TechNet MAGAZIN

2008. MÁJUS – JÚNIUS

MÉLYVÍZ, CSAK ÚSZÓKNAK!

Rendszerfelügyelet

IAG 2007
Távoli elérés egy igazi nagyvaddal, SSL VPN alapon

ALKALMAZÁSVIRTUALIZÁCIÓ
A virtualizációs megoldások általában azt ígérik, hogy a tűzoltás mellett másra is jut idő

SQL SERVER 2008: ADATTÁRHÁZ-ÚJDONSÁGOK
Hatalmas mennyiségű adatokkal operáló adattárházakat tölthetünk fel és kezelhetünk

WINDOWS DEPLOYMENT SERVICES
Bármelyik Microsoft operációs rendszer telepíthető tömegesen

ACTIVE DIRECTORY: INFORMÁCIÓK LEKÉRDEZÉSE
A PowerShell ereje az üzemeltetési feladatok során

Ára: 999 Ft

9 771586 518005 08003

Microsoft TechNet

Milyen elvárásai vannak Önnek, ha
adattárolási feladatokhoz,
fájl- és nyomtatási feladatok kiszolgálásához
vagy egy webkiszolgálóhoz keres megfelelő eszközt

?

– Legyen megfelelő teljesítménye!
Az Intel Xeon Dual Core processzor és a PC2-5300 DDR2 memória biztosítja a kívánt számítási teljesítményt.

– Legyen elegendő tárolókapacitása!
A tárolókapacitásról a 3 GB/s Serial ATA-II-es diszkek gondoskodnak, akár 750 GB diszkenkénti kapacitással.

– Legyenek biztonságban a tárolt adatok!
Az adatok biztonságáról az integrált Raid vezérlő gondoskodik, amely Raid-0-ra és Raid-1-re képes. Kiegészítő adapterrel a Raid szintek tovább növelhetők!

– Legyen üzembiztos a működése!
A memóriahibákat az ECC hibajavítás küszöböli ki. A diszkek meghibásodását a rendszer előre jelzi (Predictive Failure Analisys). A diszkek egyes modellekben üzem közben cserélhetők. Továbbá a hűtés és a tápegységek is redundánsak lehetnek!

– Legyen egyszerű az installálása és a menedzsmentje!
Az installálást az IBM ServeGuide szoftver könnyíti meg. A rendszermenedzsment az IBM Director szoftveren keresztül történhet, ezt kiegészíti az integrált menedzsmentvezérlő vagy az opcionális menedzsmentadapter.

– Legyen megfelelő garanciája!
Az eszköz 3 év garanciával rendelkezik, következő munkanapi megjelenéssel. Igény esetén ez a szolgáltatási szint opcionálisan tovább bővíthető!

A megoldás: az IBM System x3200 szerver konfigurációja!

Intel Xeon 3040 Dual Core 1,87 GHz processzor

2x 512 MB PC2-5300 DDR2 memória

2x 250 GB üzem közben cserélhető, SATA-II diszk

Integrált Raid-0, Raid-1 funkcionalitás

CD-RW/DVD-ROM meghajtó

1 GB ethernetadapter

Integrált menedzsmentkontroller

400 W fix tápegység

3 év helyszíni garancia

A konfiguráció bruttó ára:

269 900 Ft

Vásárlási szándéka esetén keresse IBM partnereinket: http://www.ibm.com/hu/partners/systemx/
További információ: http://www.ibm.com/hu vagy http://www.ibm.com/businesscenter/smb/hu/hu/drumbeat
Az akció érvényessége: 2008. február 6. és 2008. március 31. között, vagy a készlet erejéig!
A megoldás: az IBM System x3200 szerver konfigurációja!

Az ajánlat 2008. március 31-ig érvényes. Az árfolyamváltozásból eredő árváltozás jogát az IBM fenntartja. A viszonteladók egyéni árképzése miatt a végső ár változhat. Jelen akció kizárja bármely más promóció vagy különleges feltétel érvényességét. Az IBM fenntartja a jogot az ajánlat előzetes bejelentés nélküli megváltoztatására vagy visszavonására.

Avnet Technology Solutions Kft.
H - 1117 Budapest, Budafoki út 91-93. IP West Irodaház
Telefon: 06 1 888 2333
Fax: 06 1 888 2334
E-mail: ats.hu@avnet.com
Web: www.avnet.hu

AVNET

IBM Premier Business Partner

ÚTRAVALÓ

A nyári napok a szakmai gyűjtőmunkára is megfelelő alkalmat kínálnak.

Somogyi Csaba
(Csaba.Somogyi@microsoft.com)
Microsoft Magyarország

Mozgalmas évadot zárunk a TechNet Magazin mostani számával. Ritkán van lehetőségünk egyetlen éven belül ennyiféle technológiai újdonsággal megismertetni az olvasót, mint a Windows Server 2008 és az SQL Server 2008 megjelenéséhez kapcsolódóan. A rengeteg új információ közül igyekeztünk mindig olyanokat kiragadni, amelyek nemcsak újdonság mivoltukban érdekesek, hanem a napi gyakorlatban is hasznosíthatók, legyen szó biztonságról és megbízhatóságról, magas rendelkezésre állásról vagy éppen a virtualizációban rejlő lehetőségekről.

Arra biztatok minden olvasót, hogy forgassa a lapszámokat a kicsit nyugalmasabb nyári napokban és a nyári szabadságok ideje alatt is, akár újraolvasva azokat a cikkeket, amelyekre év közben csak a metrón vagy a buszon jutott idő. Egy-egy írást újraolvasva én magam is minden alkalommal találkozom olyan új gondolatokkal, amelyeket érdemes továbbgondolni, olyan ötletekkel, amelyeket érdemes kipróbálni.

Jó alkalom lehet ez a nyugalmasabb időszak arra is, hogy listát készítsünk magunknak a számunkra legfontosabb technológiákról, további információkat gyűjtsünk bevezetésük előtt, vagy éppenséggel mielőtt vizsgát teszünk belőlük. A magunk részéről igyekeztünk sokféle tartalommal és hasznosítható háttérinformáció-gyűjteménnyel segíteni az elindulást.

A szerzői csapat nevében kellemes nyarat és sikeres gyűjtögetést kívánok!


Windows Server 2008
Tanfolyami koktél a nyárra...
Kedvezményes csomagok, ajándék Ms Press könyvek
Áttérés Windows Server 2008-ra, Active Directory és hálózati újdonságok
2008. augusztus 4-8.
• Windows Server 2008 Active Directory és hálózati újdonságok Windows Server 2003-as vagy Windows 2000-es rendszergazdáknak
• Frissítse ismereteit az új verzióra és készüljön fel a hivatalos MCP vizsgára!
Rendelje meg a képzést kedvezményes, e-learning tanfolyami utalványokat és MCP vizsgát is tartalmazó tanfolyami csomagban!
Windows Server 2008 cluster szolgáltatások
2008. július 7-9.
• Haladó tanfolyam rendszergazdáknak, cluster üzemeltetőknek a téma egyik legjobb szakértőjének előadásában
• Windows Server 2008 failover cluster, storage, NLB, virtualizáció, iSCSI és Fibre Channel megoldások áttekintése, multi-site clustering
Most ajándék Windows Server 2008 Inside Out Microsoft Press kiadvány!
Szakképzési hozzájárulás, SA utalvány felhasználható. További információk a kedvezményekről weboldalunkon találhatók!
SZÁMALK
SZÁMALK Továbbképzés – Telefon: 203-0304/4122 mellék
www.szamalk.hu/tisza/windows2008

Címlapon

Rendszerfelügyelet

Infrastruktúra

Alkalmazásplatform

TechNet MAGAZIN

SZERKESZTŐSÉG
Főszerkesztő
Sziebig Andrea – *asziebig@itbusiness.hu*
Szakmai lektor
Budai Péter – *i-pbudai@microsoft.com*
Vezető szerkesztő
Varga János – *jvarga@itbusiness.hu*
Nyomdai előkészítés
Graffaello Kft.
Korrektor
Matula Zsolt
Lapterv és címlap
Emotion Bt.

Szerkesztőség és kiadó címe
IT-Business Publishing Kft.
1072 Budapest, Rákóczi út 28.
Tel.: 577-7970, fax: 577-7995

KIADÓ
Kiadja a Microsoft Magyarország megbízásából az IT-Business Publishing Kft.

A kiadásért felel
Sziebig Andrea ügyvezető
asziebig@itbusiness.hu
Tel.: 577-7999, fax: 577-7995

Hirdetési igazgató
Tóth-Haász Gabriella – thgabi@itbusiness.hu, tel.: 577-7972

Médiareferensek
Németh Krisztina – knemeth@itbusiness.hu, tel.: 577-7973
Rátóti Sarolta – sratoti@itbusiness.hu, tel.: 577-7971

Fax: 577-7995

Terjesztés
Terjesztett példányszám: 4700

Előfizethető a kiadó ügyfélszolgálatán:
terjesztés@itbusiness.hu
Az éves előfizetés díja 4990 forint.
MCP-k számára ingyenes!

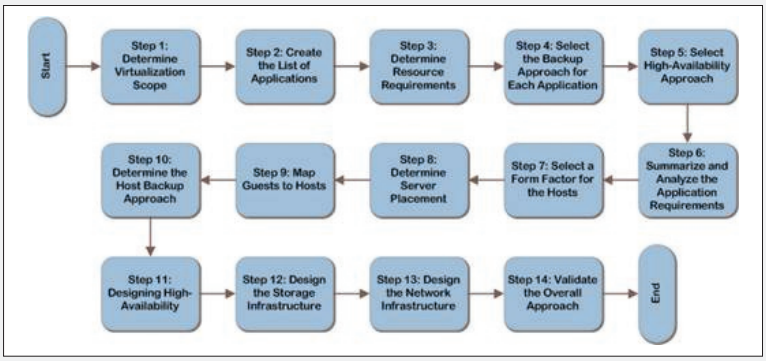
Nyomda:
Pauker Nyomdaipari Kft.
1047 Budapest, Baross utca 11-15.
Felelős vezető: Vértes Gábor ügyvezető igazgató
ISSN 1586-5185

SZERVER-KONZOLIDÁCIÓ

Apró ötletek az első lépésekhez.

A virtualizációs technológiák megítélése korántsem egységes az informatikusok körében. Vannak, akik számára már teljesen természetes, hogy a felügyelt rendszereik egy része virtuális gép. Az óvatosabbaknak ez a technológia még inkább csak játék. A cikkel továbbgondolós játékra csábítom az utóbbi tábor tagjait is, bízva abban, hogy a technológián túlmutató érvek közül is találnak megfontolandót.

Csaknem két évvel ezelőtt jelent meg a TechNet Magazinban Lepenye Tamás kollégám Tervezz merészen! című cikke. Ajánlom mindenkinek újraolvasásra. Akármekkorát változott ugyanis időközben a virtualizációs technológia, a megközelítés, a konszolidációs feladat megvalósítására alkalmazott módszer a mai napig megállja a helyét. Talán csak annyi változott, hogy megjelentek olyan alkalmazások, amelyekkel kevesebb fáradsággal, rendszerezett és jobban használható adatokat gyűjthetünk a konszolidálásra jelölt rendszerekről. Amikor pedig nekivágunk a feladatnak, akkor használhatjuk például a System Center Virtual Machine Managert, hogy fizikai gépeinket virtuális gépekké konvertáljuk. No de ne szaladjunk ennyire előre! Haladjunk nagyjából abban a sorrendben, ahogy a konszolidációs folyamat halad!



1. ábra. A konszolidáció tervezési folyamata a TechCenteren

Ha szerverkonszolidációra adjuk a fejünket, készüljünk fel arra, hogy hosszú ideig nem fogunk tudni látványos technológiai fejlesztéseket felmutatni. Ez a folyamat ugyanis hosszadalmas és gyakran nehézkes tervező-szervező munkával kezdődik. Legelsőként saját magunkban, illetve az IT-n belüli kollégák körében kell közös nevezőre, egységes álláspontra jutnunk a virtualizációt illetően. Tanulságos olvasmány a témában a Kusnetzky Group tanulmánya a virtualizációhoz kapcsolódó 10 legfontosabb mítoszról. Csak a négy legérdekesebb pontot emelném ki ezzel kapcsolatban:

1. Az én cégem máris készen áll a virtualizáció alkalmazására.
2. Nincs szükség különösebb szakértelemre az implementációhoz.
3. Nincs szükség részletes tervezésre, minden megoldható néhány kattintással.
4. A virtualizáció csökkenti a rendszer komplexitását.

A négy felvetett kérdést akár egymással összefüggésben is vizsgálhatjuk: a virtualizációs technológiák alkalmazása rövid távon egészen biztosan nem csökkenti a rendszer komplexitását, hiszen teljesen új, korábban nem használt technológiai rétegek kerülnek a rendszerbe, amelyeket egyrészt meg kell ismernünk, másrészt a felügyeletükhöz szükséges folyamatokat ki kell fejlesztenünk és dokumentálnunk. Hosszabb távon jelentkezhet a komplexitás csökkenése, ha a virtualizáció egyfajta katalizátorként hat, és ösztönöz bennünket a felügyeleti folyamatok pontosítására, jobb összehangolására és a lehető legtöbb feladat automatizálására. Ha többen dolgozunk az IT-szervezetben, akkor szükség lehet az egyes munkatársak feladatainak újragondolására és összehangolására is, hiszen a folyamatokat nekik kell végrehajtaniuk. Tehát igazából akkor állunk készen, ha az alapvető munkafolyamataink már rendben vannak, és van tartalék kapacitásunk a virtualizációs technológia bevezetésének idején jelentkező többletfeladatok elvégzésére. Kiemelten figyeljünk erre, ha egymagunk vagyunk „az IT”: mérjük fel, mennyi idő alatt, milyen kisebb lépéseken keresztül tudjuk elérni a kitűzött célt úgy, hogy közben nem hanyagoljuk el az élő rendszereket sem. Ezen a ponton máris a tervezésnél tartunk. Tervezni pedig kötelező! Mégpedig több irányban egyszerre. Mindenképpen szükség lesz egy pénzügyi tervre. Elég nagy a valószínűsége annak, hogy a konszolidációs folyamatot nem tudjuk véghezvinni pénzügyi befektetés nélkül, tehát feltétlenül meg kell győznünk azokat a döntéshozókat, akik finanszírozzák terveinket. Meggyőzésükhöz több helyről is gyűjthetünk érveket, néhány példát a következő fejezetben bemutatok.

A következő lépés a konszolidációs útiterv: milyen gépeket, szolgáltatásokat mozgatok

virtuális platformra, milyen sorrendben, milyen módszerrel, és mi a tervem, ha valami nem sikerül (ez az a bizonyos gyakran elfelejtett roll-back plan). E lépések tervezéséhez remekül használható a fentebb említett cikk. A másik vonulat a hosszabb távú tervezés: ha gondot okozott a fizikai szerverek túlzott elszaporodása (egyedi feladatokra, hosszabb távú koncepció nélküli beszerzések), akkor mit fogunk kezdeni az elszaporodó virtuális gépekkel? A virtuális gépek olcsók, nem lesz meg az a beépített korlát, amit a pénzügyi lehetőségek jelentettek a fizikai gépek beszerzésekor. Sőt, virtuális gépek esetén még a licencelés is egyszerűbb. Megvannak-e az eszközeink a hibriddé vált számítóközpontunk felügyeletére, van-e eljárásunk a licenckövetésre? Megfelelően nyilvántartjuk, dokumentáljuk-e virtuális tárolóinkat (storage) és hálózatainkat? Több olyan kérdés, amire megnyugtató megoldást kell találnunk, és ez még mind mindig csak a tervezés.

A szakértelem kérdése talán a legegyszerűbb, ezen a területen van ugyanis talán a legnagyobb motiváló erő: a technológiai érdeklődés. Aki virtualizációs megoldás bevezetésére adja a fejét, azt már valamilyen mélységben megragadta a technológia és a benne rejlő lehetőség. Szerencsére a bevezetéshez szükséges eszközök használata nem túl bonyolult. A szakértelem kérdése sokkal inkább a tervezési és az üzemeltetésbevezetési fázisban nyer jelentőséget, ahol rendszerszinten kell gondolkodni a siker érdekében. Hogy csak egyetlen üzemeltetési példát említsek: meg kell változtatnunk a javítócsomagok telepítésére vonatkozó eljárásainkat. A szokásos eljárás nagy valószínűséggel használható marad a virtuális gépek többségére és a virtualizációba nem bevont fizikai gépekre, a virtuális gépeket futtató szülőpartíciók frissítése viszont jobban szervezett eljárást igényel, hiszen amikor ezeket frissítjük, a rajtuk futó virtuális gépek szolgáltatásai is kiesnek. Ha technológiai szempontból nem is, de szervezési szempontból mindenképpen összetettebb a feladat. (Értjük már, miért ajánlott a Windows Server 2008 Core-telepítés mint szülőpartíció? Kevesebb frissítési ciklus, kevesebb fejfájás a szervezés körül.)

Nézzünk bele néhány fontosabb folyamatba, amellyel a szerverkonszolidáció során találkozhatunk, és ismerkedjünk meg néhány olyan eszközzel, ami segíthet a kezdeti fázisok nehézségeit legyűrni.

Tanuljunk közgazdászul!

Szó esett róla korábban, hogy nagy valószínűséggel szükség lesz valamennyi beruházásra, ha szerverkonszolidációba fogunk. Az üzleti döntéshozók meggyőzése, támogatásuk elnyerése nem könnyű feladat, hiszen nem a saját szakmánk fegyvertárából kell érveket kerítenünk, hanem olyan gazdasági tényekkel kell előállnunk, amelyekhez a meggyőzendő fél valószínűleg sokkal jobban ért. Hogyan keressünk tehát muníciót? Milyen érvekkel induljunk pénzügyi források megszerzésére?

Az egyik lehetséges út a jelenlegi rendszerekben kimutatható erőforrás-pazarlás kimutatása, noha ezzel óvatosan kell bánni, nehogy a mi hibánkként vagy hozzá nem értésünk jeleként jelenjen meg a tény, hogy erőforrások állnak kihasználatlanul. Mutassuk meg inkább a dolog pozitív oldalát: eddig műszaki megfontolások, például biztonsági okok miatt külön gépekre kellett telepítenünk bizonyos alkalmazásokat, ami pazarláshoz vezetett, mert nem tudtunk a feladathoz éppen elégséges hardvert vásárolni, hanem csak jóval nagyobb teljesítményűt. Most az új technológia lehetővé teszi az erőforrások koncentrálását és átcsoportosítását, amellyel hosszabb távon beszerzések válhatnak szükségtelenné. Az érveinket alátámasztó mérési adatok összegyűjtése eddig sem volt lehetetlen, hiszen régóta elérhetők a megfelelő teljesítményszámlálók, amelyekből ezeket kinyerhetjük. Szerencsés esetben készen is kaphatjuk az adatokat a System Center Operations Manager (vagy kisebb cég esetén System Center Essentials) jelentéseiből, de mit tegyünk, ha egyik rendszer sincs bevezetve?

Mielőtt tömeges performanciamérésbe kezdünk, nézzünk inkább körül például a Microsoft weblapjain. Egy nemrégiben közzétett ingyenes kis eszköz sok egyéb dolog mellett a szerverkonszolidációhoz szükséges adatok összegyűjtésében is segíthet. A Microsoft Assessment and Planning Toolkit nagyon hasznos információkat képes nyújtani például Vista-, Windows Server 2008- vagy Office 2007-bevezetés előtt, amikor megmutatja, melyik eszközünk áll készen az adott szoftver futtatására. Szerverkonszolidációhoz pedig még a teljesítményadatokat is összegyűjti a számunkra. A telepítőcsomag mindössze 50 megabájt, de a működéshez szükség van egy SQL-szerverpéldányra is, ami lehet egy más célra is használt SQL valahol a hálózaton, vagy kérhetjük a telepítőt, hogy töltse le és telepítse az SQL Express egy példányát. A telepítés másik előfeltétele az Office csomagból a Word és Excel jelenléte, mert az alkalmazás ebben a két formátumban készíti el (makrókon keresztül) a jelentéseit, amelyeket akár közvetlenül nyomtathatunk is, ha az angol nyelv nem akadály.

A telepítést követően készítsünk egy olyan adatbázist, ahová a szoftver a leltárt és a teljesítményjelentéseket elkészíti, majd állítsuk össze a felmérendő gépek listáját. Míg a bevezetési feladatokhoz szükséges lépésekben sokféle bemeneti adatforrás között választhatunk (Active Directory, alhálózat, SNMP community stb.), addig a konszolidációs felméréshez nekünk kell egy géplistát gyártanunk. Ezt legegyszerűbben a DNS-rekordok listájából generálhatjuk, kihagyva az érdektelen gépeket (például asztali gépek és notebookok). Ezzel a módszerrel viszont akár tartományon kívüli gépeket is felmérhetünk, például az IP-címük megadásával a listában. A következő lépésben meg kell adnunk a felméréshez használandó jogosultságokat. Célszerű először egy, a legtöbb gépen rendszergazdai joggal rendelkező fiókot megadni, és azt mondani, hogy ez mehet minden felmérendő gépre, majd megadni a kivételeket (például a nem tartománytag gépek rendszergazda-fiókjait). Használható eredményhez az is szükséges, hogy a megfelelő tűzfalszabályok érvényesüljenek (WMI, Remote Performance Logging, Remote Registry). Ezután már csak meg kell adnunk, mennyi időn keresztül szeretnénk gyűjteni az adatokat. Az alapértelmezés egy óra, ami elég is lehet, ha például a napközbeni tipikus terhelésre vagyunk kíváncsiak. Ha az átlagos terhelést szeretnénk látni, hagyjuk futni a felmérést legalább 24 órán át! Az eredmény a korábban említett összegző fájlokban jelenik meg. A *2. ábrán* a virtualizáció előtti átlagos kihasználtsági mutatókra láthatunk példát. Ezek azok az értékek, amiket érdemes az üzleti döntéshozó figyelmébe ajánlani, illetve azt a lehetőséget, hogy például a processzorok kihasználtságát 10 százalék alatti értékekről 40-50 százalék feletti értékre tudjuk emelni a virtualizációs technológiákkal.

A másik két terület, ahol érveket találhatunk a pénzügyi döntéshozók meggyőzésére: a villamosenergia-felhasználás és a licencgazdálkodás. Kezdjük az energiával! Csak próbaképpen kikerestem az egyik hardvergyártó

	A	B	C	D
1	**Pre-Consolidation Utilization Report**			
2	This report provides details of the current utilization of machines in your network based on performance details collected earlier. The values indicate the utilization before each machine is virtualized for consolidation.			
3				
4	**Machine Name**	**CPU Utilization (%)**	**Disk Utilization (MB/s)**	**Network Utilization (MB/s)**
5	Server_110	2,49	5	0,19
6	Server_108	2,51	5,19	0,18
7	Server_100	2,75	6,92	0,19
8	Server_109	3,13	4,72	0,18
9	Server_113	1,29	0	0,19
10	Server_111	1,17	3,47	0,21
11	Server_101	2,6	4,04	0,18
12	Server_106	2,92	2,8	0,25
13	Server_115	2,87	25,7	0,15
14	Server_104	0,83	6,82	0,31
15	Server_116	2,74	4,98	0,16
16	Server_103	1,56	39,98	0,24
17	Server_105	0,83	2,95	0,31
18	Server_117	6,09	6,33	0,14
19	Server_107	2,74	4,45	0,21
20	Server_114	1,53	36,59	0,27
21	Server_112	1,17	5,48	0,22
22	MAP2-2K3SP2-Off	69,24	2,37	3,46

2. ábra. Kiszolgálók terhelésjelentése

néhány belépőszintű és középkategóriás kiszolgálóját. Az alacsonyabb osztályt tekintsük tipikusnak, mint egyedi funkciót ellátó szervereket (ezek amúgy is csaknem asztali PC kategóriájú gépek). A középkategóriát jelöljük meg mint virtuális gépek futtatására alkalmas kiszolgálót, és mindkét kategória esetén számoljunk a maximális kiépítettségre vonatkozó energiafelhasználással és hőleadással. A gép által termelt hővel kissé bonyolult számolni, hiszen a gyártók általában BTU/h mértékegységben adják meg az értéket, amit elég nehéz kezelni. Az egyszerűség kedvéért konvertáljuk át az értéket wattra, és számoljunk vele mint többletfogyasztással (hiszen tulajdonképpen a felesleges hő elvezetésére ezt az energiamennyiséget valóban be kell vinnünk, csak éppen a klímaberendezés oldalán). A képlet elég egyszerű: 1000 BTU/h = 293 W. A gyártói műszaki dokumentációra épülő átszámított adatokat az *1. táblázat* mutatja.

Az eredmény egészen érdekes. Ha legalább három kisebb teljesítményű gépet össze tudunk vonni egy középkategóriás kiszolgálóra, akkor az energiafelhasználás már csökkenthető. Csak egy kis ellenőrzés: a kettes típusú szerverből három darab 2751 W energiát fogyaszt óránként, ha ezeket konszolidáljuk a négyesre, ez az érték 2339 W-ra csökken, a megtakarítás 15 százalék körüli. Egy negyedik kettes típusú gép konszolidálása pedig már 35 százalék feletti megtakarítást jelenthet.

A szoftverköltségek számításához emlékezzünk arra a szabályra, hogy a Microsoft szerveroperációs-rendszerei verziójuktól függően tartalmaznak további licenceket virtuális gépek operációs rendszereihez. Így a Windows Server 2003 és Windows Server 2008 Standard változatai egy, az Enterprise változatok négy, a Datacenter változatok korlátlan számú további operációsrendszer-licencet tartalmaznak. Ráadásul csak az aktív példányokkal kell számolnunk, amelyek aktuálisan futnak. A további számításokban online kalkulátorok segíthetnek bennünket. (Figyelem, az árakat csak viszonyítási alapként használjuk, a pontos adatokért kérdezzük szoftverszállítónkat!)

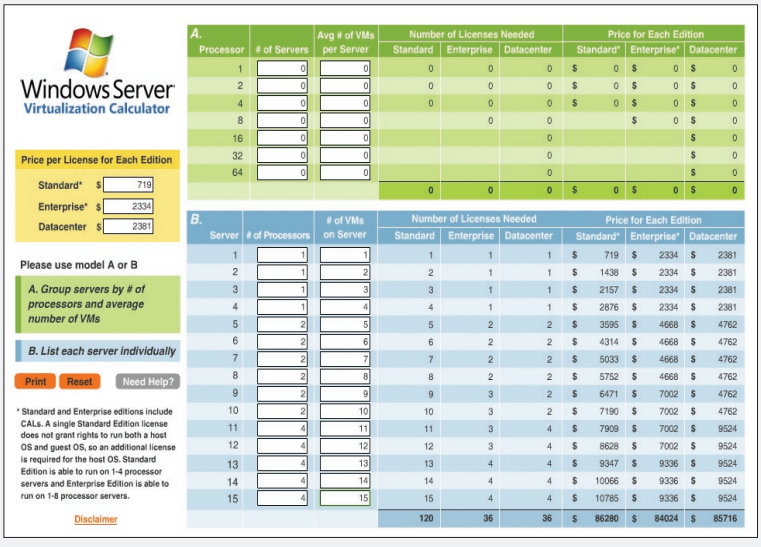
A mellékelt képen csak az operációs rendszerek költségét számoltattam ki (B. mező – világoskékkel) egy-, két- és négyprocesszoros konfigurációkra, ami a gyakorlatban akár négy-nyolc-tizenhat processzormagot is jelenthet, hiszen a költségeket fizikai processzorra kell számítani. Az eredmény itt is érdekes: négy virtuális gép egy egyprocesszoros rendszeren (akár négy processzormaggal) már olcsóbb lehet Windows Server Datacenter Edition verzióval, mint ha a szükséges Standard változatokat vásárolnánk meg. Nem beszélve arról, hogy a Datacenter változat korlátlan számú további operációs rendszert enged meg, amíg a processzorok száma nem változik (ez a verzió ugyanis processzoronként licencelt). Érdemes a kalkulátorokkal néhány „mi lenne, ha" játékot eljátszani, hogy aztán a leggazdaságosabb megoldással állhassunk elő.

Én túl kicsi vagyok ehhez…

Gondolkodjunk-e szervervirtualizációról, ha egészen kis cégnél dolgozunk, ahol kiszolgálóból is esetleg csak egy van? Amikor először tettem fel magamnak a kérdést, akkor szinte azonnal rávágtam: nem. Ne erőltessünk egy megoldást ott, ahová nem való. Aztán feltá-

1. táblázat

	Tápegység (W)	BTU/h	Energiaigény (W/h)	Megjegyzés
Szerver 1	600	1915	1161	Belépő szint, álló formátum.
Szerver 2	365	1885	917	Belépő szint, álló formátum.
Szerver 3	630	1794	1156	Belépő szint, álló formátum.
Szerver 4	1170	3990	2339	Középkategória, álló formátum.
Szerver 5	500	1194	850	Belépő szint, rack formátum.
Szerver 6	640	2174	1277	Belépő szint, rack formátum.
Szerver 7	1172	3990	2341	Középkategória, rack formátum.



3. ábra. Webhely a szoftverköltségek számításához

madt bennem a kisördög, és arra gondoltam, tervezzünk itt is merészen. Tegyük fel, van a vállalatunknál egy Small Business Server 2003 kiszolgálónk. Három-négy éve teszi a dolgát a sarokban, néha már bizonytalankodik a hardvere, fontolgatjuk a cseréjét, hiszen már a garanciája is rég lejárt.

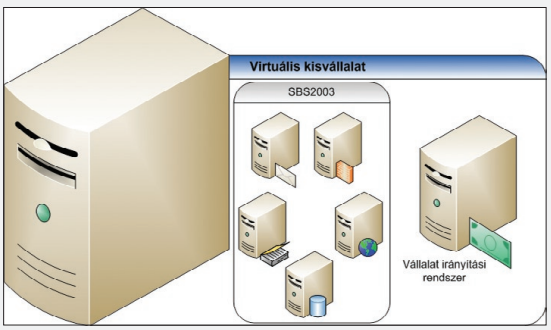
Felmerül az igény, hogy szükség lenne egy másik gépre, például egy könyvelési vagy vállalatirányítási szoftver számára. Milyen megoldások közül választhatunk? Vásárolhatunk például két új belépőszintű szervert: a megszokott SBS-ünket migráljuk az egyikre, a pénzügyi szoftvert feltesszük a másikra. Nincs is ezzel a megoldással semmi baj, ha csak az nem, hogy a szép új processzoraink alig-alig dolgoznak 5 százalék feletti teljesítményen. Ennél még a legelső gőzmozdonyok is jobb hatásfokkal működtek a XIX. század közepén.

Van más lehetőség? Gondoljunk egy merészet, és mondjuk azt, hogy igen. Vásároljunk egy kicsit komolyabb szervert, de csak egyet. Ha jól választunk, a hardverköltségben még meg is takaríthatunk egy keveset. Célozzunk meg, mondjuk, egy négymagos processzort, 4 GB memóriát és néhány SATA csatolófelületű lemezt (legyen ebből több, mondjuk 2×80 GB és 4×250 GB) tartalmazó konfigurációt. Vegyünk hozzá egy Windows Server 2008 Standard Edition operációs rendszert a 64 bites szériából, ami tartalmazza a Hyper-V virtualizációs megoldást. Licencekkel máris rendben vagyunk, hiszen a Small Business Server 2003-licencünk megvan, a Windows Server 2008 pedig önmagán felül megenged egy további példányt, így a pénzügyi rendszerünk alá is megvan az operációs rendszer.

Mielőtt nekifognánk a rendszerek átszervezésének, készítsünk legalább egy vázlatot arról, hogy hová szeretnénk eljutni: lesz egy gazdagépünk két virtuális géppel, valahogy úgy, ahogy az *4. ábrán* látható.

Ha már látjuk az alapstruktúrát, kezdjük el az erőforrások elosztását. Induljunk természetesen a gazdagép operációs rendszerével, hiszen sok szempontból ez a komponens is csak egy előfizető a rendszer erőforrásaira. Ha nagyon bátrak vagyunk, telepíthetjük a Windows Server 2008-at Server Core telepítési opcióval, de erre csak akkor vállalkozzunk, ha járatosak vagyunk ezen a területen, és még a Hyper-V távoli (nem tartományi) felügyeletével is boldogulunk. (Ennek konfigurálása elég bonyolult, lásd a hivatkozott blogsorozatot.) Bátorságunk jutalma lehet a gazdagép kisebb erőforrásigénye és a kevesebb alkalmazandó frissítés. Akármelyik változatot is választjuk, a gazdagép operációs rendszerét telepítsük a 80 gigabájtos diszkekből kialakított RAID1 kötetre.

A virtuális gépek méretezését kezdjük a Small Business Server 2003 irányából. Az alap egy Windows Server 2003 R2, 32 bites architektúrával. A Hyper-V-ben rendelkezünk megfelelő integrációs komponensekkel, azt kell csak a kis rajzunkra felvezetni, hogy összesen két virtuális processzort adhatunk gépünkhöz. Memóriából valószínűleg most sem rendelkezünk két gigabájtnál többel, ennél több nem kell a virtuális gépünkbe sem. A lemezek konfigurációjánál több választási lehetőségünk van: létrehozhatunk egyetlen RAID5 kötetet vagy két RAID1 kötetet a rendelkezésre álló lemezekből. A két külön kötet előnye, hogy az SBS így nem „közösködik" a pénzügyi alkalmazással, ha bármelyik virtuális gép intenzív lemezműveleteket végez, akkor legfeljebb a közös lemezvezérlő bizonyulhat szűk keresztmetszetnek. Az egyetlen nagyobb kötet viszont lehetőséget ad arra, hogy bármelyik gép tárolókapacitását növeljük szükség esetén a VHD-fájlok kiterjesztésével. Hogy a virtualizált SBS-kiszolgálónk nagy sebességgel érhesse el a neki szánt tárolókapacitást, most válasszuk azt a megoldást, hogy egy úgynevezett pass-through diszk formájában rendeljük a virtuális géphez a dedikált RAID1 kötetet. A pénzügyi alkalmazásunk virtuális gépéhez rendeljünk egy virtuális processzort és egy gigabájt memóriát. (A Windows Server 2008 operációs rendszerhez rendelhetnénk akár négy virtuális processzort is, de a kisvállalatoknál szokásos alkalmazásoknak általában nincs ekkora igényük, sőt csak ritkán képesek több processzort használni).



4. ábra. Egy virtuális kisvállalat sematikus felépítése

Az adattárolásra használjunk rögzített méretű VHD állományt, mert ennek a teljesítménye alig marad el a fizikai diszkétől (többek között, mert a rögzített méret miatt nem töredezik). Az induló méret legyen 80 gigabájt, ez valószínűleg hosszabb időre elegendő lesz. A fennmaradó kapacitást szükség esetén mentésekre használhatjuk, vagy akár újabb VHD állományt hozhatunk létre rajta, és felcsatolhatjuk az SBS alá kiegészítő tárhelyként. A végső diszkkonfiguráció valahogy így alakulhat.

A Small Business Server migrálása nem könnyű feladat, viszont könyvtárnyi az irodalma, csak keressünk rá néhány kulcsszóra az interneten, és már találunk is néhány alternatívát. Ha konzervatív (és biztonságos) utat szeretünk járni, akkor a migráció történhet egy biztonsági mentés visszatöltésével a virtuális gépbe, vagy választhatjuk a „swing migration" technikát, amikor egy ideiglenes (például egy PC-n futó) gépet vonunk be ideiglenes tartományvezérlőként, amíg a virtuális gépben újratelepítjük az SBS-t. Vállalkozó szelleműek kísérletezhetnek blokkszintű lemezmásoló szoftverekkel is, és átmásolhatják az SBS kiszolgáló diszkjének tartalmát a virtuális géphez rendelendő fizikai kötetre. Ez a megoldás akár működhet is, ha a másolás után a kötetet a virtuális gépünk IDE csatolójához rendeljük. A virtuális gép hardvere alapvetően szabványos, tehát van esélye annak, hogy a Plug&Play pótolja vagy lecseréli a hiányzó hardverkomponenseket, és a gép elindul. Gondosan vigyázzunk viszont az SBS eredeti lemezére (lemezeire), hogy legyen visszatérési lehetőségünk, ha mégsem sikerülne a migráció.

