Budapesti Műszaki és Gazdaságtudományi Egyetem

Virtuális gép image tárolási technológiák összehasonlítása, átjárási lehetőségek

Virtualizációs technológiák és alkalmazásaik

Házi Feladat

2009. ősz

Készítette: Tangl András

Bevezetés

A házi feladat elkészítése során négy elterjedt, szélesebb körben használt virtualizációs megoldást vizsgáltam meg. Ezek rendre: A VMware cég VMware Workstation és VMware Player, a Microsoft Windows Virtual PC illetve újabban a Microdoft Virtual PC, a Sun VirtualBox programok, és a QEMU emulátor által használt virtuális lemez tárolási technikák. A házi feladatom során elsőként ezeket vizsgáltam meg, majd foglalkoztam a nemrég elkészített és elfogadott Open Virtualization Format (OVF) elképzeléssel, mely igyekszik ezeket a technológiákat egy szabvány keretei közt átjárhatóvá tenni. Végül néhány szóban ismertetek egy szoftver, és library csomagot, melyek segítségével az egyes virtuális image-ket lehet egymás között konvertálni, illetve egyszerűen lehet ehhez hasonló saját programokat írni.

Virtuális lemezek

A virtuális lemezek fontos tulajdonsága, hogy egy vagy több fájlként vannak jelen a fizikai meghajtón. Bennük valamilyen módon a virtuális gép számára szükséges adatok tárolódnak, de a tárolási eljárás minden gyártó esetén más. Abban közös, hogy a virtuális lemezhez tartozó információkat, tárolási metódusokat, indexeket és offset értékeket egy leíróban helyezik el, mely lehet egy külön fájlban a lemez mellett, de sokszor magában az image-ben szerepel. Ez leginkább a virtuális gépet futtató programnak segít értelmezni a lemezt.

A .vmdk image

Ezt az image tárolási technológiát az 1998-ban alapított VMware cég hozta létre. Alapvetően a VMware Workstation, Player és Server termékeihez készült, de manapság már szélesebb körben támogatott, többek között a Sun xVM platform családja is képes használni.

A VMware virtuális lemez koncepció:
A VMware alapvetően két szempont szerint osztályozta a virtuális lemezeket:

- Egy virtuális lemez tárolható egyetlen nagy, összefüggő fájlban, vagy több kisebb fájl egy gyűjteményeként.
- A virtuális lemez méret szempontjából lehet előre allokált vagy dinamikus. Előbbi esetben a virtuális gép számára a szükséges összes lemezterületet a létrehozáskor lefoglalja szoftver, míg utóbbi esetben mindig csak akkora tárterületet sajátít ki, amennyire szüksége van.

Egy újonnan létrehozott virtuális lemez ezeknek a lehetőségeknek egy kombinációjaként áll elő.

A VMware-ben létrehozott virtuális lemez linkekből áll. A létrehozáskor mindig egy alap link jön létre. Ha meglévő image-ről a felhasználó készít egy snapshot-ot, akkor a virtuális lemez kiegészül egy delta linkkel (ezt néhol redo log linkként említik), mely a pillanatfelvétel óta történt változásokat tartalmazza. Így a virtuális lemez az alap és az egyes delta linkek láncolatából állhat össze:

Links in the chain that makes up the virtual disk

Egy alap és delta linkhez természetesen több delta link és tartozhat, így az ábrán megjelenítettnél jóval bonyolultabb láncolat is kialakítható.
Minden egyes link egy vagy több extent sorozatából áll. Egy extent a fizikai lemeznek egy régiója, rendszerint egy, a virtuális lemez által használt fájl.

A leíró:
Az alábbi képen a leíróra található egy példa (A ’#’ utáni szövegek megjegyzések):

```
% cat test.vmdk
# Disk DescriptorFile
version=1
CID=fffffffe
parentCID=ffffffff
createType="twoGbMaxExtentSparse"
# Extent description
RW 4192256 SPARSE "test-s001.vmdk"
RW 4192256 SPARSE "test-s002.vmdk"
RW 2101248 SPARSE "test-s003.vmdk"
# The Disk Data Base
#DDB
ddb.adapterType = "ide"
ddb.geometry.sectors = "63"
ddb.geometry.heads = "16"
ddb.geometry.cylinders = "10402"
```

Egy leíró mindig három különálló részre tagolódik, egy fejlécre (header), egy extent leíróra, valamint egy lemez adatbázisra (DDB – Disk DataBase).

A header részei rendre:

- a leíró verziója, melynek értéke alapbeállítás szerint 1,
- egy CID (Content ID), mely minden esetben egy 32 bites véletlen szám, és minden virtuális lemezhez egy egyedi érték tartozik.
- szülő CID, akkor tartozik hozzá érvényes bejegyzés, ha a linkhez tartozik egy szülő link. Ekkor a parentCID a szülő link CID-jével azonos 32 bites azonosító. Ha a linkhez nem tartozik szülő, akkor ez a mező a „CID_NOPARENT" konstans értéket veszi fel. A CID-k szerepe kettős, segítségükkel vizsgálható, hogy a szülő vagy gyermek link nem változott e meg. Lényeges szerepe csak az első esetben van, akkor ugyanis a parentCID-t érvényteleníteni kell.
- végül egy típus tartozik a leíróhoz. Ezek a következők lehetnek: monolithicSparse, vmfsSparse, monolithicFlat, vmfs, twoGbMaxExtentSparse, twoGbMaxExtentFlat, fullDevice, vmfsRaw, partionedDevice, vmfsRawDeviceMap, vmfsPassthroughRawDeviceMap, streamOptimized. A monolithic kulcsszó arra utal, hogy a virtuális lemez egyetlen fájlban tárolódik, míg a twoGbMaxExtent kulcsszó esetén több kisebb fájlból áll. A Sparse szó egy igény szerint növekvő lemezt jelöl. Flat lemez esetén a szükséges lemezterület előre le van foglalva. A vmfs csak ESX Server esetén használatosak. A fullDevice, vmfsRaw és partitionedDevice közvetlen hozzáférést biztosít egy fizikai lemezhez, míg a vmfsRawDeviceMap és a vmfsPassthroughDeviceMap ESX Server esetén valósítanak meg eszköz hozzárendelést.

