetését (pl. VMXON, VMXOFF), és egy új privilégium szintet hoz be a rendszerbe, amit nevezhetünk Ring -1 –nek. Ezt hivatkozhatjuk root módként, illetve az ide került a VMM úgynevezett hypervisor módba kerül, mely még egy szinttel erősebb, mint a supervisor mód. Így nem kell azzal vesződni, mint a teljes CPU virtualizációnál, hogy a rendszerhívásokat, amik Ring 0-ba vittek, visszaadjuk az operációs rendszernek. Ez az új réteg kezeli a supervisor mód szükségességét megkívánó utasításokat, a user módúak továbbra is közvetlenül futtathatóak a hardveren. A root vagy hypervisor módba váltás (VMCALL) és a visszaváltás (VMRESUME) költsége azonban még igen nagy, így jelen pillanatban a legtöbb esetben ez a CPU virtualizáció sebességben elmarad a teljes CPU virtualizációtól, azonban a gyártók minden újabb processzorral lefaragnak egy kicsit a hátrányból [7, 19, 20].

2.4.3. CPU virtualizáció paravirtualizációval

A paravirtualizáció feladja azt az előnyt, hogy nincs szükség az operációs rendszer kernelét módosítani. Ha ezen változat mellet döntünk, akkor bizony fel kell készíteni a kernelt, hogy ne adjon ki olyan utasításokat, melyhez Ring 0 privilégiumszint szükséges. Ezt úgy oldja meg, hogy az ilyen utasításokat új, speciálisra cseréli, de megkapja közvetlenül a Ring 0-ba való hívás lehetőségét is a hypercall hívással. Az operációs rendszer a Ring 1-be kerül, a VMM pedig a Ring 0-ba, mint a teljes CPU virtualizáció esetében, és tudatában van annak, hogy ő virtualizált állapotban fut. Mivel a Windows kernelét nem tudjuk módosítani, így tisztán paravirtualizációt nem lehet alkalmazni a processzorra, az új Linux disztribúciók viszont már fel vannak készíve rá. Sebesség tekintetében nagyon jó megoldás, csak néhány százalék a teljesítményveszteség [4, 20].

4. ábra. Paravirtualizáció [4]

2.5. Memória virtualizálása

A memóriát első körben az operációs rendszerek virtualizálják. A fizikai memóriánál így sokkal nagyobb memóriamennyiséget hozhatunk létre virtuálisan, a címtér a többszörösére növelhető. Minden modern architektúra része az úgynevezett Memory Managment Unit (MMU), amelynek feladata elvégezni a leképezést a virtuális memória és a fizikai memória címtere között. A virtualizált gépeknek is megvan a maga virtuális címtere, így a tényleges memóriát elérni a virtuális gépekből kétszeres címátfordítást igényelne, ám ez költséges. Ehelyett támadt az ötlet, hogy virtualizálni kell az MMU-t. Létre kell hozni egy új táblacsoportot, az úgynevezett árnyék laptáblákat (shadow paging tables), amik tartalmazzák

a virtuális gépek memóriája és a valós fizikai memória közötti kapcsolatokat. Ha egy virtuális gép módosítani akarja a laptábláját, akkor 3 módszer van a konzisztencia megőrzésére.

Az első szerint engedélyezett a módosítás, majd utána a megfelelő árnyéktáblákat szinkronizálni kell. Ez a legegyszerűbb, de legköltségesebb megoldás. A második verzió, hogy a virtuális gép nem módosíthatja közvetlenül a laptábláit, csak a VMM-en keresztül. Végül a legújabb technológia hardveresen kezeli le az árnyéklapok konzisztenciájának megőrzését, de ez a funkció csak a legmodernebb processzorokban érhető el (Intel Core i7). Az árnyéktábla szinkronizálás a szoftveres virtualizációnak, a VMM-en keresztüli a paravirtualizációnak, a harmadik pedig a hardveresen támogatott memória virtualizációnak felel meg, ha hűek akarunk maradni a fogalomkörhöz [7].

2.6. Perifériák virtualizációja

Perifériák virtualizálásakor is kirajzolódik ez a három lehetőség, amit eddig megfigyelhettünk a CPU és a memória esetében. Az analógiához híven elsőként a szoftveres megoldást említem, amit periféria emulációnak nevezünk. Ehhez szükséges, hogy megfelelően utánozzuk az igazi hardver működését, a megszakításait, regiszterkészletét, és a közvetlen memória elérési módú működését (DMA). Ezzel van a legkevesebb probléma, mert kihasználható az, hogy sok eszközmeghajtó van beépítve az operációs rendszerbe, így nem kell azzal foglalkozni, hogy telepítsük. Az viszont megint csak igaz erre a szoftveres technikára, hogy eléggé erőforrás-igényes megvalósítás.

Paravirtualizált perifériák is léteznek, ekkor az történik, hogy nem létező hardverelemeket emulálunk, amiket azzal a céllal hozunk létre, hogy minél egyszerűbb legyen a használatuk, és a vezérlésük minél hatékonyabb legyen. Magasszintű műveleteket biztosítunk a virtuális gép és a host között. A baj ezzel a technikával, hogy ezeket a drivereket a host gépre telepítenünk kell (pl. Hyper-V szintetikus eszközei).

Végül pár szó a perifériák hardveres virtualizációról. Az igaz hogy gyorsabb a másik kettőnél, ám felvetődnek bizonyos kérdések, amelyekre érdemes odafigyelni. Nem elhanyagolható az a tény, hogy az ily módon használt periféria csak egyetlen virtuális gép számára lesz elérhető. A másik, hogy nem minden eszköz képes támogatni a hardveres virtualizációt, és ráadásul alkalmazása új biztonsági kockázatot jelent a rendszerben. Ugyanis bizonyos eszközöket úgy lehet programozni, hogy a fizikai memória olyan részéhez is

hozzáférjenek, ami nem az adott virtuális géphez tartozik, ami akár végzetes rendszerhibához is vezethet. Ennek ellenére előfordul, hogy csak ez az út járható, például ha egy fizikai eszközhöz nincsen a virtualizált operációs rendszerben meghajtó [2, 5, 7].

3. A VMware, a Microsoft és a Citrix

A VMware után a Citrix és a Microsoft is szerette volna kivenni a részét a virtualizációban rejlő lehetőségek megvalósításában, és az ebben rejlő piac kiaknázásában. Vannak server és desktop megoldások, a szerver megoldások pedig tovább oszthatók. Általában elmondható, hogy a kis és középvállatoknak nem mindig ugyanaz a szoftver az előnyös, mint az óriásoknak. Ettől azonban bonyolultabb a helyzet, ugyanis nem lehet éles határvonalat húzni, vagy megállapítani csupán a cég méretét tekintve, hogy melyik a célravezetőbb beruházás virtualizáció terén. Inkább az figyelhető meg, hogy vannak elméletben elfogadott irányelvek, de hogy konkrétan ki mit választ, rá van bízva.

A desktop illetve a server termékek között az a különbség, hogy míg előbbieknek a lényege a könnyű kezelhetőség, az átlátható grafikus felület és a kompatiblitás növelése, addig utóbbi egész más célokat helyez előtérbe, mint például klaszterizáció, magas rendelkezésre állás (high availability, HA). A VMware termékpalettáján található a GSX utódja a VMware Server, a VMware ESX, és a legújabb a VMware ESXi a szerverre szánt alkalmazások között, míg a kliens oldaliak a Workstation, a VMware Fusion, és a VMware Player. A Microsoftnál a kliens oldalinak Virtual PC, szerver oldalon pedig a Hyper-V, valamint a Virtual Server található. A XenServer inkább szerverre szánt alkalmazás.

3.1. Klaszterek

Klaszterek segítségével a számítógépeinket elméleti csoportokba rendezhetjük, erőforrásaikat bizonyos szinten egyesíthetjük, és képesek lehetünk átcsoportosítani a feladatokat egyik szerverről a másikra. Igaz, ez nem azt jelenti, hogy a gépeinket egy nagyként tudjuk használni, de a különálló szerverektől ez mégis sokkal rugalmasabb és dinamikusabb rendszer. Egy klaszter létrehozásához ezekből kifolyólag legalább két hostra van szükség, és egy külön programra, amely vezérli.

Kiemelten fogok foglalkozni a guest clusteringgel, ami azért különleges, mert nem fizikai gépeket, hanem virtuálisakat szervez klaszterbe. Feltehetjük a kérdést, hogy vajon mi értelme van egy ilyen klaszternek? Hiszen a klaszter lényege, hogy a meghibásodások esetén az erőforrások átcsoportosíthatóak, és ha virtuális gépekből építünk klasztert akkor tulajdonképpen egyetlen fizikai szerveren zajlik minden, így hiba esetén semmivel sem jutottunk előrébb. Igen ám, de felmerülhetnek olyan esetek, mikor maguk a virtuális gépek jelentik a szolgáltatást (virtuális gép hosting), és ilyen esetben kiemelten fontos, ha az egyik host kiesik, akkor egy másikon képes legyen újraindulni a virtuális gép. Ha csak egy fizikai gépünk van, akkor ez probléma lehet. Vagy ha olyan operációs rendszert futtatunk a szervergépen, aminél stabilitási problémák jelentkeznek, akkor jól jöhet egy tartalék, ami át tudja venni a feladatát. Ha telik az erőforrásból, akkor mindig jó, ha van egy kis redundancia, amennyiben minőségi szolgáltatást szeretnénk nyújtani. Végül pedig megemlítem, hogy a guest clustering nem jelenti azt, hogy kizárólagosan virtuális gépeket szervezünk a klaszterbe, virtuális és fizikai gépek keverése is megvalósítható, így ez a technika nem zárja ki a megszokott klaszter megoldást.

3.2. A VMware ESX és a Hyper-V összehasonlítása

AZ ESX első verziója hét évvel korábban (2001) jelen volt már, mint a Microsoft terméke a Hyper-V, ami a Windows Server 2008-al érkezett. Az ESX jelenleg a 4.0 verziónál jár már, amely 2009 decemberében látott napvilágot. Mind a két megoldás Type 1, azaz a bare metal típusú, tehát a VMM mint hypervisor jelenik meg, ennek ellenére mégis van egy fontos különbség a felépítésükben. A VMware ESX úgynevezett monolitikus, míg a Hyper-V microkernel alapú hypervisor. A különbségük az, hogy a fizikai hardvert vezérlő eszközkezelők a monolitikus hypervisornak részei, a microkernel alapúnál viszont nem, ott a host operációs rendszerben találhatóak. Ennek következménye, hogy az ESX hypervisora több kódot tartalmaz, illetve az eszközmeghajtókat a VMware-nek kell szolgáltatnia. Ezzel szemben a Hyper-V esetében az eredeti driverek használhatóak, és a hypervisor csak a processzorütemezésért, valamint a memóriakezelésért felelős. Egyesek számára fontos különbség lehet, hogy a Hyper-V csak hardveres virtualizáció támogatással fut (AMD-V, Intel VT-x, Intel VT-i). Túl azon, hogy a régebbi processzorok nem rendelkeznek ilyen

tulajdonsággal, szeretném felhívni a figyelmet az esetleges BIOS beállításokra, ugyanis előfordulhat, hogy az alaplapban le van tiltva ez a képesség.

Egyszer velem történt meg, hogy egyik napról a másikra a Hyper-V-vel futó virtuális gépek nem voltak hajlandóak elindulni, semmit nem lehetett kezdeni velük, a hibaüzenet a virtuális BIOS-ra hivatkozott. A teljes operációs rendszert újrainstalláltam, viszont nem engedte feltenni a Hyper-V role-t. A BIOS-ban minden úgy volt, ahogy lennie kell, hiába kapcsoltam ki-be az opciót, nem segített a problémán. Végül a CMOS resettel sikerült újra életre keltenem a hardveres virtualizációt. VMware esetében ez úgy jelentkezhet, hogy csökken a teljesítmény.

Processzorhasználat terén azt mondhatjuk, hogy mindkét termék egyre több processzort képes kezelni, ahogyan jönnek az újabb verziók, ezért csak tájékoztató jelleggel írom, hogy jelenleg legalább 64 logikai processzort támogatnak. Van azonban olyan pont, amiben még nem egy úton jár a két termék. Ez pedig a processzor maszkolás, ami a VMotion illetve Live Migration technikához elengedhetetlen. Hogy megvalósítható legyen, a két fizikai processzornak nagyon hasonlítani kell, legalábbis egy családba kell, hogy tartozzanak. A VMware ESX kidolgozottabb CPU maszkolással rendelkezik, ezért nála a VMotion szélesebb körben alkalmazható, mint a Live Migration a Hyper-V esetében.