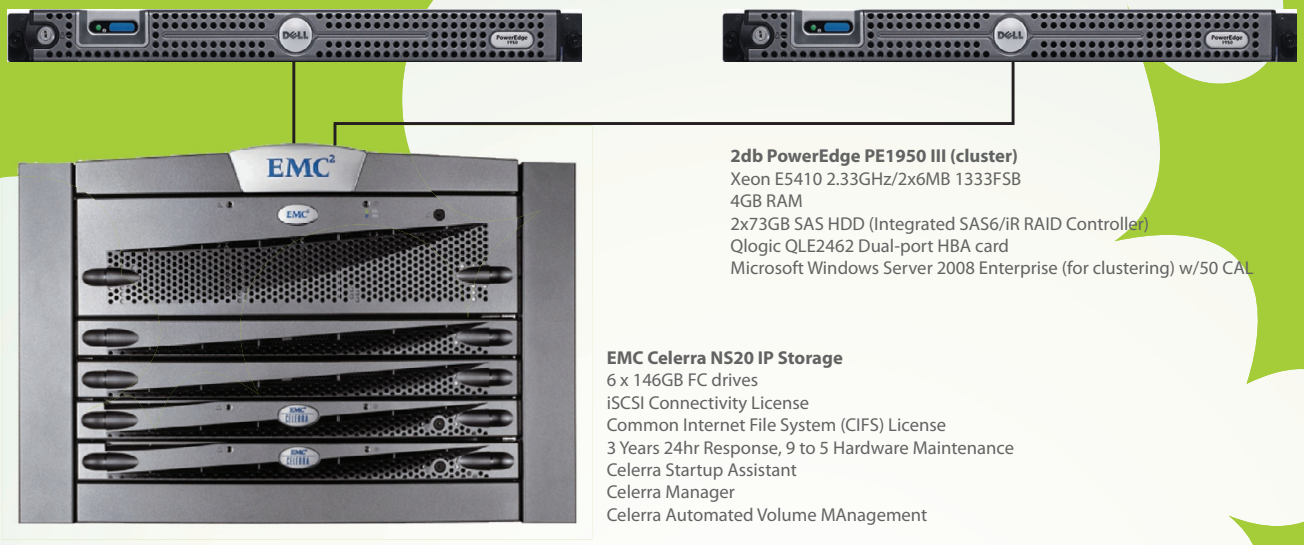
A Hyper-V integrációs komponensek a megfelelő teljesítményhez mindenképpen kellenek, tehát ha a migráció sikeres (járjunk bármelyik úton), ezt a csomagot telepítenünk kell. Ezekkel a komponensekkel pedig már élvezhetjük a szintetikus hálózati csatoló és a többi szintetikus eszköz előnyeit és a VMBus nyújtotta gyors kommunikációs csatornát. A migrációhoz képest a pénzügyi rendszer „zöldmezős" telepítése már valóságos felüdülés lesz, viszonylag kevés hibalehetőséggel.

Somogyi Csaba
(Csaba.Somogyi@microsoft.com)
Microsoft Magyarország

High-Availability Cluster

DELL-EMC HA Cluster Bundle

Teljeskörű DELL-EMC megoldás kis- és középvállalatok informatikai kihívásaira



2db PowerEdge PE1950 III (cluster)
Xeon E5410 2.33GHz/2x6MB 1333FSB
4GB RAM
2x73GB SAS HDD (Integrated SAS6/iR RAID Controller)
Qlogic QLE2462 Dual-port HBA card
Microsoft Windows Server 2008 Enterprise (for clustering) w/50 CAL

EMC Celerra NS20 IP Storage
6 x 146GB FC drives
iSCSI Connectivity License
Common Internet File System (CIFS) License
3 Years 24hr Response, 9 to 5 Hardware Maintenance
Celerra Startup Assistant
Celerra Manager
Celerra Automated Volume MAnagement

Főbb előnyök

- magas megbízhatóságú hibatürő megoldás a kis- és középvállalatok részére (file kiszolgáló , levelező rendszer)
- bevizsgált és minősített megoldás
- magas megbízhatóságú komponensek
- egyszerű menedzselhetőség
- rugalmasan bővítető
- központi adattárolóba integrált Microsoft file kiszolgáló
- alap infrastruktúra a Microsoft külső kiszolgálói számára (MS Exchange 2007, MS SQL 2008…)

4.999.000 Ft
(bruttó: 5.998.800 Ft)*

DELL EMC² where information lives Windows Server 2008

intel Xeon inside

HUMANSOFT INFORMATIKA

Microsoft GOLD CERTIFIED Partner

EMC² VELOCITY² PARTNER

Microsoft Gold és EMC Velocity Partner
– teljeskörű integrációs és támogatási szolgáltatások –

Dell | Authorised Service Provider

*Az árak 175 Ft/USD árfolyamig érvényesek és tájékoztató jellegűek, a változtatás jogát fenntartjuk! Az árak a hardver installációs szolgáltatásokat tartalmazzák, a további egyéni telepítés-igényével forduljon hozzánk bizalommal!

HUMANsoft Kft.
1037 Budapest,
Montevideo u. 8.
Tel.: (1) 270-7600
Fax: (1) 270-7679
www.humansoft.hu

ALKALMAZÁS-VIRTUALIZÁCIÓ

Már megint virtualizáció? Nem puszta divat ez? És mi az, hogy „alkalmazásvirtualizáció”? Nem operációs rendszereket szokás virtualizálni? Nem elég csupán egyfajta virtualizáció?

A rendszerüzemeltető informatikusok többsége minden szándék ellenére az állandó „tűzoltással” foglalkozik: folyamatosan esnek be hozzájuk az újabb és újabb panaszok, hibajegyek, miközben erejüket megfeszítve és gyakran automatizmus nélkül igyekeznek teljesíteni az olyan legitim kéréseket, mint a szoftvertelepítés, a gépcsere, a szoftverfrissítés és még sorolhatnánk. A virtualizációs megoldások általában – az alkalmazásvirtualizáció pedig konkrétan ebben az írásban – azt ígéri, hogy a tűzoltás mellett már másra is jut idő, mégpedig azért, mert a javasolt technológia az elvégzendő tevékenységek számát csökkenti, és emellett még az előforduló hibák valószínűsége is kisebb lesz.

Alapozás

Az alkalmazásvirtualizáció olyan technológia, amely elválasztja egymástól az operációs rendszert és a rá telepített szoftvert. Ahogy a szerver- vagy hardvervirtualizáció esetén a teljes operációs rendszerből, pontosabban azok virtuális lemezeiből egyetlen állomány keletkezik, az alkalmazásvirtualizáció is egyetlen állományba zsúfolja a virtualizált szoftvert. Ezt a trükköt persze az alkalmazás „nem veszi észre”, előttünk viszont, ahogy azt majd látni fogjuk, futurisztikus lehetőségek nyílnak meg.

A Microsoft és az alkalmazásvirtualizáció

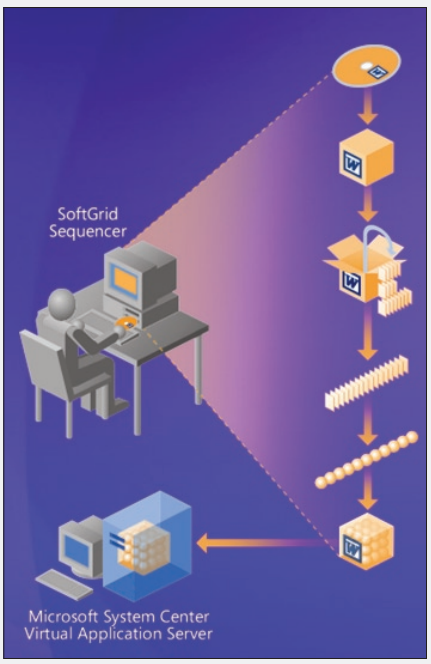
A virtualizáció elvét a telepítendő szoftverekre alkalmazni viszonylag új keletű megoldás. 1999 júniusában négy IT-szakember, *Harry Ruda, David Greschler, Stuart Schaefer* és *Owen Mysliwy* megalapította a Softricity nevű céget, s azt tűzték ki célul, hogy a szoftvert úgy lehessen elérni, mint ahogy ma az elektromosságot. (Software + Electricity = Softricity) A Softricity első terméke volt a SoftGrid, az első alkalmazásvirtualizációs megoldás. A dotcom-lufi kipukkadását a cég túlélte, az ígéretes termék szépen fejlődött, a vállalat nevét felkapták.

A Microsoft, amely 2003-ban felvásárolta a Connectixet (a Virtual PC és a Virtual Server szoftverek gyártóját), egy komplex, az adatközpontban folyó tevékenységek mindegyikére kiterjedő megújítási stratégiát dolgozott ki Dynamic System Initiative (DSI) néven. A kezdeményezésben kiemelkedő szerepet szánt a virtualizációnak, szinte minden változatának. Így került a képbe a Softricity. 2006 júniusában létrejött az egyezség a két cég vezetői között, és a Microsoft felvásárolta a Softricityt.

A SoftGrid zászlóshajóterméke lett a később kialakított, előfizetéses jelleggel igénybe vehető Microsoft Desktop Optimization Pack (MDOP) szoftvercsomagnak. A fejlesztés természetesen nem állt le, és 2008. nyár közepére várható a legfrissebb, 4.5-ös verzió, immár a SoftGrid név nélkül. Az új termék neve Microsoft Application Virtualization 4.5 (MAV 4.5).

A MAV működési elve és komponensei

A MAV működése négy fő részre osztható. Az első fázis a virtuális alkalmazás elkészítése. Ezt a folyamatot „sequencing”-nek hívják, jellegét tekintve egyfajta csomagolás, intelligens „pillanatfelvétel-készítés”. Azért intelligens, mert a sequencer – tehát a műveletet



A Sequencer teszi virtualizálhatóvá az alkalmazásokat

végző segédprogram – a kezdeti és végállapoton túl rögzíti, hogy a virtualizálandó szoftver betöltésekor mely állományokra milyen sorrendben van szükség (innen a sequencing, a „sorrendbe rakás” kifejezés). Ráadásul még olyan műveletek is elvégezhetők, mint a Microsoft Update-ről való alkalmazásfrissítés, aktiválás stb. Látható, mindez sokkal több, mint egy egyszerű különbségképzés.

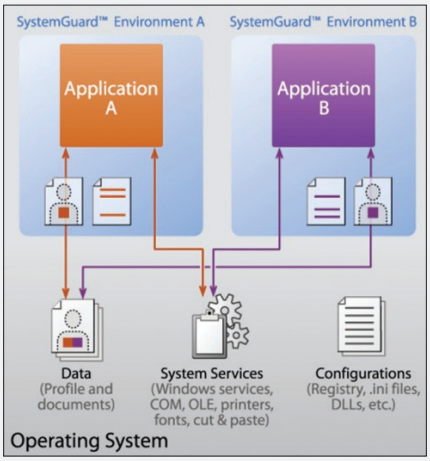
A végeredmény egy SFT kiterjesztésű állomány, amely az alkalmazás összes komponensét tartalmazza, továbbá még néhány, a csomag közzétételéhez szükséges konfigurációs fájl.

A második fázis a publikáció és a szoftver eljuttatása a megfelelő eszközre. A MAV a regisztrált csomagokhoz adott Active Directory-csoportoknak ad hozzáférést. Aki egy ilyen csoport tagja, az elérheti az alkalmazást, mások nem. A publikáció az Application Virtualization Management Serveren végezhető el a MAV MMC konzol segítségével. Sok más mellett itt állíthatók (opcionálisan) a csomagokra vonatkozó licencelési szabályok. E funkció segítségével lejárati időt adhatunk egy csomagnak, meghatározhatjuk az összes lemásolt SFT-fájl számát vagy éppen az egyidejűleg használt szoftverpéldányok mennyiségét.

Ha a csomagot regisztráltuk, és a megfelelő AD csoportnak kiajánlottuk, a célba juttatásról az Application Virtualization Streaming Server gondoskodik. Ez a harmadik fázis. Bármilyen furcsán hangzik elsőre, a szoftver – vagyis most már az SFT-fájl – úgy érkezik, mint egy internetes rádióadás-folyam, sőt még a protokoll sem különbözik (RTSP – Real Time Streaming Protocol, TCP 554). Mivel a sequencing fázis során az SFT-fájlban a szoftverindításnak megfelelő módon helyezkednek el az állományok, az adatfolyam célgépre juttatása során mód van csupán az „éppen elégséges” szoftverkód lejuttatására. A gyakorlatban ez azt jelenti, hogy az SFT-nek mindössze 20-30 százaléka kerül a végfelhasználó gépére, és máris elindul a szoftver. A koreai súgó meg letöltődik, ha majd szükség van rá.

Eltekintve a furcsának tűnő streaming-technológiától, a szoftver lényegében fájlmásolással kerül a desktopokra, ami egyben azt az érdekes helyzetet idézi elő, hogy az alkalmazás „azt hiszi”, egyes egyedül „ő” települt és fut az operációs rendszeren (pedig nem), az operációs rendszer ugyanakkor „azt hiszi”, hogy érintetlen, és semmilyen szoftver nem került rá (pedig de).

A negyedik fázis főszereplője a munkaállomásokra telepített MAV-kliens, amelynek háromféle feladata van. A MAV-szoftverpublikációkat észleli, és az operációs rendszert ennek megfelelően konfigurálja. Ha egy felhasználó jogosult egy adott virtuális alkalmazás futtatására, akkor a MAV-kliens gondoskodik az alkalmazást reprezentáló parancsikonok, környezetérzékeny menük, fájlasszociációk megjelenéséről, mintha az alkalmazás már a gépen lenne. Mikor aztán a felhasználó kettőt kattint az alkalmazás ikonján, a MAV-kliens adatfolyamként letölti a streaming-szerverről a szükséges mennyiségű kódot a desktopra, pontosabban a MAV-cache-be, és a programot elindítja.



A virtuális alkalmazások nem zavarják egymást

Végül a MAV-kliens felelős a virtuális környezet biztosításáért. Szemben a hardvervirtualizációval, itt nincs szó emulációról. A MAV-kliens figyeli a virtuális alkalmazás rendszerhívásait, és meghatározott hívások esetén átirányítja azokat. A virtualizált alkalmazás az operációs rendszer és a sequencing során rögzített rendszerparaméterek egymásra vetített változatát látja, anélkül, hogy ezt az „egymásra vetítést” érzékelné.

A fenti módon a következő komponensekre való hivatkozást lehet átirányítani:

- Fájlok (rendszerfájlok is!)
- Registrykulcsok és -értékek
- Betűállományok
- .ini-állományok
- COM/DCOM-objektumok
- Szolgáltatások
- Névterek
- Szemaforok, mutexek

A felsorolt rendszerelemek elérésekor a MAV-kliens átirányíthatja a hívásokat az SFT-fájlba. Az átirányítás igény szerinti. Ha például az alkalmazás telepít magával egy betűkészletet, akkor futáskor úgy látja, mintha az adott desktopon fent lenne a betűkészlet. Amikor ténylegesen meghívja valamelyiket, akkor a MAV-kliens – az alkalmazás számára nem észlelhető módon – átirányítja az SFT-fájlba, ahonnan a készlet betöltődik. Hasonlóan működik a többi átirányítás is. A művelet olyan gyors, hogy a futás teljesítmény-vesztés még az 1 százalékot sem éri el.

Természetesen nem minden műveletet kell átirányítani. Az SFT-fájlban nem található objektumok (rendszerfájlok- és szolgáltatások, COM-névterek stb.) rajta vannak a gazdagépen, amit az alkalmazás természetes módon, változtatás nélkül képes elérni. Éppígy működnek a lemezműveletek: a virtualizált WINWORD.EXE a valóságos mappákból valóságos dokumentumokat nyit meg, sőt a felhasználó beállításait is a valóságos profiljából tölti be. Fontos hangsúlyozni, hogy a virtualizált alkalmazások helyben futnak, éppúgy, mintha telepített alkalmazások lennének. Szépen látszanak például a feladatkezelőben.

Itt jegyezzük meg, hogy bár működhet/elindulhat egy alkalmazás az SFT-fájl 20-30 százalékának letöltésekor is, ez nem zárja ki azt, hogy az SFT-t teljes egészében letöltsük, sőt erre szükség is van kapcsolat nélküli üzemmód esetén. Vagyis, ami nagyon fontos, a MAV képes futtatni a virtuális alkalmazásokat hálózat nélküli állapotban. (Sőt, még nem 100 százalékos letöltöttség esetén is, de ekkor persze rizikós egy-egy funkció elérése.)

A MAV technológiai korlátai

Nem lenne teljes a kép, ha nem mondanánk el, milyen korlátai vannak a fenti megoldásnak a technológia jelen állása szerint.

Nem minden alkalmazás virtualizálható. Ökölszabályként azt érdemes megjegyezni, hogy a kernel módú komponenst, eszközmeghajtót tartalmazó szoftverek nem csomagolhatók be. Ebbe a kategóriába tartoznak az antivírusszoftverek vagy a CD/DVD-író programok. Ugyancsak nem virtualizálhatók a Com+ komponenseket használó szoftverek, a hardverhez kötődő szoftverek (például a gép MAC-címét ellenőrző megoldások). Teljes bizonyossággal csak az SFT-fájl alapos

tesztelése után mondható ki, hogy az alkalmazás virtualizálható.

Végül érdemes megemlíteni, hogy a 4.5-ös megoldás még csak 32 bites változatban érhető el, tehát a 64 bites terminálszerverek nem lehetnek MAV-kliensek.

Felhasználási területek

A MAV-komponensek rövid áttekintése után nézzük meg, hogy ez a viszonylag egyszerű technológia milyen lehetőségeket teremt az üzemeltetők számára. Az itt felsorolt felhasználási területeket ne tekintse az olvasó teljes listának, hiszen csak a fantázia szab határt az alkalmazhatóságnak.

Gyors alkalmazáskiajánlás (provisionálás). A már meglévő csomagok esetén a kiajánlás egy Active Directory-csoporttagság beállítására egyszerűsödik. Amint a csoporttagság változását érzékeli a MAV-kliens, az alkalmazás ikonja, hivatkozásai, fájlhozzárendelései megjelennek a felhasználó számára.

Gyors, rugalmas „telepítés". A virtualizált alkalmazást nem kell telepíteni. Amikor a felhasználó elindítja, magától letöltődik és elindul.

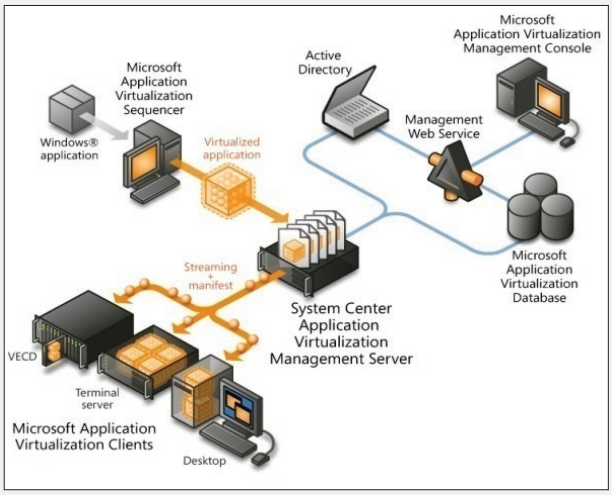
Inkompatibilis alkalmazások párhuzamos futtatása. Az átirányítási technológia lehetővé teszi, hogy minden virtualizált alkalmazás a maga kívánta környezetet lássa, ezáltal párhuzamosan „telepíthetővé" és futtathatóvá válnak egymás létezését kizáró szoftverek is. Két eltérő Java virtuális környezetet igénylő szoftver a maga Java futtatójával összecsomagolva párhuzamosan futhat, de ez a helyzet két (sőt akár három vagy négy!) Office-verzióval is.

Gyors és biztonságos verzióváltás. A fenti jellemző kihasználásával egy alkalmazás újabbal való lecserélése sokkal gyorsabbá, és – ami még ennél is fontosabb – biztonságosabbá válhat. Egy hagyományosan telepített Office 2003 mellé gond nélkül kiajánlhatunk egy virtualizált Office 2007-et. Probléma esetén a gépen maradt korábbi verzió használható, az Office 2007 biztosan nem rontja el az COM/DCOM-névtereket. A megfelelő idő elteltével pedig az Office 2003 eltávolítható.

Teljes verzióváltás a maradék szoftverek megtartásával. Az előző példa fordítva is működik. Telepíthetünk hagyományos módszerrel Office 2007-et, és ha például egy részlegen egy régi Access 97-re írt partizánszoftvert futtatnak, annak virtualizálásával megoldható az új vállalati Office-szabvánnyal nem kompatibilis alkalmazások megőrzése is.

Eltérő beállítások alkalmazása. Az Internet Explorer maga nem virtualizálható, de a beállításai igen. Így lehetséges például házirenddel minden ActiveX-vezérlőt letiltani, de egy olyan virtuális példányt mégis el lehet indítani, amelybe becsomagoltunk egy, általunk fontosnak ítélt vezérlőt.

Terminálszerver- (Citrix-) silók megszüntetése. Az inkompatibilis alkalmazások komoly méretezési problémát jelentenek a terminálszervereknél. A hagyományos megoldás általában az, hogy két-három ún. silót (szervercsoportot) hoznak létre: egy-egy silóban csak olyan terminálkiszolgáló található, amelyik azonos alkalmazásokat futtat. Más silókban más, az előző silóval nem kompatibilis alkalmazások futnak. Ez a szerverek számát jócskán növeli. A MAV lehetővé teszi, hogy a silókat megszüntessük, és csupán egyetlen csoportot hagyjunk meg úgy, hogy az inkompatibilis alkalmazásokat virtualizáljuk.



Teljes kiépítésű rendszer

Többgépes/barangoló felhasználók. Az MAV a felhasználókhoz rendeli a szoftvereket. Mivel az AD-csoporttagság minden gépen azonos, ezért MAV-kliens esetén az alkalmazások automatikusan követik a felhasználót.

Gépcsere. Ha a személyes beállítások mappaátirányítással központilag letölthetők, a végfelhasználói munkaállomások állapot nélkülivé válnak, vagyis a gépek cseréje, tartalékgépek átmeneti kiadása lényegében nem igényli a rendszergazdák közreműködését, miközben minden felhasználó mindig ugyanolyan környezettel találkozik

A többdesktopos üzemmód megszüntetése. Az inkompatibilis alkalmazások problémájának egyszerű megoldása az adott felhasználóknál a két (három?) PC megjelenése, terminálszerver alkalmazása, VDI-megoldás implementálása. Ezek a megoldások teljes mértékben kiküszöbölhetők. Meglehetősen furcsa, de ebből egyenesen következik, hogy az alkalmazásvirtualizációval akár a felhasználói gépek száma (és azok minden költsége) is csökkenhet.

Kevesebb operációsrendszer-lemezkép. A telepített, de egymást kizáró alkalmazások jócskán okoznak többletfeladatot az operációsrendszer-lemezképek létrehozásánál és karbantartásánál. Elvileg minden egyes telepített komponenst minden mással le kell tesztelni kompatibilitás szempontjából. Ez a regressziótesztelés annál hosszabb folyamat, minél több szoftverről van szó, de csak így lehet bizonyosan hibátlan lemezképet előállítani. A MAV lehetővé teszi, hogy ad absurdum egyetlen felhasználói szoftvert se telepítsünk az operációs rendszerre. Akár egy több ezer PC-vel rendelkező szervezet is elboldogulhat egyetlen, jól kialakított lemezképpel. A költséges és hosszadalmas regressziótesztelés teljes egészében elhagyható.

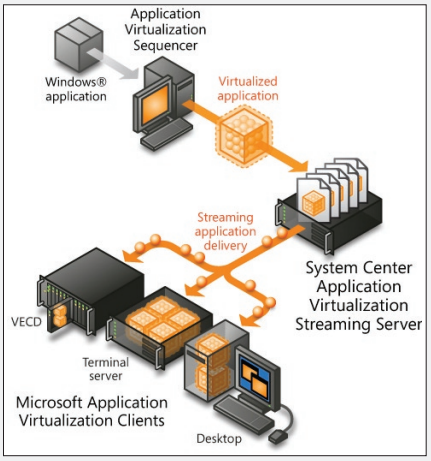
Active Update-technológia. Még nem esett róla szó, de a sequencing támogatja a csomagok frissítését. Egy már elkészített Office 2007-csomagot újra meg lehet nyitni, frissíteni SP1-re, majd lezárni. A menedzsmentkonzolon megadható, hogy az új csomag az előző frissítése. A felhasználó a már elindított alkalmazást gond nélkül használhatja, a következő indításkor pedig – hip-hip, barbatrükk – már az SP1-gyel frissített csomag indul el. Tehát nem 3000 Office-t kell frissíteni, hanem csak egyet! (A frissítési technológia ennél még sokrétűbb, de a cikken belül ezt nem taglaljuk.)

Felmerülő problémák

Ez idáig szép és jó. Öreg rókákat azonban nem vakít el talmi csillogás: a MAV – bármi-

lyen sok problémát old is meg – számos kérdést generál.

Mindenekelőtt mi lesz az olyan gépekkel, amelyek sohasem találkoznak a hálózattal? Eldugott helyen üzemelnek, vagy éppen biztonsági okokból sohasem kerülnek fel oda.



Egyszerű kiépítésű rendszer

Hogyan lehet ezután szoftverleltárt készíteni? Ha egy alkalmazás pusztán egy SFT-állomány, akkor sem a Control Panel Programok részében, sem a fájlrendszer átfésülésével, sem a regisztrációs adatbázis böngészésével nem lehet megállapítani, milyen alkalmazások vannak ténylegesen egy gépen.

Mi lesz a gépeket megcélzó szoftverdisztribúciós mechanizmusokkal? Nem telepíthetünk géphez kötődő szoftvert?

Mi a sorsa a WSUS-nak?

Végül – és talán ez a legnehezebb kérdés – ha nekünk van már egy jól bejáratott szoftverdisztribúciós mechanizmusunk, akkor azt most le kell cserélnünk? El kell dobnunk mindazt, amit eddig felépítettünk? (A szoftverterítő alkalmazások megnevezése angolul „Electronic Software Distribution”, és az ESD rövidítés használatos, mi is ezt követjük.)

Ahhoz, hogy a fenti kérdésekre megnyugtató válaszokat adhassunk, meg kell ismernünk a MAV-építőelemekből létrehozható szoftverterítési modelleket.

MAV-architektúrák

MAV-infrastruktúrát a négy fázis alkotóelemeiből építhetünk. Csak ismétlésképpen: Csomagolás (Sequencing) – Közzététel (Publishing) – Célba juttatás (Streaming) – Futtatás (Launching and running). A fázisokat megfelelő szoftverkomponensek támogatják. A MAV 4.5-től ezek a komponensek külön-külön felhasználhatók. Ezek alapján három alapstruktúra képzelhető el.

Teljes kiépítés. Minden komponenst felhasználunk, ahogy azt a fentiekben leírtuk. Ezt a modellt többnyire az adatközpontok követik.

Egyszerű kiépítés. A csomagolás után kihagyjuk a publikációt, és csak a streaming-kiszolgálót meg a klienst használjuk.

A megoldás előnye, hogy sem AD-re, sem SQL-kiszolgálóra nincs szükség (ideális megoldás távoli telephelyek esetén), viszont le kell mondanunk a szoftverhasználat méréséről, és gondoskodni kell a publikálásról. Utóbbira jó lehet a központban felállított MAV Management Server, vagy megoldható scripttel, de többnyire az ESD-szoftverek végzik majd el.

Szerver nélküli kiépítés. A MAV 4.5-ös csomagolója az eljárás végén felajánlja, hogy az elkészült MAV-csomagot még egy MSI-kerettel is körbeveszi. A funkció opcionális, de a szerver nélküli kiépítésnél nagyon hasznos. Az MSI-állomány CD-re, DVD-re másolható, és bármikor „telepíthető”, sőt remekül felhasználhatják azok az ESD-termékek, amelyek nem ismerik az alkalmazásvirtualizáció fogalmát. Az MSI-keret összesen két feladatot lát el: a MAV-csomagot 100 százalékban betölti a MAV-kliens cache-ébe, továbbá egy regisztrációs kulcs segítségével megjelenteti a szoftvert a Control Panel „Programok” részében, hogy a szoftverleltár helyes adatokat mutasson. Mindez azonban nem telepítés, a szoftver továbbra is virtualizált, és élvezi a MAV-kliens hordozta előnyöket: az egymással inkompatibilis alkalmazások futtatását, az intakt operációs rendszer megőrzését stb.

Az opcionális MSI-keret egyébként nem a 4.5-ös verzió újdonsága: 2007 decemberében a SoftGrid 4.2-höz jelent meg egy ingyenesen letölthető segédprogram, amely még különálló módon, de a fenti funkcionalitást biztosította.

Az MSI-keret, ahogy láttuk, átvágja a gordiuszi csomót. Ha egy IT-szervezet csak a kliensen tapasztalható alkalmazásvirtualizációs előnyökre vágyik, de meg akarja tartani a hagyományos ESD-rendszerét, és nem szeretne dupla disztribúciós mechanizmust, ezt könnyűszerrel megteheti. Az integrációnak azonban van ennél magasabb szintje is: a System Center Configuration Manager 2007 R2 egyik legfontosabb képessége a MAV-val való igen szoros együttműködés.

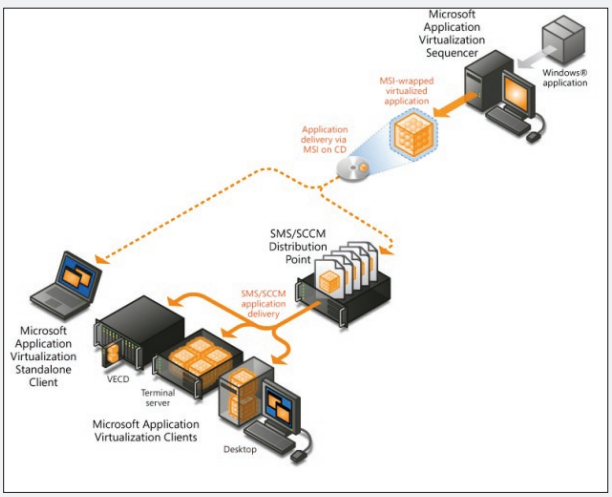
SCCM 2007 R2-integráció

A MAV és az SCCM együttműködése négy területre terjed ki.

1. Csomagolás és alkalmazásdisztribúció
2. Csomagterítés az SCCM-hirdetéses (advertisement) mechanizmusával
3. Alkalmazásfuttatás
4. Leltár- és jelentéskészítés

Az SCCM 2007 R2 szerver- és kliensoldali komponensei is „ismerik” a MAV 4.5-öt. Ha például előzőleg az Application Virtualization Streaming Servert telepítettük, egy disztribúciós ponton tulajdonságként állítható, hogy a kliensekre hagyományos módszerrel vagy streaminggel juttassa el a csomagokat. (Olyasfajta integráció ez, mint a WSUS- és a WDS-integráció: az SCCM az eredeti szoftvert további képességekkel ruházza fel.)

A SCCM-kliens is ismeri a MAV-os „kollégáját”, regisztrálja magát, és intelligensen eldöntik, melyik csomag letöltését melyik kliens hajtja végre. Sőt! Az is előfordulhat, hogy az MSI-vel keretezett virtuális alkalmazásokat az SCCM-kliens tölti le (hagyományos módon), de azután a MAV-cache-be másolja, az indításról pedig a MAV-kliens gondoskodik.



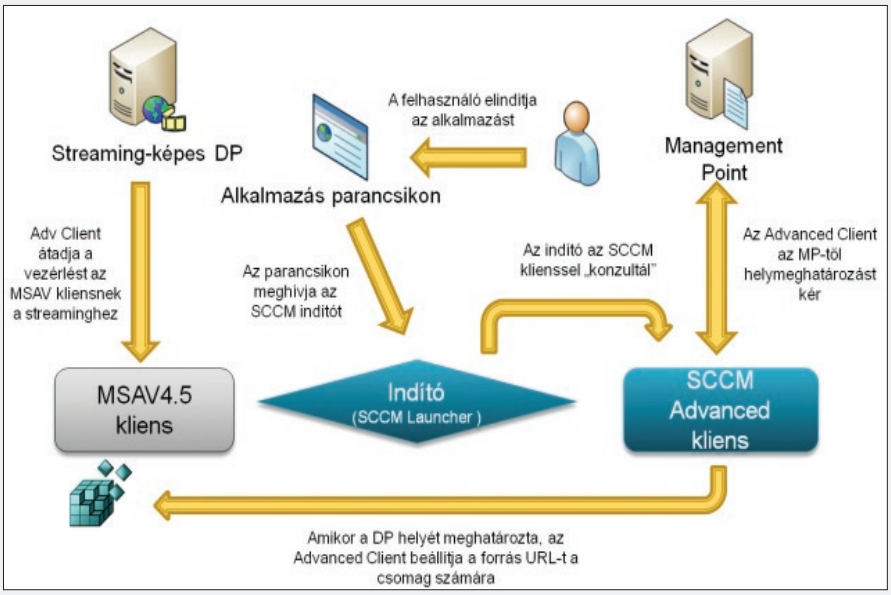
Szerver nélküli kiépítés

Az integráció fő célja az, hogy a MAV-infrastruktúrából az aktuális igényeknek megfelelően mindig annyit használjunk, amennyire szükség van.

A szerverek vonatkozásában a legizgalmasabb együttműködési terület kétségkívül a disztribúciós pontok integrációja. A virtuális alkalmazások meghirdetésekor meghatározhatjuk, hogy a kliens milyen módon indítsa a virtuális alkalmazást. Ha „Stream from DP" (Streaming delivery) opciót állítunk be, a felhasználói asztalra telepített ikonok a disztribúciós pontra mutatnak, és az SFT-állományok onnan is indulnak. (Hacsak le nem töltődtek korábban 100 százalékban.) A „Download and Execute" (Local Delivery) ezzel szemben a parancsikonokat mindig a helyi MAV cache-ben lévő, 100 százalékig letöltött SFT-állományokra irányítja.

Az SCCM funkcionalitását felhasználva a MAV-kliens mindig a legközelebbi disztribúciós pontról indítja a letöltést. Ehhez „csupán" arra van szükség, hogy az alkalmazások indításakor az SCCM-kliens lekérdezze az SCCM menedzsmentpont-szervert, amelyik a telephelynek megfelelő legközelebbi disztribúciós pont. A kapott eredmény alapján az SCCM-ügynök módosítja a MAV-kliens paramétereit, amely azután a helyes streaming-képes disztribúciós ponthoz fordul.

De nemcsak a szoftverdisztribúció, hanem a szoftverfrissítés is integrált. Miután a MAV-sequencerrel elkészítettük a frissített SFT-t, az SCCM bináris deltareplikációval csak a különbséget replikálja át előbb a telephelyekre, majd a disztribúciós pontokra. A frissítésről a munkaállomások akkor értesülnek, amikor a rendszergazdák újrafuttatják az SCCM-hirdetést. Ha a szoftvert a disztribúciós pontról indítjuk (Streaming Delivery), az új verzió azonnal elérhető. A helyi futtatás (Local Delivery) opciónál a különbség BITS-szel kerül a MAV cache-be, és csak akkor indul el a frissített verzió, amikor a teljes csomagfrissítés befejeződött.



Integráció a Configuration Managerrel

Az SCCM 2007 R2-nek már nem kell az MSI keretekre hagyatkoznia, amikor a virtuális alkalmazásokat szeretné számba venni. A MAV 4.5 egy új kliens WMI-providerrel rendelkezik, így már le lehet kérdezni a cache tartalmát, a letöltött programokat, azok verzióját, használatukat stb.

Licencelés

Számtalan alkalmazásvirtualizáció témájú megbeszélés után biztosan állíthatom, a legnagyobb kavarodás a fejekben a licencelés körül alakul ki. A villámgyors szoftverdisztribúció és gyors alkalmazáseltávolítás miatt úgy tűnhet, mintha a MAV alapvetően befolyásolná a megvásárolandó szoftverek számát. Minderre még ráerősít a termékben található licencelési funkció, a csomagok használatának időbeli korlátozása. Öntsünk tiszta vizet a pohárba ezzel kapcsolatban.