A következő sorok mindegyike egy-egy extentet ír le. Definiálni kell a hozzáférés típusát (RW, RDONLY, NOACCESS), a szektorok méretét (ez alapbeállítás szerint 512 bájt lesz), az extent típusát (FLAT, SPARSE, ZERO, VMFS, VMFSSPARSE, VMFSRDM, VMFSRAW), és az extentet tartalmazó fájl nevét. A sor végén FLAT típusok esetén szükség van egy eltolás értékre (offset), mely a vendég operációs rendszer adatainak a helyét mutatja.

Az utolsó sorokban a virtuális lemezhez tartozó további információk találhatóak. Ezeket egyaránt a következő formában kell megadni:
ddb.<nameOFEntry>=”<value of entry>”
Az adapter típusai lehetnek „ide”, SCSI lemezek esetén „buslogic” vagy „lsilogic”, illetve ESX szervereknél „legacyESX”, ha az adapter típusa előre nem ismert. Az itt leírt mértani értékek (cilinderek, fejek, szektorok) mind az adaptertől függenek.

Pár szó az extentekről:
Négy típusuk van, lehetnek egyszerűek (fix méret esetén), „hosted sparse” (főleg változó méret esetén), „ESX Server Sparse” valamint „Stream-Optimized Compressed Sparse”. Az egyszerű extentek mindig egy fájl egy régióját vagy egy blokk eszközt jelölnek, az adatokhoz való hozzáférés pedig lineáris. A másik három típus a kisebb optimalizációs eltérésektől eltekintve azonos elven alapulnak. Ezek az extentek más-más tartalmú blokkokból épülnek fel. Közös bennük, hogy egy fejlécet, egy „Grain directory”-t, több „Grain table”-t és „Grain”-ket tartalmaznak. A header rész a blokk struktúráját írja le, eltolás értékeket és méreteket definiál. Az adatokhoz több szintű hozzáférést valósítanak meg. Az első szinten a

Grain Directory 32 bites bejegyzései, a GDE-k (Grain Directory Entry) szerepelnek, melyek mindegyike egy Grain table-re mutat. Ezek szintén 32 bites bejegyzésekből, GTE-kből (Grain Table Entry) állnak, melyek eltolás értékek, és az adatokat tartalmazó Grain-ekre mutatnak. Egy Grain Directory mérete mindig akkora, amekkorára szükség van, egy Grain Table 2KB, míg egy Grain 128 szektorból áll, így 64KB.

A .vhd image

Ez a Microsoft Windows Virtual PC program image tárolási technikája. Eredetileg a Connectix cég fejlesztette, egészen 2006 júliusáig, amikor a Microsoft felvásárolta. Ez a szoftver az alapja a Windows 7 XP módjának.

A Virtual PC virtuális lemez koncepció:
A virtuális lemezeknek három fajtáját írja le. Lehet fix, azaz a lemez mérete előre meghatározott és lefoglalt, lehet dinamikus, amikor a lemez mérete igény szerint nőhet, és lehet differenciális, mely esetben egy már létező szülő lemezhez viszonyított eltéréseket tartalmazza.
A leíró alapja egy lábjegyzet (footer), ez minden image típusnál megtalálható. Dinamikus és differenciális esetben a lábjegyzet mellett egy fejléc (header) és BAT (Block Allocation Table) is található.
A szektorok hosszát a Virtual PC fixen meghatározza, mérete 512 bájt.
A Virtual PC egy külön fájlban (.vmc) tárolja a vendég operációs rendszer hardverleírásait, és néhány részletet a gazda operációs rendszerről.

A footer:
A lábjegyzet részei rendre:

- egy cookie (8 bájt), mely egyértelműen azonosítja a virtuális merevlemez készítőjét. Ha a készítő a Microsoft Virtual PC vagy Virtual Server terméke, akkor itt egy „connectix” sztring található (utalva a termék eredeti készítőjére).
- egy tulajdonság (feature, 4 bájt) mező. Két tulajdonság írható le a segítségével: a legalacsonyabb helyi értékű biten „No Features Enabled”, a tőle eggyel balra levő biten „Temporary” (a lemez egy átmeneti lemez, ebben az esetben a lemez tartalma használat után törlődik). Ezután minden esetben egy egyes következik, majd nullák.
- egy fájl formátum verzió (File format version, 4 bájt). Ez a mező két, egyenként kétbájtos részre oszlik, egy minor és egy major verzióra. Előbbi a fájl formátum verziójával egyezik meg, mely jelenleg 0x0001000, míg utóbbi csak azt mutatja, ha a fájl formátuma úgy változott meg, hogy már nem kompatibilis a régi verzióval. Ekkor növelik eggyel a major értékét
- egy adat eltolás (Data Offset, 8 bájt) mező, mely dinamikus és differenciális lemezeknél a következő leíróhoz tartalmaznak egy eltolást a fájlon belül. Fix lemezek esetén kihasználatlan, értéke 0xFFFFFFFF.
- egy idő bélyeg (Time Stamp, 4 bájt), mely a virtuális lemez létrehozásának dátumát mutatja. Ez a 2000 jan. 1 12:00:00 AM UTC/GMT óta eltelt idő másodpercekben.