Ha a memóriahasználat szempontjából vagyunk kíváncsiak a két termékre, akkor azt látjuk, hogy a Hyper-V 1 terrabájt RAM-ot képes használni, szemben az ESX 256 gigabájtjával. Virtuális gépenként mindkét esetben 64 gigabájt engedélyezett, és számolni kell egy bizonyos mértékű plusz memóriaigénnyel virtuális gépenként. A Hyper-V esetében ez a többlet 1 gigabájt alatt 32 megabájt, felette pedig minden gigabájthoz 8 megabájt pluszt kell számolni. Tehát 2 gigabájtos virtuális gép memóriaigénye 2088, 6 gigabájtosé 6192 megabájt. Az ide tartozó megfelelő értékek az ESX esetére a 1. számú mellékleten találhatóak. Igaz az ESX 3.0 még kisebb overheaddel működött, de még az se körözte le a Hyper-V-t. Rögtön látszik, hogy a Hyper-V–vel kedvezőbbek az értéket, viszont a személyes tapasztalatom, hogy adott erőforrások mellett mégis az ESX-szel lehet több virtuális gépet futtatni. Ennek az az oka, hogy a VMware kifejlesztett néhány speciális képességet a memóriafoglalás terén, melyek az előnyére válnak.

Az első a memória túlfoglalás képessége, ami lehetővé teszi a virtuális gépek számára, hogy több memóriát foglaljanak le maguknak, mint a valós fizikai memória. Hogyan lehetséges ez? A VMware úgy oldja meg, hogy a virtuális gépek egymástól elvehetnek

memóriát, amennyiben erre szükség és lehetőség van, vagyis ha egy virtuális gép kevesebb memóriára tart igényt, mint amennyit kijelöltünk számára, akkor a felesleget a hypervisor képes átadni másiknak, ami éppen szűkében van ennek az erőforrásnak. Ilyen szempontból az ESX rugalmasabbnak mondható, ám van ennek veszélye is. Mégpedig az, ha a memória kihasználtsága nagy, akkor előfordulhat, hogy miután elvettünk egy géptől memóriát, de később mégis szükség lenne rá, viszont akkor már nem lehet visszaadni neki, mert közben minden memóriát elhasználtunk. Ez által az operációs rendszer nagymértékű lapozásba kezdhet, melynek eredménye a nagyfokú sebességcsökkenés! Ezért a VMware is arra kéri a felhasználóit, hogy óvatosan bánjanak ezzel a képességgel.

A másik figyelemreméltó technika az úgynevezett osztott memória módszer, avagy memória megosztás, ami akkor válik hasznunkra, ha a virtuális gépek azonos operációs rendszert futtatnak. Az ESX képes kihasználni, hogy ezen operációs rendszerek ugyanazokat a fájlokat használják a működésükhöz, és így bizonyos memórialapokat nem kell annyi példányban eltárolni, ahány virtuális gép fut. Ezt kihasználva akár 10-20%-al is csökkenhet egy virtuális gép memóriaigénye, ami ha sok példányt futtatunk, fontos szempont lehet.

Érdemes ejteni egy pár szót a támogatott vendég operációs rendszerekről. A Hyper-V-ről sokan úgy tartják, hogy főleg Windows vendégek számára készült, és ez így is van, de Linux is támogatott, mégpedig a SUSE Linux Enterprise Server 10 SP3 és a Red Hat Enterprise Linux 5.2–től kezdődő verziók, továbbá tervben van a Solaris x86-ra írt változata is. Miért is fontos ez? Igaz, hogy ha nem támogatott egy operációs rendszer, annak ellenére néha lehet ügyeskedni, hogy fusson a Hyper-V alatt, viszont nagyon fontos a teljesítmény szempontjából, hogy a szintetikus eszközök, amiket a Hyper-V nyújt a virtuális gépeknek, kompatibilisek legyenek a vendég operációs rendszerrel. Ezek a szintetikus eszközök azok, amik biztosítják a kapcsolatot a VSP-vel (Virtual Service Provider) egy úgynevezett VMBuson keresztül, ami a hypervisoron nem megy át. Tehát Hyper-V esetében amennyiben létezik az operációs rendszerhez IC csomag, akkor ezek a szintetikus eszközök gyors I/O alrendszert alkotnak, ami nem érintkezik a hypervisorral, mikor a virtuális gép eléri a szülő partíció tényleges drivereit. Ha nincs megfelelő IC csomag, akkor viszont marad az emuláció, ami lassú mind a szintetikus eszközök, mind a monolitikus ESX-hez képest. A VMware terméke szélesebb kínálatot nyújt a támogatott operációs rendszerek esetében, a pontos táblázatot a http://www.vmware.com/resources/compatibility/search.php linken tekinthetjük meg.

A VMware és a Microsoft is rendelkezik olyan technológiával, amellyel magas rendelkezésre állású rendszert hozhatunk létre. A Microsoftál Failover Cluster, a VMwarenél pedig High Availability (illetve DRS) névvel illetik. Mindkettő célja, hogy a virtuális gépek elválaszthatóak legyenek a futtató hardvertől, ezzel egy magasabb absztrakciós szintet hozva létre. Az mindkét oldalon előfeltétel, hogy a virtuális gépek fájljai egy olyan helyen legyenek tárolva, amely hozzáférhető minden résztvevő host számára. Az ilyen tárolók tipikusan drága eszközök. Az így kialakított adattárakat SAN-oknak (Storage Area Network) szokták nevezni. Azonban aki nem engedheti meg magának, vagy szimplán nincs szüksége ezekre az eszközökre, azoknak szerencsére léteznek programok, amik segítenek a helyzeten. Hogy hogyan? Természetesen virtualizációval [6, 7, 15].

3.3. A virtuális SAN

Ha egy cég úgy dönt, hogy számára fontos az IT részleg megbízhatósága, és gyorsasága, akkor szüksége van egy minőségi adattárházra is. Hogy milyen szempontokat vehet figyelembe a választáskor, illetve hogyan alakítsa ki a környezetet, miközben a legjobb teljesítményt és a legbiztonságosabb infrastruktúrát kapja, ez külön szakdolgozati téma lehetne. Így én most nem feszegetem mélyebben ezt a témakört, helyette egy példán keresztül bemutatom, hogyan hozhatunk létre egy virtuális SAN-t.

Több szoftver is kínál hasonló megoldást, én az Openfiler nevű szoftvert fogom használni, amely ingyenesen letölthető, az x86 és x64 platformot egyaránt támogató szoftver, és az rPath Linux-ra épül. Hogy virtuális SAN készítésére használhassuk, létre kell hozni számára egy virtuális gépet, amiben futhat.

3.3.1. Az Openfiler telepítése

Járjuk végig hogyan kell telepíteni az Openfilert VMware Workstation-re. Az új virtuális gép varázslóban az első ablakban válasszuk a Custom (advanced) módot, majd a Hardware compatibility-hez az alapértelmezett beállítás (Workstation 6.5) megfelelő. Állítsuk be az openfiler iso fájlt az optikai meghajtóhoz. Ezután operációs rendszernek a Red Hat Linux-ot kell kiválasztani és 1 db processzort, valamint tapasztalatom szerint 256 MB memóriával tökéletesen működik. A hálózatot Bridged network-re állítsuk, és a SCSI adaptert pedig LSI

Logic-ra, erre különösképpen figyeljünk. Érdemes két virtuális merevlemezt létrehozni számára: először amire feltelepítjük az Openfilert, ez egyébként elég, ha 3 gigabájt méretű, a másik pedig a létrehozandó adattár legyen igény szerinti mérettel. Mivel a varázslóval csak egy virtuális lemezt lehet létrehozni, így ha elkészült, szerkesszük a kész gépet, és adjunk még hozzá egy lemezt, a leendő adattár virtuális lemezét. Ez utóbbi SCSI típusú legyen, mert úgy gyorsabban felismeri majd az ESX, mintha IDE lenne, az operációs rendszeré viszont lehet IDE is. Mindkettőt egy fájlként tároljuk (Store virtual disk as a single file). A teljesítményt valamivel növelhetjük, ha lefoglaljuk előre a merevlemez méretét, viszont ez kis időbe kerülhet, függően a merevlemez méretétől. Adjuk meg a helyüket, majd fejezzük be a varázslót.

Mielőtt folytatnám, két megjegyzést fűznék ide. Ha nem jól konfiguráltuk a merevlemezeket, akkor a Linux telepítő megállhat, mikor a lila háttér és a fehér openfiler felirat megjelenik. Ez esetben ellenőrizzük le a virtuális merevlemezeket, szükség esetén készítsünk újat. A második megjegyzés, hogy a telepítő a nyelv kiválasztása után „Searching for Openfiler NSA installations” felirattal megállhat. Ez esetben állítsuk le a virtuális gépet, szerkesszük: távolítsuk el a virtuális floppyt, majd próbáljuk újra.

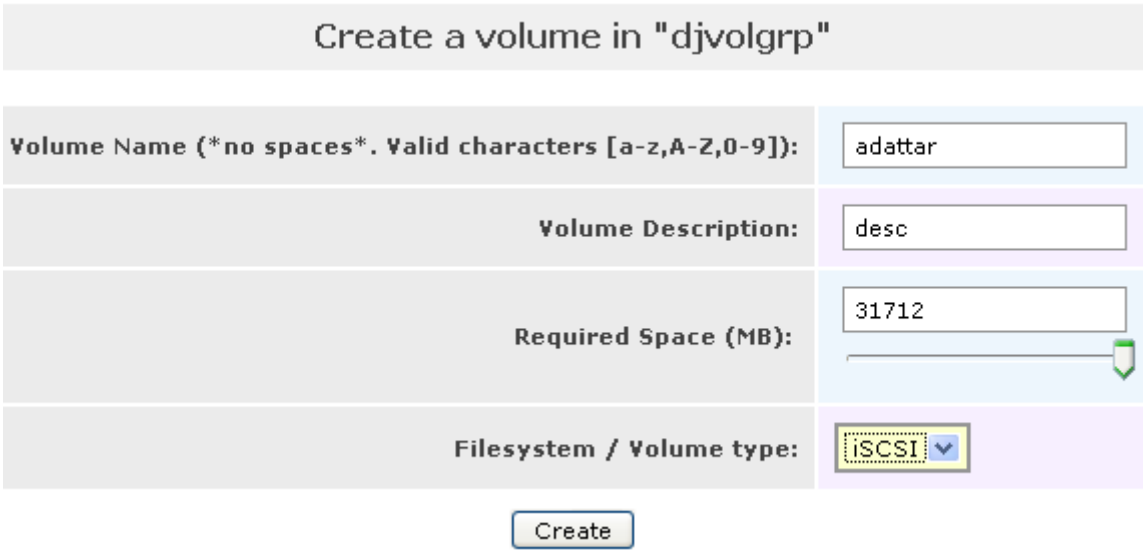
Ezután a lemezek partícionálása következik, ahol a manuális módszer javasolt. Itt egy /boot illetve Linux lévén egy /, és egy swap csatolási pontot is létre kell hozni ilyen sorrendben az Openfiler rendszerlemezére. Ha nem hozunk létre /boot csatolási pontot, akkor előfordulhat, hogy nem tud betöltődni a rendszer. A swap terület legyen legalább akkora, mint amennyi virtuális memóriát biztosítottunk ennek a gépnek. Amikor új partíciót hozunk létre, figyeljünk arra, hogy csak azon merevlemez mellé tegyünk pipát, amelyiken létre szeretnénk hozni. A leendő adattárra most még nem szükséges partíciókat létrehozni. Ezt követően a területi beállítások következnek, végül elindíthatjuk a telepítést. Ha elkészült, a Finish-re kattintva a virtuális gép újraindul, és betölt az Openfiler, a képernyőn végül megjelenik webes elérési címe, ahol látható, hogy https protokollal a 446-os porton elérhető.

A Hyper-V kevésbé rugalmas, ami a támogatott operációs rendszereket illeti, ezért grafikus felületet használva nem lehet feltelepíteni az Openfilert. Viszont ha a neki szánt virtuális gépet „Other Linux (32 bit)” operációs rendszerre konfiguráljuk, és a virtuális optikai meghajtójába az Openfiler azon a verzióját csatoljuk, ami kizárólag x86 platformra készült, akkor text módban felinstallálhatjuk. Ezt úgy érhetjük el, hogy boot utáni képernyőn nem azonnal entert ütünk, hanem előbb begépeljük a „Linux text” szöveget. Így kicsit tovább kell

foglalkozni a telepítéssel, de megéri a fáradtságot. Hyper-V alatt az Openfiler nem ismeri fel a SCSI típusú merevlemezeket, mivel nem adható meg az LSI Logic típus, ezért ilyenkor IDE módot válasszunk [25].