Az első és legfontosabb szabály, hogy a MAV egy technológia, tehát semmilyen módon nem változtatja meg a becsomagolt szoftverre vonatkozó licencelési szabályokat. Konkrétan: ha a licenc egy adott géphez köti a szoftvert (tipikusan OEM-konstrukciók), akkor a felhasználó alapú disztribúció megsérti a licencjogokat. Ha viszont a licencszerződés konkurens használatra szól, vagy adott felhasználóhoz kötött, akkor a MAV kifejezetten segíti a konstrukció betartását. Jogi szempontból ugyancsak problémamentes azoknak a szoftvereknek a használata, amelyek egy teljes telephelyre vagy a teljes vállalatra vonatkoznak. (A Microsoft Enterprise Agreement tipikusan ilyen.) A MAV az ilyen konstrukciókat korlátlan (Unlimited) típusnak ismeri. A szoftverhasználatról gyűjtött statisztika azonban még ekkor is hozzásegítheti az IT-szervezetet, hogy a licencköltségeit csökkentse.

Külön megér néhány mondatot a MAV licencelése is. A termék önmagában nem érhető el, hanem ahogy a cikk elején már említettük, a Microsoft Desktop Optimization Pack (MDOP) része, egyik komponense. Maga az MDOP egyéves előfizetéses konstrukcióban vásárolható, de csak az olyan szervezetek számára, amelyek érvényes desktop-frissítésvédelemmel (Desktop Software Assurance, vagy Desktop SA) rendelkeznek. A MAV-komponensek külön nem igényelnek licencet. Amennyiben a munkaállomások lefedettek MDOP-licenccel, szerverkomponenst akárhányat telepíthetünk. Az MDOP-licencigény az SCCM 2007 R2 integráció esetén is érvényes.

Zárszó

Az alkalmazásvirtualizáció legalább akkora potenciállal rendelkezik, mint a nála ismertebb hardvervirtualizáció, és éppúgy gyökeresen átalakítja az üzemeltetők életét. A változás- és konfigurációkezelési (change and configuration management) feladatokat teljes mértékben újra kell gondolni. Sajnos? Szerencsére? A választ nem nekem kell megadni. Egyben viszont bizonyos vagyok: a ma végfelhasználója borzasztóan kiábrándult az informatikából. Lassúnak, rugalmatlannak, nyögvenyelősnek tartja. Az alkalmazásvirtualizáció visszaadhatja a hitet a rendszerüzemeltetőknek és felhasználóknak egyaránt: lehet IT-t sokkal jobban is csinálni.

Lepenye Tamás
(tamasl@microsoft.com) MCSE, Microsoft Magyarország

Windows Deployment Services

A Windows Vistával érkező rugalmas telepítési lehetőségek kis fejtöréssel lényegében bármelyik Microsoft operációs rendszer tömeges telepítését lehetővé teszik.

Amikor 2001-ben először hallottam hálózaton keresztüli operációsrendszer-telepítésről, nem tulajdonítottam neki túl nagy jelentőséget. Az akkoriban még csak stabil 10 megabites hálózati kapcsolaton keresztülpumpálni 2-3 gigabájt mennyiségű adatot? Badarság, kinek van erre szüksége? Az operációs rendszernek telepítő CD-n a helye. Bizony, néhány évvel később meg kellett állapítanom, hogy immáron pont telepítőlemezt használok a legkevésbé a feladat elvégzéséhez. Ezzel a cikkel az a szándékom, hogy a rengeteg új fogalom és eszköz közül kiválaszthassuk azt, amellyel leghatékonyabban elérhetjük célunkat! A májusi Technet-konferencián hallottakat most kiegészíteném egy általános előkészítői leckével, illetve jó néhány hasznos praktikával, ötlettel. Sok kisebb-nagyobb, egyszerűbb-bonyolultabb eszköz áll rendelkezésre, a választást a cél határozza meg, tehát nem kell feltétlenül mindet használnunk.

Az etalonpéldány

Szoktam még donor, vakpéldány, vagy mesterpéldány néven is hivatkozni arra az operációs rendszerre, amely lényegében egy univerzális, sok szempontnak megfelelő alaprendszer, a további telepítések alapanyaga. Létrehozásakor az a cél, hogy minél kevesebb manuális beavatkozással tudjuk „személyre”, vagyis számítógépre szabni beállításait. Mesterpéldány készítésekor a főbb szempontok jellemzően a különböző gép-(hardver-)típusokra történő felkészítés, a fontosabb alkalmazások előtelepítése, licenctípusok meghatározása stb. Ez a fázis a legidőigényesebb feladat, de ne sajnáljuk az időt tesztelésére, rengeteg bosszúságot és főleg időt takaríthatunk meg később egy jó etalonpéldánnyal. A sysprep eszközzel „konzervált” operációs rendszer terjesztése után (vagyis az egyedi információktól megfosztott operációs rendszert valamely módon kimásoljuk a telepítendő gépre) a közel 10 perces, ún. mini-setup segítségével életre kel. Egyedi paraméterek megadása után (például gépnév, sorozatszám, tartományi tagság) PNP-ellenőrzést hajt végre (tehát nem nekünk kell bíbelődnünk gépenként esetlegesen egyedi meghajtóprogramok telepítésével, természetesen csak akkor, ha megfelelően felkészítettük erre az etalonpéldányt). Lényegében ezután azonnal működőképes lesz a rendszerünk, hisz a tesztelés részét már letudtuk, amikor elkészítettük a vakpéldányt. Itt kell megjegyeznem, hogy ha nincs szükségünk egyedi paraméterekre, megtehetjük, hogy élő operációs rendszerről készítünk másolatot, ekkor beszélünk a „klónozás” folyamatáról, hiszen az operációs rendszer „SID”-je, vagyis egyedi azonosítója ekkor minden újonnan telepített gépen ugyanaz lesz, és ez tartományi környezetben problémákat okozhat.

Válaszállomány(ok)

Míg az XP ötféle válaszállományt használt, addig a Vista mindössze egyet (unattend.xml). Ezeket a fájlokat arra találták ki, hogy meghatározhassuk egyedi beállításainkat, illetve azért, hogy mennyire szeretnénk a telepítési folyamatot automatizálni (értsd: mikor, mennyit kell manuálisan konfigurálni). A Vista sokkal több lehetőséget ad erre (több mint 300-féle érdemi beállítási lehetőség), eligazodni a rengeteg lehetőség között a Windows System Image Manager (WSIM) segít. A kimenet egy szöveges állomány, amelyet akár egy notepaddel is módosíthatunk, tehát látható: ha további eszközöket is bevetünk a válaszállomány manipulálására (például programozott stringcserét), akkor a kézi beavatkozást gyakorlatilag a számítógép ki- és bekapcsolására korlátozhatjuk.

Hol találjuk a válaszfájl-szerkesztő programot? A Vista esetén a WSIM az Automated Installation Kit (AIK) része, amely egy ingyenesen letölthető szoftver- és dokumentációgyűjtemény.

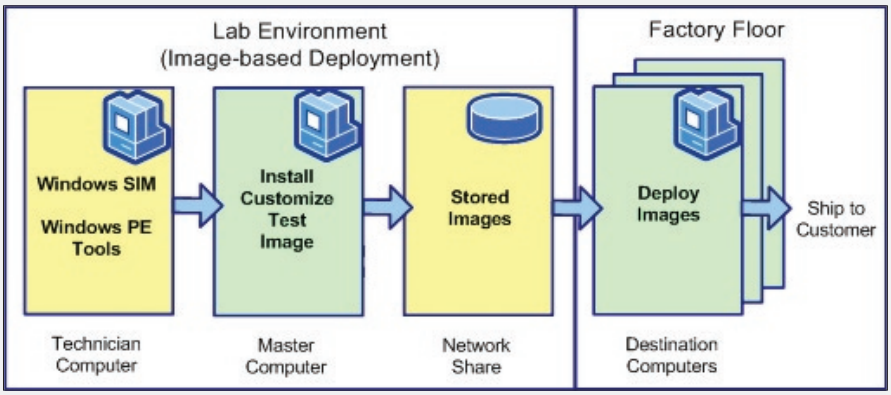
Íme egy részlet egy Vistához készült unattend.xml-ből, azaz konkrétan arról, hogyan lehet automatikusan beléptetni tartományba a SYSPREP-pel konzervált operációs rendszert:

```
<Identification>
   <Credentials>
      <Domain>wds.local</Domain>
      <Password>Win2008</Password>
      <Username>administrator</Username>
   </Credentials>
   <JoinDomain>wds.local</JoinDomain>
</Identification>
```

Windows Image format (.WIM)

A Vista új telepítő fájlformátuma, a tömörített WIM-fájl alkalmas saját etalonpéldányaink tárolására is. Valójában teljesen mindegy, mit tárolunk ebben a fájl alapú lemezképtároló állományban, tehát a sysprep segítségével konzervált vagy csak úgy röptében másolt, „klónozható” más verziójú

operációs rendszer éppen úgy megfér benne egymás mellett, mint ahogy pusztán halott adatot tároló (például profilkönyvtárak, virtuális gépek) partíciók is. Az „egymás mellett” úgy értendő, hogy a .WIM formátum indexek segítségével független partíciókat tárol egyetlen .WIM fájlban. Ez lehetővé teszi, hogy egy állomány csak először, első alkalommal tárolódjon fizikailag (single instance store), ezután, ha egy másik partíció rögzítésekor ez az állomány ismét tárolódik, már csak egy hivatkozás készül rá. Tehát egy-egy



Ahogy a nagykönyvben meg van írva – a valóságban sokkal több lehetőségünk van

újabb partíció hozzáadásakor fizikailag csak a különbség kerül bele a .WIM fájlba, emiatt adattárolási szempontból a WIM igen hatékony formátum.

A WIM-ben tárolt lemezképeinket az IMAGEX.EXE program segítségével tudjuk kezelni (ami ugyancsak az AIK része), beleértve az újabb lemezképek hozzáadását vagy a meglévők módosítását is. Ugyanezzel a programmal lehet visszaállítani a tárolt lemezképeket, de erre lesznek még további lehetőségeink.

.WIM-tartalom terjesztése 1. – fapados, ám hatékony módszerek

Ha már rendelkezünk terjeszthető operációs rendszerrel, valami módon a .WIM állományunk tartalmát fel kell másolnunk a célgépre. Nos, erre elég sokféle lehetőségünk van, és a cél fogja meghatározni a módszert. Néha megfelel, ha egy USB-csatlakozóval a célgép merevlemezét csatlakoztatjuk ahhoz a PC-hez, ahol a célpartíciót elkészítjük, aktiváljuk, megformázzuk, majd az IMAGEX /APPLY paranccsal az etalonpéldányt kimásoljuk a végleges helyére.

Jellemzőbb formája az automatizált telepítésnek, ha az AIK-dokumentáció segítségével létrehozunk egy Windows Preinstallation Environment (WinPE) lemezt, majd ezzel bootoljuk be a célgépet. A WinPE igen sokoldalú megoldás, és van például 64 bites változata is. Bár a kezelőfelülete többségében mindössze egy parancssor, mégis képes USB 2.0-eszközöket kezelni, hálózati kártyán keresztül kommunikálni, dinamikus lemezt kezelni. Sőt, mivel eszközvezérlő bővítési lehetőségünk is van a PEIMG (AIK, micsoda meglepetés!) program segítségével, bővíthetjük a kezelhető hardverek körét is. A WinPE-t arra találták ki, hogy fájlokat másolgassunk teljesen üres vasakra. (Érdemes megjegyezni: ezt a 200 megabájtnyi operációs rendszert rávehetjük, hogy közvetlenül merevlemezről is elinduljon, de ez inkább perverzió, mint alapfelhasználás.) Ezután az IMAGEX már ugyanolyan körülmények között használható, mint az előző példában. A .WIM állomány tehát ugyanúgy tárolható egy USB-s merevlemezen, mint egy hálózati megosztáson, így a szükséges etalonpéldányt, illetve az egyediséget elérő válaszállományt felmásolhatjuk a célgép merevlemezének gyökerébe (vagy éppen RAID0 tömbjére).

.WIM-tartalom terjesztése 2. – szolgáltatással támogatott

Az előző bekezdés végén utaltam rá, hogy a hálózati telepítés fogalma nem feltételezi speciális szolgáltatások üzemeltetését, hiszen egy hálózati megosztáson tárolt .WIM állománnyal és egy rugalmasan kezelhető USB-s merevlemezzel ki is válthatjuk a körülményes boot-CD vagy -DVD lemezeket. Ha tovább szeretnénk egyszerűsíteni telepítőműhelyünket, a következő lépés, hogy magát a telepítőklienst (WinPE) is hálózaton keresztül juttatjuk el a telepítendő számítógépre. Erre a PXE (Preinstallation Executable Environment) szolgáltatás képes, mely része a Windows Deployment Services-nek (WDS). Működésének feltétele IP-alhálózatonként egy Windows Server 2003 (bármelyik verzió), PXE-képes hálózati kártyák a telepítendő gépeken, illetve a WDS boot store-ban tárolt WinPE-kliens megfelelő felkészítése (például hálózatikártya-driver, megfelelő alkalmazások).

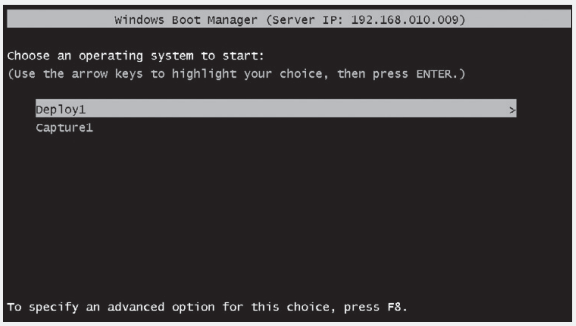
Miután a kiválasztott kliens RAMdiskként letöltődött a célgépre, funkciójától függően kimásolhatunk .WIM lemezképeket a WDS image store-jából, készíthetünk lemezképet a célgépről. Sőt, ha további alkalmazásokat is elhelyezünk a WinPE RAMdisken, akár vírust is ellenőrizhetünk, vagy éppen terminálkliens segítségével vékonyklienskent használhatjuk ezt a számítógépet.

Hálózat: az erő és a multicast velünk van

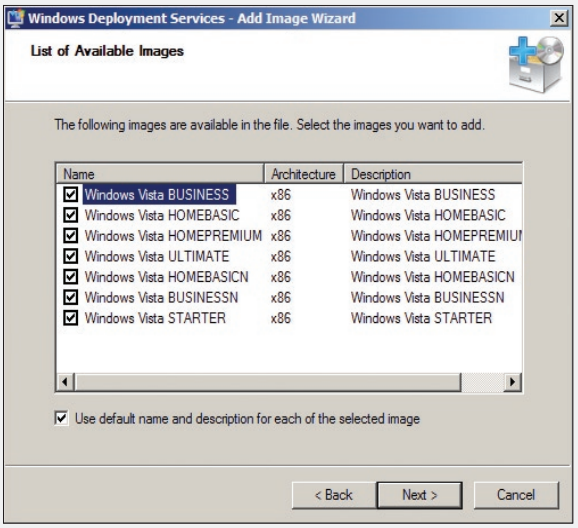
Az előzőekben felsorolt eszközök közül a WDS ad lehetőséget arra, hogy célszámítógépünket a legkényelmesebben, illetve a legkevesebb hibalehetőséggel telepítsük. Az utóbbit arra értem, hogy esetleg a témában annyira nem járatos személyre is rábízhatjuk a telepítést, mivel a beállításokat központilag adjuk meg, az opciók közül csak választani kell.

A WDS-szolgáltatásnak otthont adó számítógépnek tartományi tagnak kell lennie. Az igazi rugalmasságot adó PXE bootlehetőség használatához a WDS-szerver IP-hálózatán szükséges dinamikus IP-címosztás (DHCP). Ezután már csak azt vagy azokat a WinPE-klienseket kell létrehoznunk, amelyek közül a célgépen a PXE-kliens letöltése előtt választhat az operátor. Több boot image-t is felvehetünk, de a célgép választóképernyőjén maximum 15 különböző boot image-t jeleníthetünk meg egyszerre. Alapvetően kétféle boot image-t tudunk itt használni, mégpedig Deploy (telepítő) és Capture (lemezkép-készítő) RAMdisket. A discovery image lényege, hogy nem PXE módon indítjuk el a célgépet, hanem valamilyen lemezes médiáról, de a feladata ugyanaz lesz, mint az előbb említett két alaptevékenység.

Előzetesen el kell döntenünk, milyen architektúrán (x86, x64) szeretnénk futtatni



Maximum 15 „indítólemez"-ből válogathatunk hálózaton keresztül



WDS store importja esetén tovább válogathatunk a .WIM állomány tartalmából

RAMdiskünket, majd a megfelelő BOOT.WIM állományt kell importálnunk, amelyet érdemes egy Windows Server 2008 telepítőlemezről beszerezni (Sources mappa). Ezen a WinPE-példányon fog futni az a kliensprogram, amely hitelesítés után képes lesz a WDS-szerveren tárolt lemezképek közül a célgépre kimásolni a szükséges etalonpéldányt. A capture image készítése boot image-ből lehetséges, a különbség annyi, hogy letöltődés után az adott számítógépen sysprepelt (!) operációs rendszert feltölthetjük a WDS tárolóterületére. Megjegyzem, a művelet ugyanaz, mint ha kézzel (Imagex) elkészítenénk a .WIM állományt, majd importálnánk közvetlenül a telepíthető lemezképek közé.

.WIM-importálás

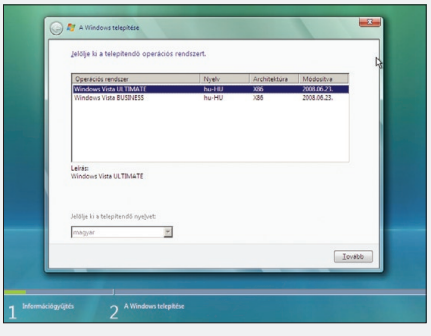
Miután beállítottuk, hogy a PXE segítségével telepítendő gépeink hozzáférhetnek a WDS szolgáltatásaihoz, és képesek a telepítőkliens letöltésére, jöhet a jól letesztelt etalon-operációsrendszereink importálása a WDS tárolóterületére (WDS image store). A WDS szabványos .WIM állományt vár bemenetként, amelyben tárolt lemezképekből még utólag válogathatunk is jelölőnégyzetek segítségével. ImageGroup-onként fog keletkezni egy tárolóegység, ahol hasonlóan, a SIS (Single Instance Store) lehetőségeit kihasználva, helytakarékosan fognak tárolódni az importált állományok. Figyelem: az alapértelmezett C:\RemoteInstall könyvtár tartalmát közvetlenül csak a WDS szolgáltatáson keresztül kezeljük! Ha módosítani szeretnénk valamelyik Image-ünket, az Image-en kattintva exportáljuk .WIM állományba, ekkor ImageX-szel már kezelhető, majd módosítás után a Replace-szel kicserélhetjük az eredeti lemezképet. Ekkor ismét csak a módosítások tárolódnak.

Mi újság az egyediesítéssel?

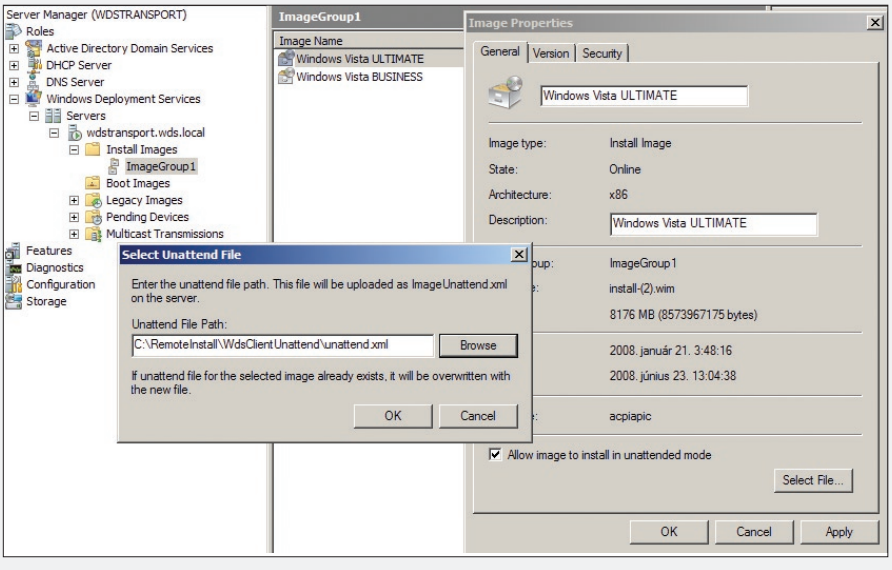
Aki telepített már Vistát, annak a PXE-boot után már nem lesz ismeretlen a felület, amely első lépésében választanunk kell, melyik tárolt operációs rendszert szeretnénk a célgépre helyezni. Ha az eredeti telepítőlemezről indítanánk ebben a fázisban a telepítést, akkor is van lehetőségünk a válaszállományt mondjuk egy pendrive segítségével megadni az operációs rendszernek (vigyázat: XP-nél ez kissé körülményesebb). A WDS-nél viszont köthetjük az adott lemezképhez a válaszállományt, így kényelmesen, központilag választhatunk a rendelkezésre álló válaszfájlok közül (tehát nem kell a kliensgépnél sorban állni a cserélhető médiával, bár a lehetőség adott erre is).

Multicast

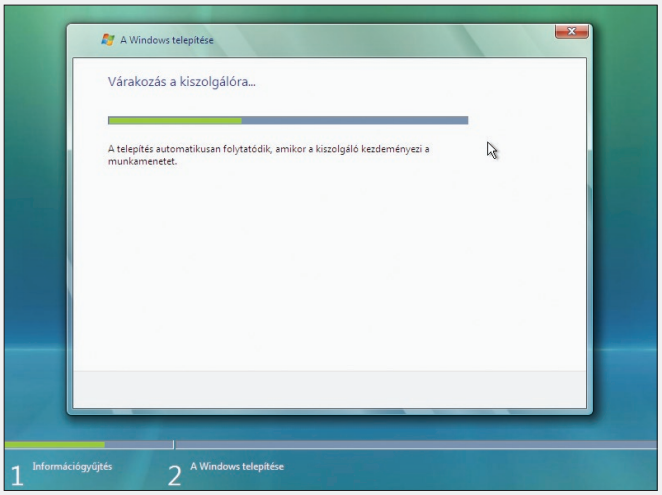
Ha WDS-szerverünk Windows Server 2008-on fut, és a terjesztendő lemezképünk is ilyen verziójú, lehetőségünk van a telepítőcsomagokat multicast hálózati forgalom használatával célba juttatni. Finoman szólva is nagyobb hatásfokkal kerülnek ki ezek a 4-5 gigabájt méretű csomagok a célgépekhez. Aki esetleg nem tudná, mi is a multicast, annak egy egyszerű példával hadd tegyem szemléletessé. Mekkora sávszélességet visz el egy darab operációs rendszer telepítése hálózaton keresz-



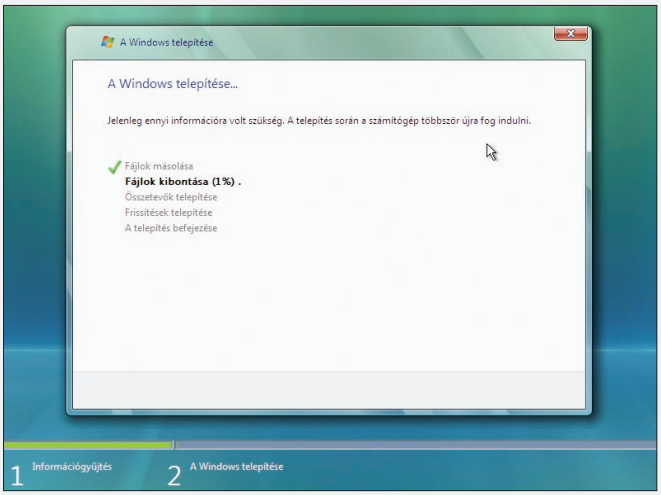
A telepítés ismerős…



Etalonpéldányonként központilag választhatjuk az egyedivé tételt



Ha van multicast munkamenet definiálva a választott lemezképhez, ennél a képernyőnél várják be egymást a számítógépek



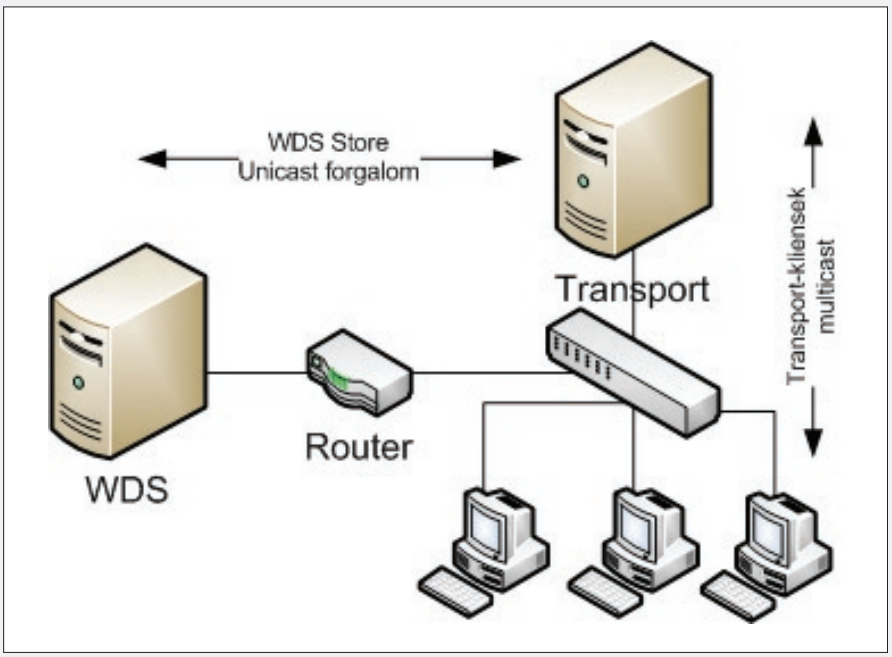
Maga a telepítés viszont teljesen ugyanúgy folyik a továbbiakban, mintha a klasszikus módon, telepítőmédiáról tennénk

tül? Legyen X. Mekkora sávszélességet visz el húsz darab oprendszer egyidejű telepítése multicast segítségével? 20X? Nem! Csupán X! Teljesen mindegy, hány gépet telepítünk egyszerre, a telepítőcsomag egy példányban megy át a hálózaton. Így szoktunk mi péntekenként komplett tantermeket újratelepíteni (egyszerre akár négyet-ötöt) friss operációs rendszerrel akár egy órán belül...

A multicast beállítása igen egyszerű: a telepítendő lemezkép kiválasztása után időponthoz vagy kliensszámhoz ütemezve kell munkamenetet létrehozni, majd amikor minden feltétel teljesül, a forgalom nem unicast, hanem multicast formában indul el. Mivel a válaszállomány lemezképhez köthető, az ilyen formájú telepítés esetén az egyediesítést kézzel vagy egyedi állománnyal (gépenként külön pendrive, ahol az egyedi beállításokat tároló unattend.xml foglal helyet) kell megoldani.

Transport server

Végül – csak hogy teljes legyen a paletta – felhívnám a figyelmet a Transport server lehetőségeire. Ennek a WDS-től függetlenül telepíthető szolgáltatásnak a feladata az állományok másolása multicast segítségével. Ebben az esetben nem az operációs rendszer telepítéséről van szó, hanem élő operációs rendszerek csoportjára utólag egy menetben ki lehet másolni így bármit. Hogy milyen állományokat érdemes kimásolni működő operációs rendszerbe? Természetesen nagyméretű állományokat, például virtuális gépeket vagy komplett műsoros DVD-állományokat, esetleg operációs rendszereket. Belátható, hogy viszonylag kicsi az alkalmazási terület, illetve nem is teljesen a tömeges telepítés témaköre, ezért a részletesebb beállítási lehetőségeket más alkalommal tárgyaljuk – ergo, most csak röviden emlékezünk meg erről. Ez a meglehetősen robusztus szolgáltatás a WDS erőforrásait képes használni (Image store), és a saját IP-alhálózatán terjeszteni azt. Konfigurálni a WDSUTIL-lal lehet (amellyel egyébként a WDS is konfigurálható). A célgépeken bármilyen operációs rendszer futhat (Windows XP, WinPE is), mivel parancssorból kell kezdeményezni minden adatátvitelt (WDSMCAST, az AIK része...).



A Transport Server működése

Remélem, a cikk végére érve jó néhány misztikus felhőt sikerült szétoszlatnom, amely a hálózatos telepítést övezi.

A rengeteg rövidítés (WIM, WDS, AIK, PXE stb.) elsőre valószínűleg zavaró lehet, viszont ha az ember pár órát eltölt a WDS-sel, hamarosan fejből fogja fújni ezeket. A telepítőlemezek számának csökkenésével pedig hozzájárulunk a Föld faállományának védelméhez.

Réczi Gábor
MCSE, MCT, SBS MVP
(reczi. gabor@netacademia.net)
NetAcademia

Active Directory: információk lekérdezése

Ebben a cikkben a PowerShell erejét szeretném bemutatni az Active Directoryval kapcsolatos üzemeltetési feladatok során.

Azoknak, akik már készítettek VBScripttel ADSI-szkripteket, lesz sok ismerős elem, számukra a PowerShell-megvalósítás egyszerűsége lesz az érdekes. Akik korábban nem éltek a VBscript lehetőségével, mert a bonyolult szintaxis elvette a kedvüket, azok most a PowerShell egyszerűsége által kaphatnak kedvet a parancssoros környezet használatához.

A PowerShell segítségével az Active Directoryval kapcsolatos felügyeleti tevékenységek is hatékonyan automatizálhatók. Miután a PowerShell alatt a .NET keretrendszer található, így fontos ismerni az ezzel kapcsolatos fontosabb osztályokat.

Mivel az Active Directory-hibák 110 százaléka valamilyen DNS-hibára vezethető vissza, így elsőként ellenőrizzük a névfeloldást. Az ezzel kapcsolatos .NET-osztály a `System.Net.Dns`, amelynek `GetHostEntry` statikus metódusával tudjuk ellenőrizni például a tartományvezérlőnk nevének feloldását:

```
PS C:\Users\Administrator> [System.Net.Dns]::GetHostEntry("adds.iqjb
.w08")

HostName                Aliases                 AddressList
--------                -------                 -----------
adds.iqjb.w08           {}                      {192.168.1.2}
```

Ha ez helyes eredményt ad, akkor folytathatjuk az AD felderítését, ellenőrzését az erdő legfontosabb objektumaival. Erre a célra a `System.DirectoryServices.ActiveDirectory.Forest` osztály alkalmas, annak is a `GetCurrentForest` statikus metódusa:

```
PS C:\Users\Administrator> [System.DirectoryServices.ActiveDirectory.
Forest]::getcurrentforest()

Name                 : iqjb.w08
Sites                : {Default-First-Site-Name}
Domains              : {iqjb.w08}
GlobalCatalogs       : {adds.iqjb.w08}
ApplicationPartitions : {DC=ForestDnsZones,DC=iqjb,DC=w08, DC=DomainDnsZone
                       s,DC=iqjb,DC=w08}
ForestMode           : Windows2008Forest
RootDomain           : iqjb.w08
Schema               : CN=Schema,CN=Configuration,DC=iqjb,DC=w08
SchemaRoleOwner      : adds.iqjb.w08
NamingRoleOwner      : adds.iqjb.w08
```

Láthatjuk, hogy ez a metódus a legfontosabb adatokat megadja az erdőről: a *root* tartomány és a többi tartomány nevét, a globális katalógusok listáját, az erdő működési szintjét és az erdő szintű egyedi szerepeket hordozó tartományvezérlők neveit.

Hasonló módon megvizsgálhatjuk az aktuális tartomány adatait is a `System.DirectoryServices.ActiveDirectory.Domain` osztály `getcurrentdomain` statikus metódusának segítségével:

```
PS C:\Users\Administrator> [System.DirectoryServices.ActiveDirectory
.Domain]::getcurrentdomain()

Forest                  : iqjb.w08
DomainControllers       : {adds.iqjb.w08}
Children                : {}
DomainMode              : Windows2008Domain
Parent                  :
PdcRoleOwner            : adds.iqjb.w08
RidRoleOwner            : adds.iqjb.w08
InfrastructureRoleOwner : adds.iqjb.w08
Name                    : iqjb.w08
```

Itt a legfontosabb pluszinformációnak a tartományi szintű egyedi szerepeket hordozó tartományvezérlők nevei számítanak.

Nem mellékes, hogy milyen AD site- (telephely-) beállításokkal dolgozunk, hiszen ez befolyásolja az ügyfélgépek tartományvezérlő választását és a címtár-replikációt. A telephely-információk kiolvasására a `System.DirectoryServices.ActiveDirectory.ActiveDirectorySite` osztály `GetComputerSite` statikus metódusa használható:

```
PS C:\> [System.DirectoryServices.ActiveDirectory.ActiveDirectorySite]::
GetComputerSite()


Name                          : Default-First-Site-Name
Domains                       : {iqjb.w08}
Subnets                       : {192.168.112.0/24}
Servers                       : {w2k8.iqjb.w08}
AdjacentSites                 : {}
SiteLinks                     : {DEFAULTIPSITELINK}
InterSiteTopologyGenerator    : w2k8.iqjb.w08
Options                       : None
Location                      : Budapest
BridgeheadServers             : {}
PreferredSmtpBridgeheadServers : {}
PreferredRpcBridgeheadServers  : {}
IntraSiteReplicationSchedule   : System.DirectoryServices.ActiveDirectory.
                                 ActiveDirectorySchedule
```

Ezzel a néhány kifejezéssel tehát elég jól át lehet tekinteni, hogy milyen az AD-infrastruktúránk, fájlba történő átirányítással akár az AD-infrastruktúránk dokumentálásához is segítséget kapunk.

Csatlakozás az Active Directoryhoz

Az előzőekben a .NET keretrendszer osztályainak statikus metódusaival dolgoztam, amelyek segítségével általános információkat lehetett kinyerni az Active Directory környezetről. Ha konkrét, adott tartományra vagy címtárelemre vonatkozó információkhoz akarunk hozzájutni, akkor csatlakozni kell az adott címtárobjektumhoz. A címtár kezelésében fontos szerepe van a gyorsítótárnak, egy ilyen csatlakoztatás előkészíti az adott címtárobjektum memóriabeli reprezentációját. Ha ezek után változtatunk a beolvasott objektum valamely tulajdonságán, akkor ez csak a memóriában hajtódik végre, külön metódussal kell ezt a változást a címtárba visszaírni, amint ezt látni fogjuk.

Elsőként azonban nézzük meg a legegyszerűbb csatlakoztatást:

```
PS C:\> $domain = [ADSI] ""
```

Az `[ADSI]` típusjelölővel hivatkozunk a címtáras elérésre, és ha egy üres sztringet adunk meg a „konstruktor” paramétereként, akkor az adott tartomány tartományobjektumához csatlakozunk. Ez a típusjelölő a `System.DirectoryServices.DirectoryEntry` .NET osztály rövidített neve. Ezt akár ki is írhatjuk, és ekkor további paramétereket is megadhatunk, ha szükséges:

```
PS C:\> $domain = new-object DirectoryServices.DirectoryEntry(„","iqjb\
Administrator", „Password1")
```

A fenti példában megadtam azt, hogy kinek a nevében csatlakozom, és mi ennek a fióknak a jelszava.

Olvassuk ki, hogy mi került a `$domain` változónkba:

```
PS C:\> $domain

distinguishedName
-----------------
{DC=iqjb,DC=w08}
```

Ez még nem túl sok, ennek a részletes tulajdonságait később mutatom be. Most nézzük, hogyan tudunk egy nevesített objektumhoz, mondjuk, egy felhasználói fiókhoz csatlakozni:

```
PS C:\> $user = [ADSI] „LDAP://cn=János Vegetári,ou=Demó,dc=iqjb,dc=w08"
PS C:\> $user

distinguishedName
-----------------
{CN=János Vegetári,OU=Demó,DC=iqjb,DC=w08}
```

Fontos!

Az `[ADSI]` utáni sztringben csupa nagybetűs az LDAP, és normál perjelek vannak utána. Ha nem csupa nagybetűs az LDAP, vagy fordított perjelet használunk, akkor nem rögtön kapunk hibajelzést, hanem csak akkor, amikor először használjuk az objektumot.