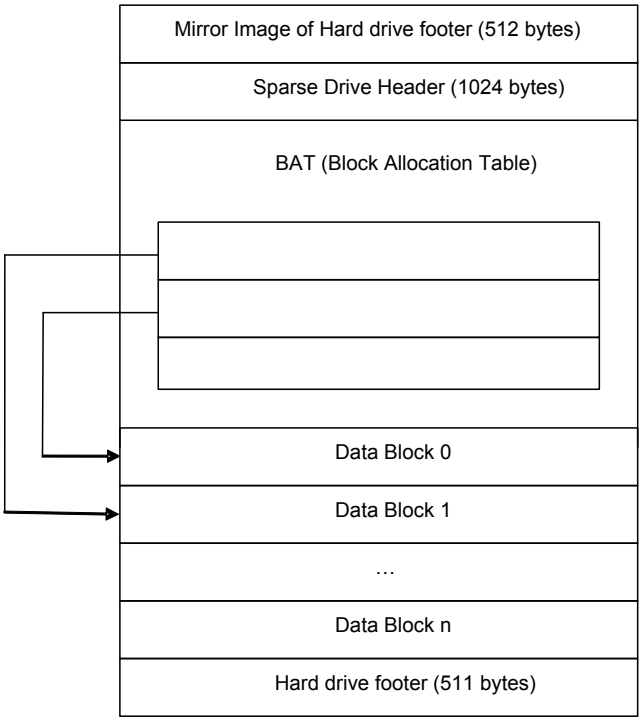
- egy létrehozó alkalmazás (Creator Application, 4 bájt) mező. Microsoft Virtual PC esetén az értéke „vpc”, Microsoft Virtual Server-nél „vp”.
- egy létrehozó verziója (Creator version, 4 bájt) mező, mely a virtuális lemezt készítő alkalmazás minor és major verzióját tartalmazza. Az értéke Virtual Server 2004 esetén 0x00010000, Virtual PC 2004-nél 0x00050000.
- a gazda gép leírója (Creator Host OS, 4 bájt). Például Windowsnál az értéke 0x5769326B (Wi2k).
- eredeti méret (Original Size, 8 bájt), mely információs célokra tartalmazza a virtuális lemez kezdeti, létrehozás utáni méretét bájtokban.
- jelenlegi méret (Current size, 8 bájt),
- lemez adatai (Disk Geometry, 4 bájt), tárolja a virtuális lemez jellemzőit, cilinderek, fejek, szektorok sávonkénti számát. ATA lemezek esetén ezen értékeket az ATA kontrollerből nyeri ki a Virtual PC.
- lemez típusa (Disk Type, 4 bájt), lehet „none”, foglalt, fix, dinamikus, differenciális.
- ellenőrzőösszeg (Checksum, 4 bájt), ez a lábjegyzet összes bájtjának egyes komplemense az ellenőrzőösszeg mező nélkül. Verifikálás során először a lábjegyzetet nézi a Virtual PC, ha hibás, akkor a header következik, amennyiben az adott lemezhez tartozik ilyen szekció. Ha ez is rossz, akkor a fájlt a program hibásnak ítéli.
- egyedi azonosító (Unique ID, 16 bájt), minden virtuális lemezhez egy egyedi 128 bites azonosító (UUID – Universally Unique IDentifier) tartozik. Differenciális lemezeknél a szülő lemezhez társításnak ez a mező az alapja.
- mentett állapot (Saved State, 1 bájt), ami egy egybites flag, a virtuális lemez mentett állapotára utal. Ha a lemez mentett állapotban van (a mező értéke 1), akkor nem lehet rajta tömörítő és bővítő műveleteket végezni.
- foglalt (Reserved, 427 bájt), nullákat tartalmaz.

A header:
Dinamikus és differenciális lemezek esetén az adat eltolás (Data Offset) mező a lábjegyzeten belül egy másodlagos struktúrára mutat, ami kiegészítő információkat tartalmaz a lemezről. Ez a fejléc, mely egy szektort foglal:

- cookie (8 bájt), értéke „cxsparse”.
- adat eltolás (Data Offset, 8 bájt), mely szintén egy következő struktúrára mutatna, de ez jelenleg még kihasználatlan, így értéke 0xFFFFFFFF.
- tábla eltolás (Table Offset, 8 bájt), tartalmazza az eltolást a BAT-hoz (Block Allocation Table).
- fejléc verziója (Header Version, 4 bájt), mely szintén egy major (felső két bájt) és egy minor (alsó két bájt) részekre oszlik. Viselkedésük azonos a lábjegyzet fájl formátum verziójával.
- maximális tábla bejegyzések (Max Table Entries, 4 bájt), mely a BAT-ban található összes bejegyzések számát jelöli. Ennek egyenlőnek kell lennie a diszken található blokkok számával.
- blokkok mérete (Block Size, 4 bájt), mindig kettő hatványának kell lennie. Az alapérték 2MB, azaz 0x00200000.