3.3.2. Az Openfiler beállításai

Az IP címet, ha mást nem állítottunk, DHCP osztja ki számára. Betöltés után már elérhető a helyi hálózaton, és ha egy böngészőből megnyitjuk, akkor egy bejelentkező ablak fogad minket, ahol a kezdeti adatok: username: openfiler
password: password
Tehát nem rootként kell bejelentkezni! Ezután létre kell hozni az iSCSI célpontot. Az adminisztrátor felületen felül a „Volumes”-ra kattintva jobb oldalon egy másik menü jelenik meg, ezen kell most navigálnunk. Ha még nem hoztunk létre partíciókat a leendő adattárnak, akkor a Block Devices menüpontot kell választanunk, ahol kapunk egy listát a csatlakoztatott, jelen esetben virtuális eszközökről. Az Edit oszlopban a nevére kattintva szerkeszthetjük, az új ablakon pedig cilinderek szerint hozhatjuk létre az új partíciókat. Ha kész van, akkor a jobb oldalon először a Volume groups alatt kell egy kötetcsoportot létrehozni, majd Add Volume menüpont segítségével egy kötetet ebben a kötetcsoportban. Személyes tapasztalatom, hogy az a legjobb, ha minden kötetnek külön kötetcsoportot hozunk létre.



6. ábra. Új virtuális iSCSI kötet létrehozása

A kötet neve és mérete mellett különösen ügyeljünk a Filesystem / Volume Type mezőbe iSCSI-t megadni, máskülönben nem lesz működőképes. Ha kész a kötetünk, akkor felül a Services–re kattintva egy listát kapunk, ahol az „iSCSI target server” funkciót kell engedélyezni. Ha ezzel is kész vagyunk, akkor végül a Volumes fülre navigálva jobb oldalt az iSCSI Targets-et kiválasztva egy IQN–t adhatunk meg, majd a belső fekete LUN Mapping fülön engedélyezzük a Mapping-et write-thru és blockio módban minden kötetre. Ezzel készen vagyunk, létrehoztuk egy virtuális SAN-t [23, 25].

4. Guest Clustering

Általában a klaszterben szereplő hostok fizikai gépek, ám lehetőség van virtuális gépeket is felvenni hostnak, így alakul ki a guest clustering, amelyben szemben a szokványos klaszterrel, mikor a fizikai hostok futtatják a virtuális gépeket, virtuális hostok futatják a virtuális gépeket. Pár szót a rendelkezésemre álló hardverről:

alaplap: Gigabyte P55-UD3R

processzor: Intel Core i5 750 @ 2.66 GHz

memória: 4GB 1600 Mhz CL7-8-7-8

A processzor támogatja a hardveres virtualizációt, és 4 maggal rendelkezik, a Hyper-Threading technológia viszont nem támogatott, így csak 4 szálat képes futtatni.

4.1. Guest clustering VMware ESX-szel

A most következő részben bemutatom, hogy miként lehet a VMware ESX szerver segítségével megvalósítani virtuális gépek magas rendelkezésre állású hálózatát, melyben a HA, DSR és a VMotion is működik. A VMware termékek licenckötelesek, 60 napos próbaidővel volt lehetőségem használni őket. Virtuális gépeket a Workstation 6.5-el hoztam létre, ezen kívül a programok a Virtual Infrastructure 4 (VI 4) termékcsalád (vSphere) részei. Mindezt a host operációs rendszeremen hajtom végre, amely Windows XP SP3, MSDNAA licenccel.

Hogy a VMware speciális képességeit használni tudjuk, szükség van a vCenter Server telepítésére a host gépen. Ennek előfeltétele egy adatbázis kezelő szoftver. A telepítő egy SQL Server 2005 példányt telepít fel, ha más nem található a gépen. Ebben kerül létrehozásra

egy adatbázis, amiben tárolódnak a vCenter Server objektumai. A menedzseléshez pedig a vSphere klienst kell feltelepíteni, amely szintén része a termékcsomagnak.

4.1.1 ESX telepítése

Következhet az ESX Server 4.0 feltelepítése egy virtuális gépre. A virtuális gépet úgy készítsük el, mint az Openfiler esetében, két különbséggel. Az egyik, hogy az ESX 4.0 memóriaigénye 2 GB, illetve ennél egy picivel többet foglaljunk le, különben a telepítő az elején a hibát fog dobni és leáll. Szerencsére ha a virtuális memória engedélyezett, hiába állítunk be 2 GB memóriát, nem foglalja le az egészet a rendszerből, így 4 gigabájttal akár két ESX is futtatható egyszerre. A SCSI adaptert szintén LSI Logic-ra állítsunk be, és itt a másik különbség, hogy most elég egy merevlemezt készíteni, a mérete pedig legyen legalább 10 GB, ugyanis az ESX csak 9.5 gigabájtnál nagyobb merevlemezre hajlandó települni. Ha ESX 3.5 verziót telepítünk akkor muszáj SCSI lemeztípust választani, a 4.0 már elfogadja az IDE csatlakozót is.

Ha kész a virtuális gép, zárjuk be a programot, ugyanis szerkeszteni kell a létrehozott virtuális gép .vmx kiterjesztésű fájlját. Keressük meg az *ethernet0.present = "TRUE"* sort, majd szúrjunk be 5 sort utána az alábbi módon:

```
ethernet0.present = "TRUE"
ethernet0.virtualdev = "e1000"
ethernet0.connectionType = "bridged"
monitor_control.restrict_backdoor = TRUE
monitor_control.vt32 = TRUE
monitor.virtual_exec = "hardware"
```

7. ábra. Az ESX virtuális gépének szerkesztése

Erre azért van szükség, hogy olyan virtuális hálózati kártyát adjunk az ESX-nek, amit felismer, illetve hogy futtathassunk virtuális gépeket a virtuális ESX-en belül. Ez a módszer az ESX 3.5 verziójára is alkalmazható. Csatlakoztassuk a telepítő képfájlját, és már indíthatjuk is a Workstationben az ESX Server 4.0 telepítését. Ha a driverek betöltésekor hibaüzenetet kapunk, akkor valószínűleg rosszul szerkesztettük a vmx fájlt. A sikeres install után a Finish-re kattintva a virtuális gép újraindul, majd ha betöltött kijelzi az IP címét, amit itt láthatunk meg először, ha DHCP-től kéri, így hasznos, hogy megjelenik.

8. ábra. Szerver a szerverben

4.1.2. ESX menedzselése vCenterrel

Csatlakozni a vSphere Client-el (Virtual Infrastructure Client) tudunk rá az IP címén „root” felhasználónévvel, és a telepítéskor megadott jelszóval. Az ESX 4.0 mostani build-je (164009) még nem támogatja a magyar billentyűzetet, így az y és z betűkre érdemes figyelni, ugyanis a klienssel csatlakozva megfordulnak a virtuális géphez képest.

A vSphere Client-el kell csatlakoznunk a vCenter Server–hez is, ha saját gépre telepítettük, akkor „localhost”-ra és a telepítéskor megadott adatokat használva. Mivel a vCenter Server-re csatlakozva elérhetjük az ESX szervereket is, ezért a továbbiakban úgy folytatom a bemutatót, hogy a vCenter Server-re csatlakoztunk a vSphere Client-el, illetve a „szerver” és „kliens”

szavakat fogom használni helyettük. Ám egy rövid megjegyzés a folytatás előtt, hogy amennyiben nem tudunk csatlakozni a szerverhez, akkor ellenőrizzük a szolgálatások között, hogy fut-e a VMware Virtual Center Server nevű szolgáltatás.

A szerverre csatlakozva kiindulásképpen egyetlen feladatot tudunk végrehajtani, ez az új Datacenter létrehozása. Ebben tudunk létrehozni egyszerűen (nevére jobbklikkel kattintva) egy új klasztert.

4.1.3. A magasszintű képességek

A klaszter létrehozásakor kell definiálnunk annak tulajdonságait, többek között a HA és DRS engedélyezését, az automatizmus fokát, a javított VMotion bekapcsolását. Mikor elkészült, akkor ugyanezzel a játszi könnyedséggel adhatjuk hozzá az ESX szervereket, mint hostokat. Ha egy host nem teljesíti a klaszterben beállított követelményeket, akkor a vSphere figyelmeztet minket, és ha nem sikerül megoldani a problémát, akkor a klaszter szabályai sérülhetnek, amit piros négyzet jelöl a klaszter, illetve a problémás host ikonján. A hibaüzeneteket részletesen megtekinthetjük, ha a hostra kattintunk, a jobb oldali ablakon a „Tasks & Events” fület választjuk, majd az Events nézetet választjuk ki.

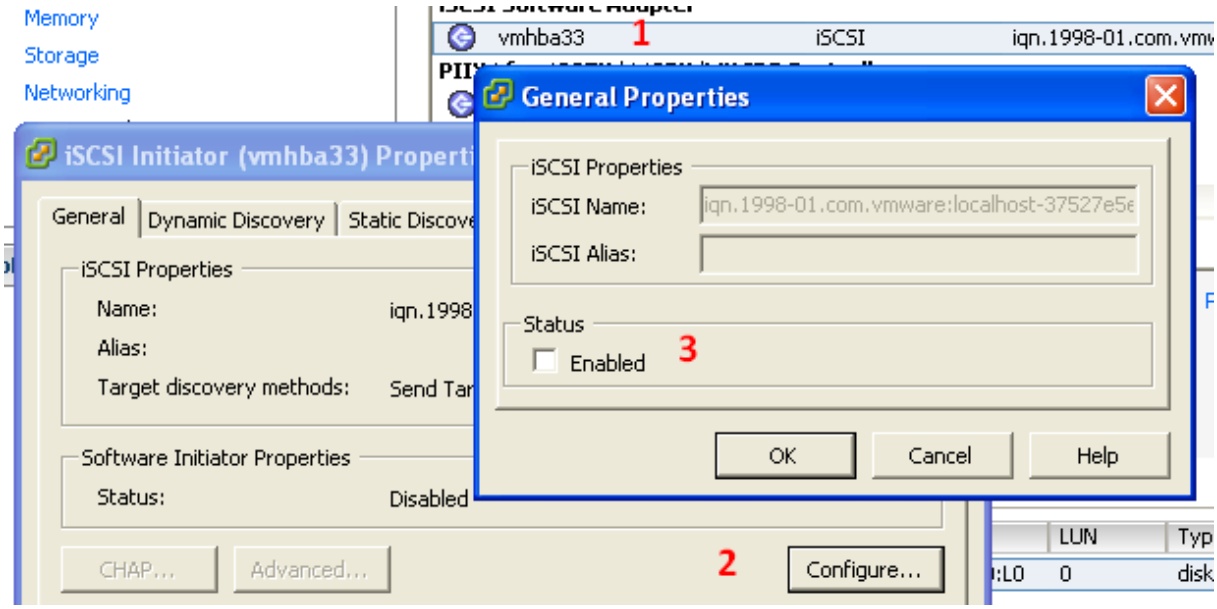
Többször találkoztam azzal a problémával, ezért azt hiszem érdemes megemlíteni, hogy az ESX szervereken ilyen módon beállítva sokszor még nem tudjuk engedélyezni a HA-t, mert a hostname –s parancsra nem jó eredményt ad vissza a Linux. Ezt én a Workstation-ben orvosoltam, bejelentkezve az ESX-re, a vi programmal szerkesztettem a /etc/hosts fájlt, mégpedig úgy hogy az első sorba egy hasonló új bejegyzést vettem fel:

192.168.2.139 egyik.djackson.hu egyik

ahol a sor végén az „egyik” a hostname rövidítettje (short), ezt adja vissza a kérdéses parancs. Ezután újra kell indítani az ESX Servert. Akkor lehetünk biztosak, hogy jó lesz, ha újraindítás után bejelentkezünk az ESX-re konzolon (ALT+F1 segítségével), majd kiadjuk a hostname-s parancsot. Ha „unknown host” vagy „localhost” üzenettel tér vissza akkor ismételjük meg az előbbi eljárást. Újraindításig ideiglenesen esetemben a „hostname 192.168.2.139 egyik.djackson.hu egyik” parancsot kiadva orvosolhatjuk a problémát, de biztosan csak akkor lesz jó később is, ha újraindítás után már jó értékkel tér vissza beavatkozás nélkül.