AD-objektumok létrehozása

AD-objektumokat létrehozni a korábban már VBScriptből megszokott ADSI-szintaxishoz nagyon hasonló módon lehet. Először egy AD-konténerre kell csatlakozni, ahova létre szeretnénk hozni az új objektumot. Ez a csatlakozás a már látott módon megy, ezzel, ugye, „átemeljük” a PowerShellbe az adott konténert mint objektumot. Az így „átemelt” AD-objektum `Create` metódusával lehet létrehozni az új AD-elemet. A `Create` paramétereként meg kell adni a létrehozandó objektum típusát és a *„relative distinguished name”-et*, azaz az adott konténeren belüli megkülönböztető nevét.

Az alábbi példában magán a tartományvezérlőn (`localhost`), közvetlenül a tartományobjektum alá hozok létre egy szervezeti egységet (`organizational unit`):

```
PS C:\> $konténer = [adsi] „LDAP://localhost:389/dc=iqjb,dc=w08"
PS C:\> $adObj = $konténer.Create(„OrganizationalUnit", „OU=Emberek")
PS C:\> $adObj.Put(„Description", „Normál felhasználók")
PS C:\> $adObj.SetInfo()
```

Ebben, ugye, az az újdonság, hogy az LDAP kifejezésbe beillesztettem a tartományvezérlő nevét és a portszámot is, ahol a címtárszolgáltatás elérhető. A szemléltetés kedvéért még a „`Description`” attribútumát is kitöltöttem. Vigyázat, amit idáig tettem, azt mind a memóriában végeztem el, ahhoz, hogy mindez ténylegesen bekerüljön a címtár-adatbázisba, meg kell hívni a `SetInfo` metódust!

AD-objektumok tulajdonságainak kiolvasása, módosítása

Ha PowerShell, akkor objektumok. Az előzőekhez hasonlóan csatlakozzunk a most létrehozott szervezetiegység-objektumhoz, és nézzük meg a tagjellemzőit a `get-member` cmdlet segítségével:

```
PS C:\> $adou = [ADSI] „LDAP://OU=Emberek,DC=iqjb,DC=w08"
PS C:\> $adou | get-member


   TypeName: System.DirectoryServices.DirectoryEntry

Name                  MemberType Definition
----                  ---------- ----------
description           Property   System.DirectoryServices.PropertyValueC...
distinguishedName     Property   System.DirectoryServices.PropertyValueC...
dSCorePropagationData Property   System.DirectoryServices.PropertyValueC...
instanceType          Property   System.DirectoryServices.PropertyValueC...
name                  Property   System.DirectoryServices.PropertyValueC...
nTSecurityDescriptor  Property   System.DirectoryServices.PropertyValueC...
objectCategory        Property   System.DirectoryServices.PropertyValueC...
```

```
objectClass          Property   System.DirectoryServices.PropertyValueC...
objectGUID           Property   System.DirectoryServices.PropertyValueC...
ou                   Property   System.DirectoryServices.PropertyValueC...
uSNChanged           Property   System.DirectoryServices.PropertyValueC...
uSNCreated           Property   System.DirectoryServices.PropertyValueC...
whenChanged          Property   System.DirectoryServices.PropertyValueC...
whenCreated          Property   System.DirectoryServices.PropertyValueC...
```

Hát elég furcsa, amit kaptunk. Látjuk a szervezeti egységünk tulajdonságait – de hol vannak a metódusok? Hol a `Create`? Sajnos, a PowerShell 1.0-ba még nincsen 100 százalékban adaptálva a `System.DirectoryServices` osztály. Ennek több oka is van. Az egyik, hogy valójában itt nem színtiszta .NET-osztályról van szó, hanem COM-objektum is meghúzódik a felszín alatt, és annak metódusait nem olyan egyszerű átemelni. Gondoljunk csak arra, hogy egy ilyen `DirectoryEntry` típusú objektum lehet felhasználói fiók, számítógépfiók, telephely, csoport stb., ezeknek mind más és más metódusuk van, ezeknek az adaptálása a PowerShell-környezetbe nem olyan egyszerű. Ebből származik a második ok is, hogy a fejlesztők az 1.0 megjelenését nem akarták ezzel késleltetni, várhatóan a 2.0 verzió már precízebb AD-támogatást fog nyújtani.

Szerencsére van egy kis menekvési ösvényünk, azaz kikapcsolhatjuk a PowerShell adaptációs rétegét, és megnézhetjük a „színtiszta” .NET objektumot is a `psbase` nézeten keresztül:

```
PS C:\> $adou.psbase | get-member


   TypeName: System.Management.Automation.PSMemberSet

Name                 MemberType Definition
----                 ---------- ----------
...
MoveTo               Method     System.Void MoveTo(DirectoryEntry n...
RefreshCache         Method     System.Void RefreshCache(), System....
remove_Disposed      Method     System.Void remove_Disposed(EventHa...
Rename               Method     System.Void Rename(String newName)
...
ToString             Method     System.String ToString()
AuthenticationType   Property   System.DirectoryServices.Authentica...
Children             Property   System.DirectoryServices.DirectoryE...
Container            Property   System.ComponentModel.IContainer Co...
Guid                 Property   System.Guid Guid {get;}
Name                 Property   System.String Name {get;}
NativeGuid           Property   System.String NativeGuid {get;}
NativeObject         Property   System.Object NativeObject {get;}
ObjectSecurity       Property   System.DirectoryServices.ActiveDire...
Options              Property   System.DirectoryServices.DirectoryE...
Parent               Property   System.DirectoryServices.DirectoryE...
Password             Property   System.String Password {set;}
Path                 Property   System.String Path {get;set;}
Properties           Property   System.DirectoryServices.PropertyCo...
...
```

A fenti, kicsit megvágott, de még így is hosszú listából látszik, hogy az objektumot valójában lehet például mozgatni, átnevezni, és néhány újabb tulajdonság is feltárul a szemünk előtt. De például még mindig nem látjuk a `SetInfo` és a `Create` metódust, mert ezek az ADSI COM interfészéből jönnek, és a .NET nem kérdezi le, így nem is mutatja meg, viszont meghívni, használni ennek ellenére lehet őket.

Vagy nézzük a következőket:

```
PS C:\> $d = [ADSI] „"
PS C:\> $d

distinguishedName
-----------------
{DC=iqjb,DC=w08}
```

A fenti módon például nagyon egyszerűen lehet az aktuális tartományunkhoz csatlakozni. Próbáljuk meg ennek megnézni a „rejtett” `children` tulajdonságát:

```
PS C:\> $adou = [ADSI] „LDAP://OU=Demó,DC=iqjb,DC=w08"
PS C:\> $adou.psbase.Children

distinguishedName
-----------------
{CN=Csilla Fájdalom,OU=Demó,DC=iqjb,DC=w08}
{CN=Csoport,OU=Demó,DC=iqjb,DC=w08}
{CN=group1,OU=Demó,DC=iqjb,DC=w08}
{CN=János Vegetári,OU=Demó,DC=iqjb,DC=w08}
{CN=Márton Beléd,OU=Demó,DC=iqjb,DC=w08}
```

Hiszen ez megadta az adott konténerobjektumban található alobjektumokat! Mindebből az következik, hogy nem érdemes még kidobni korábbi ADSI-ismereteinket, illetve ismernünk kell az AD-objektumok tulajdonságainak neveit, hogy ezeket a tulajdonságokat lekérdezhessük és módosíthassuk. Nézzünk erre egy példát egy felhasználói fiókkal kapcsolatban. Van egy már létező felhasználóm, annak szeretném kiolvasni és beállítani a telefonszám-tulajdonságát. Ehhez kell nekünk az, hogy tudjuk: mi is a belső elnevezése az AD-ben a telefonszám-tulajdonságnak. Ennek felderítésére nézzünk egy PowerShell-módszert:

```
PS C:\> $user = [ADSI] „LDAP://cn=János Vegetári,OU=Demó,DC=iqjb,DC=w08"
PS C:\> $user.psbase.properties

PropertyName       Value                    Capacity             Count
------------       -----                    --------             -----
objectClass        {top, person, o...              4                 4
cn                 János Vegetári                  4                 1
sn                 Vegetári                        4                 1
telephoneNumber    2008                            4                 1
givenName          János                           4                 1
distinguishedName  CN=János Vegetá...              4                 1
...
```

A fenti listában látjuk, hogy a telefonszám-attribútum neve – meglepő módon – `telephoneNumber`.

Nézzük meg, hogyan lehet ezt a telefonszámot kiolvasni, majd átírni. Az első megoldás a „PowerShell-stílusú”:

```
PS C:\> $user.telephoneNumber
1234
PS C:\> $user.telephoneNumber=2008
PS C:\> $user.setinfo()
PS C:\> $user.telephoneNumber
2008
```

Jóllehet a `Get-Member`-rel nem lehetett kiolvasni, hogy a `$user`-nek van `telephoneNumber` tulajdonsága, mégis lehet használni.

A második megoldás a hagyományos, ADSI-stílus:

```
PS C:\> $u.Get(„telephoneNumber")
1234
PS C:\> $u.Put(„telephoneNumber",9876)
PS C:\> $u.SetInfo()
```

```
PS C:\> $u.Get(„telephoneNumber")
9876
```

A Get metódussal tudjuk az adott tulajdonságot kiolvasni, a Put-tal átírni. Egyik esetben sem szabad megfeledkezni a SetInfo-ról, ami az objektum memóriabeli reprezentációját írja be ténylegesen az AD-ba.

Megjegyzés

Sajnos nem minden attribútum kezelhető a PowerShell-módszerrel. Ilyen például a *Company* attribútum:

```
PS C:\> $u.company
PS C:\> $u.get(„company")
Cég
```

Az első esetben nem kaptam semmilyen választ az attribútum kiolvasására, de get-tel mégis működött.

Mi van akkor, ha kiolvastuk egy felhasználó adatait egy változóba, majd ezután valaki egy másik gépről vagy egy másik alkalmazással módosítja a felhasználónk valamely attribútumát. Ilyenkor a getinfo metódussal lehet frissíttetni az adatokat a memóriában:

```
PS C:\> $u.get(„company")
Egyik
PS C:\> $u.getinfo()
PS C:\> $u.get(„company")
Másik
```

A fenti példában az első kiolvasás után az ADUC eszközzel átírtam a felhasználó *Company* attribútumát, és a getinfo-val ezt frissítettem a memóriában, így az új érték már a PowerShellből is látszik.

Munka többértékű (multivalued) attribútumokkal

Az Active Directory egyik jellegzetes attribútuma a „multivalued property". Ez olyan tulajdonság, ahova az értékek listáját, tömbjét tehetjük. Legtipikusabb ilyen attribútum az Exchange Server bevezetése után a felhasználók e-mail-címeit tartalmazó ProxyAddresses vagy a csoportok Member attribútuma, de ez utóbbit külön kezeljük speciális metódusokkal. Maradnak mondjuk az other... kezdetű különböző telefonszámok tárolására szolgáló attribútumok, mint például az otherMobile vagy az otherTelephone.

Ezeket ki lehet olvasni az eddig megismert módszerekkel is, de nézzük, hogy milyen problémákkal szembesülhetünk. Ha a get metódust használom, és csak egy értéket tárol a „multivalued propery", akkor nem egyelemű tömböt kapok, hanem sima skaláris értéket:

```
PS C:\> $user.get(„otherMobile")
othermobil1234
PS C:\> $user.get(„otherMobile").gettype()

IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     True     String                                   System.Object
```

Ezzel szemben, ha több értéket tárolunk, akkor már tömböt kapunk:

```
PS C:\> $user.get(„otherMobile")
othermobil2345
othermobil1234
PS C:\> $user.get(„otherMobile").gettype()

IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     True     Object[]                                 System.Array
```

Ez nem biztos, hogy jó nekünk, mert így a szkriptünket kétfajta esetre kell felkészítenünk: külön arra az esetre, ha csak egy értéket tárolunk, és külön arra az esetre is, ha többet. Ez bonyolítja a programjainkat. Ha konzisztensen, mindig tömbként akarjuk kezelni az ilyen *multivalued propertyket*, akkor vagy használjuk a PowerShell-stílust:

```
PS C:\> $user.otherMobile
othermobil1234
PS C:\> $user.otherMobile.gettype()

IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     False    PropertyValueCollection                  System.Collec...
```

Vagy használjuk a GetEx metódust:

```
PS C:\> $user.getex(„otherMobile")
othermobil1234
PS C:\> $user.getex(„otherMobile").gettype()

IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     True     Object[]                                 System.Array
```

Nem tökéletesen egyforma a két kimenet típusa, de mindkettő tömb (collection) típusú.

Az ilyen multivalued propertyk írása sem egyértelmű, hiszen több lehetőségem is van:

- a meglevő értékekhez akarok egy újabbat hozzáfűzni,
- a meglevő értékek helyett akarok egy vagy több újat betölteni.

Ezeket a lehetőségeket én magam is tudom programozni a szkriptemben. Ha az első változatra van szükségem, akkor előbb kiolvasom az attribútum aktuális tartalmát egy változóba, hozzárakom az új értéket, és így rakom vissza a put-tal vagy egyszerű értékadással. Ha pedig a második változatra van szükségem, akkor egyszerűen felülírom az attribútumot az új értékkel.

Sokkal elegánsabb, ha ezt már maga az objektum tudná egy „okosabb" metódussal. Ilyen létezik, mégpedig a PutEx:

```
PS C:\> $user.getex(„otherMobile")
othermobil1234
PS C:\> $user.putex(3,"otherMobile",@(„othermobilPutEx2"));$user.setinfo()
PS C:\> $user.getex(„otherMobile")
othermobilPutEx2
othermobil1234
PS C:\> $user.putex(2,"otherMobile",@(„othermobilPutEx3"));$user.setinfo()
PS C:\> $user.getex(„otherMobile")
othermobilPutEx3
```

A fenti példában a kiinduló állapotban egy mobilszámunk van. Ezután hozzáfűzök egy újabbat a putex használatával, a hozzáfűzést az első paraméterként szereplő 3-as jelzi. Fontos, hogy a hozzáfűzendő értéket tömbként kell kezelni, ezért van ott a kukac-zárójelpár! Ezután egy újabb putex-et hívok meg, immár 2-es paraméterrel, ez a felülírás művelete, hatására már csak ez a legújabb mobilszám lesz az attribútumban.

Használhatok még 1-es paramétert is, ez ekvivalens az attribútum értékeinek törlésével, vagy használhatok 4-es paramétert, ez egy elemet töröl az értékek közül:

```
PS C:\> $user.putex(3,"otherMobile",@(„Append"));$user.setinfo()
PS C:\> $user.getex(„otherMobile")
Append
othermobilPutEx3
PS C:\> $user.putex(4,"otherMobile",@(„Append"));$user.setinfo()
PS C:\> $user.getex(„otherMobile")
othermobilPutEx3
```

A fenti példában elsőként hozzáfűzök egy értéket, majd ugyanezt eltávolítom.

Speciális tulajdonságok kezelése

Van néhány olyan attribútum, amelyek az eddig megismert módszerek egyikével sem kezelhetők:

```
PS C:\> $user.AccountDisabled
PS C:\> $user.get(„AccountDisabled")
Exception calling „get" with „1" argument(s): „The directory property canno
t be found in the cache."
At line:1 char:10
+ $user.get( <<<< „AccountDisabled")
```

A PowerShell-szintaxis meg se nyikkan, a get meg még hibát is jelez. Ilyen esetekben használhatjuk a psbase nézeten keresztül az InvokeGet és InvokeSet metódusokat:

```
PS C:\> $user.psbase.invokeget(„AccountDisabled")
False
PS C:\> $user.psbase.invokeset(„AccountDisabled","TRUE")
PS C:\> $user.SetInfo()
```

Jelszó megváltoztatása

Speciális attribútum a jelszó, hiszen tudjuk, hogy valójában nem (feltétlenül) tárolja a címtár a jelszavakat, hanem csak a belőlük képzett hasht. Így a jelszó kezelésekor nem egyszerűen egy attribútumot kell beállítani, hanem ezt a hasht kell képezni. Szerencsére erre a célra rendelkezésünkre áll két metódus, a SetPassword, illetve a ChangePassword:

```
PS C:\> $user.SetPassword("UjPass2")
PS C:\> $user.ChangePassword("UjPass2","MégújabbPass3")
```

A SetPassword felel meg a Reset Password műveletnek. Ezt, ugye, csak megfelelő rendszergazdai jogosultságokkal tudjuk meghívni. A ChangePassword a meglevő jelszó birtokában módosítja a jelszót, ehhez már nem kell külön rendszergazdai jogosultság. Mindkét metódus ténylegesen be is írja az új jelszót a címtárba, tehát nincs szükség a SetInfo-ra.

Csoportok kezelése

Az Active Directoryban csoportokat leginkább a rendszer üzemeltetésének megkönnyítésére veszünk fel. Segítségükkel osztunk ki hozzáférési jogokat, felhasználói jogokat, de még a csoportos házirendek érvényre jutását is szabályozhatjuk csoportokkal. Miután ilyen széles körű a felhasználásuk, fontos lehet a csoportok kezelésének automatizálása.

Erre is kiválóan alkalmas a PowerShell, nézzük meg a leggyakoribb műveleteket.

Csoportot létrehozni a már látott módszerrel lehet:

```
PS C:\> $target = [ADSI] „LDAP://ou=Demó,DC=iqjb,DC=w08"
PS C:\> $group = $target.create(„group","CN=Csoport")
PS C:\> $group.setinfo()
```

Ez alaphelyzetben globális biztonsági csoport. A későbbi, összetett példában majd bemutatom, hogyan lehet másfajta csoportokat is létrehozni.

Ezután kétféleképpen lehet tagokat adni a csoportokhoz. Az első módszer a hagyományos „ADSI"-s módszer, ahol a csoport Add metódusát hívom meg, paramétereként a berakni akart felhasználó LDAP-os szintaxisú elérési útját kell megadni. Vagy ha már megragadtam a felhasználói fiókot, akkor vissza kell alakítani az LDAP-os elérési úttá, mint ahogy ebben a példában tettem:

```
PS C:\> $user = [ADSI] „LDAP://CN=János Vegetári,OU=Demó,DC=iqjb,DC=w08"
PS C:\> $group.add(„LDAP://$($user.distinguishedname)")
PS C:\> $group.setinfo()
```

Hasonlóan lehet tagot eltávolítani, csak az Add helyett a Remove metódust kell meghívni.

A második módszer kicsit PowerShell-szerűbb, itt nem kell ide-oda alakítgatni, elég a felhasználó distinguishedname tulajdonságát használni:

```
PS C:\> $user = [ADSI] „LDAP://CN=Csilla Fájdalom,OU=Demó,DC=iqjb,DC=w08"
PS C:\> $group.member += $user.distinguishedname
PS C:\> $group.setinfo()
```

Természetesen a két megoldás egyenértékű, csak stílusbeli különbség van közöttük. A második módszer hátránya talán, hogy egyszerűen nem lehet csoporttagot eltávolítani, külön képezni kellene a nemkívánatos tag nélküli tömböt, és azt betölteni a csoport member tulajdonságába.

Keresés az AD-ben

Az igazán profi kereséshez a .NET keretrendszer egyik osztályát, a System.DirectoryServices.DirectorySearcher-t hívjuk segítségül, ennek egy objektuma lesz a keresőnk, és ennek különböző tulajdonságait beállítva adjuk meg a keresésünk mindenféle feltételét. Nézzünk egy nagyon egyszerű feladatot, egy konkrét felhasználói fiókra keressünk rá:

```
[6] PS I:\>$objRoot = [ADSI] „LDAP://OU=IQJB,DC=kfki,DC=corp"
[7] PS I:\>$objSearcher = New-Object System.DirectoryServices.DirectorySearc
her
[8] PS I:\>$objSearcher.SearchRoot = $objRoot
[9] PS I:\>$objSearcher.Filter = „(&(objectCategory=user)(displayName=So-
ós Tibor))"
[10] PS I:\>$objSearcher.SearchScope = „Subtree"
[11] PS I:\>$colResults = $objSearcher.FindAll()
[12] PS I:\>$colresults

Path                                    Properties
----                                    ----------
LDAP://CN=Soós Tibor,OU=Normal,OU=... {homemdb, distinguishedname, count...
```

Elsőként definiálom, hogy az AD adatbázis-elemek fastruktúrájában hol is keresek majd ($objRoot). Majd elkészítem a keresőt ($objSearcher), amelynek SearchRoot tulajdonságaként az előbb létrehozott keresési helyet adom meg.

Majd definiálom az LDAP-formátumú szűrőt, amely ebben az eset-

ben a „Soós Tibor” nevű felhasználókat jelenti, és ezt betöltöm a kereső Filter tulajdonságaként. Végül meghatározom a keresés mélységét, ami itt Subtree, azaz mélységi, mert nem pont közvetlenül a kiindulópontként megadott helyen van a keresett objektum. Nincs más hátra, ezek alapján ki kell listázni a feltételeknek megfelelő objektumokat a FindAll metódussal.

A $colResult változóban tárolt eredmény nem közvetlenül DirectoryEntry típusú elemek tömbje, hanem egy hashtábla-szerűség, ahol a Path oszlop tartalmazza a megtalált objektum LDAP formátumú elérési útját, a Properties meg a kiolvasható tulajdonságait. Azaz ahhoz, hogy kiolvassuk például az én nevemet és beosztásomat, egy kicsit trükközni kell:

```
[25] PS I:\>"$($colResults[0].properties.displayname) az én nevem,
beosztásom $($colResults[0].properties.title)"
Soós Tibor az én nevem, beosztásom műszaki igazgató
```

Megjegyzés

PowerShell-ténykedésem során ez a második eset, amikor kis—nagybetű érzékenységet tapasztaltam! (Az első az LDAP:: kifejezésnél volt, de ez félig-meddig betudható az ADSI-örökségnek.) A második ez: ha $colResults[0].properties.displayName-et írok (nagy „N” az utolsó tagban), akkor nem kapok semmit. Ez azért is furcsa, mert eredetileg a címtárban nagy az „N”.

A következő példában egy függvényt hozok létre, amellyel felhasználói fiókok tetszőleges attribútumát lehet tömegesen lecserélni valami másra. A kód megfejtését az előzőek ismeretében az olvasóra bízom. Csak egy kis segítséget adok: a középrészen található If vizsgálat Else ágában azt érem el, hogy ha a kicserélendő attribútumérték üres, azaz azt szeretnénk, hogy a ki nem töltött attribútumokat töltsük ki, akkor az LDAP-filterben a !$Attr=* kifejezést kell szerepeltetni, ennek az a jelentése, hogy *„az $Attr változó által jelzett attribútum nem egyenlő akármi, azaz van értéke”*.

```
function ModifyUserAttrib
{
param (
$domain =
        [System.DirectoryServices.ActiveDirectory.Domain]::getcurrentdo-
main().Name,
$Attr = $(throw „Melyik attribútumot?"),
$sValue = $null,
$cValue = $(throw „Mire változtassam?")
)

$root= [ADSI] „LDAP://$domain"
$Searcher = New-Object DirectoryServices.DirectorySearcher
$Searcher.SearchRoot = $root
if($sValue)
{
        $buildFilter = „(&(objectClass=user)($Attr=$sValue))"
}
else
{
        $buildFilter = „(&(objectClass=user)(!$Attr=*))"
}
$Searcher.Filter = $buildFilter
$users = $searcher.findall()
Foreach ($i in $users)
{
        $dn=$i.path
        $user = [ADSI] $dn
        write-host $dn
        $user.Put($Attr,$cValue)
        $user.SetInfo()
}
}
```

Keresés idő típusú adatokra

A címtárban nemcsak szöveges adatok vannak, hanem például dátum típusúak is. Ezekre nem triviális a keresés. Például keresem az utóbbi 2 napban módosított AD-felhasználói objektumokat:

```
PS C:\> $tól=get-date ((get-date).AddDays(-2)) -Format yyyyMMddHHMMss.0Z
PS C:\> $tól
20080530110514.0Z
PS C:\> $searcher = New-Object directoryservices.directorysearcher
PS C:\> $searcher.searchroot = [ADSI] „"
PS C:\> $searcher.filter = „(&(objectCategory=person)(objectClass=User)(when
Changed>=$tól))"
PS C:\> $result = $searcher.findall()
PS C:\> $result

Path                                Properties
----                                ----------
LDAP://CN=János Vegetári,OU=Demó,D... {samaccounttype, lastlogon, dscore...
```

Az egészben a lényeg a $tól változó generálása, látjuk, hogy egy speciális formátumra kell hozni a dátumot: ÉÉÉÉHHNNÓÓPPMM.0Z. Ilyen formázást szerencsére a get-date format paraméterével könnyen elvégezhetünk.

Sajnos nem mindig ilyen egyszerű a helyzetünk, hiszen néhány dátumot jelző attribútum nem ilyen formátumban tárolja az időt, hanem „tick”-ekben, ketyegésekben. Ez 1600. január 1. 0:00 időponttól eltelt 100 nanoszekundumokat jelenti, mindez *long-integer* formátumban. Ilyen attribútum például a *lastLogon*, *lastLogonTimestamp*, *lastLogoff*, *pwdLastSet*. Ha ilyen attribútumok valamely értékére akarunk rákeresni, akkor elő kell tudnunk állítani ezt az értéket. Szerencsére a .NET keretrendszer ebben is segít. Elsőként nézzük, hogy dátumból hogyan tudunk ilyen *long-integer*-t előállítani:

```
[5] PS C:\> $most = get-date
[6] PS C:\> $most

2008. június 1. 20:49:41
[7] PS C:\> $most.ticks
633479501812656250
```

Látjuk, hogy elég a ticks tulajdonságát meghívni a dátumnak. Nézzük visszafelé, azaz hogy *long-integer*-ből hogyan kapunk dátumot! Ez sem nagyon bonyolult:

```
[8] PS C:\> $MyLongDate = 633479501812656250
[9] PS C:\> $mydate = New-Object datetime $MyLongDate
[10] PS C:\> $mydate

2008. június 1. 20:49:41
```

Egyszerűen kell kreálni egy dátum típusú objektumot, az objektum konstruktorának kell átadni ezt a számot, és ebből a .NET automatikusan létrehozza a dátumot.

Soós Tibor
(soost@iqjb.hu)
MCT, IQSOFT – John Bryce Oktatóközpont

FEFO COMPUTER Storage szerver megoldások

http://www.fefo.hu

Microsoft® Windows® Storage Server® Operációs rendszerrel

intel Xeon® inside™

☑FEFO NetiX Base Storage m1v1

- Belépő szintű Storage (NAS) megoldás kis és középvállalatok számára
- Opcionális DPM

☑FEFO NetiX Workgroup Storage m1v1

- Storage (NAS) megoldás középvállalatok számára
- Opcionális DPM

☑FEFO IntiX Base Storage m1v1

- Belépő szintű Rack kivitelű Storage (NAS) megoldás kis és középvállaltok számára
- Opcionális DPM

☑FEFO IntiX Workgroup Storage m1v1

- Rack kivitelű Storage (NAS) megoldás középvállalatok számára
- Opcionális DPM

Cím: 1135 Budapest, Jász u. 33-35. telefon: 1-412-3580 e-mail: ertekesites@fefo.hu
Cím: 6722 Szeged, Szentháromság u. 27. telefon: 62-422-386 e-mail: szeged@fefo.hu
Cím: 7621 Pécs, Munkácsy M. u. 9. telefon: 72-516-316 e-mail: pecs@fefo.hu
Cím: 9026 Győr, Ady E. u. 10. telefon: 96-519-062 e-mail: gyor@fefo.hu

Windows (Unified Data) Storage Server

NAS, azaz Network Attached Storage. Ebbe a kategóriába sorolhatjuk a Windows Storage Servert, ami jelenleg a 2003 R2 családot erősíti, de talán még ebben az évben meglesz az új, 2008-as változata. Mit tud egy ilyen Windows Server alapú NAS-megoldás? Mit jelent a „Unified" kiegészítés? Menjünk le a gépházba, és érintsük meg a gőzölgő csöveket és szelepeket!

Hogyan készül a NAS-kiszolgáló? Íme, a recept: vegyünk egy fájlszerverkiszolgálásra optimalizált hardvert. Ebben a dobozban olyan módon illesztett alrendszereket és adatsíneket találunk, amelyek az adott feladathoz és a várható terheléshez méretezett módon nem, vagy csak igen kis mértékben tartalmaznak szűk keresztmetszetet az adatáramlás útjában. Telepítsünk fel rá egy olyan operációs rendszert, amelyik mind teljesítményében, mind szolgáltatásait tekintve kifejezetten a fájlkiszolgáló szerep betöltésére lett kialakítva. Ha ez az operációs rendszer a Windows Server család tagja, akkor azzal sem kell bajlódnunk, hogy egy újabb célgép vagy operációs rendszer kezelését, menedzsmentjét megértsük és elsajátítsuk.

Az így kialakított kiszolgálót illesszük a hálózatba, engedjük rá tárhelyre és teljesítményre éhes (no és persze többnyire meglehetősen fegyelmezetlen) felhasználóinkat, és várjunk türelemmel. Átlagos esetben mi történik? Minél gyorsabb egy kiszolgáló, minél több a felhasználó, annál hamarabb megtelik még a leggigászibb tárhely is. Igen, kerül oda a munkával összefüggő állományokból is, de sajnos leginkább filmek, zenék, képek és azonosíthatatlan forrásból származó applikációk – sokszor ugyanaz, több példányban. Mit tehet ilyenkor a rendszer gazdája? Van, aki a nyugtatók hosszú távú hatásaival számolva inkább beletörődik. Mások elkeseredetten lobbiznak az adattárolási házirend kialakításáért céges szinten, ami szigorúan büntetné a...

Van viszont egy olyan, irigyelt típus, aki észre sem veszi. Hogyan csinálja? A dolog egyszerűnek tűnik: megismerkedik a Windows Storage Server szolgáltatásaival. Alkalmazza a lehetőségeket, és néha-néha ránéz a kezelőfelületre, hogy minden úgy működik-e, ahogy megálmodta.

És úgy működik! Nézzük meg, melyek a Storage Serverben rejlő lehetőségek! Először is jöjjön a „feketeleves"!

1. táblázat

	Express	Workgroup	Standard	Enterprise
Processzor (fizikai)	1	1	1-4	1-8
32 bit és 64 bit	Igen	Igen	Igen	Igen
Diszkek száma	2	4	Nincs korlát	Nincs korlát
Hálózatok	1	2	Nincs korlát	Nincs korlát
Nyomtatási szolgáltatás	Nem	Nem	Igen	Igen
CAL szükséges?	Nem	Nem	Nem	Nem
iSCSI target	Opcionális	Opcionális	Opcionális	Opcionális
Failover Clustering	Nem	Nem	Nem	Igen

2. táblázat

Tiltott funkcionalitás	Példa	Kivételek, megjegyzések
Autentikációs szolgáltatás, címtárszolgáltatás	Microsoft Active Directory	AD Application Mode (ADAM), AD Federation Service (ADFS)
Network-infrastruktúra	RRAS, WINS	DHCP-kiszolgáló engedélyezett
Terminálszolgáltatás	Windows Server 2003 Terminal Services	„Remote Admin" mód engedélyezett
Network Load Balancing	Windows Server 2003 Network Load Balancing network driver	DFS Load Balancing engedélyezett
Nagyvállalati adatbázis-kezelő motor	Microsoft SQL Server	MSDE engedélyezett
Levelező- és csoportmunka-kiszolgáló, beleértve azok web alapú elérhetőségét is	Exchange Server 2003 Lotus® Notes SharePoint Portal Server Outlook Web Access	Windows SharePoint Services engedélyezett
Egyéb, a kiszolgálón futó szerveroldali applikációk		Például víruskeresők kiszolgálóoldali moduljai

Vallomások – az igazság 3 perce

A Windows Storage Server olyan, mint a többi Windows Server. Azaz hogy – majdnem! Ugyanúgy van belőle Standard és Enterprise kiadás, de létezik még két „kisebb" változat is, különböző megkötésekkel *(l. táblázat)*.

A tisztesség úgy kívánja, hogy feltárjak néhány „titkot", amelyeknek tudatában érdemes elgondolkozni a bevezethetőség módjáról vagy egyáltalán annak lehetőségéről. A Storage Server elsősorban fájlkiszolgáló. Ebben, mint látni is fogjuk, nagyon jó. Viszont van a Windows-világban néhány olyan tulajdonság és szolgáltatás, amelyeket vagy nem tudunk feltelepíteni rá, vagy a licencben foglaltak szerint nem szabad. Ez a lista a *2. táblázatban* olvasható.

A táblázat átböngészése után mindent tudunk, ami esetleg megköti a kezünket tervezéskor. Érdemes azonban a „tiltott" szolgáltatások körmére nézni, hiszen igaz, hogy a Storage Server nem lehet tartományvezérlő, de attól még örömmel válik egy már meglévő domain tagjává, és ugyanúgy végrehajtja a csoportházirendet, mint a többiek. Igaz, hogy nem telepíthetjük fel rá például az Exchange szervert, de minden további nélkül képes iSCSI targetként tárolni a levelezőrendszerünk adatbázisát.

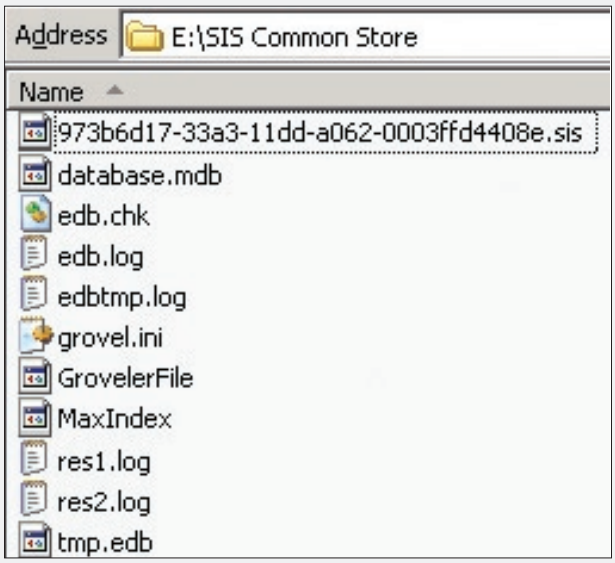
A továbbiakban pedig következzenek a jó hírek!

Ha a Storage Server mellett döntünk, számos olyan szolgáltatást kapunk, amelyek nincsenek, vagy csak részben vannak meg a többi Windows-kiszolgálóban.

Melyek ezek?

SIS – Single Instance fáljrendszer

Az első és talán a legfontosabb szolgáltatás a SIS, azaz a Single Instance fáljrendszer. A fenti példában vázolt probléma, ami miatt a tárhely végtelen kapacitását kívánja az üzemeltetők többsége, egyszerűen megoldható a segítségével. Arra szolgál, hogy a fájlrendszerben elhelyezett azonos tartalmú állományok, amelyek különböző könyvtárban és esetleg más-más néven mentettek, ne foglalják a drága tárkapacitást a példányszámnak megfelelő mennyiségben. Gondoljunk csak egy e-mailben kapott vicces filmecskére, ami beérkezik 25 felhasználónkhoz! Közülük minimum 18-an elmentik azt, valószínűleg a hálózati saját adatterületükre, a home directoryba. Ha ezek a könyvtárak a szerveren ugyanazon a köteten vannak, a SIS ezt észreveszi, és az állománynak egyetlen példányát helyezi el egy rejtett, közös területre. A felhasználói mappába viszont pointert, linket illeszt. Mennyi a megtakarítás? A vicces videó méretének 17-szerese! Mi van akkor, ha különböző néven mentik el az állományokat? Ugyanez, a SIS nem a fájlnevet, hanem a tartalmat figyeli. Nagyszerű! – kiáltanak fel sokan. – Ugyan már! – mormognak az Exchange-rendszergazdák –, ilyen a levelezőrendszerben is van évek óta. Ez igaz, de a fájlrendszerben? Bizony, a Storage Server NTFS-kötetein a SIS egy kattintással aktiválható!



1. ábra

Hogyan működik a SIS?