- ellenőrzőösszeg (Checksum, 4 bájt), mely a fejléc bájtjainak egyes komplemense az ellenőrzőösszeg mező nélkül. Ha az ellenőrzés meghiúsul, a fájl hibásnak minősül.
- szülő egyedi azonosítója (Parent unique id, 16 bájt), mely azonos differenciális lemezeknél a szülő lemez UUID-jével.
- szülő időbélyege (Parent time stamp, 4 bájt), a szülő lemez utolsó módosításának idejét mutatja, mely a 2000 jan. 1 12:00:00 UTC/GMT óta eltelt idő másodpercekben.
- foglalt (reserved, 4 bájt), minden biten 0 érték található.
- szülő neve (parent unicode name, 512 bájt).
- szülő helymeghatározója (parent locator entries, 8x24 bájt), ezek egy, a platformra jellemző elérési utat tartalmaznak, melyek segítségével a gyermek virtuális lemez azonosítani tudja a szülő lemezt a fizikai meghajtón.
- foglalt (reserved, 256 bájt), nullákat tartalmaz.

Block Allocation Table:
Eltolásokat tartalmaz az adatokat tartalmazó blokkokra. A nem használt bejegyzések értéke 0xFFFFFFFF. A méretét a rendszer a lemez létrehozásakor számolja ki, ugyanis dinamikus és differenciális lemez esetén is meg adni egy maximális méretet, melyet elérve a lemez már nem terjeszkedik tovább.



Az adatblokk adatokból és szektor bittérképből áll. Dinamikus lemeznél a bittérkép jelzi, hogy mely szektor tartalmaz érvényes adatot (1-sek), és mely szektorokat nem módosították még soha (0-k). Differenciális lemez esetén a bittérkép mutatja, mely szektorok vannak a differenciális lemezen (1) és melyek a szülőben (0). A bittérkép az 512 bájtos szektor határhoz simul.

Dinamikus Lemez létrehozása:
A blokkok igény szerint jönnek létre. Az újonnan létrehozott virtuális lemez csak az adat struktúrát tartalmazza. Ha adatot írnak a lemezre, akkor a dinamikus lemez kiterjesztődik, új blokkot hozva létre. Ezzel egy időben a BAT frissül, mutatókat hozva létre az új blokkokra.

Differenciális lemezek:
A differenciális lemez megnyitásakor mind a lemez, mind pedig a szülő lemez megnyílik. Ha a szülő lemez is egy differenciális lemez, akkor egy egész láncolat nyílhat meg.
Fontos, hogy a szülő lemezt ne módosítsuk. Ha mégis megtesszük, akkor az alatta levő differenciális lemezek érvénytelenné válnak.

Lemezek felosztása:
A fejléc és a lábjegyzet különállósága itt kap értelmet. Ha ugyanis a virtuális lemez mérete nagyobb lesz, mint a gazdagép által maximálisan támogatott fájlméret, akkor lehetőség van a lemez darabokra tördelésére. A darabolt részeknek se header, se footer része nincsen. Csak az utolsó rész végén van egy footer. A darabolt részeknek ugyanabban a könyvtárban kell lenniük. Legfeljebb 64 különálló rész hozható részre.

A QCOW2 formátum

Ez a formátum egy kicsit kilóg a sorból, ugyanis nem egy tisztán virtualizációs image tárolási technológiája, hanem egy processzor emulátorhoz tartozik. Ez a QEMU, ami ingyenes szoftver és Fabrice Bellard írta. A QEMU-ban egyaránt lehetőség van processzor és operációs rendszer emulációjára. Az emulátor egyaránt futtatható UNIX alapú rendszereken (Linux disztribúciók, Darwin/MacOSX) valamint különböző Windows verziókon is.

A QEMU virtuális lemez koncepció:
Egy QCOW2 virtuális image fájl jellemzően a következőket tartalmazza:

- egy fejléc,
- egy L1 tábla,
- egy hivatkozás számláló (refcount) tábla,
- egy vagy több refcount blokk,
- snapshot fejlécek, 8 bájtokban tárolva,
- L2 táblák, minden egyes tábla egy klaszter hosszú,
- adat klaszterek.

A fejléc:

- Az első négy bájt (magic) foglalt, a Q, F, I karaktereket és 0xfb értéket tárolja,
- a következő négy bájtban található a verzió száma, mely jelenleg 1 vagy kettő lehet,
- egy nyolcbájtos eltolás (backing-file-offset), mely a fájl elejétől addig a sztringig tart, ami ennek a fájlnak az elérési útját tárolja. Copy-on-write image esetén a szülő lemezhez tárolja az útvonalat. (CoW image esetén a lemez csak egy másik lemezhez viszonyított eltéréseket tartalmazza.)

- egy négybájtos mező (backing-file-size), mely az előbb említett útvonal hosszát tartalmazza,
- egy négybájtos címzés (cluster-bits), mely két részre oszlik, az alacsonyabb helyi értékű bitek klaszteren belüli címzésre használhatók, míg a magasabb helyi értéken levőn az L2 tábla címzésében segítenek.
- a következő 8 bájt tartalmazza a virtuális lemez méretét bájtokban mérve,
- egy négybájtos mező (crypt-method) a titkosítás módját írja le. Ha az értéke 0, nincs titkosítás, ha pedig 1, akkor a lemez 128 bites AES kódolással van titkosítva. Ebben az esetben egy 16 karakteres jelszóval védett a lemez.
- az L1 tábla mérete (l1_size, 8 bájt), az L1 táblában található bejegyzések számát tartalmazza,
- a nyolcbájtos l1_table_offset mező egy eltolást definiál, mely az L1 táblára mutat,
- egy nyolcbájtos refcount_table_offset tartalmazza a hivatkozás számláló (Refcount) táblához az eltolást, míg a következő négy bájt (refcount_table_clusters) ennek a táblának a klaszterekben mért nagyságát tárolja,
- végül egy négybájtos mező (nb_snapshots) tartalmazza a lemezről létrehozott pillanatfelvételek számát, egy nyolcbájtos mező (snapshot_offset) pedig minden snapshot fejlécéhez tartalmaz eltolást.