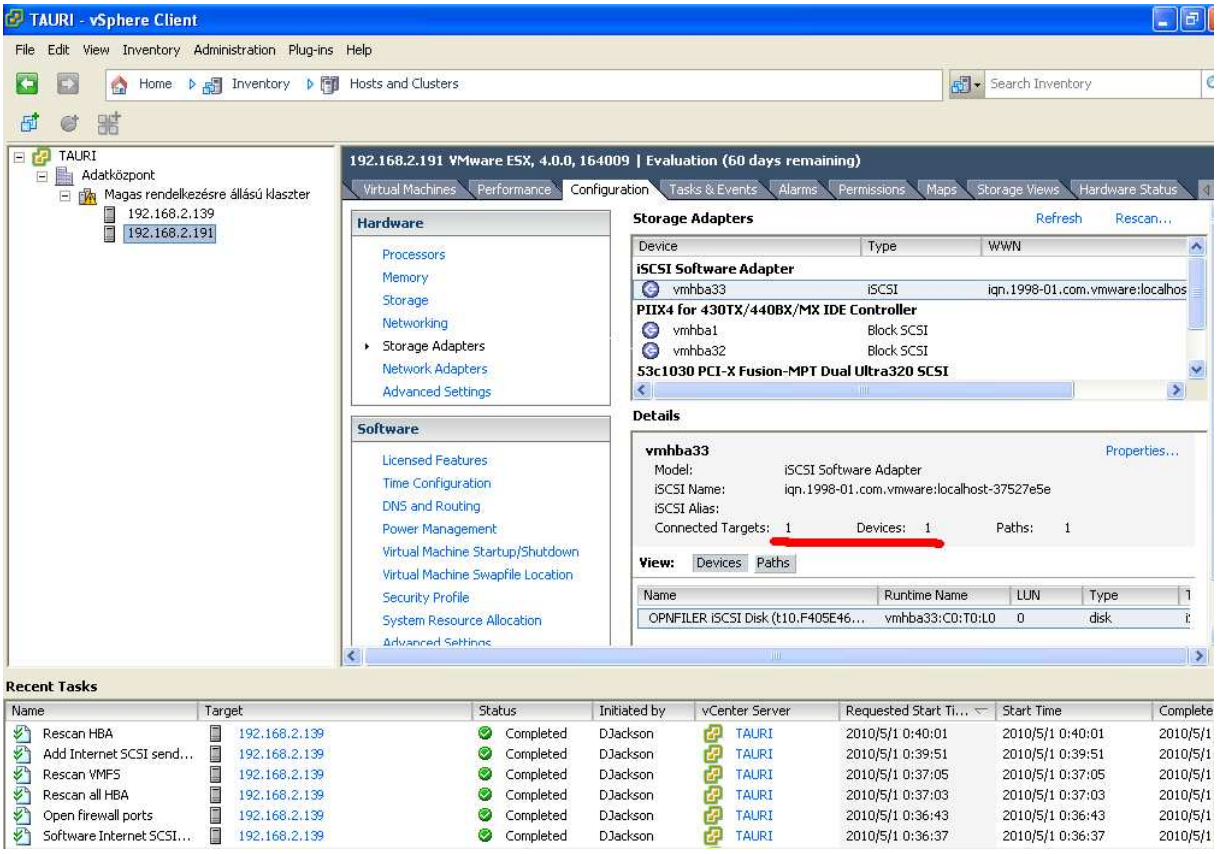
4.1.4. A háttértároló rendszer csatlakoztatása

Azonban ezzel még nem vagyunk kész, hiszen ha sikeresen engedélyeztük a HA-t, azzal még nem biztosítottuk, hogy működni is fog. Mint korábban említettem még létre kell hozni a közös tárolót. Pontosabban a virtuális SAN fejezetben ez már megtörtént, viszont még nem tettük elérhetővé az ESX szerver számára. Ehhez a hostot kell kijelölnünk a kliensben, és a jobb oldali fülek közül a Configuration-t kiválasztani. Kattintsunk az Add networking… linkre, válasszuk ki a VMkernel-t, majd a következő ablakban a rádiógombbal válasszuk ki a már létező switchet. Az ez utáni ablakban tegyünk pipát a „Use this port for VMotion” mellé, és végül az utolsó lapon állítsuk be az IP címet. A VMkernel az iSCSI forgalomhoz és a VMotion működéséhez is szükséges, ami azt jelenti, hogy mivel a VMkernel egy hálózati interfész, így saját IP címet kap. Ez például akkor lehet fontos számunkra, ha a közös tárolóhoz biztonsági megszorításokat adunk hozzá, ugyanis ha IP alapján korlátozunk és engedélyezünk, akkor a VMkernel interfésszel kell dolgoznunk. Az előbbi pipával engedélyezhetjük a VMotion-t, de később bármikor kikapcsolhatjuk a VMkernel szerkesztésével. Ezt követően a Storage Adapters-be navigáljunk és ott az ábrán látható módon engedélyezzük az iSCSI Initiatort.



9. ábra. Az iSCSI engedélyezése

Még ugyanitt a Dynamic Discovery fülön vegyük fel a virtuális SAN IP címét, és ha felajánlja, akkor scan-eljük újra az adaptert. A VMkernel felvételét, és az iSCSI Initiator-t minden hoston ugyanígy végig kell csinálni. A Security profile-ban ellenőrizzük, hogy a Software iSCSI Client engedélyezve van-e. Ha a Connected Targets és a Devices értékek változtak (nőttek), akkor a művelet sikerült.

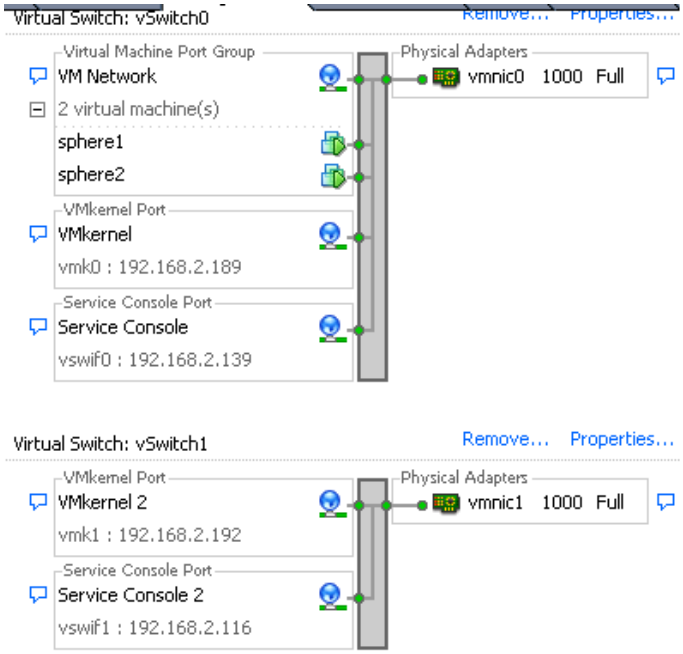


10. ábra. Az iSCSI működőképes

Az Openfilerrel létrehozott struktúra csak az iSCSI célpontokat tartalmazza, rajta partíciókat kell még létrehozni, hogy megjelenjenek a Storage menüpont alatt. vSphere esetén mi sem egyszerűbb: a Storage menüpontba navigáljunk, majd az Add Storage varázslót használva hozzuk létre a közös adattárat. Típusnak a Disk/LUN opciót válasszuk, és a következő ablakon meg kell jelenniük az elérhető iSCSI célpontoknak, a varázsló végeztével pedig az új adattárnak az összes olyan hoston, amin az Openfilert beállítottuk a Dynamic Discovery-ben.

A klaszter rendelkezésre állását úgy lehet növelni, hogy a hostok között redundáns hálózati kapcsolatot alakítunk ki. Amíg ezt nem tesszük meg, addig ott lesz egy sárga háromszög a

klaszteren, jelezvén a hiányosságot. Hogy ezt pótoljuk a leállított host virtuális gépéhez adjunk hozzá még egy hálózati adaptert, majd szerkesszük a vmx fájlt. Most az ethernet1.present = "TRUE" sort kell megkeresni, és elég egy sort beszúrni utána a következő tartalommal: ethernet1.virtualdev = "e1000". Ha ez elkészült indítsuk el, és csatlakozzunk rá a klienssel. A Configuration fülön Networking menüpont alatt az Add Networking varázslóval adjunk hozzá egy Service Console-t, majd az új virtuális switchre egy VMkernel-t is, amin engedélyezzük a VMotion-t. Az ábra mutat egy helyes konfigurációt. Ezzel technikailag lehetővé tettük a VMotion, HA és DRS teljes értékű használatát a klaszteren belül [23].



11. ábra. Példa a redundáns internetkapcsolat helyes beállítására

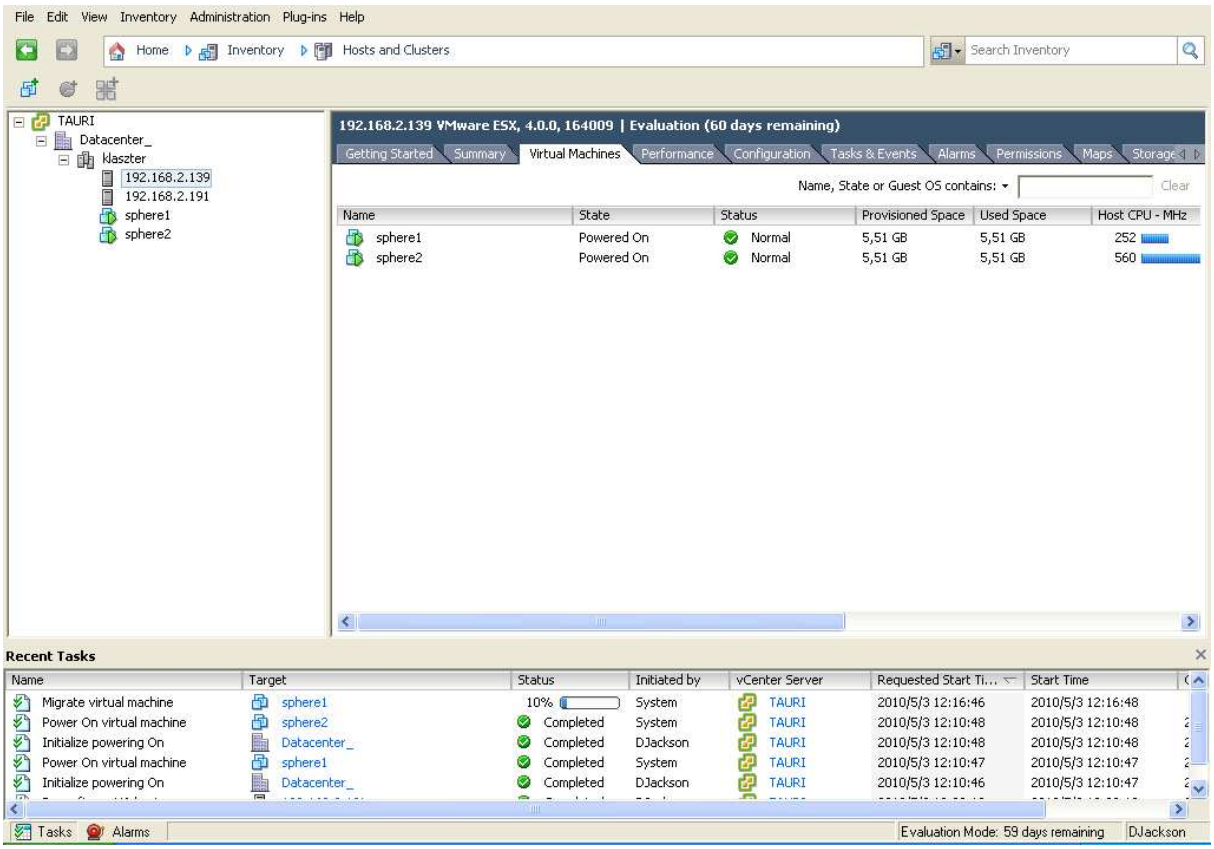
4.1.5. A munka gyümölcse

Már csak annyi a dolgunk, hogy létrehozzuk a virtuális gépeket, figyelve arra, hogy a közös tárolót használjuk. A VMotion-t úgy próbálhatjuk ki, hogy egy futó virtuális gépre jobb klikkelünk a bal oldali fa szerkezetben, és a Migrate… opciót választjuk, majd a Change host rádiógombot. Miután kiválasztottuk a célhostot a szerver végrehajtja az ellenőrzést, miszerint a két host processzorának hasonlónak kell lennie, s ha sikeres volt a validáció, akkor elindítja az áthelyezést. A Migrate… menüpont alatt azonban nem csak a hostot változtathatjuk meg működés közben, hanem magát a virtuális gépet is áthelyezhetjük másik tárolóra. Itt természetesen csak olyan tárolót tudunk majd kiválasztani, amelyet lát a virtuális gép aktuális

hostja. Ennek a technikának a neve Storage VMotion. Egyszerre nem valósítható meg a VMotion-nel, de egymás után végrehajthatóak, illetve ha leállítjuk a virtuális gépet, akkor egyszerre is működik.