A SIS főbb komponensei közül az első és legfontosabb a ***SIS Groveler.*** Ez egy rendszerszerviz, mely addig nem aktív, amíg az első köteten be nem kapcsoljuk a szolgáltatást. (Megjegyzendő, hogy a SIS egy időben maximálisan 6 köteten képes működni.) A Groveler az, amelyik figyel. Figyeli az NTFS fájlrendszer változásait, és folyamatos összehasonlítgatást végez a tartalmakban. Hú, nem túl nagy munka ez? Nyugalom, ésszel csinálja. Minden állománytörzsben, valahol középtájon mintát vesz, és abból hash-t számít. Ha észlel két olyan állományt, amelyeknek az így képződött hash-értékei megegyeznek, sóhajt egyet, és nekilát egy igencsak derekas, bitszintű összehasonlításnak. Ha a két fájl *tényleg* megegyezik, felriasztja álmából legjobb barátját, a ***SIS Storage Filtert.*** Ez egy kernel módú fájlrendszeri komponens (driver), amelyik a Groveler kérésére a megjelölt állományokból egy példányt kimásol a ***SIS Common Store*** rejtett rendszerkönyvtárba az adott köteten, majd a fájlok eredeti helyére létrehozza az eredeti fájlneveknek megfelelő ***SIS Linket.*** A SIS Link az, ami kísértetiesen hasonlít a Unix-rendszerekben és immár a Windows Vistában és Windows Server 2008-ban is használatos Symbolic Linkre,

```
C:\Documents and Settings\Administrator>sisadmin /v e:
Analyzing volume 'e:'.
Analyzing the SIS Common Store directory...
Analyzing the SIS Common Store directory... 1 processed, complete.
Sorting SIS common store file list...
Sorting SIS common store file list... complete.
Analyzing reparse points...
Analyzing reparse points... 2 processed, 0 not analyzed, complete.
Analyzing data...
Analyzing data... complete.
=== Analysis of volume 'E:\' on STORAGESERVER ===
Common store files:       1
Link files:               2
Inaccessible link files: 0
Space saved:              320 KB
```

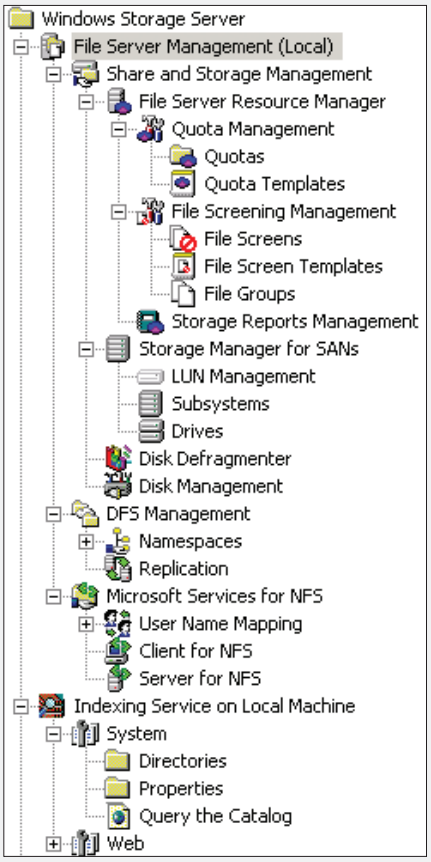
2. ábra

azzal a nagy különbséggel, hogy nem manuálisan jön létre, és nem igényel beavatkozást az eltűnése sem, mindent a SIS Storage Filter végez vele kapcsolatban. Mit észlelnek ebből a felhasználók? Egészen pontosan: semmit! (Az *1. ábrán* a SIS Common Store látható a működéshez szükséges adatbázissal és egy, már azonosított „.sis" kiterjesztésű állománnyal. Az eredeti könyvtárakban változatlan névvel mutatnak a SIS Linkek erre az állományra. Az összefüggések pedig a database .mdb-ben rejtőznek.) – Ez igazán szép! – dünnyögik a szkeptikusok. – De mi van a mentéssel? Hány példányban mentjük éjszaka az így tárolt állományokat? – Ezt bizony mentőszoftvere válogatja, a SIS rendszerén nem múlik. A Storage Server tartalmazza a ***SIS Backup API-t,*** amelyen keresztül a mentés is megértheti a tárolás miként jét, a Common Store és a SIS Link összefüggéseit. Ha megérti, csak nyerünk vele: a szalagon tárolt adatunk fizikai valójában kevesebb, a mentés pedig gyorsabb lesz. Ez nem túl nagy baj, ugye? Végezetül gondoljunk a **{ Hősök }** -re is egy kicsit, a rendszergazdákra, akik a SIS-t üzemeltetik. Milyen felülettel kényezteti őket a Storage Server? Papír zsebkendőket elő, lehet, hogy sírni fogunk: a felület parancssori! Első látásra elég meghökkentő, hogy egy ilyen horderejű funkciónak nincs minimum egy MMC-be épülő modulja, de ha egy kicsit is belegondolunk, talán nem is baj. Mit kell a SIS-en adminisztrálni? Alig valamit. Be lehet ugyan kapcsolni a grafikus felületen, azután maximum statisztikákat nézegetnénk színesben... talán majd egyszer. Addig is itt a ***SISADMIN.EXE!***

A *2. ábrán* a kiszolgáló E:\ meghajtóján 2 különböző könyvtárban 1-1 példányban lakik egy 320 kilobájt méretű dokumentum. A megtakarítás mértéke meggyőző. Az arányok hasonlóak akkor is, ha nem tesztrendszerben vizsgálódunk, hanem élesben, adott esetben terabájtos nagyságrendű állománymennyiséggel. Egy utolsó kérdés merül fel bennem, bár már meg vagyok győzve: mi történik akkor, ha az egyik SIS Linkre kattintva a felhasználó módosítja a saját „példányát"? Ebben az esetben a Storage Filter elkészíti a felhasználónk állományának a valós változatát a munkakönyvtárban, a SIS Linket pedig lebontja. Jó-jó, de mi lesz akkor, ha valaki kiad egy törlési utasítást egy SIS Linkre? Ebben az esetben csak a link törlődik. Viszont az utolsó link törlésével együtt a Common Store tartalmát képző valós fájltest is elutazik a Recycle Binbe. A SIS jelentősége megkérdőjelezhetetlennek látszik! Honnan került elő? Most következik az utolsó vallomás vele kapcsolatban. A RIS-ből. A Remote Installation Services környezetéből, ami már igen régóta része a Windows kiszolgálóknak. Emlékszünk a RIPrep image szerkezetére? Onnan emelték ki és tették önálló rendszerkomponenssé itt, a Storage Serverben. Ha valamire, hát a SIS-re tényleg komoly szükség lehet egy valamirevaló fájlkiszolgálóban! Arról nem is beszélve, hogy a fájlrendszeri műveleteket gyorsító cache kihasználtsága is nagymértékben javulhat általa! Hogyan? Ha az azonos tartalmú fájlokat klienseink egyszerre nyitogatják meg, nem kell mindegyik példányt a gyorsítótárban tartani, bőven elég csak egyet – a Common Store könyvtárban lévő valós állományt. Ez pedig egy erőteljesen kihasznált kiszolgáló általános teljesítményére feltétlenül pozitív hatással van!

File Server Management-funkciók

Most pedig következzenek azok a kiváltságos szolgáltatások, amelyek – ellentétben a SIS-sel – a grafikus felületen elérhető felügyelettel bírnak *(3. ábra)!* A Storage Serverben lé-

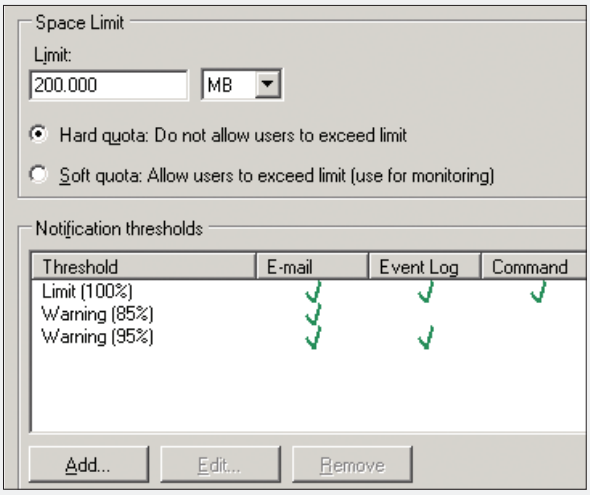


3. ábra

tezik egy olyan MMC beépülő modul, amely összefogja az extra funkcionalitás nagy részét: a File Server Management, Quota és Screening, DFS és NFS, az Indexing és a SAN-eszközök kezelése lehetséges a menün keresztül, tessék választani!

Quota Management – ésszerűbben

Quota, azaz a felhasználónkénti tárhelyfoglalás mérése és az evvel összefüggő intézkedések a Windows Serverekben is vannak, az R2 óta pedig már nemcsak a komplett kötet, illetve partíció szintjén, hanem már könyvtárra bontott korlátozást is beállíthatunk. Hogyan? Segíthetnek a Quota Template-ek, de saját elképzeléseinket is megvalósíthatjuk. A Management Console-ban elérhető lehetőségek igen széles körűek. Csak a példa kedvéért: A kiválasztott könyvtárra alkalmazzunk egy template-et, amelyben megszabjuk a felhasználók által lefoglalható terület mértékét. Saját template-et is menthetünk tetszés szerint. Eldönthetjük, hogy a benne foglalt érték valós korlát (hard limit), vagy csak monitorozásra való (soft limit). A figyelmeztetésre és a végleges limitre beállított százalékos értékek



4. ábra

elérésekor különféle akciókat is foganatosíthatunk. Ezek között szerepel az e-mail (*nem* a „net send"!) a felhasználónak és/vagy az adminisztrátornak, valamint naplóbejegyzések és szkriptek futtatása mint opciók. A szkriptelés érdekes lehetőséget rejt, meghívhatjuk vele a quota parancssori felügyeletét (**DirQuota .exe**), amit megfelelően paraméterezve automatikus quota-váltásra is használhatunk. Mire jó ez? A felhasználót a limit előtt többször is figyelmeztethetjük, majd amikor eléri a tényleges korlátot, még egy utolsó lehetőségként egy kicsivel több helyfoglalást biztosító kvótát biztosítunk számára. Ettől kezdve viszont már nem vagyunk engedékenyek. Az abban foglalt limitet már tényleg komolyan kell venni *(4. ábra)!* Egy további lehetőség az **AutoQuota,** amely a limitált könyvtár meglévő és jövőbeni alkönyvtáraira is automatikusan beállítja a kívánt korlátozást.

File Screening – állománytípusok szűrése

Örömhír! A File Screening segítségével rá tudjuk venni a Storage Servert, hogy az általunk megjelölt kiterjesztésekkel bíró tartalmakat ne lehessen bizonyos területekre elmenteni. Jó, tudom, a kiterjesztések megváltoztatása régi user-trükk, de számos felhasználó nem bajlódik vele. Hogyan működik? A kezelőfelület nagyon hasonlít az előzőekben tárgyalt kvótáéhoz. Adjunk meg egy tetszőleges kötetet vagy könyvtárat, és állítsuk be, hogy milyen állománytípusokat nem látunk szívesen. Abban az esetben, ha a vizsgált típus, azaz az annak megfelelő kiterjesztéssel bíró fájl megjelenik, annak mentését a File Screening blokkolhatja, vagy csak naplóz, értesít, szkriptet futtat – igény szerint. Az *5. ábrán* láthatóak a template-ek, amelyeket megváltoztathatunk, vagy saját elképzeléseink szerint újakat is készíthetünk. Gondoljunk csak a cikk bevezetőjében említett rendszergazdára! A SIS, a Quota és a File Screening együttes használatával olyan eszközök vannak a kezében, amelyekkel könnyen gátat szabhat a felhasználók féktelen tárhelyhasználatának. Ha viszont (csak) tájékozódni szeretne a helyfoglalás számtalan jellemzőjéről, akkor van egy roppant hasznos része is a File Server Managementnek – a ***Storage Reports Manager.*** Milyen riportokat tud készíteni? A *6. ábra* magáért beszél!

File screen Template	Screening Type	File Groups
Block Audio and Video Files	Active	Block: Audio and Video Files
Block E-mail Files	Active	Block: E-mail Files
Block Executable Files	Active	Block: Executable Files
Block Image Files	Active	Block: Image Files
Monitor Executable and System Files	Passive	Warn: Executable Files, System
TXT Screen	Active	Block: Text Files

5. ábra

A riport számtalan formában megjeleníthető, és rendkívül részletgazdag. Tortadiagramok és táblázatok segítenek felmérni a tárhelynek és fogyasztóinak jellemzőit.

Indexing – keresés tartalomra is

A Windows Server 2003/2008 is magában rejti az ***Indexing Service*** szolgáltatást, akárcsak a Storage Server. Igen ám, de meglátásom szerint ezzel a szervizzel az üzemeltetők többsége kétféle kapcsolatban van: vagy nem használja, mert tény, hogy egy nagyon kevéssé dokumentált és messze alulreklámozott dolog, vagy pedig feleslegesnek tartja, és az első adandó alkalommal tiltja a működését. De miért? Akár komolyan is vehetjük a létét egy olyan fájlkiszolgálón, ahol állományok százezreivel dolgozunk (ez bizony nem ritka). Egy kis memóriát áldozunk rá, és bízvást meghálálja. Meg bizony, már ha tényleg használjuk is. E cikk terjedelmi lehetőségein messze túlmutat a szerviz beállítási lehetőségeinek, optimalizálásának és alkalmazhatóságának hatalmas halmaza, de a Storage Serveren csak kapcsoljuk be egy adott területre, ezáltal az ott található állományok a katalógusba kerülnek. Ha ismert szöveges típusokról van szó, akkor azoknak a ***tartalmában is*** kereshetünk – akár felhasználói oldalról is. Barátkozzunk meg vele, hiszen ebből a szempontból (lehet, hogy szentségtörés ilyet kijelenteni, de meggyőződésből teszem) a SharePoint alternatívája! Jó, nincs verziókövetés és csili-vili felület, de szegény ember vízzel főz.

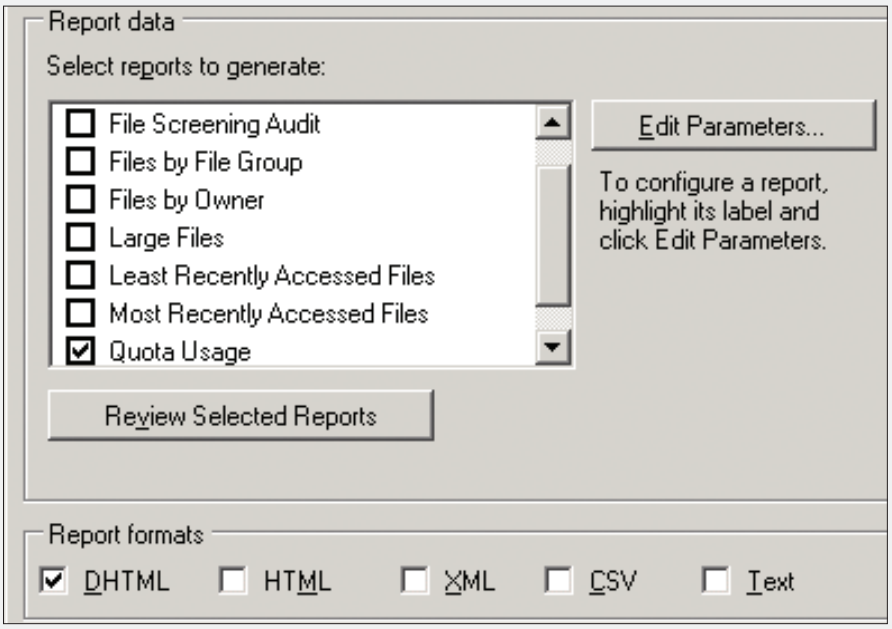
DFS – az elosztott fájlrendszer és a megújult replikáció

A Technet Magazin előző számában találhatunk egy részletes ismertetőt a megújult DFS tulajdonságairól és működéséről, ezért ezt a szolgáltatást itt nem fejteném ki. Fontos szolgáltatás, de nem csak a Storage Serverre jellemző. Benne van a Windows Server 2003 R2-változatokban ugyanúgy, mint a Windows Server 2008-ban. Funkciói közül röviden a virtuális hálózati adatszervezést, az Active Directoryval integrált elérhetőségeket és annak hibatűrését, a szinte bármire rávehető replikációs mechanizmusokat emelném ki, persze a teljesség igénye nélkül. Olyan – főleg sokszerveres, és esetleg sok telephelyes – környezetben, ahol az adatok nehezen átlátható módon vannak elhelyezve és csoportosítva, komoly segítséget nyújt a felhasználók számára.

Services for NFS – látszódjunk NFS-kiszolgálónak!

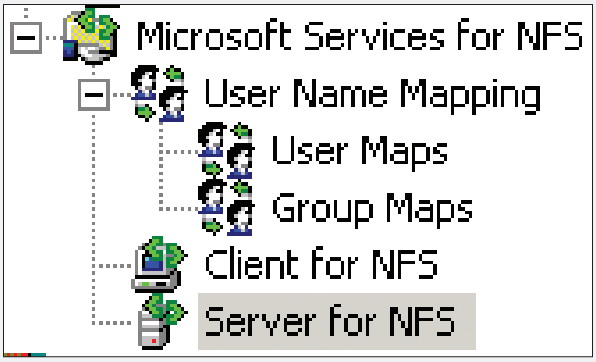
Nagyon fontos, hogy ez a szolgáltatás nem teljes SFU, azaz a Services for Unix, hiányzik ugyanis belőle a gateway-funkció! Miről is van szó? Ha a teljes SFU-t telepítjük egy

átlag Windows-kiszolgálóra, akkor három főbb kapcsolattípussal találkozhatunk. Ezek a Client for NFS, a Server for NFS és a Gateway for NFS. Az első lehetőséget ad egy szerint leginkább diagnosztikai célból van jelen a „de szeretnék adatterületeket látni az NFS-kiszolgálón" kívánságot kielégíteni tudó ***Client for NFS*** komponens a maga parancs-



6. ábra

Unix/Linux-kiszolgálóhoz kliensként való csatlakozásra, a második – itt a legfontosabb – esetben a mi Windows-szerverünk NFS-kiszolgálóként (is!) funkcionál, a harmadikban pedig a Windows-szerver által az NFS-kiszolgálón látható adatterületeket az SMB-kliensek (Windows-kliensek) számára teszi elérhetővé úgy, hogy azokon Unix/Linux-specifikus komponens telepítésére nem kerül sor. A Storage Server NAS és ezáltal fájlkiszolgáló



7. ábra

szerepkörével így elsősorban a „de szeretném kiszolgálni a Windows-kliensek mellett az esetleges NSF-klienseket is" – ***Server for NFS*** – képesség egyeztethető össze. Érzésem sori ***mount*** utasításával együtt, amitől rögtön egy másik világban érezhetjük magunkat. A beállítások között a „User és Group Mapping" az első, hiszen a Unix rendszerében autentikált felhasználókat a saját környezetünkben is reprezentálnunk kell valahogy. Hja, Kerberos Realm Trust kapcsolat híján ez kézenfekvő megoldásnak látszik. Ezután következhet az NFS-megosztások kialakítása, a jogosultságok definiálása, majd végül a teszt: el tudják érni az NFS-kliensek a számukra közzétett adatterületeket? El bizony!

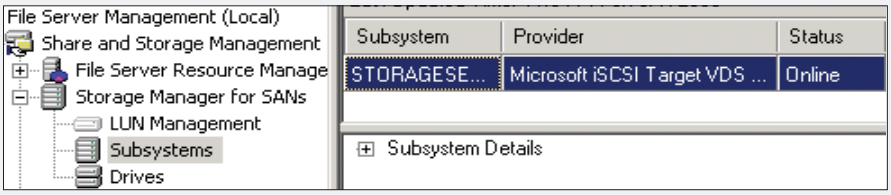
Storage Manager for SANs – egységesített storage-kezelőfelület

VDS, azaz ***Virtual Disk Service***. Mire jó ez nekünk? Vizsgáljuk meg a szerverünket! Van benne RAID-vezérlő? Nincs? Csatlakozik közvetlenül a Storage Area Networkbe Fibre Channel- vagy iSCSI-eszközökkel? Nem? Ebben az esetben maradjunk a jól megszokott Disk Managementnél, azt a helyi 1-2 diszket biztonsággal elkezelgethetjük vele. Viszont egy olyan környezetben, ahol a tárhelyre, sőt annak teljesítményére és hibatűrésére komoly hangsúlyt fektetünk, hamarosan azt vesszük észre, hogy szép lassan kezdenek elszaporodni a fent vázolt komponensek: RAID-vezérlők és SAN elérhetőségű diszk-alrendszerek. Sok esetben ezek több, különböző gyártó dobozai és megoldásai, mindnek sajátos kezelőfelülete és kezelési módszertana van/lehet. A VDS egy olyan szabványosított illesztőfelület, amelyik megadja a lehetőséget a külső storage-gyártóknak arra, hogy a portékájuk menedzsmentjét beilleszthessék egy, a Windowsban egységes rendszer alá. A gyártónak „nincs más dolga", mint az adott storage eszköz mellé csomagolni egy úgynevezett ***VDS Providert.*** Ezt telepítjük a rendszerünkbe a VDS alá – és máris működőképes a ***Storage Manager for SANs*** *(8. ábra)*! Egy időben akár több VDS Provider is lehet a rendszerünkben, így ugyanarról a felületről hozhatunk létre mondjuk Logikai Unitot egy iSCSI Target mögött, vagy növelhetjük menet közben a méretét egy virtuális diszknek a Fibre Channel Arrayben. Fontos megjegyezni, hogy a szerverre telepített VDS Provider segítségével nemcsak azokat az objektumokat változtathatjuk, amelyek a mi szerverünk számára lettek „prezentálva" a SAN-ban, hanem az adott storage-ban az összeset! Lehet, hogy a felület barátságos és egységes, de megnyitásával már a SAN-adminisztrátorának felelőssége is a mi vállunkat nyomja – csak *óvato**SAN***! A VDS Provider megszerzése sok esetben némi kutatómunkát igényel. A gyártói weboldalak, dokumentációk és az adott diszkalrendszer mellé adott CD-k, DVD-k sokat segítenek!

Windows Unified Data Storage Server

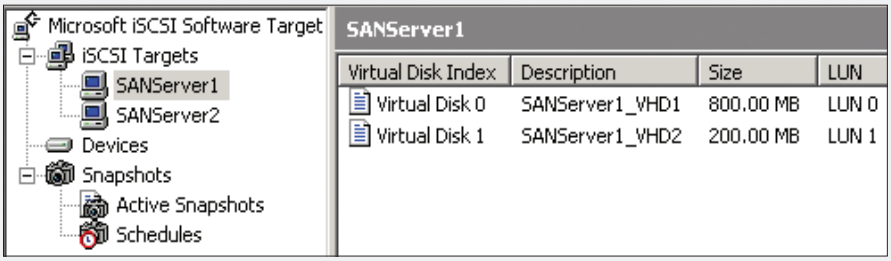
Unified? Mit és mivel egyesít a Storage Servernek ez a verziója? Először is nem külön verzió. De hát mégis az, hiszen a neve... Hogy is van ez? Öntsünk tiszta vizet a pohárba! Bármelyik Storage Server-kiadásból (lásd *1. táblázat*) készíthetünk Unified Data Storage Servert úgy, hogy megvesszük és feltelepítjük a „Unified Add-on" kiegészítést. Aha... és abban mi van? Mi is? Hát az ***iSCSI Target!*** Az iSCSI Target szerepkör viszont már egy másik kategória, azaz nem NAS, hanem ***SAN!*** Szinte az összes eddig felsorolt szolgáltatás a szerverünk fájlkiszolgáló-képességét dicsérte.

Lássuk csak... A fájlszerverhez, azaz a NAS-hoz kik csatlakoznak? Jellemzően a felhasználók! Hogyan, milyen módon? Hálózaton. Hálózaton, ami a modell szerint ***network kapcsolat.*** Host kapcsolódik hosthoz, Linux



8. ábra

Windowshoz, applikáció applikációhoz... az adat elérése pedig jellemzően ***fájlszintű.*** Hogy működik mindez a SAN-ban? SCSI-adapter SCSI Storage-hoz, host egy szalagos egységhez, host egy diszkhez. Jellemzően a szerver az adattárhoz. Ez pedig maga a ***csatornakommunikáció!*** Az adat elérése pedig ebben az esetben *e*Akkor mit is egyesít



9. ábra

a Windows Unified Data Storage Server, amelyik abban más, mint a többi Storage Server, hogy telepítünk rá iSCSI Target komponenst? Igen, az adatok fájl- és blokkszintű elérhetőségét! A NAS és SAN kiszolgáló-szerepköröket. Ezt jól megfejtettük! Vizsgáljuk meg, hogy állja meg a helyét szerverünk iSCSI Targetként!

Microsoft iSCSI Target

Tisztázzunk alapfogalmakat az iSCSI kommunikáció rendszerében. Melyik a kliens? Az, amelyen úgynevezett ***iSCSI Initiator*** komponens van. Ha Windows Server 2008 kiszolgálónk van, akkor az Initiator az operációs rendszer része. Más – régebbi – kiszolgálók esetén nyugodtan telepíthetünk ilyet, ingyenesen letölthető a Microsoft oldalairól. Hova csatlakozik egy Initiator? Először is jól nevelt módon regisztrálja magát az iSNS kiszolgáló adatbázisába (Windows 2008-ban az operációs rendszer része, más gépekre ingyenesen letölthető), majd felkeresi a rendszer gazdája által benne beállított ***iSCSI Target Portalt*** *(9. ábra).* A Portal jelen esetben nem más, mint a mi Unified Data Storage Serverünk, azaz a rajta futó iSCSI-szolgáltatás. Mit láthat az Initiator a Target Portal mögött? Kis szerencsével ***Targetet.*** A Target nem más, mint egy olyan logikai objektum, amely az iSCSI-kiszolgáló számára leírja, hogy mely kérelmezők mely logikai (virtuális) diszkekhez férhetnek hozzá. Így magától értetődő, hogy az iSCSI SAN-ban minden kérelmező szerver számára a kiszolgálóban létrehozzunk egy targetet. A target mögé csatlakoztatott virtuális diszkeket fogja a kérelmező – iSCSI csatornakapcsolaton – saját diszkként a Disk Managementben megjeleníteni. Az, hogy minden kérelmező egy saját targetet lát, és természetesen (csak) a target mögé csatolt diszkeket, nem más, mint a SAN-okban olyan jól bevált prezentáció (selective storage presentation, LUN masking) módszerének alkalmazása. A targethez való csatlakozáskor a kiszolgáló azonosítja a kérelmezőt (lehetőségek: IP, IQN, FQDN, MAC), autentikációt is beállíthatunk (CHAP), és ha minden stimmel, a virtuális diszkekhez való hozzáférést biztosító zöld lámpa kigyullad. Végezetül tapogassuk meg a kérelmezők felé prezentált virtuális diszkeket! Mik ezek? Ugyanolyan VHD állományok, mint amilyenek a virtualizációs rendszerekben is széles körben elterjedtek a Virtual PC-től a Hyper-V-ig. Az MS iSCSI kiszolgáló ezekből snapshoto(ka)t is képes készíteni, illetve helyi csatolásra (mount) is lehetőség van. Támogatja a targetekhez való redundáns hozzáférést is, ha a kérelmezőoldalon feltelepítjük és konfiguráljuk a ***MultiPath I/O*** (MPIO) komponenst. A hibatűrő SAN-kapcsolatok kialakítása ebbe az irományba terjedelmi okokból nem fér bele, sőt egy újabb cikkért kiált. Kedvcsinálóként viszont – remélem – megállja a helyét.

Storage Server-szerepkörök

A Storage Servert a felé irányuló kapcsolatok tekintetében különböző kategóriákba sorolják. Zárószóként nézzük át ezeket! Jellemzően háromféle logikai szerepet vállalhat. Az első a ***NAS-kiszolgáló***. Ebben az esetben kiszolgálónk a saját diszkjein fellelhető adatokból juttat jószívűen a klienseknek – akárcsak egy fájlszerver. Sokszor elhangzik azonban a ***SAN Gateway*** fogalom is a mágusok szájából, amitől ne jöjjünk zavarba: ez ugyanaz, mint a NAS, csak ebben az esetben a Storage Server nem a saját helyi diszkjein, hanem az általa a SAN-kapcsolatok valamelyikén elérhető diszke(ke)n tárolja a felhasználói állományokat. A harmadik fogalom a ***SAN Target,*** itt a Windows Storage Server – a Unified Add-On kiegészítéssel – blokkszintű diszkelérést biztosít, jellemzően a többi kiszolgáló részére.

Szeretnék Storage Servert! Hogyan jutok hozzá?

Az a módszer, hogy földi halandóként bebaklagunk az első szoftverdisztribúcióba, és megvillantjuk dombornyomott bankkártyánkat – sajnos nem működik. Két járható út van. Az egyik, hogy neves gyártók előre telepített „NAS Appliance" kategóriájú célgépét megvásároljuk. Többségük még ki is egészíti a szolgáltatások körét némi adatmentő-, adatreplikációs- vagy felügyeleti szoftverrel. A másik módszer, hogy mi magunk rakjuk össze a NAS-kiszolgálót. Kössünk szerződést a Microsoft Embedded termékeket forgalmazó csatorna képviselőjével! Így az általunk gyártott (összerakott) hardverre telepíthetjük a Storage Servert, és tovább értékesíthetjük a végfelhasználóknak.

Székács András
(andras@edupro.hu)
MCSE, MCT, MCTS
Számalk Zrt.

IAG 2007

Távoli elérés egy igazi nagyvaddal, SSL VPN alapon.

Rendszereink távoli elérésének hagyományai messzire nyúlnak vissza az időben – és ez a múlt igencsak küzdelmesnek bizonyult, nem vitás. Ennek ellenére a távoli elérés vonzó lehetőség az informatika világában, amit ha józan ésszel nézek, akkor nem is értek, hiszen például a vasutasok nem viszik haza a munkát, mi meg szinte nem is bírjuk ki anélkül, hogy otthon, illetve bárhol máshol ne férjünk hozzá például a leveleinkhez vagy más erőforrásainkhoz.

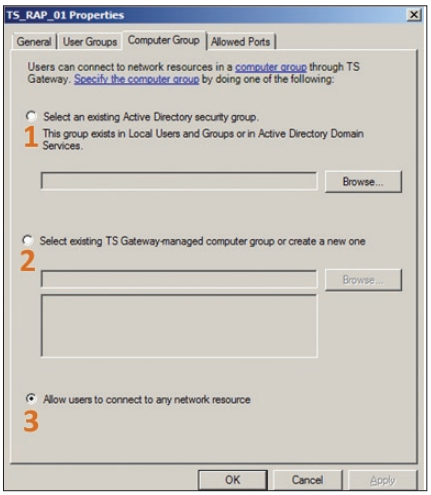
Kezdetben, az analóg modemes korszakban nem kevés nehézség állt előttünk, iszonyú drága volt és borzalmasan lassú (amikor én kezdtem, az első modemem 9,6 K-s volt, de én viszonylag későn kezdtem, és még nem számítok nyugdíjas korúnak), hardveresen erősen korlátozott volt a kapcsolatok száma, és a maximum szolgáltatás, amit komoly kompromisszumok nélkül használhattunk, az a postaládánk elérése volt (persze csak POP3, vagy inkább IMAP, semmi extra), meg még esetleg a telnet, no meg az automatikus visszahívás lehetősége, és az ezzel járó széttárt kezek a céges telefonszámlák érkezésekor. Aztán jött az ISDN, és ekkor a kapcsolódás fő jellemzője immár nem az egyperces, idegborzoló sípolás volt, a sebesség pedig nőtt, ha ügyesek voltunk a multilink segítségével akár az akkor csillagászati mértékűnek számító 128 K-s sebességet is összekalapálhattuk. Így aztán már több belső szolgáltatás használata is belefért, sőt, az ISDN-központok révén akár a felhasználóknak is bátrabban ajánlgathattuk a távoli elérés lehetőségét, anélkül, hogy ez mély fájdalmat okozott volna nekünk. De nagyjából ekkor két új probléma is felmerült:

1. Kliensoldalon bonyolult egy kapcsolatot létrehozni a megfelelő szaktudástól el nem vakítva.
2. Ekkor már az internet nem az volt, ami korábban: enyhén szólva is erősen ügyelni kellett a biztonságra.

Az első pont nehéz ügy, de a másodikra megoldás volt (és ma is az) a klasszikus VPN használata (persze a távoli elérés tiltása is jó módszer, de talán nem elég kreatív), amely új dimenziókat nyitott. A távoli ügyfél számítógépe saját internetkapcsolat birtokában, teljes körű résztvevője lehetett a belső hálózatnak, gyakorlatilag a megszokott sávszélességen kívül minden lehetőséget megkaphatott, amit a benti hálózatban élvezhetett – és ezt bárhonnan, ahol volt internetelérés. Ezzel aztán át is estünk a ló túlsó oldalára, hiszen a vezetékes hálózatban, a tartományi rendszerben hatékonyan működő Csoportházirend, a kötelező frissítések letöltése és telepítése (például WSUS), az esetleges belső központi antivírusrendszer adatbázisa és sok más korlátozó, rendet teremtő megoldás alól a VPN-ügyfelek mentesültek, ráadásul tényleg, miért is kell elméletileg mindent elérnie kívülről egy átlagos felhasználónak? Nem kell, de nem volt rá életképes megoldás, még a Windows Server 2003-ban bevezetett, majd az ISA 2004-ben „kifényezett" karanténszolgáltatás sem volt az. Ma már más a helyzet, az NLA miatt a távoli Vistánál lehetséges dinamikusan a Csoportházirendet alkalmazni, a Windows Server 2008-ban debütáló Network Access Protection mindenféle közegben (DHCP, IPSec, VPN, RDP, 802.1x) megállja a helyét, sőt a teljesen újnak számító Terminal Server Gatewayen beállított házirendek segítségével megszabhatjuk, hogy ki, mely gépekről mely belső gépeket érheti el.

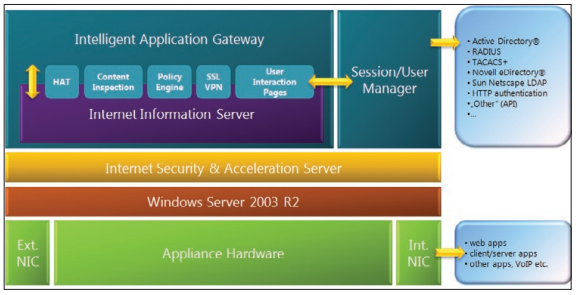
Aztán a sávszélesség egyre csak nőtt és nőtt, immár az intranetportálok, a „gazdag" szolgáltatást nyújtó belső kiszolgálók elérése sem volt lehetetlen, és a tűzfalbarát elv mentén elkezdtek megjelenni a HTTPS porton működő vagy abba beágyazott szolgáltatások (xxx over HTTPS), de a megmaradt régiek mellett újabb problémák is jelentkeztek:

- Még mindig bonyolult, sőt még bonyolultabb beállítani a távoli elérést, a VPN mellett gondoljunk az Outlook RPC over HTTPS-re vagy a TSG kliensoldali beállítására, vagy akár csak arra, hogy a felhasználónak minden egyes alkalommal először kapcsolódnia kell a VPN-ikon segítségével.
- Még az olyan komplex VPN-szerverekkel, mint az ISA 2006 sem tudunk igazán lényegesen változtatni azon, hogy a külső ügyfél több fizikai elérést kapjon a belső hálózatban, mint amennyi kellene.
- Továbbra sem biztonságos az internet, és bár a VPN és a HTTPS ezt a problémát többnyire áthidalja, de sajnos nem lehet mindenhonnan a szokásos VPN-protokollokat használni (erre viszont passzolhat a szintén a Windows Server 2008-ban megjelent SSTP VPN típus – pongyolán fogalmazva: „VPN over HTTPS" –, de csak erre).
- A PPTP VPN-nél lényegesen többet nyújtó megoldás az L2TP/IPSec, de ehhez az összes köztes résztvevőnek támogatnia kell ezt, és például a NAT vs. integritás problémára megoldást nyújtó NAT-Traversalt, beleértve az OS-eket is (az XPSP2 előtt ez nem volt jellemző).
- Nemcsak a HTTP/S-n elérhető szolgáltatásokat szeretnénk használni, és persze nem akarunk és nem is lehet x darab kliensprogramot telepíteni minden egyes távoli



TSG Resource Authorization Policies, azaz mely gépeket érhetjük el a hálózatban? (1. AD csoport alapján, 2. TSG csoport alapján, 3. bármelyiket)

elérésre használt gépre, hogy aztán VPN-en kapcsolatba kerülhessen a belső, nem HTTP alapon működő kiszolgálókkal.