Kétszintű keresés:
A QCOW virtuális lemez tárolási technológia alapja a klaszter, ami 512 bájtos szektorok összessége. Egy bejegyzés megtalálásához szükség van a klaszterek megtalálására, valamint ezen belül további keresés indokolt. A gyors és hatékony kereséshez egy kétszintű indexelést valósítottak meg. Legfelső szinten található az L1 tábla, melynek bejegyzései eltolás értékek, és fejléc cluster_bits értéke szerint három részre oszlik. Első rész a klaszteren belüli cím, középső része az L2 táblákhoz eltolás, míg a fennmaradó részével az L1 táblában indexelhetünk. Következő szinten L2 táblák sokasága szerepel, ezek mind eltolás értékeket tartalmaznak, melyek klasztereket címeznek meg. Végül a legalsó szinten maga az adattároló klaszterek vannak, melyekben a keresés a már említett cluster_bits mező megfelelő részének használatával történik.

Hivatkozás számlálás (RefCount Table):
A szabad helyek növelésére a QCOW2 technológiában egy rendkívül egyszerű megoldást választották. Egy külön, erre a célra fenntartott és folyamatosan frissített táblában tárolja minden egyes klaszterre a rá történt hivatkozások számát. Ha túl régen nem olvasták vagy írták az adott klasztert, akkor a terület felszabadul.

Snapshot:
Hasonlatos a CoW megoldáshoz, azzal a különbséggel, hogy egy snapshot nem írható, viszont az eredeti lemez igen. CoW-nál az eredetit kell read-only módba állítani, és a differenciális lemez az, ami írható. Minden snapshot egy csak olvasható mentés az image egy korábbi állapotáról. Snapshot fejléc:

- a snapshotnak van neve (16 bájt) és azonosítója (16 bájt), amiket egy-egy sztring reprezentál

- a snapshotnak van egy másolata az eredeti L1 tábláról egy l1_table_offset (64 bájt) és egy l1_size (32 bájt) formájában,
- a dátumokkal azt tároljuk, hogy mikor készült a snapshot, 32 bájtos date_sec és date_nsec, valamint egy 64 bájtos vm_clock_nsec mezők szolgálnak erre.
- vm_state_size (32 bájt) az image méretét adja, amikor a mentés elkészült,
- extra_data_size (32 bájt) az adatbájtok számát tárolja.

A .vdi image

A VirtualBox image tárolási technológiája. Eredetileg egy német cég, az Innotek készítette, míg 2008 februárjában megvette a SUN, és jelenleg az xVM platform része.

A VirtualBox lemez koncepció:
Egy VirtualBox virtuális lemez méret szerint lehet fix vagy dinamikus, hozzáférés szerint pedig normál, állandó (csak olvasható), vagy „writethrough”. Fix méretű lemez esetén a létrehozáskor megadott lemezterület előre lefoglalódik, míg dinamikus esetben igény szerint nő.
A VirtualBox a virtuális géphez rendelt és beállított hardverek leírását egy külön XML fájlban tárolódik a virtuális lemez mellett.
A program támogatja a pillanatfelvételek készítését, és differenciális lemezeket is létrehozhatunk a segítségével. Ez utóbbi esetben egy új virtuális gépet kell készíteni, melynek alapja egy pillanatfelvételkor létrejött .vdi virtuális image fájl.
Minden virtuális lemez egy egyedi UUID azonosítóval rendelkezik, mely pillanatfelvételek megnyitásakor segít azonosítani a szülő lemezt.

Egy jellegzetes fejléc szerkezete a következő:

```
0000      3C 3C 3C 20 53 75 6E 20 78 56 4D 20 56 69 72 74    <<< Sun xVM Virt
0010      75 61 6C 42 6F 78 20 44 69 73 6B 20 49 6D 61 67    ualBox Disk Imag
0020      65 20 3E 3E 3E 0A 00 00 00 00 00 00 00 00 00 00    e >>>
0030      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0040      7F 10 DA BE                                       Image Signature
                      01 00 01 00                           Version 1.1
                                  90 01 00 00               Size of Header(0x190)
                                              01 00 00 00   Image Type (Dynamic VDI)
0050      00 00 00 00                                       Image Flags
                      00 00 00 00 00 00 00 00 00 00 00 00   Image Description
0060-001F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0150      00 00 00 00
                      00 02 00 00                           offsetBlocks
                                  00 20 00 00               offsetData
                                              00 00 00 00   #Cylinders (0)
0160      00 00 00 00                                       #Heads (0)
                      00 00 00 00                           #Sectors (0)
                                  00 02 00 00               SectorSize (512)
                                              00 00 00 00    -- unused --
0170      00 00 00 78 00 00 00 00                           DiskSize (Bytes)
                                  00 00 10 00               BlockSize
                                              00 00 00 00   Block Extra Data (0)
0180      80 07 00 00                                       #BlocksInHDD
                      0B 02 00 00                           #BlocksAllocated
                                  5A 08 62 27 A8 B6 69 44   UUID of this VDI
0190      A1 57 E2 B2 43 A5 8F CB
                                  0C 5C B1 E3 C5 73 ED 40   UUID of last SNAP
01A0      AE F7 06 D6 20 69 0C 96
                                  00 00 00 00 00 00 00 00   UUID link
01B0      00 00 00 00 00 00 00 00
                                  00 00 00 00 00 00 00 00   UUID Parent
01C0      00 00 00 00 00 00 00 00
                                  CF 03 00 00 00 00 00 00   -- garbage / unused --
01D0      3F 00 00 00 00 02 00 00 00 00 00 00 00 00 00 00   -- garbage / unused --
01E0      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   -- unused --
01F0      00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   -- unused -
```