A High Availability (HA) biztosítja a képességet, hogyha egy host kiesik a klaszterből, akkor a rajta futó virtuális gépek újrainduljanak egy másik, még működő hoston. Az igaz, hogy nem ott folytatják működésüket, ahol abbahagyták, de legalább újraindítja őket a szerver. Ezáltal a váratlan hibákra is képesek vagyunk valamilyen szinten felkészülni.

A Distributed Resource Scheduling (DRS) funkciója a klaszterben szereplő hostok erőforrásainak menedzselése, elosztása, eszközül felhasználva a VMotion-t. A DRS-t bármikor személyre szabhatjuk, a klaszter tulajdonságainál. Szabályokat adhatunk meg a virtuális gépek külön vagy egy hoston történő futtatására, vagy engedélyezhetjük automatikus be- és kikapcsolásukat az erőforrás-tartalékok rendelkezésre állásának függvényében (Distributed Power Management, DPM). De a leghasznosabb funkciója a kiegyensúlyozottság biztosítása, amelyet automatikusra állítva önmagában dönthet a hostok közötti mozgatásról, törekedve arra, hogy minden host nagyjából azonos mértékben legyen terhelve. Ezáltal egy igen rugalmas és dinamikus hálózat alakítható ki [28].



12. ábra. DRS működés közben. Látható,hogy az éppen futó migrációt System kezdeményezte

4.2. Guest clustering Hyper-V-vel

Ha Hyper-V-t szeretnénk hypervisornak használni, akkor arra több lehetőségünk is van. A Microsoft zászlóshajója a Windows Server 2008 operációs rendszer, amely most az R2 verziónál tart. Ezen kívül a Hyper-V Server nevű termékkel is elérhető, ami egy Windows licenc nélküli operációs rendszer és egyetlen „role”-lal rendelkezik: a Hyper-V-vel. Dolgozatomban a Windows Server 2008 R2 operációs rendszer segítségével mutatom be a guest clustering-et, melyhez az MSDNAA program keretében jutottam hozzá.

4.2.1. Követelmények

Hyper-V esetében egy kicsit más a felépítése a megvalósítható magas rendelkezésre álló rendszernek, a Failover Clusternek. A fizikai gépre (szülő partíció) kell először a Windows Server 2008-at telepíteni, és engedélyezni rajta a Hyper-V role-t. A klaszter megvalósításához még szükséges, hogy a résztvevő virtuális gépek domain tagjai legyenek, így ha a semmiből állítunk össze egy klasztert, akkor a szülő partíció operációs rendszerére még egy Active Directory Domain Controllert is létre kell hozni, és futnia kell a DNS szervernek is. Ha ezzel megvagyunk, akkor a Server Manager-ben a Hyper-V Manager alatt hozhatjuk létre a virtuális gépeket. A példában két Windows Server 2008 R2 lesz a virtuális gépeken, illetve a szülő partíción is Windows Server 2008 R2 fut mint Domain Controller és DNS Server.

Az Openfiler a Hyper-V alá is telepíthető (23. oldal), de bemutatok egy másik szoftvert alternatíva gyanánt, amely nem kevéssé elterjedt az Openfilernél. Ez a szoftver pedig a Starwind, szintén ingyenesen letölthető, és otthoni használatra nem kell fizetni érte. Talán egyszerűbb a kezelése, mint párjának, viszont ha céges környezetben szeretnénk használni, akkor licencköteles.

4.2.2. Egy virtuális SAN alternatíva

A Starwind segítségével tehát létre tudunk hozni egy iSCSI célpontot a gépen, amire felkerült. Telepíthetjük a szülő partícióra is, vagy létrehozhatunk számára egy virtuális gépet, ha kedvünk tartja, én az előbbi mellett döntöttem. A Starwind egy rendszerszolgáltatást hoz

létre, amit a services.msc indításakor láthatunk a többi között. Ha ez nem fut, akkor nem lesz elérhető az iSCSI célpont. Félreértések elkerülése végett a többi gépre nem kell feltelepíteni, kivéve, ha onnan szeretnénk menedzselni a célpontokat, azonban ebben az esetben is elég csak a Management Console-t. Ha elkészült, akkor indítsuk is el, és adjunk hozzá egy új hostot, az IP címével, vagy ha helyiről csatlakozunk, akkor 127.0.0.1-el. Ha fut a célgépen a Starwind iSCSI SAN Software szolgáltatás, akkor ez nem dobhat hibát. Csatlakozni hozzá alapértelmezetten a „root" felhasználónévvel, és „test" jelszóval lehet, de előfordulhat, hogy nem fogadja el a jelszót. Ekkor navigáljunk a Starwind könyvtárába, és keressük meg a starwind.cfg fájlt, majd nyissuk meg szerkesztésre. Keressük meg az Authentication szekcióban a password value sort, és értékének adjunk meg: test. Ezután indítsuk újra a Starwind iSCSI SAN Software szolgáltatást.

Ha sikerült belépnünk az iSCSI szerverre, akkor az első dolgunk hozzáadni egy új célpontot, ezt az Add Target gombbal tudjuk megtenni. Választhatunk számos lehetőség közül, például optikai meghajtót, vagy merevlemezt választhatunk ki, legyen az fizikai, vagy virtuális eszköz, de akár RAID-1 es tömböt is építhetünk. Virtuális merevlemez esetében csak img formátumot támogat, virtuális optikai meghajtó pedig iso vagy mds lehet. Fontos, hogy engedélyezzük a Clustering-et, hogy egyszerre több gép is csatlakozhasson rá.

4.2.3. A Starwind V2V Image Converter

A kizárólagos img formátum támogatása szigorú megkötés, ám a Starwindnek van egy másik programja is, melynek neve a V2V Image Converter. Ezzel a programmal tudunk konvertálni VMware vmdk és Microsoft vhd között, illetve az egyszerű img formátum között is. Igen hasznos lehet, ha P2V (phisical to virtual) konverziót hajtottunk létre valamelyik cég termékével, majd a kész virtuális merevlemezt a másikkal is szeretnénk használni, vagy ha csak egyszerűen váltani szeretnénk. Működése elég megbízható, és jó munkát végez, illetve sok esetben életmentő lehet ingyenessége is.

4.2.4. Az iSCSI csatlakoztatása

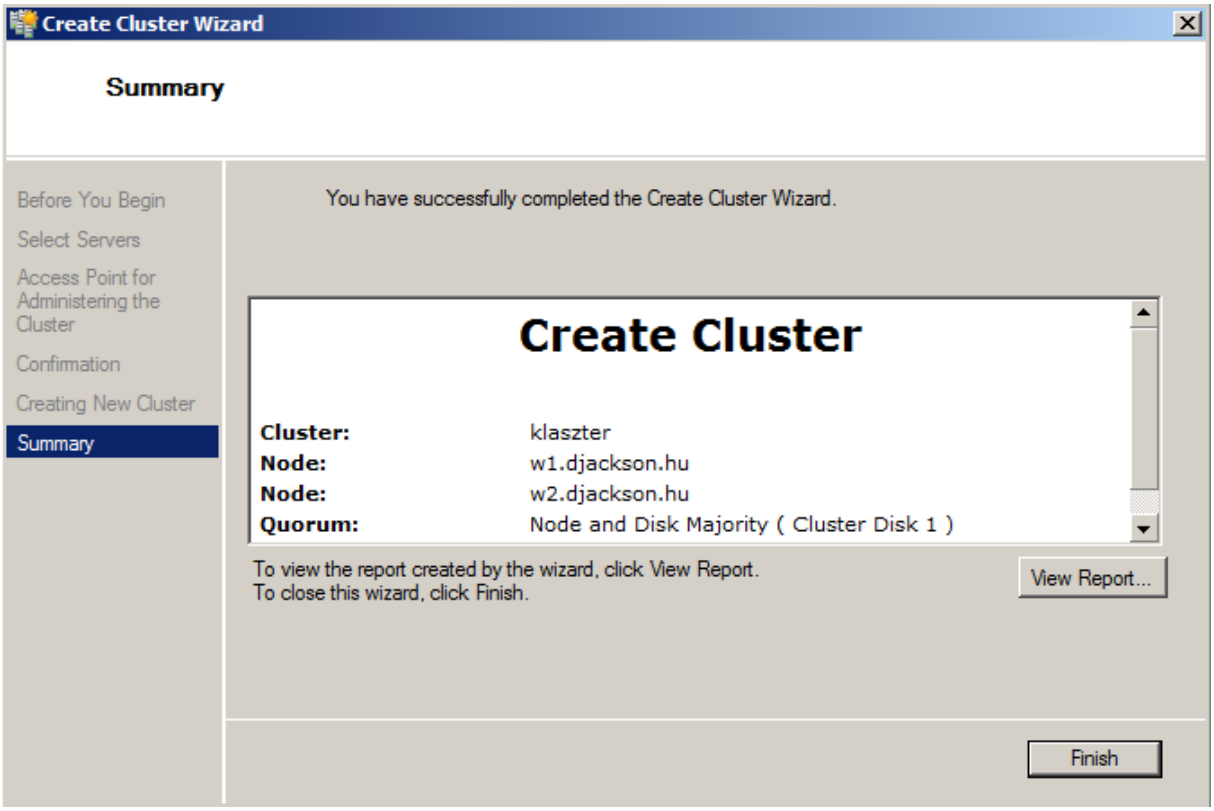
Ha sikerült a célpont létrehozása, akkor a leendő klaszter minden tagja számára elérhetővé kell tennünk. A Windows Server 2008-ban található egy eszköz, ami pontosan ezt a funkciót

szolgálja, nevezetesen hogy csatlakozzon egy iSCSI célponthoz. Neve az ISCSI Initiator, és a %windir%\system32\iscsicpl.exe útvonalon érhető el. Indítás után rögtön látjuk a Target mezőt, ami kitöltendő a szülő partíción futó rendszer IP címével, majd a Quick Connect-re kattintva már meg is jelenik az IQN neve a célpontnak. Ha felvesszük a Favorite Targets közé, akkor minden újraindításkor megpróbál majd csatlakozni hozzá, így ez erősen ajánlott. A Volumes and Devices fülre navigálva a használjuk az Auto Configure-t, és már el is készültünk. Ezt minden gyerek partíció operációs rendszerén el kell ismételni. Az eredményt leellenőrizhetjük a Server Managerben a Storage alatt, hogy megjelent-e az új lemez eszköz. Itt hozhatjuk Online állapotba, ha kell, inicializálhatjuk. Amit viszont csak egyszer szükséges megtenni, az a kötet létrehozása az eszközön. Mivel minden klasztertag ugyanazt a lemezt látja, így ha változtatunk rajta, akkor mindenki értesül a változásról automatikusan.

Szeretnék pár kiegészítést tenni ehhez a folyamathoz. Ha az iSCSI Initiatorral nem sikerül kapcsolódni a szülő partíció IP címére, akkor a szülő tűzfalában kell keresni a problémát. Engedélyezzük az iSCSI Service-t az „allow a program or feature through Windows Firewall” alatt, és ha ez sem elég akkor nyissuk meg a portját (3260 alapértelmezetten) a tűzfalon. Ha Starwind helyett Openfilert használunk, akkor is az iSCSI Initiatorral kell csatlakozni rá, csak ilyenkor értelemszerűen a Linux IP címét kell a Target mezőbe megadni.