Az IAG 2007 komponensei és felépítménye

Nos, egy ideális (azért csak óvatosan ezzel a szóval) megoldásnak tehát minimum a következőket kellene nyújtania ezen a területen:

- Problémamentes, biztonságos kapcsolódás bármilyen tűzfalon és hálózati eszközön keresztül.
- A szolgáltatások ügyfélbarát megjelenése, mindenféle kliensoldali beállítás nélkül.
- Univerzális (azaz nem csak webes) alkalmazáspublikálási lehetőség az üzemeltetők számára.

A lehetséges megoldás már egy ideje itt van köztünk, neve kb. annyi van, mint Ságvári Endrének, de a legjobb valószínűleg az SSL VPN, de még ennél is fontosabb, hogy már kilépett a gyerekcipőből, azaz most már tényleg ideje megismerkednünk vele. Kezdetben viszont az eszközeink (akár hardveres, akár szoftveres területen) kevésbé voltak sokoldalúak, így az ismerkedést kezdjük meg első körben az SSL VPN alapjául szolgáló (vagy ezekhez nagyon hasonló) technikai megoldások választékával:

Reverse proxy megoldások a webes alkalmazások számára. A legjobban ezt az ISA-kiszolgálók webpublikáló szabályai mutatják, amelyeknek semmi közük sincs a VPN-hez, van viszont az ezen a területen alapfeltételnek számító előzetes hitelesítéshez, illetve az URL-ek átirányításához, ergo, a különböző web filterek igencsak hozzájárulnak a biztonságossághoz, sokkal jobban, mint egy átlagos reverse NAT-megoldás.

Protocol tunneling. Egy SSL-kapcsolaton belüli alkalmazásprotokoll-kapcsolat. Mindenki ismeri a jó példát: Outlook RPC/HTTP-kliens RPC/MAPI-hívásokat gyömöszöl bele egy SSL-lel védett alagútba, a másik oldalon pedig ezt egy RPC/HTTP-proxy fogadja, majd szépen továbbítja az Exchange-kiszolgálónak.

Socket / port forwarding. A kliensoldali alkalmazás figyel, majd a spéci portra/socketre igyekvő forgalmat (például TCP 25/110) átirányítja egy SSL-linken keresztül az SSL VPN-átjáróhoz, amely aztán szépen kibontja az alagútból, és elküldi a levelezőkiszolgálónak.

Ezek esetleges kombinációin kívül ma már a valóságban sokkal több mindent is elvárhatunk egy komplex SSL VPN-eszköztől. Konkrétan a következőkre gondoljunk:

- **Tunneling.** Minden SSL VPN-átjárónak képesnek kell lennie arra, hogy a webes és a nem-webes alkalmazások összes forgalmát bele kell tudni pakolni a HTTPS sessionbe.
- **Kliensoldali biztonság.** Akármilyen állapotú kliens nem érheti el a publikált szolgáltatásokat, épp ezért működnie kell egy ún. endpoint detectionnek, amely a feltételeink alapján képzett biztonsági házirend pontjaival szembesíti a klienst, és ha nem felel meg, nem kaphat hozzáférést (lásd VPN-karantén, de inkább a NAP vagy inkább egy, a kettő közötti megoldás, ám finomabb ellenőrzési lehetőségekkel).
- **Előzetes hitelesítés.** Nincs közvetlen hitelesítés a belső szerverekkel, az SSL VPN-átjáró kéri be a hitelesítési adatokat, elbandukol a belső szerverhez, és továbbadja ezt az információt, majd bekéri az adott erőforráshoz szükséges jogokat is, és ezzel vándorol vissza a külső ügyfélhez – akkor is, ha nem egy webes protokollról van szó. Ráadásul kismillió névtérből tudnia kell ezt.
- **Jogosultság-ellenőrzés.** Magának az SSL VPN-eszköznek is képesnek kell lennie előzetesen engedélyezni vagy megtagadni a hozzáférést a hostolt alkalmazásokhoz, akkor is, ha az egy másik kiszolgáló segítségével érhető el.

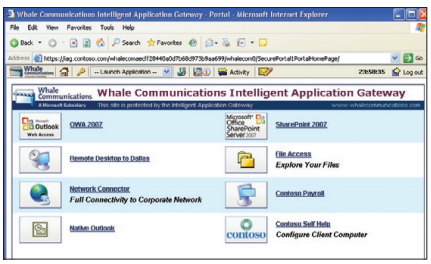
Maga az eszköz, amin az IAG fut

- **Portál.** Az egyszeri, űrlapos belépés után a felhasználó egy könnyen kezelhető weblapot lát, rajta az összes alkalmazással és hozzáférési lehetőséggel, amelyeket elérhet, de tényleg csak azokkal, amelyeket elérhet. Így a felhasználónak csak ezt az egy URL-t kell megjegyeznie, és a szintén kötelező SSO miatt (Single-Sign On) elég egyszer (interaktívan) belépnie. Ráadásul követelmény, hogy ezt a portált a lehető legnagyobb mértékben lehessen testre szabni az üzemeltetők által.

Minimum Requirements	
Operating System	**Supported Browsers**
Windows XP, Windows Server 2003	Internet Explorer 6, Internet Explorer 7, Netscape Navigator 7.1.x, Netscape Navigator 7.2.x; Mozilla 1.7.x; Firefox 1.0.x
Windows Vista	Microsoft Internet Explorer 7, Firefox 2.0
Windows Mobile 2003 for Pocket PC	Pocket Internet Explorer
Mac OS X	Safari 1.2.4, Safari 1.3, and Safari 2.0, Netscape Navigator 7.1.x, Netscape Navigator 7.2.x; Mozilla 1.7.x; Firefox 1.0.x; Camino 0.83
Linux (Red Hat, SUSE, Debian)	Netscape Navigator 7.1.x, Netscape Navigator 7.2.x; Mozilla 1.7.x; Firefox 1.0.x

A lista igazán nem mondható szűknek…

- **Szűrés az alkalmazásrétegben.** Az egy igazán előremutató elképzelés, hogy a sokféle hálózati forgalmat és protokollt egy tűzfalbaráttal helyettesítsük (legalább is a szervezetünk belépési pontjáig), de ha logikusan belegondolunk, akkor ez azzal is jár, hogy minden más káros vagy nem engedélyezhető alkalmazás – spyware, trójai stb. – is erre fog rákapni. Ebben az esetben a dilemma nagy: vagy engedélyezzük szűrés nélkül a HTTP/S-portokat, és akkor a webes MSN-t, ICQ-t, a torrent-alkalmazásokat és sorolhatnám a további példákat azokra, amelyek vagy a népszerű portokon, vagy a HTTP/S folyamatba beágyazódva működnek, vagy ha nem, akkor bentről sincs netezés. Bármennyire is szimpatikus az utóbbi, ezt nem tehetjük meg, ergo, gondoskodni kell egy komoly HTTP-filterről, illetve emellett természete-



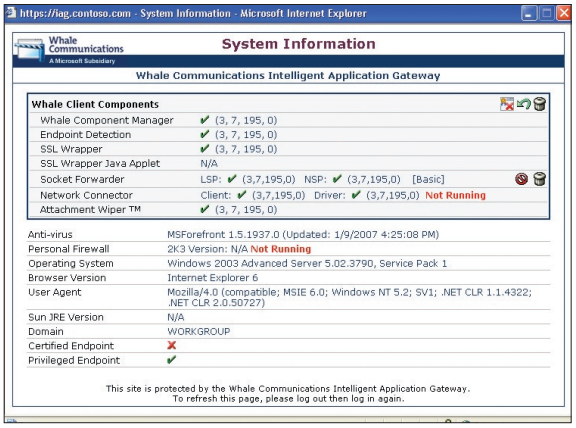
Az ikonokra kattintva indulnak az alkalmazások

sen az SMTP-, POP3-, RPC-, IMAP4- stb. portok megfelelő szűréséről is.

Szóval ma már szerencsére sokat várhatunk el egy korrekt SSL VPN-eszköztől, és mivel ez egy jó ideje létező technológia, már számos gyártó (Juniper, Celestix, F5, Checkpoint, Cisco, Citrix, SonicWall stb.) készít is ilyen eszközöket, pontosabban önmagában a szoftvert (ez a ritkább), vagy együtt a hardverrel, amelyet ilyenkor „appliance"-nek hívunk, és ez tipikusan egy-egy rackbe szerelhető, célhardvert jelent, a speciális szoftverrel, extrákkal és persze egy megfelelő OS-sel együtt.

(Megjegyzés. Az „appliance" kifejezés nem ismeretlen az ISA-rendszergazdák számára, ISA 2004/2006 appliance-ből is van számos, a HP-tól a Celestixig jó néhány gyártó készít szenzációs, Windows Server 2003 alapon működő, az ISA 2006 speciális, ún. Workgroup Editionjét tartalmazó eszközt, szokás szerint spéci filterekkel, bővítményekkel – például Websense/SurfControl termékek, vírusirtók stb. Jómagam is szeretgetek egy pár hónapja egy Celestix MSA2000i-t, igazán nagy élmény.)

Visszatérve az eredeti témához, a Microsoft és a Whale Communications együttműködéséből, 2007 tavasza óta létezik egy Windows Server 2003 R2 alapon működő, Internet Application Gateway 2007 nevet viselő SSL VPN-eszköz, amely – mint az architektúrából látható – az OS-en kívül több ismerőst is tartalmaz, például az IIS-t és az ISA-kiszolgálót.



Részletes infók a kliensről

Ez a termék kifejezetten szintén csak az említett „appliance" formában kapható, azaz csak a hardverrel egybeépítve szállítják (így nem is egy Microsoft logóval ellátott dobozt kapunk, például itt, az asztalomon egy Celestix WSA 3000-es hever, a fotót lásd a 34. oldali képen), az OEM-cég által előtelepítve. Az ára többnyire igencsak borsos, ám ha átgondoljuk, hogy mi mindent tartalmaz, és megismerjük alaposan, akkor már nem is tűnik majd olyan soknak.

Így néz ki egy SSL VPN-eszköz, amelyben belül egy IAG 2007 munkálkodik (felhívnám a figyelmet a jobb oldalon látható tekerőgombra – jog –, ezzel első lépésként a TCP/IP-konfigurációt fogjuk majd beállítani).

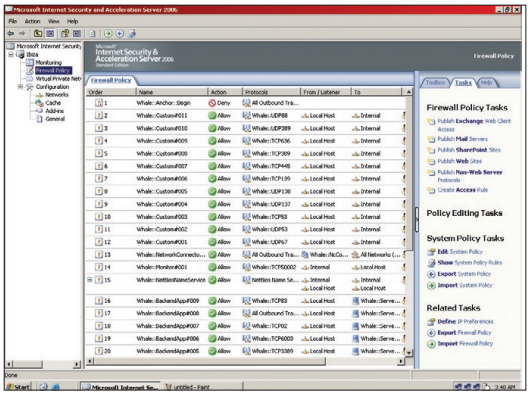
Általában a használatbavétel első lépése az OS és az alkalmazások telepítése, de mivel ezzel itt nem kell foglalkozunk, helyette tekintsük meg inkább a támogatott kliensek (böngészők) listáját.

A portálok

Ha szeretnénk látni már egy portált, akkor csak a böngészőre van szükségünk, meg a portál URL-jére, ezek birtokában el is juthatunk az adott kezdőlapra, amelyen egy űrlap fogad majd bennünket. Bejelentkezünk, majd a kliens megbízhatóságától függően (például külső gép, belső gép) különböző listát kapunk az elérhető alkalmazásokról és szolgáltatásokról. Sok más opció mellett a rendszerben eltölthető maximális időtartam is része lehet ennek a beállításnak, a jobb felső sarokban látszik, hogy mivel ez egy belső/védett kliens, ezért az alap intervallumot 24 órára állítottam.

A böngészőablak tartogat még jó pár kellemes szolgáltatást a menüsorban – a nem admin felhasználók számára is. Többek között lehetséges a jelszóváltás, az esetleges – birtokunkban lévő – további jogosultság beadagolása, az egyéb, nem ezen a kiszolgálón publikált alkalmazások tallózása, a nagyon alapos rendszerinformációk listázása, a portálaktivitás részleteinek megtekintése és az e-mail-küldés az adminnak.

A portál mint objektum nagyon részletesen konfigurálható, könnyedén testre szabható, és egyszerűen létrehozható a különböző igények alapján. Az első portált a szervezet maximálisan meglévő szolgáltatásai alapján állítottam be a belső felhasználók számára, míg a másodikat, jóval szigorúbban, kevesebb erőforrást megengedve egy elképzelt partnercég számára (természetesen különböző URL-lel rendelkeznek).

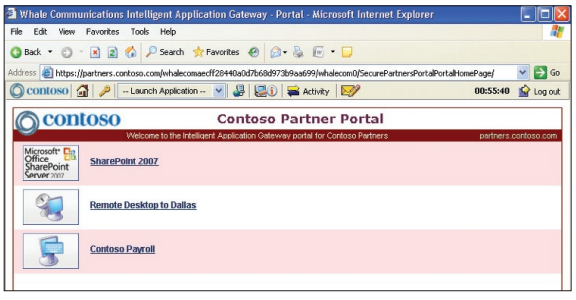


Előregyártott szabályok az ISA-ban

Az ISA Server

Ha viszont az ismerkedést nem a portállal, hanem direktben az IAG gépen kezdjük, akkor a kíváncsiságtól vezérelve megtekinthetjük rögtön az IAG mögött álló ISA Servert. A faszerkezetben sok meglepetés nem lesz, ugyanis egy teljesen szimpla (jelen esetben egy Standard) ISA 2006 MMC konzolt láthatunk, egyetlen igazi különbséggel, amely a tűzfalszabályoknál jelentkezik, de ott viszont nagyon.

A képen számos tűzfalszabály látható, azért, mert ez már egy működő IAG 2007 gép, viszont magában az ISA-ban manuálisan nem hozunk létre szabályokat, ezt az IAG a



Szűkített lehetőségek a partnerportálon

konfigurálása közben automatikusan megteszi. Szóval sok esetben a szolgáltatások egy részénél valóban az ISA van a háttérben, de a beállítása automatikus.

IAG Configuration

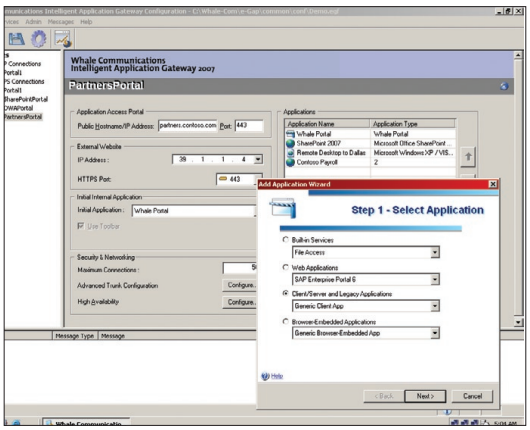
Ha már az IAG gépen vagyunk, nézzük meg az üzemeltető számára legfontosabbat, azaz azt az alkalmazást, amellyel az eddig ismertetett elvek alapján az összes publikációs, biztonsági, megjelenési, hitelesítési és minden egyéb más lényeges beállítást elvégezhetünk. Rettentően összetett eszközt kapunk a kezünkbe, amelynek menüszerkezetében eleinte simán eltévedhetünk, de ez csak azt bizonyítja, hogy az IAG 2007 nagyon sokat „tud".

Szummaként és a személyes tapasztalataim alapján bátran elmondhatom, hogy az IAG 2007 már egy rövid megismerkedés után is bizonyíthat. Aki ismeri az ISA-kiszolgálók, illetve például a Terminal Services Gateway, a TS Remote Apps, valamint a TS Web Access alkalmazás publikálási lehetőségeit, az szorozza meg ezt egy borzasztó nagy számmal, és akkor megkapja ennek a terméknek a használhatóságát és a komplexitását.

Slusszpoénként szeretném még elárulni azt is, hogy a Forefront család egyik elemeként készült termékvonal következő példánya a Unified Application Gateway (UAG), amely csak és kizárólag 64 bites Windows Server 2008-on fut majd. Már teszteljük – a 2009-es évben érkezik majd meg végleges formájában –, és a birtokomban lévő információk alapján már most is valószínűnek tartom, hogy lényegesen kívánatosabb és egyúttal könnyebben elérhető lesz egy nagyobb réteg számára, mint jelenleg az IAG 2007.

(Folytatjuk)

Gál Tamás
(v-tagal@microsoft.com)
Microsoft Magyarország



A partnerportálhoz épp egy speciális alkalmazást adunk hozzá

További információk

IAG 2007 Technical Resources
http://technet.microsoft.com/en-us/forefront/edgesecurity/bb687299.aspx
IAG 2007 portál
http://www.microsoft.com/forefront/edgesecurity/iag/default.mspx
Részletes klienstámogatási adatok:
http://www.microsoft.com/technet/forefront/edgesecurity/iag/system-requirements.mspx
Forefront Edge Security and Access Demonstration Toolkit
http://www.microsoft.com/forefront/edgesecurity/trial.mspx

IPv6 – A CALC.EXE RENESZÁNSZA

Ezt a cikket lehet, hogy nem nekem kellett volna megírnom, hanem valami hálózatos szakértő kollégának. Így egész biztosan nem lesz olyan mély – cserébe lehet, hogy a szerverüzemeltető szakemberek is érteni fogják.

Tehát IPv6. Azaz IP version 6. Gondolom, senkit sem fog meglepni, hogy most az IP version 4-et használjuk. Az 5-ös? Eltévedt valahol.

Ilyenkor a „meghalt a király, éljen a király” logikával a szerző nekiáll szidni a régit, hogy utána könnyebb legyen dicsérnie az újat. A lehetőség most is csábító, de próbáljunk meg ellenállni neki. Nem is olyan rossz ez az IPv4. Gondoljunk bele: már 1990 körül elkezdték kongatni fölötte a vészharangot. Hol volt akkor még a www? Pocok. Meg FTP. Ilyesmiket pötyögtek az emberek az akkor még nem létező böngészők helyett a parancssorba. De már megjósolták, hogy egyszer el fognak fogyni az IP-címek. Aztán most, 2008-ban is csak mondogatjuk, hogy igen, tényleg el fognak fogyni az IP-címek. Valamikor. De akkor biztosan.

Hogyan sikerült eddig megúsznunk a katasztrófát? CIDR. Meg NAT. Ők a hősök.

A CIDR (Classless Inter-Domain Routing, barátainak csak cider) egy címkiosztási nonszenszt volt hivatott orvosolni. Ezt a nonszenszt összefoglalóan címosztályoknak hívták. Egészen biztos vagyok benne, hogy valamikor mindenki szorgalmasan biflázta, hogy aszongya:

Osztály	Bal oldali bitek	Hálózatokat leíró bitek száma	Hálózatok száma	Hostokat leíró bitek száma	Hostok száma
A	0	8	127	24	16 777 214
B	10	16	16 384	16	65 534
C	110	24	2 097 152	8	254
D (multicast)	1110				
E (foglalt)	1111				

Így nézett ki a classful címkiosztás. A 32 bites IP-címből A osztály esetén az első 8 bit jelentette a hálózat címét, a többi 24 pedig a node-ot azonosította. Nyilván a matekozás adta lehetőségekből a gyakorlatban le kellett vonni valamennyit, emiatt maradnak el a maximális számok az elméleti maximálistól. (Például a B kategóriánál elméletileg 65 536 hálózat jöhetett volna szóba, de a kötelező ‘10’ indítás miatt kiestek a ‘00’, ‘11’, ‘01’ lehetőségek, azaz megnegyedeltük a plafont. A node-ok száma meg azért csökkent kettővel, mert kiesett alul a network-azonosító, felül meg a broadcast-cím.)

Nos jött egy amerikai egyetem, amelyik jókor volt jó helyen, igényelt magának egy B kategóriás címet. Megkapta. Nehogy már sínylődjenek a diákok a campusban. Jött Kovács János, megigényelte a 169.254.0.0/16 B osztályú címtartományt. Megkapta. És így teltek-múltak a napok, fogytak a címek.

Néhány későn ébredő országnak már csak C kategóriás cím jutott. Igaz, abból több is, na de milyen már, hogy egy ország 254 címenként routolgasson? Miközben az A és B kategóriás címeknél meg ott volt egy csomó használaton kívüli IP-cím. Egészen vad dolgok történtek. De azért látszott, hogy ez így nem tartható sokáig. 1993-ban színpadra is lépett CIDR – és rendet vágott. Két fontos dolgot tett lehetővé:

- Megengedte, hogy szanaszét subneteljék a nagy hálózatokat.
- Megengedte, hogy összekapcsoljuk a kis hálózatokat.

Hogyan? Elkezdtük tologatni a biteket a subnet-maszkban. Az első esetben jobbra, a második esetben balra.

Nézzünk egy példát. Kaptunk két C kategóriás címtartományt:

- 192.168.0.0/24
- 192.168.1.0/24

Ugyanezek binárisan:

```
11000000 10101000 00000000 00000000 – IP-cím
11111111 11111111 11111111 00000000 – Subnet-maszk
11000000 10101000 00000001 00000000 – IP-cím
11111111 11111111 11111111 00000000 – Subnet-maszk
```

Legyünk bátrak, toljuk el egy bittel balra a subnet-maszk határát.

```
11000000 10101000 00000000 00000000 – IP-cím
11111111 11111111 11111110 00000000 – Subnet-maszk
11000000 10101000 00000001 00000000 – IP-cím
11111111 11111111 11111110 00000000 – Subnet-maszk
```

Látjuk, mi történt? A piros karakterek jelentik az egy hálózatba tartozó node-okat. Az eltolással elértük azt, hogy a két hálózatunk immár egynek számít. Úgy is jelöljük, hogy 192.168.0.0/23.

Nézzünk egy másik példát. Kapunk egy B kategóriás címet. Illetve, mit is beszélek? Már nincs sehol sem B kategóriás címtartomány. Pontosabban van, de erről majd később.

Ez a tartományunk: 172.18.0.0/16. Igen ám, de van 50 telephelyünk, szanaszét az országban, mindenhol 100 node-dal. Hülyeség lenne ezt egy hálózatnak tekinteni, de nem rendelhetek meg 50 darab B kategóriás címtartományt sem.

Nézzük, milyen pályán játszunk:

```
10101100 00010010 00000000 00000000 – IP-cím
11111111 11111111 00000000 00000000 – Subnet-maszk
```

Mi lenne, ha most jobbra csúsztatnám el a subnet-maszk határát?

```
10101100 00010010 00000000 00000000 – IP-cím
11111111 11111111 10000000 00000000 – Subnet-maszk
10101100 00010010 10000000 00000000 – IP-cím
11111111 11111111 10000000 00000000 – Subnet-maszk
```

Rögtön két, különböző hálózatot kaptam!

- 172.18.0.0/17
- 172.18.128.0/17

És még nincs vége, a subnet-maszk határát egész bátran tologathatom még jobbra, még több, még apróbb hálózatokra tagolva a címtartományomat. Nyilván ez a másik irányra is érvényes – csak akkor apró hálózatokból rakok össze nagyobbakat.

Jó, és akkor mit csinált a NAT?

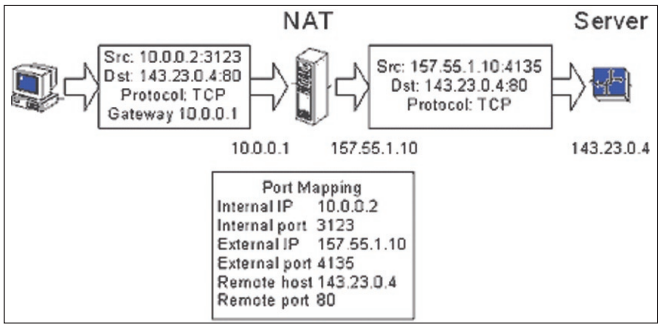
Nos, elbújtatta a hálózatokat.

A címkiosztók idejében kapcsoltak, és minden címosztályban kijelöltek egy-egy tartományt. Azt mondták, az ebből a tartományból jövő címek értelmezetlenek lesznek az interneten. Nem léteznek. Mint egy véletlenül kicsúszott büfi egy üzleti tárgyaláson: mindenki hallotta, de senki sem reagál rá.

Ezekről van szó:

Címtartomány	Alsó határ	Felső határ
A	10.0.0.0	10.255.255.255
B	172.16.0.0	172.31.0.0
C	192.168.0.0	192.168.255.255

Legyen egy vállalatunk, mondjuk, 100 000 számítógéppel. Tételezzük fel, hogy egy darab router köt össze minket az internetszolgáltatónkkal. A router belső felétől már mi vagyunk a májerek, azt csinálunk, amit akarunk. Mit akarunk csinálni? Hát, például használjuk a 10.0.0.0/8 hálózatot. Ez teljesen védett címtartomány, bárki is használja rajtunk kívül, a neten nem fog megjelenni. Ha jól csináljuk, akkor a miénk sem. Jó, jó... de akkor mi fog megjelenni? Hát az az egy IP-cím, amelyet a szolgáltatónk adott, és mi a router külső lábához illesztettünk. Mind a 100 000 alkalmazott azon az egy IP-címen keresztül fogja tolni az iwiw-et.



Végtelen mennyiségű hálózat egymás mögött

Részletesen ebbe most nem mennék bele, egy jó ábra száz szónál is szebben beszél.

Nos, nagyjából látjuk, hogyan is állunk. Az internet köszöni szépen, működik. A NAT segítségével gyakorlatilag végtelen mennyiségű hálózatot tudunk egymás mögé bújtatni.

Mi akkor a probléma?

Például a NAT:

- Azért csak erőforrás-igényes. Vessünk ismét egy pillantást az ábrára. Láthatjuk, egy kapcsolathoz a routernek el kellett raktároznia a port mapping értékeit. Mi van, ha több ezer kapcsolat pörög át a routeren percenként? Füst. Meg csökkentett felhasználói élmény.
- Mi van akkor, ha erőforrásokat akarunk publikálni az internet felé? Mondjuk, két darab webszervert? A routernek csak egy darab 80-as portja lesz. A másik webszervernek már bele kell törődnie a nem szabályos portba, vagy trükközni kell.
- Titkosítás. Gondoljunk csak az L2TP VPN-re. Ha a router belepiszkál a csomagba, akkor hiába írtam alá odabent digitálisan, az egész megy a levesbe. Arról nem is beszélve, hogy egy rendesen titkosított csomagot nem is tud értelmezni a router. (Mondjuk, a NAT traversal – IPSEC NAT-T – segítségével le lehet kezelni a problémát, viszont ez erőforrásba kerül.)

De mi a helyzet például a stream jellegű adatfolyamokkal? A mobilinternettel?

Miközben persze az igények is egyre nőnek. Nehogy már ne én mondhassam meg a kenyérpirítónak, hogy mikorra érek haza, és mennyire süsse át addigra a kenyeret! A hűtőgépem pedig küldjön figyelmeztető e-mailt, ha kevés benne a sör. Ez mind-mind IP-címet kíván. Méghozzá egyedit, mert egy benatolt kenyérpirító, lássuk be, nem az igazi. (Lehet, hogy a router mögött már a porszívó foglalta le a megfelelő portot.)

Bizony, ezek az igények az IP-mechanizmus alapjait ostromolják. Itt már nem segít a patkó továbbpatkolása – a gyökerekhez kell hozzányúlni.

Ez lesz/van az IPv6.

IPv6-alapok

De előtte egy kis történelem.

- 1990. Az IETF megállapítja, hogy baj van. Belevágnak a CIDR-be.
- 1993. Az IETF kezdeményezésére beindul az IPng (new generation) kidolgozása. Még ebben az évben kijön az RFC1719, egyfajta irányelvgyűjtemény.
- 1995. Rengeteg felmérés, ötletelés után elméletben összeáll a kép.

Az újszülöttet IPv6-nak nevezik el.

Ízlelgessük egy kicsit: 1995. Én 1997 körül kattantam rá a telefondrótra. Emlékezzünk vissza, milyen is volt akkoriban nálunk az IPv4-net? Nem merem azt mondani, hogy gyerekcipőben járt... az anyaméhben ugyanis logisztikai problémák miatt nem szoktak tipegőt hordani a csöppségek.

És akkor vágjunk végre bele.

Alapvetően az IPv6 az IPv4 továbbfejlesztett változata. Ez mindenképpen jó hír, mert azt jelenti, hogy nem lesz olyan irgalmatlan hatalmas nagy paradigmaváltás. Mint a neve is mutatja, elsősorban az IP-rész változik. A TCP például nem. De azért lesznek hullámzások.

Mire is jó az IP? A címfeloldásra. Milyen címeket is oldunk fel? Hát, ugye, az alkalmazás azt mondja, hogy őneki az sql001.cegnev.hu géppel kell hetyegnie. A DNS-szervertől meg kell kérdeznie az sql001.cegnev.hu névhez tartozó IP-címet. A DNS-szerver visszaadja. (Hoppá, egy hullámzás: a DNS-nek majd ismernie kell az IPv6-címeket.) Az alkalmazás innentől a 192.168.14.24 IP-címet fogja keresni. Az IP-protokoll dolga lesz, hogy megtalálja az ehhez a címhez tartozó hálózati kártya MAC addressét. Ennyi a történet az IP részéről. Ez fog megváltozni.

No még egyszer. Mire is jó az IP? A címfeloldásra. Ki segít neki ebben? Hát az ARP.

Na, őt kinyírták.

Maga a címfeloldás mechanizmusa az, ami gyökeresen megváltozott. Meg az IP-cím szintaxisa. Meg az IP-fejléc. Meg... na, jó. Egy napra legyen ennyi is elég.

Kezdjük az elején, nézzük az IP-címet. Az IPv4 32 bites volt. Az IPv6 128 bites lett. Ha szigorúan matematikusszemmel nézzük, ez azt jelenti, hogy a címtartomány mérete a negyedik hatványára emelkedett. Hogy ez mekkora szám? Nem kicsi. Több frappáns hasonlat is kering a neten, legyen ezek begyűjtése a házi feladat.

Az IPv4-címeket oktettekre bontottuk, majd az egybájtnyi értékeket decimális formában használtuk: például

```
11000000 10101000 00010111 00001100
azaz 192.168.23.12
```

Az IPv6-címeknél ez meglehetősen húzós lenne. Próbáljuk ki:

```
11000000 10101000 00010111 00001100 11000000 10101000 00010111 00001100
11000000 10101000 00010111 00001100 11000000 10101000 00010111 00001100
azaz 192.168.23.12.192.168.23.12.192.168.23.12.192.168.23.12
```

Látszik, hogy ezeket a számokat ember nem fogja tudni megjegyezni. Ide valamilyen tömörebb ábrázolás kell. Úgy van, látom sokan vágják egyből: igen, barátunk, a hexadecimális.

És akkor most felírok egy IPv6-címet:

Látható, hogy az elválasztójel megváltozott, pont helyett kettőspont lett. Az is látható, hogy 8 egységünk van, egységenként négy karakterrel. (Ahol ennél kevesebb van, oda nullákat kell eléjük tenni.) Mekkora szeletet jelent egy betű? Négy bitet, azaz egy fél bájtot. Naná... azért hexadecimális.

Gyakoroljunk. Hogyan nézne ki a fenti IPv4-cím IPv6-formában? Így: c0a8:170a.

A jó hír, hogy tényleg rövidebb. A rossz hír, hogy ezeket a számolgatásokat sóványmalac-vágtában el kell sajátítanunk fejben, ha IPv6-hálózatokat akarunk üzemeltetni. Immár nem az aknakereső lesz a harmadik leggyakrabban használt alkalmazásunk, hanem a calc.exe.

És persze még nincs vége. Ha az egyes blokkokban például csak nullák vannak, akkor az elhagyható. Konkrétan a fenti IPv6-cím így is írható: fe80::fd12:2f1:1234:abcd. Vegyük észre, hogy ahol kimaradtak a nullák, ott megduplázódtak a kettőspontok. Hogy mennyi maradt ki? Tudjuk, nyolc részletnek kell lennie, látunk ötöt – tehát háromblokknyi nullánk van. Vigyázat, egy címen belül csak egy ilyen összevonás lehet! Nincs olyan cím, hogy fe80::abcd::21 – tekintve, hogy ekkor nem tudnánk, mely részen hány üres blokk volt.

Aztán ebből elég vad dolgok is ki tudnak sülni. Hogy mást ne mondjak, a localhost: ::1

A subnet-maszk mint elv megmaradt – csak most már prefixnek hívják. Nagyjából ugyanúgy is használjuk: fe80:0:1234:adf:2::/64.

A /64 az, ugye, jelen esetben azt jelenti, hogy a cím fele a hálózati azonosító, a második fele pedig a hostazonosító, egész konkrétan:

- hálózat: fe80:0:1234:adf
- host: 2:0:0:0

Eddig a gyerekmedencében pancsikoltunk. Lassan itt az ideje a mélyebb víz felé venni az irányt.

Persze előtte még zuhanyozunk... azaz definiálgatunk.

Címzések

Az IPv6 alapvetően háromfajta címet ismer:

- unicast,
- anycast,
- multicast.

Nagyon durván leegyszerűsítve: a unicast címre küldve egy csomagot, csak egy node fogja megkapni azt. Multicast címre küldve egy csomagot, mindenki megkapja, akinek az a multicast címe. Az anycast cím ugyanolyan, mint a multicast, de ha valamelyik node rácsap a forgalomra, akkor arra rá lesz csapva. Másnak már nem jut belőle.

Biztos lesz olyan, akinek ez ismétlés, de nem árt az alapokat sem tisztázni.

Mi is rejlik az olyan fogalmak mögött, hogy host, node, subnet, link?

- Host: tulajdonképpen számítógép, akár több hálózati kártyával.
- Node: MAC addres-szel rendelkező hálózati pont. 1 MAC address, 1 node. Ha egy számítógépben két hálókártya van, akkor az két node.
- Subnet: alhálózat. Az egy subneten belüli hostok router közbeiktatása nélkül is tudnak egymással kommunikálni – feltéve, hogy egy linken vannak.
- Link: fizikailag egy dróton lévő hostok összessége, amelyeket egy router szeparál el a hálózat többi részétől. Az esetek nagy részében ez egy subnet is, de nem kötelezően. Minden további nélkül lehet egy fizikai hálózaton – linken – több subnet is.

Unicast címek

A következő unicast címek léteznek:

- unique global, azaz egyedi globális cím;
- link-local cím, azaz csak a linken értelmezett cím;
- site-local cím, azaz csak a lokális site-on értelmezett cím;
- unique-local, azaz lokálisan egyedi cím;

- speciális címek
- transition, azaz áttérési címek.

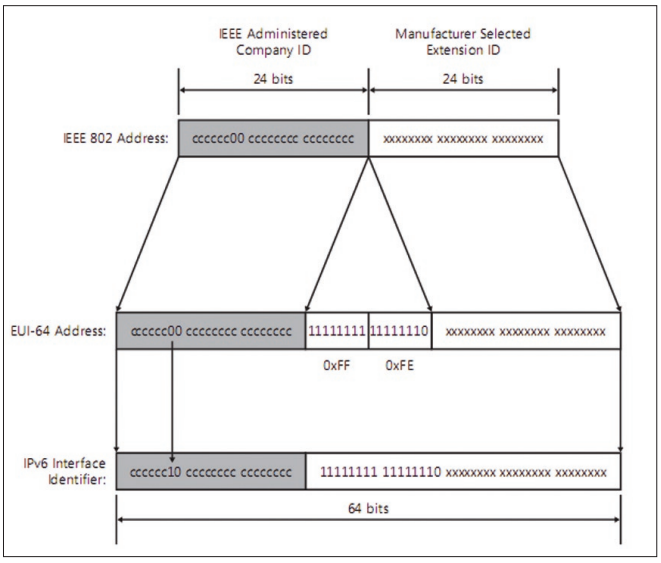
Egyenként.