Átjárási lehetőségek

Az előbbiekben ismertetett virtuális adattárolási eljárások túl specifikusak, egy korábban az egyik gyártó szoftverével létrehozott virtuális gép nem vihető át közvetlenül egy másik cég megoldásába, hiszen ezek nem tudják értelmezni a másik formátumát. (Ez alól kivételt képez a VirtualBox, mely felismeri és kezeli a VMware és Virtual PC image fájlokat is.) Ennek a problémának a feloldására két lehetőség van. Az egyik megoldás, hogy egy szabványos felületet definiálnak, melyet minden gyártó támogat, és ennek segítségével csak a szabvány ismerete szükséges, a többi virtuális lemezé nem. Ez értelemszerűen egyszerűbben implementálható, hiszen nem kell számtalan lemez specifikációját ismerni. Ez a megoldás a nemrég elfogadott OVF szabvány. Másik lehetőség egy konvertáló program készítése, mely minden, vagy több image formátumot ismer, és ezeket egymás közt szabadon lehet átvinni. Ebből több is van, például qemu-img, .Net DiscUtils. Én csak ez utóbbival foglalkoztam.

Az .ovf formátum

Az Open Virtualization Format egy nyílt forráskódú szabvány virtuális készülékek dobozolására és szétosztására. A szabvány nyílt, biztonságos, hatékony, hordozható, kiterjeszthető, platform független . A szabványt 2007 szeptemberében fogadta el a Distributes Management Task Force (DTMF), kidolgozói pedig a Dell, HP, IBM, Microsoft, VMware és XenSource cégek. 2009 áprilisa óta a VirtualBox is támogatja a szabványt.

Egy OVF csomag tartalma:

- egy OVF leírót, .ovf kiterjesztéssel,
- nulla vagy egy OVF manifest fájl (.mf),
- nulla vagy egy OVF tanúsítvány, .cert kiterjesztéssel,
- nulla vagy több diszk image,
- nulla vagy több kiegészítő forrás fájlok, például ISO image-k.

Az OVF-nek nincsen szüksége külön virtuális lemez formátumra, de annak a lemeznek, amely eleget tesz az OVF specifikációnak, tartalmaznia kell egy URI-t, amely segít a lemez értelmezésében. Ennek értelmében egy OVF támogatással rendelkező virtuális platform felismeri és képes használni az összes olyan virtuális image formátumot, melynek gyártója szintén támogatja a szabványt.