4.2.5. A Failover Custer Manager

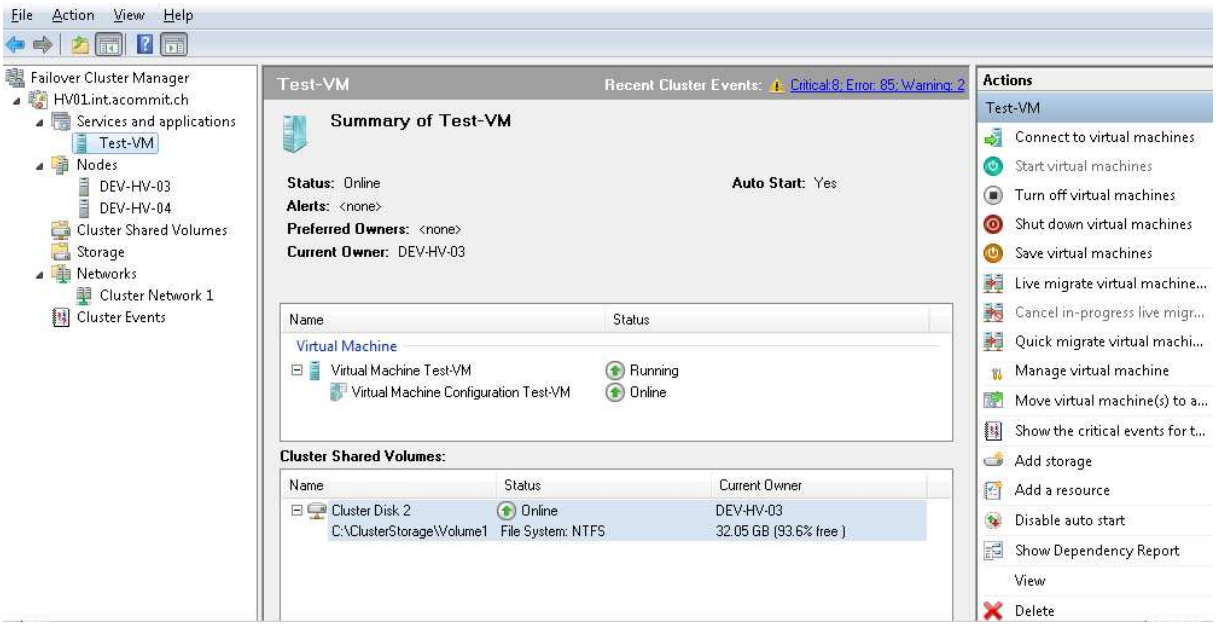
Ha mindezekkel elkészültünk, akkor magát a klasztert az adminisztrátori eszközök között található a Failover Cluster Manager varázslóval hozhatjuk létre. Ha elindítjuk, először egy ellenőrzést kell végrehajtani, amellyel megbizonyosodhatunk, hogy a hostokból létrehozható-e a klaszter. Ez a folyamat ellenőrzi a hálózati, háttértár, és egyéb erőforrások elégséges rendelkezésre állását, és a Validate a Configuration-re kattintva indíthatjuk el. Az AD segítségével megkereshetjük a kívánt virtuális hostokat, majd elindíthatjuk az ellenőrzést. Bizonyos feltételek nem teljesülése esetén a folyamat megszakad (pl. hiányzó közös tárhely) de vannak olyan tesztek is, amelyek ilyet nem okoznak, csak egy figyelmeztetést kapunk a végső jelentésben (pl. ha nem a Microsoft által aláírt eszközvezérlőket használunk). Ha sikerrel lezajlott a teszt, akkor végre létrehozhatjuk a klasztert a Create Cluster-re kattintva, ahol egyébként a tesztfolyamatot megismételhetjük. A képen egy sikeres folyamat vége látható.



13. ábra. Hibátlan klaszter virtuális hostokból

A klaszteren ezután különböző szolgáltatásokat hozhatunk létre, ha a „Configure a service or application”-re navigálunk. A felajánlott szolgáltatások között van a virtuális gép is, de ezen a ponton szembetalálkozunk a technológiai akadállyal, miszerint a Hyper-V nem képes a CPU teljes virtualizálására, ebből kifolyólag nem tudunk virtuális hostokra virtuális gépeket telepíteni. Ehhez ugyanis az lenne szükséges, hogy a Hyper-V-t telepítsük a virtuális hostokon, amit viszont nem tehetünk meg, mert a fizikai gépen futó Hyper-V olyan processzort mutat a virtuális gépeknek, melyek nem képesek a hardveres virtualizációra. A guest clustering így az egyéb szolgáltatásokban merül ki, melyek között van például a File Server, vagy a különböző adatbázisszerverek futtatása. Amennyiben azonban legalább két olyan géppel rendelkezünk, amely támogatja a hardveres virtualizációt (AMD-V, Intel VT-x, Intel-VT-i), akkor a virtuális gépeket is helyezhetünk a klaszterbe. Ennek feltétele, hogy minden hoston telepítve legyen a Hyper-V role. Az ilyen klaszter már képes az ESX-hez hasonló technikák alkalmazására. A Storage VMotion megfelelője a Quick Migration, míg a VMotion-é a Live Migration, és bár egy kicsit máshogy működnek, azt mondhatjuk, hogy funkciójuk azonos. Végül még annyit megemlítenék, hogy elvileg lehet ESX Servert is

felvenni a hostok közé, viszont Hyper-V-vel egyazon gépen a VMware Workstation nem futtatható, illetve a Hyper-V-vel létrehozott virtuális gépben bár elindul, de nem tud sikeresen betölteni az ESX 4.0 [13, 22, 31].



14. ábra. A Quick Migration és Live Migration lehetősége Hyper-V alatt

4.3. Guest clustering Citrix XenServerrel

Végül, de nem utolsó sorban a Citrix megoldását szeretném bemutatni, melynek neve XenServer, és a legújabb elérhető verziója az 5.5 update 2. A szoftver ingyenesen letölthető a Citrix oldaláról. Hogy klasztert építsek vele a VMware Workstation-t használom mint virtuális környezetet, két okból is. Az egyik, hogy Hyper-V alatt ismeretlen hibára hivatkozva megáll a folyamat, mielőtt elindulhatna a fájlok másolása. A hálózati kártya problémáját a Legacy Adapterrel korrigálni lehet, de még valami más gondba is ütközik. A másik ok, hogy a virtuális gépek menedzseléséhez a Xen egy XenCenter nevű programot biztosít, mely szintén ingyenes, viszont kizárólag Windows platformra elérhető. A XenServernek „Other 64-bit” operációs rendszer beállítás szükséges, mikor létrehozzuk számára a virtuális gépet, valamint 1 gigabájt RAM javasolt. Bootolás után a konfigurációra kevés lehetőség van, így a telepítés viszonylag könnyen és gyorsan kivitelezhető. Ha ilyen konstrukcióban telepítünk, akkor egy üzenet fogad minket arról, hogy a hardveres virtualizáció nem lesz elérhető. Ennek ellenére a

szerver feltelepül, ami a paravirtualizációnak köszönhető. Itt jelenik meg az, hogy igazából mi is a paravirtualizáció, jelen esetben hogy a virtuális processzor igaz nem rendelkezik a hardveres támogatással, ennek ellenére megoldható a futtatás. Ennek azonban ára van, gyakorlatilag a virtuális szerveren vendégként futtatható operációs rendszerek száma jelentősen korlátozódik, mégpedig azért mert a paravirtualizációra fel kell készíteni magát a vendéget is. Sajnos így a Windows rendszerek azonnal kiesnek a lehetőségek közül, vannak azonban Linux disztribúciók, amiket letölthetünk úgy, hogy nekünk már semmit nem kell módosítani rajta, mégis képes ebben a „virtuálisban virtuális" környezetben futni. Az egyik ilyen Linux a CentOS [5, 26].

4.3.1. A CentOS és a Xen

Virtuális gépeket a XenServeren is létrehozhatunk, de sokkal egyszerűbb, ha a XenCentert használjuk. Telepítés után egy új szervert kell hozzáadnunk, ezzel csatlakozhatunk a virtuális XenServerre. Ezután már létrehozhatjuk a virtuális gépeket a XenServerre egy egyszerű párbeszédablakkal. Válasszuk a legújabb CentOS típust, és ízlés szerint biztosítsunk memóriát. Arra nem sikerült rájönnöm, hogy iso-t hogy tudunk könnyen csatlakoztatni kívülről, de valószínüleg erre nincs lehetőség, viszont megoldhatjuk a gondot, ha az egyik XenServer-be csatlakoztatjuk a CentOS iso-ját, és ezt a virtuális gép létrehozásakor kiválaszthatjuk forrásnak. Amennyiben sikeresen létrehoztuk az új virtuális gépet, látható lesz a Console fül, a vSphere-hez hasonlóan, ahol beavatkozhatunk a telepítésbe. A CentOS úgy vettem észre, hogy nagyon érzékeny, ami az optikai meghajtókat illeti, ugyanis igaz sikerült bebootolnia az iso-ról, viszont a telepítő csomagokat nem találta meg rajta, akárhogy ügyeskedtem. Azonban ilyen extrém esetben is van megoldás: válasszunk http-n vagy ftp-n keresztüli telepítést, egy ftp kiszolgálót beállítani egyszerű feladat mindenki számára. A CentOS elérhető 6 lemezzel, illetve DVD-vel, az alaptelepítéshez elég az első CD lemez iso-ja, viszont ha ennél többre van szükségünk, akkor le kell töltenünk a csomagokat a sajátgépünkre egy könyvtárba, és ennek tartalmát kell átadni a telepítőnek az ftp megosztásban (távoli könyvtárnak /-t adjunk meg, ha így történt a megosztás). Ha lehet, akkor ne Anonymous módban csatlakozzunk! Saját ftp kiszolgáló esetén letöltéshez ímhol egy mirror link: ftp://ftp.is.co.za/mirror/centos/5.4/os/i386/

A telepítés olyan, mint a korábban már bemutatott, apróbb különbségekkel, és text üzemmódban, ami nem túl kényelmes, de használható. A végeredmény egy konzol, amit talán kevesellhet az átlag felhasználó, annak nem jártam utána, hogy milyen lehetőség van a grafikus felület életre keltésének paravirtualizációval, de nem is ez volt a célom. Most vizsgáljuk meg, hogy milyen lehetőségek vannak a magas rendelkezésre állás, és az erőforrás-kiegyenlítés megvalósítására a XenCenterben.

A XenMotion képesség csak akkor elérhető, ha a rá telepített virtuális gépre telepítve van az úgynevezett Xen Tools. Hogy ezt installáljuk a CentOS-re a következőket tegyük:

1. A virtuális DVD meghajtóba helyezzük be a xs-tools.iso-t
2. Adjuk ki konzolon a következő parancsokat:

   mkdir /mnt/cdrom

   mount /dev/xvdd /mnt/cdrom

   /mnt/cdrom/Linux/install.sh
3. Mikor rákérdez, hogy folytassuk, nyomjuk le az „y” betűt.
4. Várjunk, míg elkészül, majd indítsuk újra a virtuális gépet az alábbi paranccsal:

   shutdown –r now

Nagyon fontos, hogy nem fogunk sikerrel járni, ha a CentOS telepítése közben kiválasztjuk a KDE, Gnome stb eszközök mellé a Virtualization-t, ezért figyeljünk oda, hogy ne tegyünk mellé X-et.

4.3.2. A Xen mint magas rendelkezésre állású rendszer

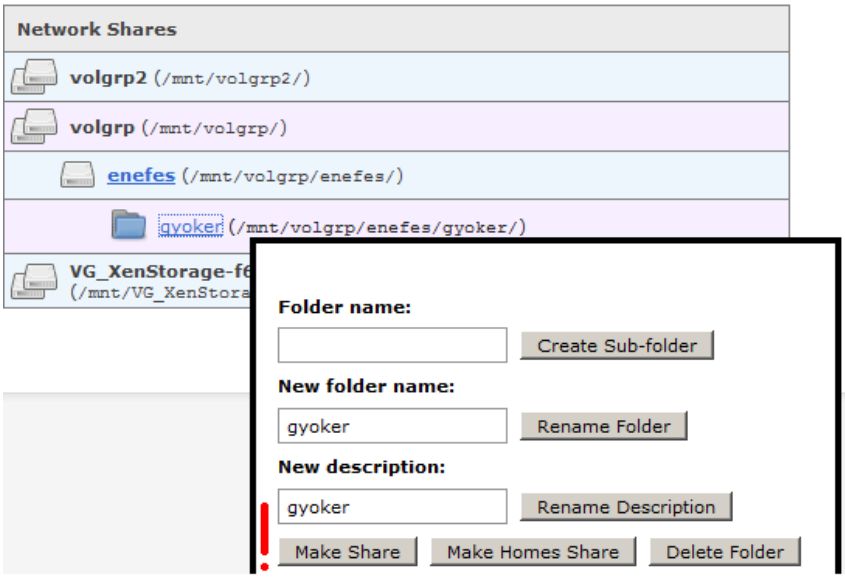
Hogy a funkciók elérhetőek legyenek egy újabb Xen programra van szükség, amit úgy hívnak Xen Essentials. Ezért már fizetni kell, viszont 1 hónapos határidővel kipróbálható. A Citrix oldalán kell ehhez regisztrálnunk, ahol mellé kapunk egy Enterprise licencet is. Telepítéséhez egy SQL Server példány szükséges, illetve .NET Framework 3.5, ezeket indítás után ellenőrzi. Amennyiben lehetőségünk van rá, használjuk az SQL Server 2008-at.