Egyedi globális címek. Az egyedi globális címek, mint a nevük is mutatja, abszolút egyediek. Az egész világegyetemben. Felépítésük nagyjából így néz ki:

```
fe80:0:0:0:fd12:2f1:1234:abcd
```

- 001: ez a fix része a címnek.
- Global routing: ettől lesz a cím globális. Tulajdonképpen ebben a tartományban fognak nyüzsögni az ISP-k.
- Subnet ID: a globális címen belüli alhálózatok. Logikusan 65 536 lehet belőlük.
- Interface ID: a node egyedi azonosítója. Mivel ez egy olyan elem, amely a legtöbb címzési technikánál visszaköszön, érdemes elmélyedni a generálásában. Létezik egy kódolás, 64-bit Global Identifier (EUI-64) a pontos neve. Ennek van egy olyan fejezete, amelyik azzal foglalkozik, hogyan is lehetne egy 48 bites MAC (IEEE 802) addressből 64 bites egyedi azonosítót fabrikálni. Le fogsz esni a székről, ha elárulom, hogyan: a MAC address közepére beszúrnak 16 bitet, méghozzá ezeket: FFFF. Pontosabban, ez a szabvány, de az IPv6-ban inkább az FFFE értéket szúrják be. Még pontosabban, ez csak az egyik lehetséges mód interface ID generálására... de a legnépszerűbb. (Apropó, azt ugye tudtad, hogy a MAC address első 24 bitje a gyártó azonosító kódja, a második 24 a kártyáé? Azaz az FFFE érték pont a kettő közé szúródik be.)

Megjegyzés: a fenti címképzés az elméleti változat. A gyakorlatban beszúrtak a tervezők néhány matyó csujjogást a koreográfiába: a MAC address gyártóspecifikus részében a konverzió során átfordítanak egy bitet. Pusztán a tömörebb számábrázolás kedvéért. A teljes folyamat az ábrán látható.



A címképzés teljes folyamata

Link-local címek. A link-local címek olyan címei egy node-nak, amelyek csak az adott linken érvényesek. A router ezeket a címeket nem fogja kiengedni.

Képzésük:

```
|1111111010|54 0 bit|Interface ID (64 bit)|
```

- 1111111010: fix érték.
- 54 üres bit: meglehetősen fix érték.
- Interface ID: már ismerjük.

Gondolom, látod te is, hogy ez az egész felírható úgy is, hogy FE80::/64 prefix + node azonosító.

Site-local címek. Amikor ilyen nagy címtartományunk van, simán előfordulhat, hogy ún. köztes alhálózati rendszert iktatunk be a prefix és az interface ID közé. Ezeket a köztes alhálózatokat hívják site-nak, a site-on belülről kitörni nem képes címeket pedig site-local címeknek.

Így néznek ki:

```
|1111111011|subnet azonosító (54 bit)|Interface ID (64 bit)|
```

Unique local address. A helyzet az, hogy a site-local címek miatt nem lehetünk 100 százalékig biztosak abban, hogy a link-local címeink abszolút egyediek a linken. Emiatt vezették be a unique-local címeket.

Speciális címek. Például a localhost: ::1/128.

Transition címek. Átállítani az internetet az IPv6-ra... igen nagy falat lesz. Nem is fog menni egyik napról a másikra. A békés átmenet érdekében szükségünk lesz egy csomó kétéltű címre. Itt csak felsorolom ezeket, nagy a net, akit mélyebben érdekel a téma, utána tud nézni:

- IPv4-compatible address;
- IPv4-mapped address;
- 6to4 address;
- ISATAP address;
- Teredo address.

Multicast címek

Általában így néznek ki:

```
|11111111|Flags (4 bit)|Scope (4 bit)|Group ID (112 bit)|
```

Ebbe most nem mennék bele túl részletesen, a flagnek is és a scopenak is meglehetősen sok variánsa van. Inkább csak felsorolom azokat, amelyekkel sűrűn fogunk találkozni a hétköznapokban. Az összes előre definiált multicast cím egyébként az FF01:: - FF0F:: címtartományban található, ide egyszerű halandó nem is definiálhat saját címeket.

- FF01::1 – interface-local scope all-nodes multicast address, azaz minden-node multicast cím, hoston belül.
- FF02::1 – link-local scope all-nodes multicast address, azaz minden-node multicast cím, linken belül. (Vegyük észre, hogy ez tulajdonképpen a broadcast! Nincs is külön, megszűnt. Ez a multicast cím van helyette.)
- FF01::2 – interface-local scope all-routers multicast address, azaz minden-router multicast cím a hoston belül.
- FF02::2 – link-local scope all-routers multicast address, azaz minden-router multicast cím a linken belül.
- FF05::2 – site-local scope all-routers multicast address, azaz minden router multicast cím a site-on belül.

Rengeteg egyéb előre definiált multicast cím létezik – például az összes DHCPv6 szerver, meg egyéb ínyencségek. Tessék utána olvasni!

Fontos megemlíteni még egy speciális multicast címet is, ezt úgy hívják, hogy solicited-node address. De erről a következő részben fogok részletesebben is beszélni.

Gondolom, elég sok ez így első körben. Ismétlésképpen nézzük végig, milyen címekre is kell hallgatnia egy mezei hálókártyának:

- link-local cím;
- unique global;
- unique local;
- loopback;
- minden-node hoston belül;
- minden-node linken belül;
- solicited-node;
- egyedi multicast címek.

Durva egy kicsit. Ez ugyanis mind egy-egy IP-cím, amely a kártyához rendelődik. Alaphelyzetben.

Most, hogy már van némi fogalmunk az IPv6-címzésről, érdemes lehet végignézni annak a gépnek az IP-konfigurációját, amelyen ezt a cikket írom:

```
Windows IP Configuration

Host Name                   :hq
Primary Dns Suffix:
      Node Type             : Hybrid
      IP Routing Enabled    : No
      WINS Proxy Enabled    : No

Ethernet adapter Local Area Connection:

Connection-specific DNS Suffix  . :
Description                     Realtek RTL8168B/8111B Family PCI-E
                                Gigabit Ethernet NIC (NDIS 6.0)
Physical Address                : 00-1E-8C-AB-2F-88
DHCP Enabled                    : Yes
Autoconfiguration Enabled       : Yes
Link-local IPv6 Address         : fe80::c5cc:1c5c:8384:4546%8(Preferred)
IPv4 Address                    : 192.168.1.101(Preferred)
Subnet Mask                     : 255.255.255.0
Lease Obtained                  : 2008. június 3. 19:14:09
Lease Expires                   : 2008. június 4. 19:14:08
Default Gateway                 : 192.168.1.1
DHCP Server                     : 192.168.1.1
DHCPv6 IAID                     : 201334412
DNS Servers                     : 84.2.44.1
                                  84.2.46.1
NetBIOS over Tcpip              Enabled

Tunnel adapter Local Area Connection* 6:

Connection-specific DNS Suffix  .  :
Description                     : Teredo Tunneling Pseudo-Interface
Physical Address                : 02-00-54-55-4E-01
DHCP Enabled                    : No
Autoconfiguration Enabled       : Yes
IPv6 Address                    : 2001:0:d5c7:a2ca:3c2f:2bc6:3f57:fe9a
                                  (Preferred)
Link-local IPv6 Address         : fe80::3c2f:2bc6:3f57:fe9a%9(Preferred)
Default Gateway                 : ::
NetBIOS over Tcpip              : Disabled

Tunnel adapter Local Area Connection* 7:

Connection-specific DNS Suffix  . :
Description           : isatap.{47CE5CAA-223F-4CD4-9A17-D91D7DDC2066}
Physical Address      : 00-00-00-00-00-00-00-E0
DHCP Enabled          : No
Autoconfiguration Enabled  : Yes
Link-local IPv6 Address    : fe80::5efe:192.168.1.101%10(Preferred)
Default Gateway            :
DNS Servers                : 84.2.44.1
                             84.2.46.1
NetBIOS over Tcpip         : Disabled

Illetve a route tábla:

===========================================
Interface List
8     00 1e 8c ab 2f 88          Realtek RTL8168B/8111B Family PCI-E
                                 Gigabit Ethernet NIC (NDIS 6.0)
1                                Software Loopback Interface 1
9     02 00 54 55 4e 01          Teredo Tunneling Pseudo-Interface
10    00 00 00 00 00 00 00 e0    isatap.{47CE5CAA-223F-4CD4-9A17-
                                 D91D7DDC2066}
===========================================

IPv4 Route Table
===========================================
Active Routes:
Network Destination  Netmask          Gateway      Interface      Metric
0.0.0.0              0.0.0.0          192.168.1.1  192.168.1.101  20
127.0.0.0            255.0.0.0        On-link      127.0.0.1      306
127.0.0.1            255.255.255.255  On-link      127.0.0.1      306
127.255.255.255      255.255.255.255  On-link      127.0.0.1      306
192.168.1.0          255.255.255.0    On-link      192.168.1.101  276
192.168.1.101        255.255.255.255  On-link      192.168.1.101  276
192.168.1.255        255.255.255.255  On-link      192.168.1.101  276
224.0.0.0            240.0.0.0        On-link      127.0.0.1      306
224.0.0.0            240.0.0.0        On-link      192.168.1.101  276
255.255.255.255      255.255.255.255  On-link      127.0.0.1      306
255.255.255.255      255.255.255.255  On-link      192.168.1.101  276
===========================================
Persistent Routes:
None

IPv6 Route Table
===========================================

Active Routes:
If     Metric   Network Destination                          Gateway
9      18       ::/0                                        On-link
1      306      ::1/128                                     On-link
9      18       2001::/32                                   On-link
9      266      2001:0:d5c7:a2ca:3c2f:2bc6:3f57:fe9a/128    On-link
8      276      fe80::/64                                   On-link
9      266      fe80::/64                                   On-link
10     281      fe80::5efe:192.168.1.101/128                On-link
9      266      fe80::3c2f:2bc6:3f57:fe9a/128               On-link
8      276      fe80::c5cc:1c5c:8384:4546/128               On-link
1      306      ff00::/8                                    On-link
9      266      ff00::/8                                    On-link
8      276      ff00::/8                                    On-link
===========================================
Persistent Routes:
None
```

Csemegézzünk! Vajon mi lehet az IPv6-címem? Vigyázat, beugrató kérdés, ugye, elég sok lehet. De jelen esetben csak link-local címeket látunk, az egyik például így néz ki: fe80::c5cc:1c5c:8384:4546%8. Ránézésre is több baj van a címmel. Először is, mi az a %8 a végén? Aztán az interface ID egyáltalán nem úgy néz ki, mintha a MAC addressből képezték volna betoldással. Árulás? Nos, a %8 formulára egyszerű a válasz: egyszerű cím esetén ez az interface azonosító száma,

úgynevezett zónaazonosító. (A route-táblánál látható, hogy konkrétan egy Realtek hálókártyáról van szó.)

Jó, akkor mi van az Interface ID-val? Az, hogy az EUI-64 nem kötelező. A Windows Server 2008, illetve a Vista alaphelyzetben például véletlenszerűen generált interface ID-t használ. Erről a következő parancs segítségével tudjuk lebeszélni:

```
netsh interface ipv6 set global randomizeidentifiers=disabled
```

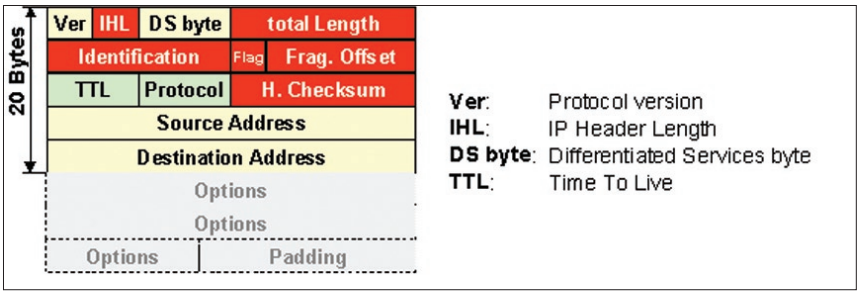
Miután kiadtam ezt a parancsot, és megújítottam az IP-címemet, a következőt kaptam:

```
Link-local IPv6 Address: fe80::21e:8cff:feab:2f88%8
```

Ugye, mindjárt más? Ez már EUI-64. A matyó csujjogatással.

Miért is nincs globálisan vagy lokálisan egyedi (global/local unique) címem? Mert a routerem azt mondta, hogy nekem olyanom nincs. Pontosabban, nem mondta, hogy van.

Aztán nagyon elszaladtunk amellett, hogy nekem nem is egy link-local IP-címem van. Mi is a többi? Nos, transition címek. Azok, amelyeket csak úgy megemlítettünk korábban. Most sem kapnak nagy szerepet... de azért vegyünk észre valamit: mindkettő, a Teredo és az Isatap is Tunnel Adapter Local Area Connection néven fut. Tunnel... valami belecsövezve valamibe. Mi van most? IPv4. Mi lesz majd? IPv6. Nyilván logikus, hogy ezeket a címeket használva tulajdonképpen az IPv6-ot csatornázzuk bele az IPv4-be. Végül egy kérdés: ha teszem azt a Magyar Telecom globális azonosítója 1100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 1, és én ezen belül a 0000 0000 0000 0011 subnetben vagyok, akkor mi lesz a világegyetemben egyedi azonosítóm? Lapátoljuk össze: 001, mert ez abszolút fix. Ehhez hozzácsapjuk az ISP globális azonosítóját és a subnet-azonosítót. Ez lesz a prefix, ehhez jön majd az interface ID, amelyet a link-local címből tudunk kibányászni. Jelen esetben ez c5cc:1c5c:8384:4546. Innen már csak matekozás: 3800::1:3:c5cc:1c5c:8384:4546/64. Long live calc.exe. (Hint: fel kell írni bitsorozatként, bájtonként csoportosítani, átváltani, két bájt egy blokk.)

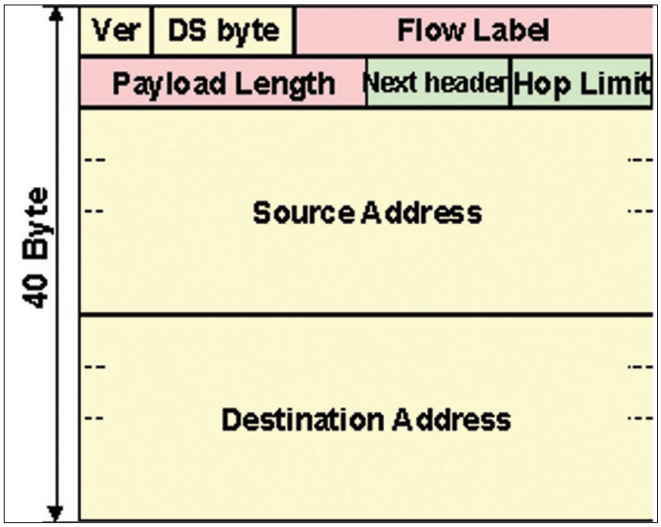


IPv4-csomag fejlécének felépítése

IP-fejlécek

Ezen a részen gyorsan át fogunk rohanni – ugyanis az egyes mezők szerepeinek kifejtése bőven meghaladja e cikk kereteit. Minden különösebb kommentár nélkül egymás mellé teszem az IPv4- és az IPv6-csomag fejléceinek a szerkezetét:

Habár azt mondtam, nem fogom kommentálni a képeket, azért egy dologra felhívnám a figyelmet: annak ellenére, hogy az IP-címek a négyszeresükre nőttek, a fejléc csak a duplája lett. Racionalizálás, kérem szépen, racionalizálás.



IPv6-csomag fejlécének felépítése

(Gondolom, nem kell külön kihangsúlyoznom, hogy az új csomagstruktúrához új TCP/IP stack is kell.)

A címfeloldás – ARP nélkül (Neighbour Discovery, ND)

Ha már rajtunk a búvárfelszerelés, ne fecséreljük az időt. Nézzük meg, hogyan is történik majd IP-szinten a névfeloldás?

De előtte lássuk azt, hogyan is történt régen?

Az 'A' node kapcsolatba akart lépni egy konkrét IP-című géppel. (Ezt a címet már megmondta neki a DNS.) Fogta, és szétküldött egy ARP broadcastot, benne a saját IP-címe, IP(A), saját MAC-címe, MAC(A) és a keresett IP-cím, IP(B). A broadcastot minden node vette a linken, felolvasták és megnézték, vajon őket keresik-e. Aztán aki birtokolta az IP(B) címet, az visszaküldte a MAC-címét, MAC(B) – és a sóvárgó MAC-címek vonzalma szárba szökkent.

Ugye, nem kell mondanom, mi az eljárás hátránya? Az, hogy a broadcast-csomaggal mindegyik node operációs rendszerének foglalkoznia kellett.

Az IPv6-ban újítottak. Bár eljátszhatták volna mindezt a minden-node link-local címre küldött üzenettel, de akkor ugyanott lennénk, mint az ARP broadcasttal.

Ehelyett egy speciális multicast címre megy ki az érdeklődő üzenet. Ezt a speciális címet hívják úgy, hogy solicited-node cím. A következőképpen képződik:

```
|FF02::1:FF00:0 (104 bit) | az interface ID jobb szélső 24 bitje|
```

A megértés kulcsa a jobb szélső mező. Kinek is az interface ID-ja? Nyilván nem a feladóé. Annak nem lenne semmi értelme. Természetesen a keresett IP-címből képződik a solicited-node cím.

Az érdeklődő – 'A' – node tehát legyártja ezt a solicited-node címet és elküldi neki a saját IP-címét, IP(A), saját MAC-címét, MAC(A) és persze a keresett IP-címet, IP(B). Mely node-ok fogják felkapni ezt a

kérést? Akiknek a – saját maguknak már korábban legyártott solicited-node – címük megegyezik a feladó által megadott solicited-node multicast címmel. Egész konkrétan: az a pár node fogja felkapni, ahol az interface ID jobb szélső 24 bitje megegyezik. A többi node békésen kérődzik tovább. Tiszta haszon az ARP-hez képest.

Egy újabb gondolkodós kérdés: vajon mi is lesz az én solicited-node címem?

Az első 104 bit, az ugye adott. A maradék 24 bit pedig kiszámolható az Interface ID-ból.

Merre is vagy, calc.exe?

Tessék, a cím: ff02:1::ff00:0084:4546. Most az egyszer nem segítek.

Elképzelhető, hogy valakiben felmerül a kérdés: mire is jó ez az egész cécó? Hiszen az interface ID a MAC addressből képződik... egyszerűen számoljunk belőle vissza, és már miénk is a MAC.

Az ötlet jó... de sajnos ez csak ajánlás. Az interface ID sokféleképpen keletkezhet, semmi garancia sincs rá, hogy pont az EUI-64 volt a keletkezésének alapja.

Autokonfiguráció

Ez megint egy szép téma. Kezdjük megint azzal, hogy milyen lehetőségünk volt autokonfigurációra IPv4 alatt? Igen, a DHCP.

Az IPv6 alatt tágabb lett a horizontunk:

- stateless autoconfiguration;
- stateful autoconfiguration (DHCPv6);
- kombinált bérlet.

Vizsgáljuk meg először a stateless változatot. Messziről fogunk elindulni.

Van egy hostunk, egy hálózati kártyával. Feldugjuk egy IPv6-hálózatra. DHCPv6 nincs. Router van.

A router meghatározott időnként küld egy üzenetet a minden-node link címre. Az üzenet sok mindent tartalmaz, többek között:

- a router MAC-címét;
- a linken élő prefixeket;
- a linken érvényes MTU-t;
- a linken érvényes maximális hop számot;
- van-e DHCP a linken?

És még egy csomó mindent, amelyek jelentőségét nem tudtam felfogni. De már ezek is izgalmasak. Központilag szabályozható MTU? Hmm... finom. A link összes prefixe? Álljunk csak meg! Ha adottak a prefixek, a hálókártya meg ismeri a saját MAC-címét, ismeri az EUI-64 módszert, simán le tudja gyártani magának a link összes prefixére a link-local címét! Mit ad még meg a prefixlista? Hát például azt, hogy mely prefixek vannak a linken – és melyek azok, amelyekhez át kell harcolniuk a csomagoknak magukat a routeren.

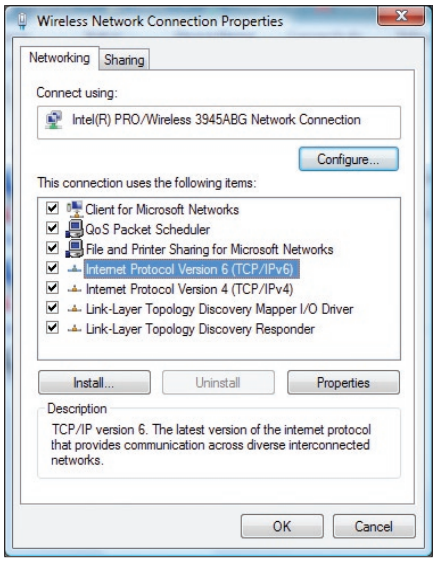
Szóval belép az új állomás a linkre. Olyan korban élünk, hogy türelmetlenek vagyunk: nem várjuk meg, hogy a router körbeküldje az üzenetét, belerúgunk: router, küldj üzenetet! És az küld. Ez alapján a node összerakja magának a link-local címeit. Elképzelhető, hogy üközni fog másik node címeivel? Bizony, igen. Ekkor jön a jó öreg ND. Megszólítjuk a solicited-node címen azt az IP-címet, amelyet magunknak kívánunk lefoglalni. Ha nem jön válasz, nyertünk. Miénk a cím.

(Ki kell rá térnem, hogy a Windows-implementációnál ez is másképp van egy kicsit. Arról már írtam, hogy az Interface ID nem EUI-64 módon generálódik, hanem véletlenszerűen. Arról viszont nem, hogy a tervezők úgy saccolták, nagyon kicsi az esélye annak, hogy a véletlen azonosítók között egyformák legyenek: ezért elhagyták az ellenőrzést.)

Mi van, ha megváltozik a MAC-címünk? Úgy csinálunk, mintha ND kérdéseket kaptunk volna, elárasztjuk a linket válaszokkal.

Mi van, ha két hálókártyával is kapaszkodunk egy linken? Hol ez, hol az fog válaszolni az ND kérdésekre – terheléselosztás.

Nézzük a stateful változatot. Nos, a DHCPv6 megint egy olyan terület, amellyel nem szándékozom túl sokat foglalkozni: nem fér már bele a cikkbe. Nyilván lesznek új kunsztjai. Például rá tud szólni az ügyfelekre, hogy változás történt a központi konfigurációban, legyenek kedvesek, indítsanak el egy új címigénylési folyamatot. Aztán a szerver képes lesz visszavonni is a kiadott címeket.



Az IPv6 alapból fent van, és eltávolíthatatlan Vistán

Zárszó

Tény, hogy az egyszerű mezei rendszergazda már évek óta ignorálja az IPv6-témát. Tény, hogy ez eddig sikeres taktika volt, mert soha nem tört még be igazán a technológia a Windows-világba. Remekül meg tudtunk élni nélküle.

Ennek a világnak lassan vége. Ma már minden valamirevaló operációs rendszer képes támogatni az IPv6-ot. A technológia sunyiban terjed. Nagyjából azzal a sebességgel, ahogyan bátor rendszergazdák barátkoznak a számrendszerek közötti átváltogatásokkal. Igény van rá, tökéletesen együtt tud működni az IPv4-világgal, fokozatosan lehet rá áttérni, ahogy gyarapodnak a Windows Server 2008, illetve Windows Vista operációs rendszerek a vadonban. De már az XP is beépítetten támogatja, igaz, külön kell telepíteni (ipv6 install). Windows Server 2003 alá szintén telepíthető. Windows 2000-hez, illetve NT-hez letölthető add-onként. Windows 95/98/Me számára a Trumpet gyárt IPv6 winsock implementációt.

Tényleg csak elhatározás kérdése, mikor kezdünk el vele komolyabban foglalkozni.

No meg fogékonyság az újra.

Petrényi József
MCSE+M, MCITP, MVP
(petrenyi.jozsef@sao.hu)
SAO-Synergon

Linkek

A cikkhez rengeteg forrást használtam. A linkgyűjtemény a http://www.microsoft.hu/technet blogon lesz elérhető.

SQL Server 2008: adattárház-újdonságok

Az SQL Server 2008-at erőteljesen továbbfejlesztették abba az irányba, hogy hatalmas mennyiségű adatokkal operáló adattárházakat tölthessünk fel és kezelhessünk a segítségével. Ezeket tekintjük át cikkünkben.

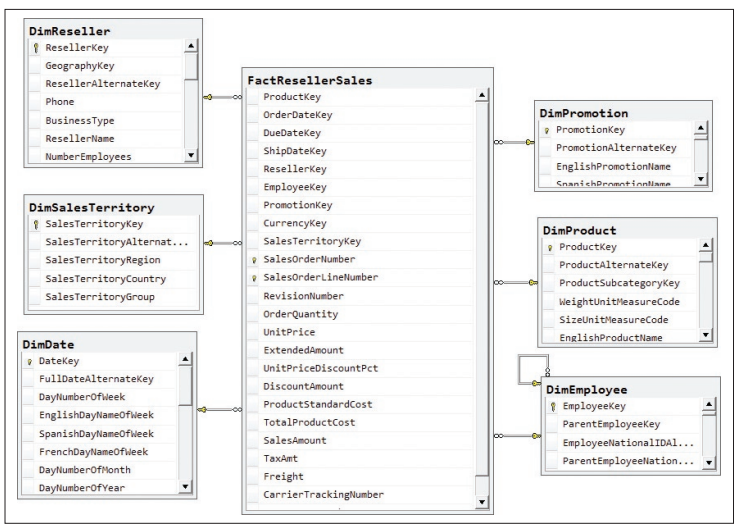
Az adattárházat sokan az OLAP-pal társítják, pedig ez sima relációs adatbázis-fogalom. Az adattárház olyan speciálisan megtervezett relációs adathalmaz, amelyből könnyen és gyorsan lehet adatokat lekérdezni *elemzés* céljára. Sokszor OLAP-kockák *bemeneteként* is szolgál, de például jelentések bemeneteként is kiválóan használható.

Egy tranzakciós adatbázist általában erősen normalizálnak, hogy az adatokat minél kevésbé redundáns módon tárolják. Emiatt sok, kevés oszlopot tartalmazó tábla keletkezik. Ez általában ideális adatok beszúrására, módosítására és törlésére, azaz tranzakcionális igénybevételre, de lekérdezéskor általában elég sok joinon keresztül tudjuk csak összerakni a szétszórt információdarabkákat, azaz lekérdezésre nem optimálisak az OLTP-adatbázis-szerkezetek.

Az adattárházak ezzel szemben tipikusan csillagsémás felépítésűek. Ez azt jelenti, hogy van egy hatalmas *ténytábla (Fact…)*, amely a tranzakciók lenyomatát tárolja tömören, és ehhez a táblához kapcsolódik sok *dimenziótábla (Dim…)*, amely az adatokat értelmezi. Például a ténytáblában csak annyi van egy megrendelésről, hogy a 3-as vevő 20080203-kor (integerként tárolva!) vásárolt a 44-es termékből 5 darabot a 34-es akciós kóddal beárazva. Ezeket az értékeket oldják fel üzletileg értelmes információkra a dimenziótáblák.

A ténytábla igen nagy tud lenni, akár milliárdsoros is (10^9 sor). A dimenziótáblák tipikusan sokkal kisebbek: pár ezer, de maximum pár millió sorosak. Ezeknek a méreteknek és arányoknak nagy szerepük lesz, amikor a szervernek tényleg össze kell joinolnia a táblákat egy lekérdezésben.

Az adattárházakat az OLTP-adatokból tápláljuk, sokszor inkrementálisan, csak, mondjuk, az aznap történt tranzakciókat átlapátolva. Az adott időszak változásait azonosíthatjuk a többek között erre a célra tervezett Change Data Capture (CDC) nevű, SQL Server 2008-ban megjelenő technológiával, vagy használhatjuk az előző részben megismert MERGE utasítást is a különbség képzésére. A CDC előnye, hogy kis költséggel, a tranzakciós naplót olvasva normál üzemi időszakban gyűjtögethetjük a tranzakciók lenyomatait, így mind a változáskövetés, mind az adatáttöltés nagyon kis költséggel valósítható meg.



1. ábra. Adattárház: csillagséma (részlet)

A már adattárházba betöltött adatok elemzése során nagyon nagy IO-költségekkel kell számolni, hisz óriási méretű adatokról van szó. Az IO-költséget csökkenthetjük az adatlapok tömörített tárolásával, némi processzorteljesítmény árán. A backup idejét és helyszükségletét a mentés tömörítése csökkenti.

A GROUPING SETS segítségével többféle szempont szerint, hatékonyan, egy menetben tudunk aggregátumokat számolni, ami például riportok bemeneteként nagyon kényelmesen bevethető.

Ha kevésbé szabályos adatokból kell tárházat építenünk, akkor az adatok előzetes elemzésére, profilozására a termék részét képező SQL Server Integration Services új szolgáltatásait is bevethetjük, amely aztán a betöltést is az iparban páratlan sebességgel képes végrehajtani (per pillanat a leggyorsabb ETL-eszköz az adatbázispiacon).

Az új, sokszálú partíciófeldolgozás gyorsabb lekérdezéseket tesz lehetővé sokprocesszoros gépeken. Az indexelt nézeteket is továbbfejlesztették, így ha particionált táblán hozzuk azt létre, a partíciót mozgathatjuk, darabolhatjuk anélkül, hogy az indexelt nézetet el kellene dobni. Mivel az indexelt nézet általában előaggregált adatokat tárol, azaz lassú létrehozni, ez nagy teljesítménynyereséget eredményezhet.

Az adattárház csillagsémás szerkezetéből való lekérdezéseket pedig a Bitmap Filter tudja jelentősen felgyorsítani. Ezeket az újdonságokat elemzem részletesen a következőkben.

Előtte azonban még egy megjegyzés. Bár adattárházi funkcióként azonosítottam ezeket az új szolgáltatásokat, valójában nagyon sok egyéb adatbázis alapú alkalmazás képes ezekből előnyt kovácsolni. Hisz a Bitmap Filter bármely nagy táblás joinban tud gyorsítani, vagy a tömörítés sok, javarészt olvasott, kevésbé módosított adatbázistábla IO-költségét tudja csökkenteni, nem csak egy adattárházét.

Minimálisan naplózott beszúrások

Korábbi SQL Server-változatokból már ismert, hogy a minimálisan naplózott vagy más néven bulk műveletek nagyon gyorsak, mert a tranzakciós logba nem kerül bele például minden egyes beszúrt sor egy bulk insertnél, hanem csak azok a lapok (8k) vagy extentek (64k) jelölődnek, amelyek módosultak. Azaz sokkal kevesebb adat kerül a logba, több tucatszor gyorsabb lehet a művelet. Nemcsak a bulk insert minimálisan logolt, hanem a truncate table, select into tábla, writetext, updatetext, sőt bizonyos index-műveletek is.

Ez eddig is így volt. 2008-ban azonban már bizonyos esetekben az insert-select is minimálisan naplózott! Ez nagy szám ám, mert sokan úgy töltenek be adatokat, hogy először nyersen behúzzák azokat egy átmeneti táblába, aztán elemezgetik, javítgatják, tisztítgatják, majd áttöltik a végleges táblába. A nyers betöltés mehetett már eddig is gyorsan bulk inserttel (akár .NETből az SqlBulkCopy osztállyal), vagy SQL Server Integration Services-zel, de a két tábla közötti adatátvitel eddig teljesen naplózott, ergo, lassú volt. Eddig. Most már megy gyors módon is az insert-select, a következő feltételekkel.

- Értelemszerűen csak simple és bulk logged recovery modell esetén működik, a full recovery modell minden bulk műveletet teljesen naplózottá alakít (ez a célja).
- With (tablock) hintre van szükség a céltáblánál.
- Ha a táblán nincs index (heap), az adatok minimálisan naplózott módon kerülnek be a táblába.
- Ha a táblán nincs clustered index (heap), de van rajta legalább egy noncusered index, akkor az adatok minimálisan logoltan kerülnek a táblába, az indexek módosítása csak akkor lesz minimálisan logolt, ha a tábla üres volt. Ha nem, akkor az indexek teljesen logolt módon rögzülnek (mint eddig), de az adatok továbbra is minimálisan naplózott módon.
- Ez a feltöltöttségről szóló kitétel azért fontos, mert ha egy üres táblába nem egyszerre, hanem több darabban töltünk be adatokat insert-selecttel, akkor csak az első insert lesz minden szempontból minimálisan naplózott.
- Ha a táblán clustered index is van, csak üres tábla esetén lesz a betöltés minimálisan naplózott.

Nézzük, hogyan lehet meggyőződni róla, hogy az insert-select tényleg minimálisan naplózott lett-e!

Készítsük elő a teszttáblákat! A forrástáblában jó hosszú sorokat készítünk elő, így jól látható lesz a kétféleképpen naplózott művelet közötti különbség.

```
create table Celtabla
(
 id int,
 adat nvarchar(1000)
)
go
create table Forrastabla
(
 id int,
 adat nvarchar(1000)
)
go

set nocount on

declare @i int = 1
while (@i < 1001)
begin
   insert into Forrastabla
   values (@i, replicate('a', 1000))
   set @i += 1
end
```

Jöhet az insert-select, először with (tablock) hint nélkül:

```
begin tran --hogy ne truncate-olódjon a log mielőtt megnéznénk

insert into Celtabla --with (tablock)
select * from Forrastabla

select top 100 [log record length] Hossz,
AllocUnitName Célobjektum,
Context Környezet
from fn_dblog(null, null)
where allocunitname='dbo.Celtabla'
order by [Current LSN] desc

rollback
```

A tranzakciós napló tartalmát az fn_dblog függvénnyel kérdeztük le:

```
Hossz  Célobjektum   Környezet
------ ------------------------
2108   dbo.Celtabla  LCX_HEAP
2108   dbo.Celtabla  LCX_HEAP
2108   dbo.Celtabla  LCX_HEAP
...
```

Látható, hogy a dbo.Celtabla objektumba történik beszúrás, ami heapen tárolt, azaz olyan tábla, amelyen nincs clustered index. A művelet során 2108 bájt íródik a naplóba, *minden egyes sorhoz.*

Ha a fenti kódban visszaállítjuk a megjegyzésbe tett with (tablock) hintet, akkor a következő lesz a log képe:

```
Hossz  Célobjektum   Környezet
------ ------------- ---------
72     dbo.Celtabla  LCX_GAM
```

```
72   dbo.Celtabla  LCX_IAM
92   dbo.Celtabla  LCX_PFS
92   dbo.Celtabla  LCX_PFS
...
```

A hossz oszlop az érdekes, bár a sor 2 kilobájtos, mégis csak 72-92 bájt kerül a naplóba, mert minimális módon történt az insert. A három betűszó egyébként a Global Allocation Map, Index Allocation Map és Page Free Space rövidítése, ezek az adatok tárolását nyilvántartó speciális lapok. Vagyis az adatokat egyáltalán nem naplózzák, csak az azok által érintett foglalási egységeket.

Azaz látható, az SQL Server 2008-ban kibővült a minimálisan naplózott műveletek listája az insert-select-tel, amely nagyon gyors adatbetöltést, -áttöltést tesz lehetővé.

Csillag-Join optimalizálása Bitmap Filterrel

Csillag-joinról beszélünk, amikor egy táblához sok másik táblát kapcsolunk hozzá. Mint láthattuk, adattárházakban pont ilyen a séma, amikor egy hatalmas ténytáblához sok kisebb dimenziótábla tartozik.

Az AdventureWorksDW példaadatbázis egy ilyen tárházi minta. Ezek az adattárházi táblák elég speciálisan vannak tervezve, nemcsak a csillagséma szerkezet miatt. Például a FactResellerSales táblában a dátumok nem datetime-ként, hanem intként vannak tárolva. 20010701, ez egy egész szám, amit dátumként értelmezünk. Azért van ez a furcsa felállás, mert integereket nagyon gyorsan lehet összehasonlítani, ami a JOIN szempontjából fontos. Emellett ráadásul 8 bájt helyett csak 4 bájtot igényel a tárolása.

Amikor ezeket a nagy táblákat joinoljuk, akkor általában szóba se jöhet a loop join a világ legjobb indexei mellett sem, mert százmilliószor nekifutni egy táblának még index mellett is nagyon költséges. Ilyen nagy tábláknál merge vagy hash join jöhet számításba. A merge csak a kulcsok azonos rendezettsége, azaz általában akkor működik, ha mindkét tábla joinolt oszlopán clustered index van. Ez ritkán jön össze mindkét oldalnál, ezért elég ritkán látni merge joint nagy táblák esetén. Kis tábláknál egyébként még arra is hajlandó a szerver, hogy a kisebb táblát lerendezi úgy, mint a nagyot, aztán merge-öl. De ezt csak pár százezer soros táblákig vállalja be, és csak akkor, ha jó sok memória van a gépben.