OVF leíró:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Generated by VMware ovftool 1.0.0 (build-166674), User: tungi, UTC time: 2009-10-27T11:15:16.661928Z-->
<Envelope vmw:buildId="build-166674" xmlns="http://schemas.dmtf.org/ovf/envelope/1" xmlns:cim="http://schemas.dmtf.org/wbem/wscim/1/common"
xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1" xmlns:rasd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingDa
xmlns:vmw="http://www.vmware.com/schema/ovf" xmlns:vssd="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <References>
    <File ovf:compression="gzip" ovf:href="1-disk1.vmdk.gz" ovf:id="file1" ovf:size="833414997"/>
  </References>
  <DiskSection>
    <Info>Virtual disk information</Info>
    <Disk ovf:allocationUnits="1073741824" ovf:capacity="8" ovf:capacityAllocationUnits="byte * 2^30" ovf:diskId="vmdisk1"
    ovf:fileRef="file1" ovf:format="http://www.vmware.com/interfaces/specifications/vmdk.html#streamOptimized" ovf:populatedSize="2374238208"/>
  </DiskSection>
  <NetworkSection>
    <Info>The list of logical networks</Info>
    <Network ovf:name="nat">
      <Description>The nat network</Description>
    </Network>
  </NetworkSection>
  <VirtualSystem ovf:id="Ubuntu">
    <Info>A virtual machine</Info>
    <Name>Ubuntu</Name>
    <OperatingSystemSection ovf:id="93" vmw:osType="ubuntuGuest">
      <Info>The kind of installed guest operating system</Info>
    </OperatingSystemSection>
    <VirtualHardwareSection>
      <Info>Virtual hardware requirements</Info>
      <System>
        <vssd:ElementName>Virtual Hardware Family</vssd:ElementName>
        <vssd:InstanceID>0</vssd:InstanceID>
        <vssd:VirtualSystemIdentifier>Ubuntu</vssd:VirtualSystemIdentifier>
        <vssd:VirtualSystemType>vmx-07</vssd:VirtualSystemType>
      </System>
      <Item>
        <rasd:AllocationUnits>hertz * 10^6</rasd:AllocationUnits>
        <rasd:Description>Number of Virtual CPUs</rasd:Description>
        <rasd:ElementName>1 virtual CPU(s)</rasd:ElementName>
        <rasd:InstanceID>1</rasd:InstanceID>
        <rasd:ResourceType>3</rasd:ResourceType>
        <rasd:VirtualQuantity>1</rasd:VirtualQuantity>
      </Item>
    </VirtualHardwareSection>
  </VirtualSystem>
</Envelope>
```

Az OVF leírója egy XML dokumentum, mely információkat tartalmaz a termékről, a hardver követelményekről és az engedélyekről. A dokumentum négy különböző névteret tartalmazhat, ezek rendre ovf, ovfenv, rasd, vssd. A leíró részei részletesebben:

1. Boríték elem (Envelope Element): ez az elem írja le a virtuális gép számára szükséges összes metaadatot. Ez tartalmaz egy verzió, hivatkozás lista, metaadat, tartalmak leírása és üzenet specifikáció mezőt.
2. Fájl hivatkozások (File References): lehetővé teszik, hogy egyszerűen meghatározza a külső eszköz az OVF csomag integritását anélkül, hogy szükséges lenne a teljes struktúrát értelmezni. Ezenkívül az OVF csomag egyszerűen manipulálható, nem jár információvesztéssel. A „File” elemet el kell látni egy egyedi azonosítóval („ovf:id”).
3. Tartalom elem (Content Element): A virtuális gép beállításai egy „VirtualSystem” vagy egy „VirtualSystemCollection” elemben vannak definiálva. Ezek szintén rendelkeznek egy egyedi azonosítóval.
4. Virtuális hardver leírása (Virtual Hardware Section): Minden „VirtualSystem” elem tartalmazhat egy vagy több „VirtualHardwareSection” gyermeket, melyben a virtuális gép számára szükséges hardverek leírása található. Minden egyes hardverleírás CIM (Common Information Model) osztályokon alapul. A virtuális hardver leírása nem

más, mint „Item” elemek sorozata, melyek jelen esetben a CIM osztályok reprezentációi.

5. Core metaadat szekció (Core MetaData Section): az OVF több ilyen szekciót is definiál, ezekből egynek sem kell kötelezően szerepelnie, és valamennyiből több is jelen lehet. Lehetőségek:
    a. lemez leírása (DiskSection): ez a rész a merevlemezről tárol metaadat információkat. Minden egyes lemezt egy „Disk” elem reprezentál.
    b. hálózat (NetwotkSection),
    c. forrás felhasználás (ResourceAllocationSection): ez tartalmazz azon erőforrások listáját, melyekre szükség van az OVF csomag telepítésekor.
    d. megjegyzések (AnnotationSection),
    e. termék (ProductSection): termék információk szerepelnek, mint például név, verzió, eladó.
    f. szerződés (EulaSection; End-user licence agreement),
    g. indítási-leállítási paraméterek (StartupSection),