A magas rendelkezésre állású rendszer létrehozásának lépéseit szeretném most ismertetni, a folyamat elejétől. Hozzuk létre a kívánt XenServer virtuális hostokat, és mindet indítsuk el. Lépjünk be a XenCenterbe, és az Add… funkcióval vegyünk fel minden szervert. Hozzunk létre egy pool-t a New pool… segítségével, majd a szervereket mozgassuk bele. Telepítsük a Xen Essentials-t, majd a letöltött licencfájlt (.xslic kitrjesztéssel) alkalmazzuk minden

virtuális hostra, ezt úgy tudjuk megtenni, hogy kijelöljük a kívánt szervert, majd a XenCenter menüjében a Server menü alatt az Install License Key…-t választjuk, majd betallózzuk a licenc fájlt. A poolban szereplő összes szerverre végezzük el ezt a műveletet, különben később nem tudjuk majd aktiválni a HA és WLB funkciókat.

Ezután csatlakozni kell az iSCSI célponthoz, ugyanis itt is igaz, hogy közös tároló szükséges. Az Openfiler iSCSI, illete Starwind konfigurációjáról már korábban esett szó, így most csak azt írom le, hogy XenCenterben milyen módon kell csatlakozni. A pool-ra kattintás után a NEW Storage gombot válasszuk, a bejövő párbeszédablakban pedig az iSCSI-t. Tapasztalatom szerint a kényes rész a következő ablakban van, mikor is meg kell adni a kiszolgáló IP címét, majd a Discover IQNs gombot kell megnyomni, és megvárni az eredményt. Ha nem találja, akkor próbálkozhatunk az Openfilerben új IQN hozzáadásával (hosszabbal, rövidebbel) a Volumes fülön az iSCSI Targets menüponton, de figyeljünk arra, hogy minden IQN-hez saját LUN Mapping tartozik. Ha megtalálta az IQN-t, akkor aktívvá válik a Discover LUNs gomb, amely után befejezhetjük a varázslót, és eredményképpen létrejött az iSCSI célpont a poolban.

Sajnos a HA és WLB számára iSCSI szükséges, azonban ha csak a Xen Motion-t szeretnénk használni, ahhoz elég egy NFS megosztást felvenni a poolba. NFS megosztáshoz az Openfiler Services fülében engedélyezni kell az NFSv3 Servert, valamint a kötet létrehozásakor Filesystem / Volume type-nál nem iSCSI-t kell kiválasztani, hanem ext3 típust! Ezután az ábrán látható módon létre kell hozni egy könyvtár nevet, majd a Make Share után a Share Access Control Mode-ot Public guest access-re kell állítani.



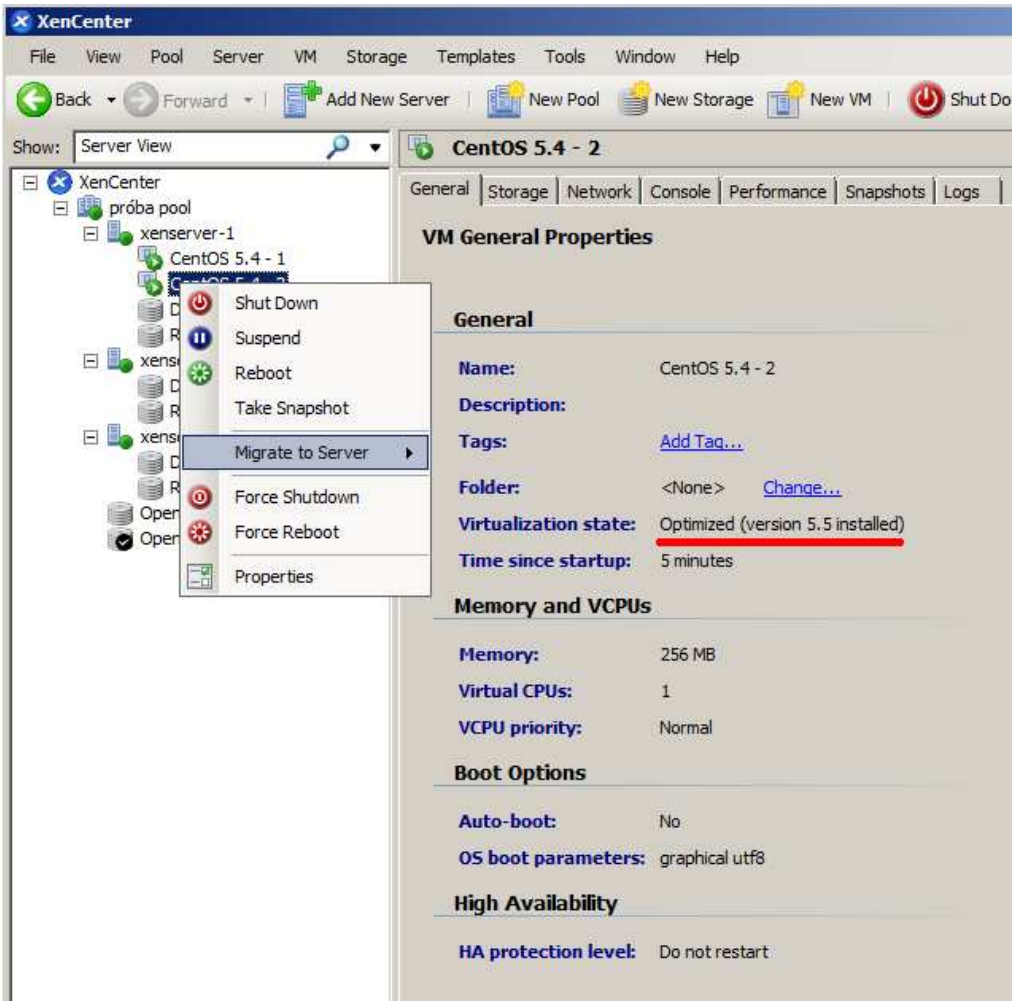
15. ábra. NFS megosztás létrehozása

Tapasztalatom szerint ennek ellenére nem fogja megtalálni a XenCenter a megosztást, ha nincs engedélyezve legalább az egyik host számára a hozzáférés, amit a System fülön tudunk állítani, felvéve a szerver IP címét és maszkját, és a típust pedig Share-nek megadni. Ennek megfelelően az előbbi Share Access Control Mode alatt megjelenő Host access configuration táblázatban az NFS gyűjtőoszlopban az RW oszlopba kell tenni a rádió gombot, majd Update-elni. Minden szervert vegyünk fel ilyen módon az engedélyezetek listájába. Ha kész az NFS tároló, akkor a poolra kattintva a HA fülön engedélyezzük a magas rendelkezésre állást. A XenServer automatikusan kiszámolja a Fault Tolerance szintet, és már kezdhetjük is a virtuális gépek létrehozását.

4.3.3. A Xen működése

Néhány szót ejtenék a Xen felépítéséről, gyakorlati megközelítésben. A poolnak mindig van egy úgynevezett master tagja, ez a szerver lesz az alapértelmezett helye a virtuális gépeknek, és a pool létrehozásakor kell beállítani. Az, hogy egy host melyik pool tagja, az magukon a hostokon tárolódik, és a mastert nem is tudjuk eltávolítani a pooljából. Ha a master kiesik a poolból mondjuk leállás miatt, akkor egy másik host veszi át a master szerepet. Ilyenkor előfordulhat, hogy a XenCenter-ben elveszítjük a kapcsolatot a poolal, de csak annyi a teendő, hogy újracsatlakozunk az új masterre. Hogy melyik host kapta meg a master címet, azt a virtuális szerverek konzolján tudjuk megnézni a Resource pool configuration alatt, ugyanitt tudunk ki-, és belépni egy poolba. De a master szerepét át is ruházhatjuk manuálisan a XenCenterben, ekkor nem veszítjük el a kapcsolatot. Vigyázni kell, hogy a master szerepet mindig ruházzuk át egy másik hostra leállítás előtt, ugyanis ha ez nem sikerül, akkor érdekes dologgal találhatjuk szembe magunkat. Ha ugyanis a master úgy esik ki a hálózatból, hogy a többiek nem tudták átvenni a master szerepet (ha egyszerre lekapcsoltuk mindet, vagy a master egyedül volt a rendszerben), akkor a többiek bootolás után nem lesznek képesek belépni a poolba. A XenServer konzolon az jelenik meg, hogy nincs használható interfész, illetve hogy a hálózat nincs konfigurálva. A XenCenterrel sem tudunk majd rájuk csatlakozni, sőt új poolt sem tudunk így létrehozni velük. Az sem jelent megoldást, ha eltávolítjuk a virtuális hálózati kártyát, vagy új virtuális gépet hozunk létre nekik. Ha a master menthetetlenül sérült (pl. letöröltük) akkor csak a többi gép újratelepítésével, és új virtuális hálózati kártyákkal lehetséges. Hogy ezt elkerüljük, sose állítsuk le a mastert a szerep

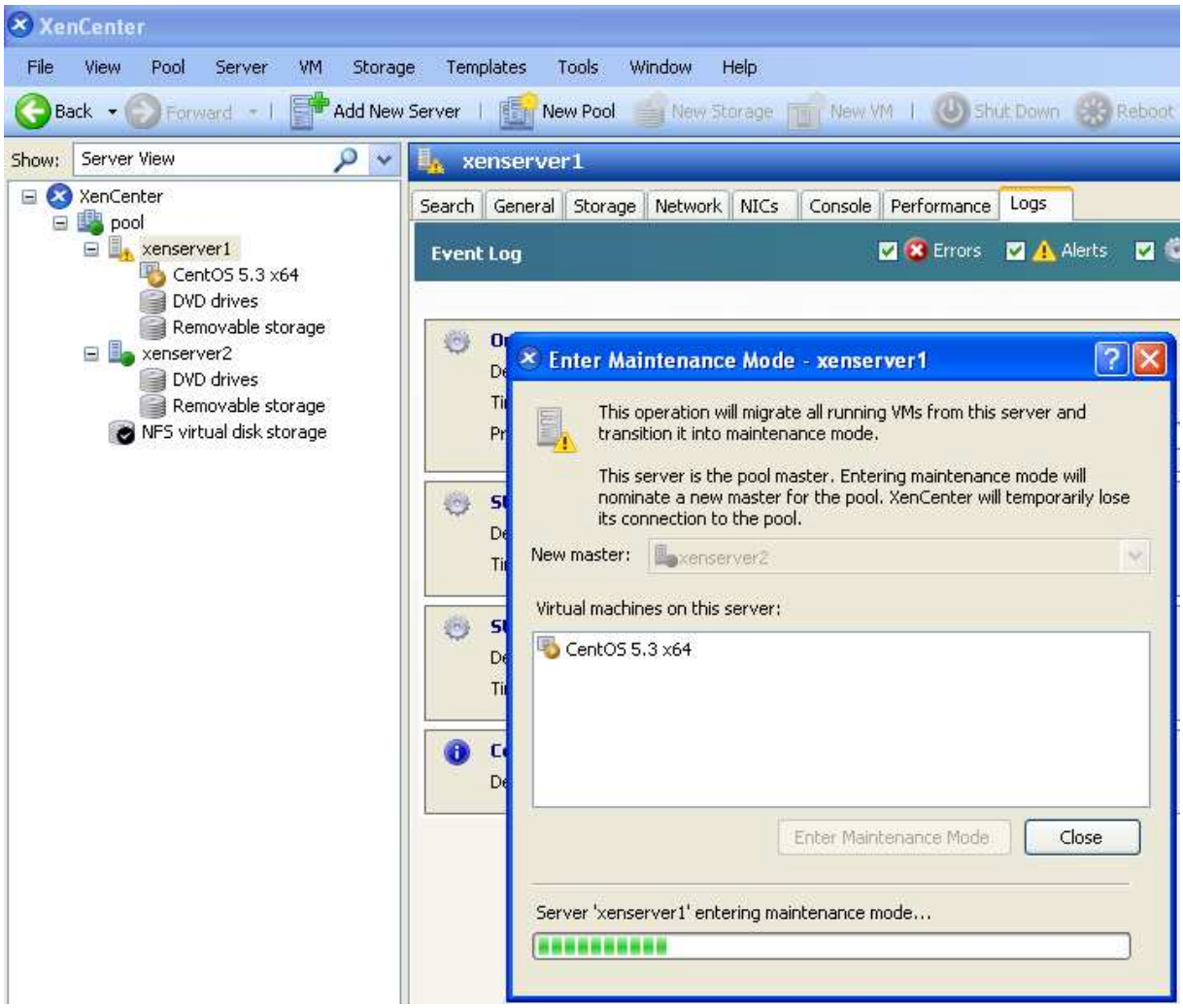
átruházása nélkül. Ha a master helyreállítható, akkor indítása után a többi gépet újraindítva ismét elérhető lesz a teljes hálózat, visszaáll a rend.