Azaz, a legtöbb esetben az adattárházakban hash joinokat fogunk látni. A merge és a hash join is azért jobb a ciklusosnál nagy táblák esetén, mert mindkettő csak maximum egyszer járja be a táblákat. Igaz, hogy akkor sokszor az egészet, de legalább nem esik neki sok milliószor, mint a loop join.

A hash join úgy működik, hogy a szerver a kisebb táblát a joinolandó oszlop mentén lehasheli, ez lesz a build input. Adattárház esetén az adott dimenziótábla. Épít egy hashtáblát a kulcs alapján. Aki valamely programozási frameworkből ismeri a hashtáblát, az tudja, hogy ez egy olyan memóriastruktúra, ami nagyon gyors keresést tesz lehetővé a hashkulcs alapján. A kiinduló értékek hashei között lesz ütközés, például az 123 és a 3455 is lehet, hogy ugyanarra a hashértékre, mondjuk, 12-re képeződik le. Az ütköző értékeket ún. vödrökbe (bucket) rakják, amiben lineáris listában tárolódnak az eredeti kulcsok, azaz egy vödrön belül a keresés már nem gyors, csak lineáris, és nem a hasheket, hanem az eredeti típusokat kell összehasonlítani.

Amikor a build input kész az egész (de a kisebb) táblára, akkor a szerver elkezd végigmenni a nagyobbik táblán, és megnézi, benne van-e a tábla adott joinkulcsa a build inputban. Ehhez lehashelik ennek is a kulcsát, majd a vödrök között felezős kereséssel gyorsan megtalálják azt a vödröt, amelyikben benne lehet a keresett kulcs. A vödrön belül lehet számtalan tétel, amelyekkel már ténylegesen össze kell hasonlítani a kulcsokat. Ha a kulcs megvan, nyertünk, a két tábla az adott kulcs mentén összeilleszthető. Ezt a folyamatot ismétlik meg a másik tábla minden egyes kulcsára, ez egyébként a probe input.

Ez a hash join, és elég gyors még indexeletlen táblákra is, de mindkét táblán egyszer végigmegy. Nagyon nagy tábláknál (százmillió soros) azonban még ez a folyamat is lassúvá válik. Képzeljük el, hogy egy százmillió soros ténytáblát kell 5 egyenként tízezer soros dimenziótáblához joinolni. A tízezer soros dimenziótáblák kulcsaiból viszonylag kis költséggel felépíthető a build input, azonban ez után végig kell menni a százmillió soros ténytáblán, és a kulcsait lehashelve minden egyes dimenziótábla hashtáblájában utána kell keresni. Ez már nagyon nagy IO- és processzorköltséget igényel.

Ezen csillagsémás joinnak a költségét igyekszik csökkenteni a Bitmap Filter operátor, eleve kidobva azokat a sorokat a probe inputból, a ténytáblából, amelyekről valamilyen mágikus módon tudjuk, hogy biztos nincs hozzájuk passzoló sor a többi táblában. Az SQL Server 2008 ún. Bloom filtert használ erre a célra. Ez egy érdekes adatstruktúra, amellyel halmazokat lehet nagyon hatékonyan kezelni. Érdekessége: egy elemre tévesen azt mondhatja, hogy benne van a halmazban, pedig nincs, de sosem mondja azt, hogy nincs benne egy elem a halmazban, amit pedig a valóságban beleraktak. Mivel a szerver ezt a garantáltan *nem* egyező sorok kiszűrésére használja a hashtábla-keresések számának csökkentése érdekében, ezért nem probléma valamennyi false pozitív (azt hiszi, hogy benne van, de nincs) sor, mert a bennmaradt sorokra úgyis megnézzük hagyományos hash joinnal, hogy tényleg egyeznek-e a joinfeltételnek megfelelően.

A Bloom filter egy bittömbben tárolja a halmaz elemeit. Úgy működik, hogy ha be akarnak rakni egy elemet a halmazba, akkor azt n féle hash-függvénnyel is lehashelik, és a hash-értékeknek megfelelő biteket egyre állítják egy bittömbben. Azaz például n=10 hash-függvény alkalmazása esetén 10 bitpozíció lesz 1-re állítva, a hashek kimenetének megfelelő sorszámú. Más elemeket hozzáadva persze egyre több olyan bit lesz, ami már eleve 1 volt a korábban behelyezett elemek miatt, így itt információ veszik el, emiatt a határozatlanság az elem tesztelése során.

Hogy egy kulcs eleme-e a halmaznak, azt úgy nézik meg, hogy a kulcsot lehashelik a példabeli, mondjuk, 10 hash-függvénnyel, és megnézik, hogy ezek a bitek 1-re vannak-e állítva. Ha mind be van állítva, akkor az elem *valószínűleg* eleme a halmaznak. De nem biztos. Nyilván minél kisebb a tömb, és minél több elemet tárolunk benne, annál többször csal, jól kell megbecsülni a méretét és a hashek számát is, ez a hibahatár nem túl bonyolult valószínűség-számítással előre belőhető.

Az SQL Server csillag-Join esetén, miközben minden egyes dimenziótáblához felépíti a build inputot, azaz lehasheli a kulcsaikat, közben építi a Bloom filtereket is, ezt hívja Bitmap Filternek. Minden egyes dimenziótáblához készül tehát egy Bitmap Filter. Ezek után nekilát a ténytáblát soronként felolvasni. Normál esetben le kellene hashelni az összes kulcsoszlop-értéket a sorból, amely

valamelyik join-feltételben szerepel, és rákeresni a megfelelő dimenziótábla hashtáblájában. Ez lenne a sima hash join. Ehelyett azt csinálja a szerver, hogy végrehajtja a Bitmap Filterhez tartozó n féle hashképzést, és ráprobálja azokat az első, általa legszelektívebbnek gondolt dimenziótábla Bitmap Filterére. Ha az azt mondja, hogy nincs benne az elem a halmazban, akkor ez garantáltan azt jelenti, hogy ez a sor kiesik a join miatt, így sem ennek a dimenziótáblának a hashtáblájában, sem a többiében nem érdemes megnézni, benne van-e az elem. Ha az első bitmap filter nem ejtette ki a sort, akkor jön a következő dimenziótábláé. Ha mindegyiken átment a dolog, akkor még mindig megvan, ha nagyon kicsi is a valószínűsége, hogy fals pozitív a sor, azaz valójában nincs is párja a többi táblában, ezért ilyenkor még meg kell csinálni a rendes hash join procedúrát erre a sorra, összepárosítva az összes táblával.

Ha olyan bitmap filtereket csinál a szerver, amelyek elég szelektívek, azaz sok sort előre kiejtenek, akkor nagy nyereséget érhetünk el a kevesebb hash join miatt.

Okos a szerver, ha menet közben észreveszi, hogy nem a legszelektívebb filter van elöl, átrendezi a listáját. Ez végül is minimális statisztikázással nyomon követhető a részéről. A bitmap filtert csak párhuzamos tervek esetén használja az SQL Server *(2. ábra)*, hisz csak nagy költségű lekérdezéseknél érdemes vesződni vele, illetve az algoritmusból látható, hogy az jól párhuzamosítható.

A *2. ábrán* a bekarikázott részekben látható az a lépés, amely felépíti a Bitmap Filtert (a Bloom Filtert) a DimDate és a DimSalesTerritory táblákhoz. Miután a szűrők készen vannak, indul az alsó Table Scan operátor, amely a FactInternetSales2 táblán megy végig, előszűrve az adatokat az előbbi két Filterrel. A szűrési feltétel így néz ki:

```
PROBE([Opt_Bitmap1007],[AdventureWorksDW].[dbo].[FactInternetSales2].[SalesTerritoryKey] as [F].[SalesTerritoryKey],N'[IN ROW]') AND PROBE([Opt_Bitmap1008],[AdventureWorksDW].[dbo].[FactInternetSales2].[OrderDateKey] as [F].[OrderDateKey],N'[IN ROW]')
```

Az SQL 2005 is ismerte már a Bitmap Filtert, de csak statikusan, a terv generálása közben döntötte el, hogy használni fog filtert, míg az SQL 2008 a közbenső joinok végrehajtása közben is dinamikusan képes dönteni róla, hogy elég volt a hagyományos hash joinból, itt bizony bitmap filtereket kell építeni.

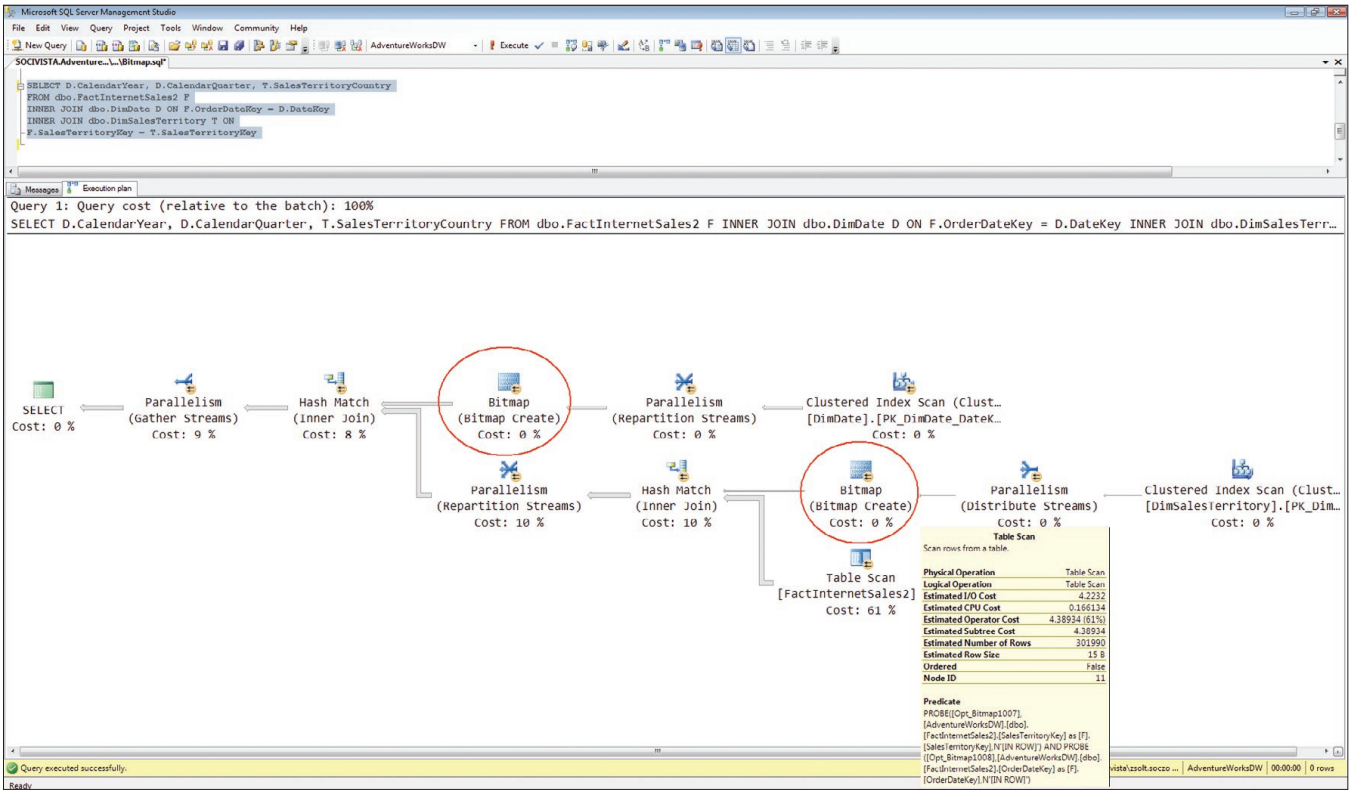
Tömörítés (adat és mentés) – Page Compression

Az adattárházak tábláit sokkal többet olvassák, mint módosítják. Erre a tényre épít az adatlapok tömörített tárolása, amely jelentős IO-költséget spórolhat meg viszonylag kevés processzorköltség árán.

Az SQL 2005 SP2 már bevezette a *vardecimal* tömörített típust, amely nem fix hosszúságon tárolja ezeket a számokat, csak olyan hosszan, amennyi az adott példány tényleges tárolásához kell. Például a 2.23 kevesebb helyet kér, mint a 2234234.23 vagy a 3.345353535345. Változó hosszúsággal ábrázolják tehát az egyébként fix hosszúságú decimal adatot, ezzel helyet spórolnak meg. Gondolom, nem kell mondanom, miért pont ezt a típust rakták be az SP2-be! Azért, mert a pénzmennyiségeket ebben szoktuk tárolni (nem kettes, hanem 10-es számrendszer alapú, ezért véges tizedes törteket pontosan tud ábrázolni, szemben mondjuk a reallel, ami 2-es számrendszer alapú).

Az SQL Server 2008 ezen a vonalon ment tovább, és már nemcsak a decimalt, hanem a többi fix hosszúságú adatot is tudja változó hosszal, azaz tömörítve tárolni.

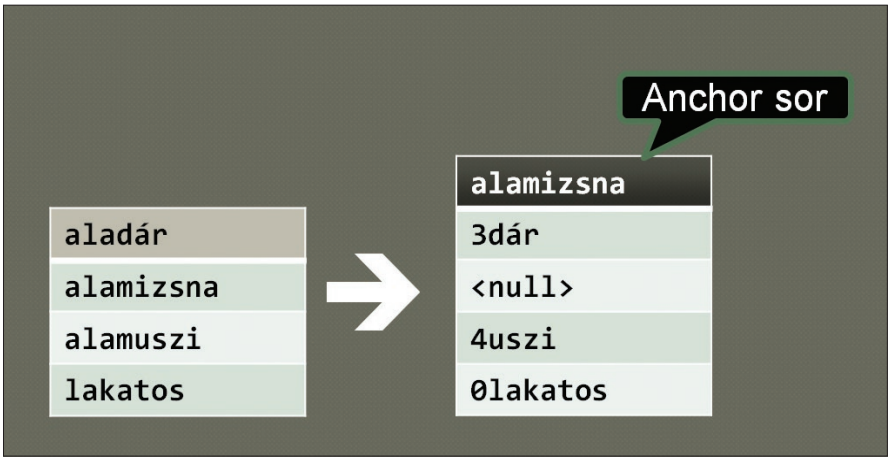
Mielőtt azonban megbeszélnénk az ösz-



2. ábra. Bitmap Filter a párhuzamos tervben

szes tömörítési módszert, nézzük, miért jó ez nekünk! A tömörítés elsődleges célja az IO-költségek csökkentése, ezáltal a lekérdezések gyorsítása. Az adatmódosítások nyilván lassulnak, ezért elsősorban általában csak olvasott adatokra érdemes használni. Nincs dráma a módosításnál, de pár százalékkal lassabb lehet. A backup is gyorsul, hisz kevesebb adatot kell kimásolni. A backup is tud tömöríteni 2008-ban, a kettő egymástól független, és használható együtt, erről is lesz szó hamarosan.

Mivel tömörebbek az adatlapok, jobban kihasználható a gép memóriája cache céljára, azaz egy 2-szeres tömörítés hasonló hatású, mint ha dupla annyi memóriánk lenne cache céljára. Ez ezért lehetséges, mert a felolvasott lapokat *nem* tömöríti ki a szerver a memóriában, hanem eredeti formájában tárolja.



3. ábra. Column-prefix tömörítése

Lehet tömöríteni adatot, indexet és akár egy tábla vagy index bizonyos partícióit is. Ez utóbbi nagyon jó, mert így az archív adatokat lehet tömörítve tárolni régebbi partíciókban, míg az éppen töltött adatokat tömörítés nélkül, hogy ne lassuljanak a DML-műveletek.

Hogyan tömörít az SQL Server 2008? Azért azt látni kell, hogy nem lehet egy zipet vagy egy rart berakni a szerverbe, mert bár az valószínűleg nagyobb tömörítési arányt érne el, de sokkal lassabb lenne tőle a feldolgozás. Olyan tömörítés kellett, ami elég sokat tömörít, de nem túl nagy költséggel. Egyféle technikáról bár beszéltem, a fix adatok változó hosszúságú kódolásáról. Ez működik a számokra és a char, nchar típusra. Kívülről persze ez nem látszik, az int továbbra is 4 bájtosként néz ki, annak ellenére, hogy belül lehet, hogy csak 1,5 bájtot igényel. **Row compression** néven érhető el ez a tömörítés.

Főleg szöveges adatok esetén azonban ez a módszer nem tudna nyereséget adni, maximum ostobán megszerkesztett hosszú char, nchar oszlopoknál, de van annyi eszünk, hogy változó hosszúságú adatokat ***n***varchar és társaiban tárolunk. Más módszer kell a tömörítésre, ez pedig a sorok közötti redundancia csökkentésével működik. Az első módszer az adatok első részében, prefixében levő redundanciát űzi el *(3. ábra)*.

Például az **ala**dár és az **ala**mizsna szavakban az alamizsna bájtokat csak egyszer írják le a lap fejlécében mondjuk 1-es indexszel, és a mezőkben csak 3dár és 3mizsna lesz. A 3 azt jelenti, hogy az anchor sor első három bájtját kell venni, majd mögé rakni a letárolt adatot (ala + mizsna). Ha egy sor értéke egyáltalán nem származtatható az anchor sor értékéből (lakatos), akkor persze nem nyerünk, hanem vesztünk a tömörítéssel. A storage engine persze észreveszi ezt, és fenntartja magának a jogot, hogy bár be van kapcsolva a tömörítés egy tábla lapjaira, egyes lapokat mégse tömörítve tároljon.

Ez a column-prefix tömörítés.

A másik módszer szótár alapú, azaz ha például a 3dár bájtok egy lapon 15 sorban is szerepelnek bármely oszlopban, akkor csak egyszer tárolják le a lap fejlécében, és a sorokba mutatókat raknak az adatszótár adott bejegyzésére.

Azaz a két módszer együtt működik, először a közös prefixeket emelik ki, majd megnézik az eredő képet, és azt a szótárazós módszerrel tovább tömörítik. Ez a két módszer együtt a page compression.

Persze a lapszintű tömörítések nemcsak szöveges, hanem bináris adatokra is mennek, csak így könnyebb volt szemléltetni a folyamatot.

Melyiket használjam? A row compressionnek jóval kisebb a költsége, ezért a gyakrabban lekérdezett vagy módosított *adatokhoz* ez megfelelőbb. Cserébe nem tud annyira tömöríteni.

A gyakran használt indexeket valószínűleg nem érdemes tömöríteni, csak azokat, amelyek nagyok, de ritkán használatosak.

Kis táblákra kár használni bármelyik módszert is, csak izzítjuk vele feleslegesen a procikat.

Index seek-eken (pontszerű lekérdezéseknél) nem sokat javít a tömörítés, mert egy-egy sor miatt akár 6-8 lapot is ki kell csomagolni, ami felesleges költség. Range seek-ekre vagy index scan-ekre már megéri, azaz, amikor nagyobb tartományokra kerestetünk, így eleve sok adatlapot érintene a lekérdezés (az előbbi példa hash joinja például ilyen volt). A sys.dm_db_index_operational_stats nézet megmutatja, melyik index mennyire és milyen módon van kihasználva, ez segíthet eldönteni, melyik indexeket érdemes tömöríteni.

A nagy adatokra, mint varchar(max) és társai *nem* működik a tömörítés, hisz az előbb leírt módszerek nyilvánvalóan nem mennek csak apró adatokra, ezeket inkább a hagyományos stream alapú tömörítésekkel lehet összepakolni. Mit lehet tenni, ha ezeket is tömöríteni akarjuk?

1. Az alkalmazás maga tömörít. A mai világban ez már nem nagy szám, szinte minden keretrendszerben megvannak hozzá a szükséges osztályok vagy függvények.

2. Tömörítő CLR-függvényt írunk, azzal tömörítünk a tárolás előtt, mondjuk, egy tárolt eljárásban.

3. Olyan CLR-típust implementálunk, ami tömörítve tárolja a belepumpált adatokat. A méretlimit feloldása miatt ez minden további nélkül lehetséges.

4. Az előző számban tárgyalt FILESTREAM oszlopot használjuk tömörített NTFS-könyvtárban. Ez nem tömörít olyan agresszíven, de elég gyors, és nincs vele semmi dolgunk a konfiguráláson kívül.

Tegyük fel, rájöttünk, kell nekünk a tömö-

rítés. Mielőtt azonban bezipelnénk az univerzumot, érdemes kicsit méricskélni, mit várhatunk el tőle, hisz az adatainktól nagyban függ, mekkora lesz a nyereségünk, ha egyáltalán lesz.

Az sp_estimate_data_compression_savings tárolt eljárással ki lehet próbáltatni, hogy egy adott tábla vagy index egy adott partícióján a row vagy page compression mennyit hozna a konyhára. Az sp persze nem áll neki a 80 exabájtos táblát betömöríteni, hanem mintavételezéssel készít egy kis mintatáblát a tempdb-ben, és azt csomagolgatja, majd ennek eredményét vetíti vissza (tudományosan: extrapolálja) az eredeti táblára.

Nézzünk egy konkrét példát (a kimenetet lerövidítettem, és átneveztem az oszlopokat):

```
exec sp_estimate_data_compression_savings 'Production',
'TransactionHistoryArchive', NULL, NULL, 'row'
exec sp_estimate_data_compression_savings 'Production',
'TransactionHistoryArchive', NULL, NULL, 'page'

object_name              index_id  eredeti méret (KB)  tömörített (KB)
------------------------ --------  ------------------  ---------
TransactionHistoryArchive  1       5136                3240
TransactionHistoryArchive  2       1144                968
TransactionHistoryArchive  3       1488                1240

object_name              index_id  eredeti méret (KB)  tömörített (KB)
------------------------ --------  ------------------  ---------
TransactionHistoryArchive  1       5136                1680
TransactionHistoryArchive  2       1144                816
TransactionHistoryArchive  3       1488                1144
```

Mit látunk? Az alapban 5,1 megabájtos adatból és rajta levő 2,7 megányi indexből row compressionnel 3,2 és 2,2 mega lesz. Az adat kb. a felére megy össze, ami nem rossz, hisz row compressionről van szó, ami nagyon gyors. Az indexet nem tudta úgy összenyomni, valószínűleg az indexekben levő int adatok jelentős része 2 bájt hosszan tárolható csak (nagy számok), így csak felére tömöríthető.

Lapszintű tömörítésnél az 5,1 megás adatok csak 1,7 megát foglalnak el, azaz 3-szoros tömörítést kapunk. Az indexek összmérete 2,7-ről 1,95 megára esik vissza, nem sokkal kisebbre, mint csak sortömörítéssel (2,2). Szóval indexnél esetünkben nem sokat ért egyik módszer sem, az adatok jellege miatt. Én lehet, hogy indexnél semmilyen vagy csak sor-, adatnál pedig lapszintű tömörítést használnék.

Az adatok tényleges tömörítése lapszinten:

```
alter table Production.TransactionHistoryArchive
rebuild with (data_compression = page);
```

Nézzük meg, mit nyerünk!

```
select index_id, index_level, index_type_desc,
page_count, compressed_page_count,
(select top 1 name from sys.indexes si
where si.object_id = s.object_id and
si.index_id = s.index_id) index_name
from sys.dm_db_index_physical_stats(DB_
ID(N'AdventureWorks'),
object_id('Production.TransactionHistoryArchive'), NULL, NULL,
        'DETAILED')
as s order by s.index_id, s.index_level desc
i_id i_level index_type_desc page_count
            compressed_pc index_name
---- ------- --------------- ---------- ------------ -----------
--------
1    1       CLUSTERED       1          0
             PK_...TransactionID
1    0       CLUSTERED       203        203
             PK_...TransactionID
2    1       NONCLUSTERED    1          0
             IX_...ProductID
2    0       NONCLUSTERED    125        0
             IX_...ProductID
3    1       NONCLUSTERED    1          0
             IX_...ReferenceOrderID
3    0       NONCLUSTERED    168        0
             IX_...ReferenceOrderID
```

Látható, hogy a clustered index 1. szintje (1. sor), azaz az index gyökérlapja nem page, csak row tömörített, de a 203 levélszintű lap, azaz az adatlapok mind page tömörítettek (2. sor).

Látható az is, hogy az indexek egyáltalán nincsenek tömörítve, az alter table clustered index esetén csak arra és az adatra vonatkozik, a nonclustered indexekre nem. Ha azokat is tömöríteni akarjuk, alter index parancsra lesz szükségünk:

```
alter index IX_TransactionHistoryArchive_ProductID
on Production.TransactionHistoryArchive
rebuild with (data_compression = page);

i_id i_level index_type_desc page_count
            compressed_pc index_name
---- ------- --------------- ---------- ------------ -----------
--------
1    1       CLUSTERED       1          0
             PK_...TransactionID
1    0       CLUSTERED       203        203
             PK_...TransactionID
2    1       NONCLUSTERED    1          0
             IX_...ProductID
2    0       NONCLUSTERED    90         89
             IX_...ProductID
3    1       NONCLUSTERED    1          0
             IX_...ReferenceOrderID
3    0       NONCLUSTERED    168        0
             IX_...ReferenceOrderID
```

Szinte minden indexlap page compressed lett (89), csak 1 maradt row compressed (a gyökéren kívül).

Hasonlítsuk össze a teljes tábla kiolvasásának IO-költségét a page tömörítéssel és simán:

```
set statistics io on
select * from Production.TransactionHistoryArchive

alter table Production.TransactionHistoryArchive
rebuild with (data_compression = none);

select * from Production.TransactionHistoryArchive

set statistics io off

Table 'TransactionHistoryArchive'. Scan count 1, logical reads
205, physical reads 0, read-ahead reads 0, lob logical reads 0,
lob physical reads 0, lob read-ahead reads 0.

Table 'TransactionHistoryArchive'. Scan count 1, logical reads
622, physical reads 0, read-ahead reads 0, lob logical reads 0,
lob physical reads 0, lob read-ahead reads 0.
```

Harmadára esik vissza az IO (633 → 205), a háromszoros tömörítés miatt.

Összegezve, a tömörítés igen hasznos szolgáltatás, amely hatalmas táblák esetén jelentősen csökkentheti az IO-költséget, így ha az a szűk keresztmetszet, akkor nem csak spórol a merevlemezzel, de még gyorsít is.

Backup Compression

A mentés tömörítésének célja a mentés méretének csökkentése, akár jelentős CPU-költség árán is.

```
backup database HungarySpatial
to disk='c:\temp\h.bck'

Processed 192936 pages for database 'HungarySpatial',
   file '...' on file 1.
```

```
Processed 2 pages for database 'HungarySpatial', file '...'
  on file 1.
BACKUP DATABASE successfully processed 192938 pages in
  153.180 seconds (9.840 MB/sec).

backup database HungarySpatial
to disk='c:\temp\hc.bck'
with compression

Processed 192936 pages for database 'HungarySpatial',
  file '...' on file 1.
Processed 1 pages for database 'HungarySpatial', file '...'
  on file 1.
BACKUP DATABASE successfully processed 192937 pages in
  73.008 seconds (20.645 MB/sec).
```

Látható, hogy ebben a konfigurációban (laptop) a tömörítés processzorigényét messze meghaladta az a nyereség, amit a kisebb backup fájl miatti kevesebb IO-val nyertünk, így a mentés kb. feleannyi idő alatt lefutott. Egyébként az eredeti mentés mérete 1508 megabájt volt, a tömörített 242. Ez igen nagy arány, jelentősen nagyobb, mint amit a laptömörítésnél láttunk (6-szoros). Cserébe sokat kellett dolgoznia a processzornak, de ha a mentés eleve kevésbé terhelt időszakban történik, amikor amúgy is ejtőzik a proci, akkor nagyon sokat nyerhetünk a tömörített mentéssel.

Érdekes kérdés lehet még, hogy a kétféle tömörítés használható-e együtt? A két szolgáltatás ortogonális egymásra, magyarul teljesen független, tetszés szerint ki-be kapcsolható mindkettő.

Egy durva tesztként tömörítsük be az előbbi adatbázis minden tábláját page compressionnel (ha van értelme, ha nincs), és készítsünk így is kétféle mentést. Egy táblázatban összefoglalom a mérések eredményét:

	Backup	Backup tömörítéssel
Adat nem tömörített	9,840 MB/s; 1508 MB	20,645 MB/s; 242 MB
Adat tömörített	11,405 MB/s; 644 MB	20,118 MB/s; 151 MB

A kétféle tömörítés együttes használatával 10-szeres tömörítést értem el ezen az adatbázison, ám nem sokkal rosszabb eredményt értem el tisztán backup tömörítéssel is. Az adattömörítés jótékony hatású a mentésre is, hisz a kevesebb „nyers” adat miatt kevesebb információt kell menteni.

Az is látható, hogy a mentés tömörítés a lehető legnagyobb nyereségre törekszik, nem törődve a CPU-költséggel, míg az adattömörítés beéri szerényebb tömörítéssel, de cserébe csak minimális CPU-időt használ.

Grouping Sets

A GROUP BY egyféle szempont, akárhány oszlop vagy kifejezés alapján csoportosít, aztán aggregáló függvényekkel dolgozhatjuk fel a többi oszlop adatait. Időnként azonban többféle szempont szerint is kellene csoportosítani. Például eladások év alapján, eladások év és termék alapján, eladások csak úgy, összesen stb. Ezt meg lehet tenni több lekérdezés UNION ALL-jával, amelyek eleje majdnem ugyanaz, csak a GROUP BY-ok mások.

Valami hasonlót csináltak már a korábbi SQL Server-verziók is CUBE és ROLLUP záradékkal a GROUP BY után. Ezekbe be volt építve, hogy ne csak a megjelölt oszlopok szerint csoportosítsanak, hanem egyre több csoportosító oszlopot elhagyva egyre durvább, egyre nagyobb átfogással dolgozzanak. A CUBE az összes GROUP BY oszlop minden kombinációját képezte, azaz N oszlop esetén 2^N-féle csoport keletkezett.

A ROLLUP N+1 szempont szerint csoportosít.

Az új GROUPING SET záradék a GROUP BY után arra szolgál, hogy mi, explicit megadhassunk több csoportosító feltételt is, így többféle dimenzió mentén aggregálhassunk. Azaz, ha akarunk, eljuthatunk a ROLLUP-ig, illetve a CUBE-ig is, de kihagyhatunk tetszés szerinti szempontokat is.

Ez nem más tehát, mint egy testre szabható átmenet a sima egyes GROUP BY és a mindenféle kombinációban aggregáló CUBE között.

Az alábbi példában zárójelben odaírtam, hogy melyik csoportosító feltétel melyik kimeneti sort generálja:

```
SELECT D.CalendarYear Y, D.CalendarQuarter Q,
T.SalesTerritoryCountry ST,
SUM(F.SalesAmount) AS SA
FROM dbo.FactInternetSales F
INNER JOIN dbo.DimDate D ON F.OrderDateKey = D.DateKey
INNER JOIN dbo.DimSalesTerritory T ON
F.SalesTerritoryKey = T.SalesTerritoryKey
GROUP BY GROUPING SETS (
(CalendarYear, CalendarQuarter, SalesTerritoryCountry), --(1)
(CalendarYear, CalendarQuarter), --(2)
(SalesTerritoryCountry), --(3)
()) --(4)

Y    Q    ST              SA
------ ---- ---------- ------------
NULL  NULL NULL          80450596 (4)
NULL  NULL Australia      1594335 (3)
NULL  NULL Canada        14377925 (3)
...
2001  3    NULL          3193633 (2)
2001  3    Canada         637982 (1)
2001  3    United States  2555651 (1)
2001  4    NULL          4871801 (2)
...
```

Egyébként nemcsak egyszerűbb a formátum az eddigi UNION-os megoldáshoz képest, hanem gyorsabb is a GROUPING SET, mert egy menetben megy végig az alaptáblán,

Az SQL Server 2008 új logója

és felhasználja a finomabb felbontású aggregált eredményeket a durvábban. A végrehajtási tervből látható lenne, hogy először kiszámolja a CalendarYear, CalendarQuarter, SalesTerritoryCountry csoportosításnak megfelelő összegeket, majd a CalendarYear, CalendarQuarter alapút ebből, és nem az eredeti adatokból adja össze, ezzel erőforrásokat megtakarítva.

Zárszó

Láthattuk, hogy az adattárházak kezeléséhez rengeteg új szolgáltatást kaptunk az SQL Server 2008-ban, amelyek legtöbbje nem csak adattárházakban, hanem tetszőleges alkalmazások esetén is jelentős teljesítménynyereséget okozhat.

Soczó Zsolt
MCSD, MCDBA, ASP.NET MVP
(http://soci.hu)
Research Engineer
Qualification Development Kft.

Kerüljön újra az élvonalba! Windows Server 2008 tanfolyamok a NetAcademiánál

Windows Server 2008

A NetAcademia Oktatóközpont a hivatalos Windows Server 2008 tanfolyamok teljes választékát kínálja: harcedzett rendszergazdák és a szakmába frissen belépők egyaránt megtalálják, amire szükségük van.

Windows 2008 MCSE Kit áttérőknek!

Frissítse üzemeltetői ismereteit Windows Server 2008-ra Windows 2008 MCSE Kit-tel! A három áttérő tanfolyamot (**6415, 6416, 6417**) így kedvezményesen végezheti el, mindössze 379 ezer Ft-ért.

A tanfolyamcsomag következő időpontjai:
augusztus 4., szeptember 9.

Windows 2008 MCSE Kit

6415
A Windows Server 2008 hálózati újdonságai
3 nap

6416
A Windows Server 2008 Active Directory újdonságai
3 nap

6417
A Windows Server 2008 alkalmazáskiszolgáló
3 nap

További Windows 2008-as tanfolyamaink

Üzemeltetés

6430
Windows Server 2008 kiszolgálók üzemeltetése
5 nap

6434
Windows 2008 PowerShell
3 nap

6427 / 6428
Windows Server 2008 IIS 7.0 és Terminal
4 nap

Infrastruktúra

6420 / 6424
Windows Server 2008 hálózat és tartomány alapok
5 nap

6421 / 6431
Windows Server 2008 haladó hálózati ismeretek és infrastruktúra kiszolgálók
5 nap

6423
Windows Server 2008 fürtözés (Clustering)
3 nap

6435
Windows 2008 hálózatok tervezése
5 nap

Tartományok

6425 / 6432
Windows Server 2008 tartományok kezelése és monitorozása
5 nap

6426
Windows Server 2008 Identity and Access Solutions
3 nap

A megjelölt tanfolyamok hallgatói egy ötfelhasználós Microsoft Windows Server 2008 Standard Edition-t (NFR) kapnak ajándékba. Az akció 2008. október 31-ig, vagy visszavonásig érvényes.

Microsoft GOLD CERTIFIED Partner

További információért kérjük, látogasson el honlapunkra (www.netacademia.net/windows2008), vagy keresse Szántó Zoltánt (szanto.zoltan@netacademia.net, tel.: 1/472-12-14).

netACADEMIA
A LEGJOBBAKAT TANÍTJUK.

A Fujitsu Siemens Computers áttörő IT infrastruktúra-megoldásai éppoly rugalmasan alkalmazkodnak a váratlan helyzetekhez, mint Ön.

A Fujitsu Siemens Computers FlexFrame™ Infrastructure rendszere tökéletesen illeszkedik az adatközponti környezetekbe. Az egyes szoftverfejlesztők által támogatott rugalmas IT-platform dinamikusan rendeli hozzá a szervererőforrásokat az egyes alkalmazásokhoz. A nagy teljesítményű, **négymagos Intel® Xeon® processzorral ellátott PRIMERGY RX300** szerverekre épülő megoldás minden speciális igényt kielégít az alkalmazások támogatása terén. Hiba esetén az érintett alkalmazások dinamikus áthelyezésével tartja fenn a zavartalan működést. Miért is jó ez Önnek? Mert a kivételes rugalmasságnak köszönhetően végre minden energiájával az üzleti működésre koncentrálhat. És ez csak egy a Fujitsu Siemens Computers Dinamikus Adatközponthoz készült innovatív megoldásai közül!

www.fujitsu-siemens.hu