    h. fejlesztői rész (DeploymentOptionSection): ide lehet leírni az OVF csomag készítőjét, fogyasztót, stb.
    i. operációs rendszer leírása (OperatingSystem Section),
    j. telepítési útmutató (InstallSection).

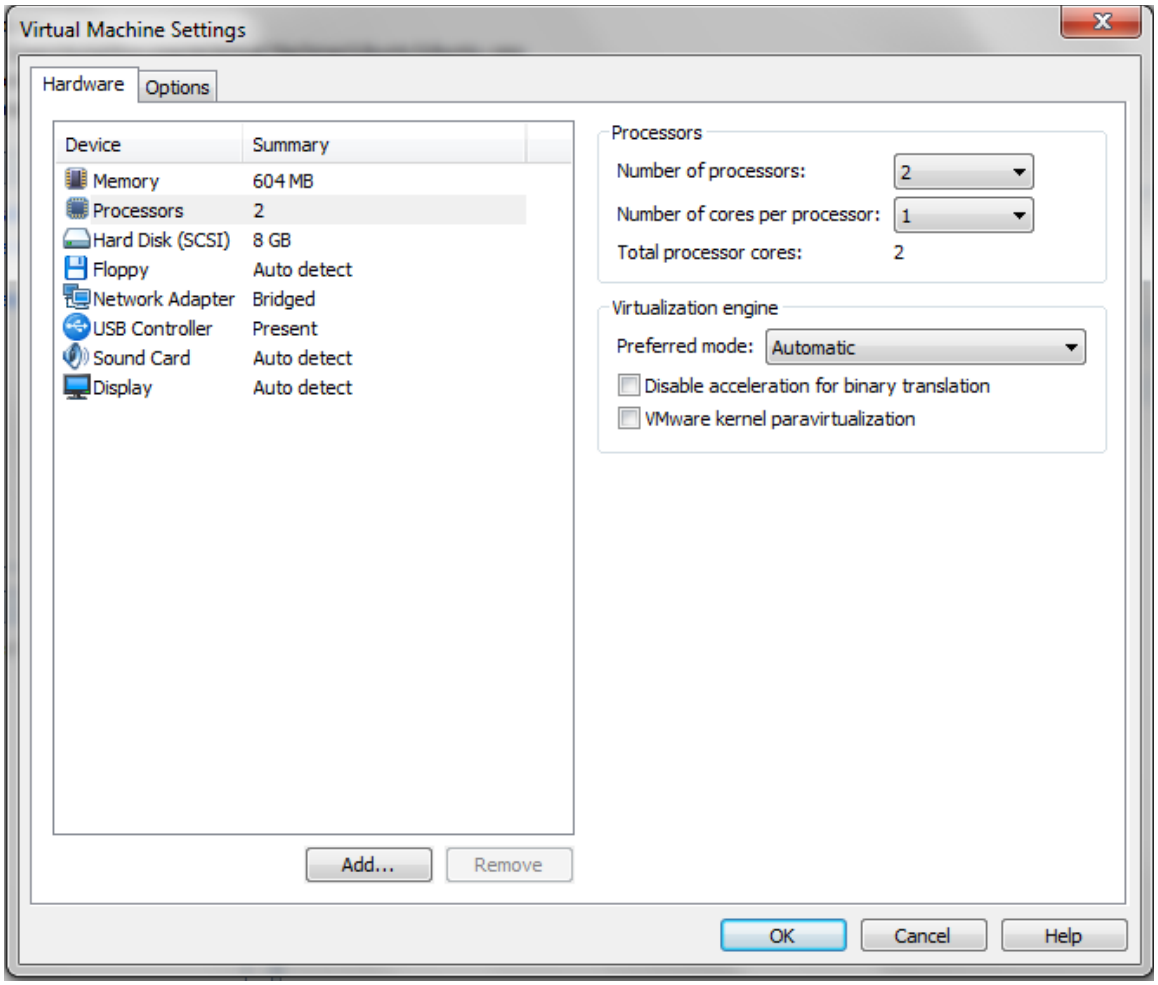
OVF környezet:
Egy „ovf-environment.xsd” nevű XML séma definiálja vendég szoftvernek és a fejlesztett platformnak a kölcsönhatását. A környezet teszi lehetővé a vendég szoftvernek, hogy a fejlesztett platformról információkhoz férjen hozzá, mint például az OVF leíró felhasználó specifikus értékei. A környezet egy „protocol” és egy „transport” részre tagolódik. Előbbi az XML dokumentum formáját és szemantikáját, míg utóbbi a vendég platform és a fejlesztett platform közti kommunikáció formáját határozza meg.

.Net DiscUtils

Jelenleg a 6. verzió érhető el. Ez egy szoftvercsomag és keretrendszer, mely segítségével egyrészt image formátumokat lehet egymást közt konvertálni, másrészt olyan programok készíthetőek, amik a támogatott fizikai és virtuális fájlrendszereket használják. Jelenleg a csomagban a következők vannak implementálva: FAT, NTFS (jelenleg csak olvasásra és írásra van lehetőség, formázásra nem), ISO, VHD, XVA (Xen Virtual Appliance), VMDK, VDI. Ezenkívül a csomag a SCSI lemezekhez is hozzáférést biztosít.

OVF és DiscUtils összehasonlítása

A fenti két megoldást próbaképpen ki is próbáltam. A kísérlet során VMware-ben egy új Ubuntu virtuális gépet hoztam létre, majd a telepítés befejeztével bezártam. Ezután az ingyenesen elérhető VirtualBox programban próbáltam megnyitni a virtuális gépet. Természetesen ebben a programban VMDK kiterjesztésű fájlok bármilyen köztes megoldás nélkül is használhatóak, de most a cél a fenti két technológia kipróbálása volt.
A VMware-ben létrehozott virtuális gép tulajdonságai:



A DiscUtils honlapjáról egy kis alkalmazás csomag is letölthető, melyben többek között egy konvertáló program is található, ezt használtam a konvertáláshoz. Ezt parancssorból kell elindítani, és paraméterül kell adni a céllemez formátumát, valamint a forrás lemez és céllemez helyét. A konvertálás végeztével rendelkezésemre állt egy VDI kiterjesztésű fájl, melyet már szabadon fel lehet használni egy új virtuális gép elkészítéséhez.

A VirtualBox-ban létrehoztam egy új virtuális gépet, melyhez virtuális lemezként az előzőekben elkészített fájlt adtam hozzá. A virtuális gép létrejött, és a következő tulajdonságokkal rendelkezett:

Az OVF kipróbálásához szükséges egy plusz program, mely a VMware oldaláról ingyenesen letölthető, ez a VMware OVF Tool. Ezzel szintén parancssort használva létrehozható az ovf csomag, a programnak paraméterül csak egy forrás VMX fájlt és cél elérési utat kell megadni. Miután létrejött a csomag, a VirtualBox-ban semmi mást nem kell tenni, mint importálni.

Az így létrejött virtuális gép tulajdonságai:

A fenti képeken látható az egyik legnagyobb előnye az OVF-nek. A konvertálás egyik legnagyobb hátránya, hogy a virtuális gép létrehozása során a hozzá rendelt hardver erőforrások elvesznek, hiszen az ehhez tartozó leírások nem a lemeznek része, hanem egy külön,(VMware esetén VMX kiterjesztésű) fájlban található. Ezen kívül egy nagy probléma, hogy szükség van konvertáló programra, amit értelemszerűen a gyártó nem fog elkészíteni, hiszen számtalan célformátumot kellene benne implementálnia. Mindkét problémára megoldás az OVF alkalmazása. Az előbbi gond megoldása maga az OVF fájl, mely már tartalmazza a hardverek leírását. Utóbbi szempontból nézve pedig OVF használata esetén a gyártónak csak egy olyan programot kell készítenie, vagy kiegészítéssel kell ellátnia a platformját, mely a korábbi virtuális gépről egy OVF csomagot készít.
Egyébként mind konvertálással mind pedig OVF használatával az eredetivel azonosan, egy működő operációs rendszer a végeredmény.

Források:

1. VMware VMDK formátuma:
   http://www.vmware.com/appliances/getting-started/learn/ovf.html
   http://www.dmtf.org/standards/published_documents/DSP0243_1.0.0.pdf
2. QCOW2 formátum:
   http://www.gnome.org/~markmc/qcow-image-format.html
3. Microsoft VHD formátuma:
   http://technet.microsoft.com/en-us/virtualserver/bb676673.aspx
   http://download.microsoft.com/download/f/f/e/ffef50a5-07dd-4cf8-aaa3-442c0673a029/Virtual%20Hard%20Disk%20Format%20Spec_10_18_06.doc
4. SUN VDI formátuma
   http://forums.virtualbox.org/viewtopic.php?t=8046
5. DiscUtils konvertáló eszköz és library csomag:
   http://discutils.codeplex.com/