16. ábra. A Xen Tools működik, így indulhat a XenMotion

A WorkLoad Balancing engedélyezéséhez tehát az Essentials szükséges a licencek miatt, valamint egy SQL Server, ahol az adatait tárolja majd a WLB Server. Ez a WLB szerver is a Citrix oldaláról tölthető le. A WLB engedélyezésekor meg kell adnunk a gép IP címét és alapértelmezetten a 8012-es portot kell szabaddá tennünk számára, valamint két felhasználó név-jelszó párost is meg kell adni. Az első az adatbázis eléréséhez szükséges (általában a Windows felhasználó adatai szükségeltetnek ide, amennyiben mást nem adtunk meg az SQL Server eléréséhez, a második helyre pedig a XenServer adatait írjuk (a masterét). Csatlakozni ne a 127.0.0.1-es címmel próbáljunk, hiába azon a gépen található a kiszolgáló, hanem a belső hálózati IP címével. Ha sikerült csatlakozni, először kiválaszthatjuk, hogy miként szeretnénk

használni a rendszert: minden virtuális hoston azonos mennyiségű virtuális gép fusson, vagy vegye-e figyelembe a teljesítményt, és az alapján helyezze el a virtuális gépeket, akár egyenlőtlen arányban is, ami a számukat illeti. Ezután különféle küszöböket állíthatunk be, amik kiváltják majd a kiegyensúlyozás folyamatát, választhatunk merevlemez és hálózati I/O műveletek, illetve processzor és memóriakihasználtság értéke által meghatározott határok közül is, illetve ezekhez fontossági sorrendet rendelhetünk.



17. ábra. A XenCenter automatikusan migrálja a virtuális gépet, szerverleállítás esetén

4.4. Tapasztalatok összegzése

Sikerült tehát mind a VMware ESX-szel, mint a Hyper-V-vel, mind a Citrix XenServerrel létrehozni egy magas rendelkezésre állású rendszert. Láthattuk, hogy a legnagyobb szabadságot a VMware adja számunkra guest clustering esetében, mivel megvalósítja a teljes processzor virtualizációt, lehetővé téve ezáltal, hogy a nem nyílt forráskódú rendszerek is futhassanak egy virtuális host alatt. A memóriatúlfoglalás képessége pedig lehetővé teszi, hogy akár mindössze 4 gigabájt RAM-mal is több ESX Servert futtathassunk. 2 gigabájt memóriaigénye első hallásra soknak tűnhet, viszont cserébe az ember egy kiváló rendszert építhet fel.

A Microsoft Hyper-V a guest clustering szempontjából nem a legjobb választás, ugyanis futásához hardveres támogatás szükséges, ami a virtuális processzorok esetében nem valósul meg. Így virtuális gépeket nem tudunk telepíteni a virtuális hostjainkra, viszont vannak szolgáltatások, amelyekkel mégis ki tudjuk használni a klaszter képességeit. Hozzá kell tenni, hogy ha nem virtuális hostokat használunk, akkor itt is elérhetővé válnak a magasszintű virtualizációs képességek, úgy mint a Live Migration és a Quick Migration.

A XenServer egy köztes megoldásnak nevezhető, ugyanis képesek vagyunk vele a virtuális hostok alatt futtatni más gépeket, azonban a lehetőségek meglehetősen korlátozottak, és a grafikus felület elővarázsolása ebben az esetben nem triviális feladat, így kezdők számára nem ajánlott. Ugyancsak elmondható, hogy fizikai hostok virtualizálásakor a Xen is alkalmazza a hardveres támogatást amennyiben lehetősége van rá, így kedvezőbb a helyzet.

A VMware ESX profizmusának azonban ára is van, a VMotion, Storage VMotion, HA és DRS képességekért alaposan a nadrágunk zsebébe kell nyúlni. A licencelés sem a legegyszerűbb feladat akármelyik technológia mellet is döntünk. A Quick és a Live Migration majdnem helyettesítik a VMware magasszintű technológiáit, azonban még nem sikerült teljesen ledolgozni a hátrányt. A Xen esetében a XenMotion-ért nem kell külön fizetni, ami pozitívum, viszont a HA és WLB már plusz költségeket varr a nyakunkba. Bármelyik mellett is dönt az ember biztos, hogy nagymértékben hozzájárul a rendszere biztonságosabb működéséhez, és a megbízhatóbb szolgáltatás biztosításához.

5. Végszó

A dolgozatomban összehasonlítottam a három legismertebb virtualizációs megoldást vendég klaszterizáció (guest clustering) szempontjából. Célom az volt, hogy megismertessem az olvasóval a telepítések folyamatát, az esetlegesen felmerülő problémákat, ezek megoldásait, és hogy érthetővé tegyem a technológiák közötti különbségeket. Remélem, hogy sikerült rávilágítanom, hogy milyen lehetőségek rejlenek a virtualizáció ezen formájában, és hogy felkeltettem az olvasó érdeklődését maga a virtualizáció iránt is. Véleményem szerint fontos, hogy ügyeljünk környezetünkre, és a virtualizáció az egyik nagy lépés, amit az informatikával foglalkozó emberek tehetnek a környezetbarát infrastruktúra kialakításának érdekében. A jövőben minden valószínűség szerint tovább hódít majd, így érdekes olvasnivaló lehet azok számára is, akik szeretnének megismerkedni az alapokkal.

Irodalomjegyzék

Magyar nyelvű irodalom

[1] Szerver virtualizációs technológiák; Beregszászi Alex, Szalai Ferenc (2006)

[2] Virtualizációs Technológiák Mérési Útmutató; Tóth Dániel (2009)

[3] Virtualizációs technológiák; Suri Gusztáv, Tóth Ferenc (2009)

[4] Virtualizáció, Novell virtualizációs megoldások; Varga Zsolt

[5] Virtualizáció, Esettanulmány: a XEN VMM; Csapó Ádám, Kasza Bálint (2007)

[6] http://www.microsoft.com/hun/technet/article/?id=8d33f817-0e07-4695-af12-f93d13fde8f7

[7] https://sauron.inf.mit.bme.hu/Edu/VirtualizacioValaszthato/virttech2009.nsf/eloadasokPage?OpenPage

[8] http://lepenyet.spaces.live.com/blog/cns!8A601C211789FCC8!3167.entry

[9] http://computerworld.hu/takarekossag-virtualizacioval.html

[10] http://hu.wikipedia.org/wiki/Alkalmaz%C3%A1svirtualiz%C3%A1ci%C3%B3

[11] http://www.microsoft.com/hun/technet/article/?id=045ee48b-700c-4p0a0-a1e4-4271cf63516b

[12] http://proserver.sodamediacenter.com/public/technet.html

[13] http://blogs.technet.com/csabi_items/archive/2007/12/10/izol-ci-s-integr-ci-hyper-v-m-lyrep-l-s.aspx

[14] http://hu.wikipedia.org/wiki/Futásidejű_fordítás

Angol nyelvű irodalom

[15] Memory Resource Management in VMware ESX Server; Carl A. Waldspurger (2002)

[16] A Comparison of Software and Hardware Techniques for x86 Virtualization; Keith Adams, Ole Agesen

[17] http://www.scribd.com/doc/11724318/Virtualization

[18] http://nexus.realtimepublishers.com/tips/Application_Virtualization/Part_1_-_What_Is_Application_Virtualization.php

[19] http://www.webopedia.com/DidYouKnow/Computer_Science/2007/hardware_assisted_virtualization.asp

[20] http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf

[21] http://www.servercare.nl/Lists/Posts/Post.aspx?ID=61

[22] http://blogs.technet.com/mghazai/archive/2009/12/12/hyper-v-guest-clustering-step-by-step-guide.aspx

[23] http://www.techhead.co.uk/how-to-configure-openfiler-v23-iscsi-storage-for-use-with-vmware-esx

[24] http://technet.microsoft.com/en-us/library/cc755059%28WS.10%29.aspx

[25] http://www.openfiler.com/learn/how-to/graphical-installation

[26] http://en.wikipedia.org/wiki/Xen

[27] http://www.usenix.org/events/sec00/full_papers/robin/robin_html/

[28] http://www.virtualizationadmin.com/articles-tutorials/vmware-esx-articles/vmotion-drs-high-availability/configure-vmware-high-availability-vmha.html

[29] http://www.macwindows.com/emulator.html

[30] http://en.wikipedia.org/wiki/X86_instruction_listings

[31] http://en.wikipedia.org/wiki/Failover

[32] http://en.wikipedia.org/wiki/Ring_(computer_security)

[33] http://en.wikipedia.org/wiki/Remote_Desktop_Protocol

[34] http://en.wikipedia.org/wiki/Operating_system-level_virtualization

[35] http://en.wikipedia.org/wiki/Hypervisor

Köszönet

Ez úton szeretnék köszönetet mondani témavezetőmnek, Krausz Tamásnak a diplomamunka megírásában nyújtott segítségéért, és szakmai támogatásáért.

1. sz. melléklet. ESX overhead

Virtual CPUs	Memory (MB)	Overhead for 32-Bit Virtual Machine (MB)	Overhead for 64-Bit Virtual Machine (MB)
1	256	87.56	107.54
1	512	90.82	110.81
1	1,024	97.35	117.35
1	2,048	110.40	130.42
1	4,096	136.50	156.57
1	8,192	188.69	208.85
1	16,384	293.07	313.42
1	32,768	501.84	522.56
1	65,536	919.37	940.84
2	256	108.73	146.41
2	512	114.49	152.20
2	1,024	126.04	163.79
2	2,048	149.11	186.96
2	4,096	195.27	233.30
2	8,192	287.57	325.98
2	16,384	472.18	511.34
2	32,768	841.40	882.06
2	65,536	1,579.84	1,623.50
4	256	146.75	219.82
4	512	153.52	226.64
4	1,024	167.09	240.30
4	2,048	194.20	267.61
4	4,096	248.45	322.22
4	8,192	356.91	431.44
4	16,384	573.85	649.88
4	32,768	1,007.73	1,086.75
4	65,536	1,875.48	1,960.52

Plágium - Nyilatkozat

Szakdolgozat készítésére vonatkozó szabályok betartásáról nyilatkozat

Alulírott (Neptunkód: ……) jelen nyilatkozat aláírásával kijelentem, hogy a

……..

című szakdolgozat/diplomamunka

(a továbbiakban: dolgozat) önálló munkám, a dolgozat készítése során betartottam a szerzői jogról szóló 1999. évi LXXVI. tv. szabályait, valamint az egyetem által előírt, a dolgozat készítésére vonatkozó szabályokat, különösen a hivatkozások és idézések tekintetében.

Kijelentem továbbá, hogy a dolgozat készítése során az önálló munka kitétel tekintetében a konzulenst, illetve a feladatot kiadó oktatót nem tévesztettem meg.

Jelen nyilatkozat aláírásával tudomásul veszem, hogy amennyiben bizonyítható, hogy a dolgozatot nem magam készítettem vagy a dolgozattal kapcsolatban szerzői jogsértés ténye merül fel, a Debreceni Egyetem megtagadja a dolgozat befogadását és ellenem fegyelmi eljárást indíthat.

A dolgozat befogadásának megtagadása és a fegyelmi eljárás indítása nem érinti a szerzői jogsértés miatti egyéb (polgári jogi, szabálysértési jogi, büntetőjogi) jogkövetkezményeket.

hallgató

Debrecen,