ATA Over Ethernet: merevlemezek a helyi hálózaton

Napjainkra az ATA felületű merevlemezek olcsóbbak lettek, mint a szalagok, a címben szereplő egyszerű, új megoldással pedig archívumok, biztonsági mentések vagy éles használat céljára építhetünk tárolótömböket.

Előbb-utóbb mindenki kifogy a tárolóhelyből. Szerencsére a merevlemezek egyre olcsóbbak, miközben kapacitásuk folyamatosan növekedik. Hiába azonban a több hely, hiszen egyre többet használunk el, így újra szűkébe kerülünk.

Bizonyos adatfajták természetüknél fogva méretesek. A mozgóképek például mindig nagy mennyiségű helyet foglalnak. A vállalkozásoknak sokszor kell mozgóképeket tárolniuk, különösen, mióta elkezdődött a videómegfigyelő rendszerek térnyerése. Sőt, még otthoni gépükön is sokan szeretnek filmeket nézni vagy szerkeszteni.

A biztonsági mentés és a redundancia minden számítógépeket használó vállalkozás számára fontos fogalom. A valóság nagyjából az, hogy bármennyi tárolókapacitásunk is van, sose árt még egy kicsit több. Elég csak arra gondolni, hogy még az elektronikus leveleknek szánt tárolóhely is folyton kevés – az internetszolgáltatók bizonyára tudnának erről mesélni.

A korlátlan tárolóhely akkor válhat valósággá, ha a meghajtók kikerülnek a számítógépházból, és ezzel a tárolóeszközök függetlenekké válnak az őket használó számítógépektől. A rugalmasságnak az egymással együttműködő összetevők szétválasztásával elért növelésére több területen is láthatunk példát, nem csak az adattárolásén. A moduláris forráskód rugalmasabban használható az előre nem látott igények kielégítésére, és egy több részből álló sztereó rendszer is érdekesebb összeállításokban telepíthető, mint egy mindent egyetlen dobozban tartalmazó.

A leginkább ismert példa a készen kapható adattárolásra talán a ***tárolóhálózat (storage area network, SAN)***. Emlékszem, amikor a *SAN*-ok megjelentek, hatalmas zsongás vette őket körül, és elég nehéz volt kideríteni, hogy valójában mik is. Amikor végre sikerült, csalódottan kellett tudomásul vennem, hogy a *SAN*-ok bonyolultak, egy-egy gyártóhoz kötődnek és drágák.

A *SAN*-ok támogatása érdekében a linuxos közösség mindennek ellenére fontos módosításokat hajtott végre a rendszermagban. A 2.6-os rendszermag újdonságainak jelentős részét a 2.4-es rendszermagsorozat nagyvállalati használatra szánt tagjainak szolgáltatásai ihlették, és napjaink üzembiztos rendszermagjai számos olyan dolgot tudnak, amit néhány éve még nélkülöznünk kellett.

Például hatalmas kapacitású blokkos eszközöket használhatunk, messze túlnyúlva a régi két TB-os határon, és nagyobb számú lemez egyidejű csatlakoztatása sem okozhat gondot. Szintén fontos fejlesztés a tárolókötetek kifinomult kezelése, ahogy a fájlrendszerek is hatalmas méretűekre hízhatnak, akár becsatolt állapotban, használat közben is.

Írásomban ismertetem, hogyan aknázhatjuk ki a rendszermag új szolgáltatásaiban rejlő lehetőségeket, hogyan vehetjük ki a lemezeket a számítógépekből, és hogyan léphetünk túl a tárolók használatával és kapacitásával kapcsolatos korábbi korlátokon. Az ***ATA over Ethernetet (ATA Ethernet felett, AoE)*** talán úgy szemlélhetjük, mint megoldást arra, hogy az *IDE* kábel helyére egy *Ethernet* hálózat kerüljön. Ha az adattárolást leválasztjuk a számítógépről, és kihasználjuk a két rendszerelem közötti *Ethernet* rugalmasságát, akkor a lehetőségeket csupán képzelőerőnk és új dolgok elsajátítására irányuló hajlandóságunk végessége fogja korlátozni.

Mi az AoE?

Az ***ATA over Ethernet*** egy az *IEEE*-nél ***0x88a2*** azonosítóval bejegyzett *Ethernet* hálózati protokoll. Az *AoE* alacsony szintű, jóval egyszerűbb a *TCP*/*IP*-nél, sőt, az *IP*-nél is. A *TCP*/*IP* és az *IP* az internet esetében fontos megbízhatóságot növelő feladatokat lát el, ám a számítógépeknek komoly munkát jelent az ebből fakadó bonyolultság kezelése.

Az *iSCSI*-t használóknak nyilván ismerős ez a *TCP*/*IP*-vel kapcsolatos gond. Az *iSCSI* egy másik megoldás be- és kivitelek *TCP*/*IP* feletti továbbítására, segítségével olcsó *Ethernet* készülékekkel lehet helyettesíteni a költséges *Fibre Channel* berendezéseket. Sok *iSCSI*-használó ***TCP tehermentesítő motorokat (TCP offload engine, TOE)*** kezdett vásárolni. A *TOE* kártyák olcsók, és alkalmasak arra, hogy az *iSCSI*-t használó gépek válláról levegyék a *TCP*/*IP* kezelésének terhét.

Érdekes megfigyelés, hogy a gyakorlatban a legtöbb esetben az *iSCSI*-t nem internet felett használják. Ha a csomagoknak egyszerűen a szomszéd szekrényben található gépbe kell befutniuk, akkor a nehézsúlyú *TCP*/*IP* protokoll használata túlzásnak tűnik.

A *TCP/IP* tehermentesítés helyett tehát nem volna jobb, ha teljesen elhagynánk az internetes protokollt? Az ***ATA over Ethernet*** protokoll pontosan ezt teszi, kihasználva a napjainkban kapható olcsó kapcsolók által kínált lehetőségeket. A korszerű kapcsolók képesek a folyamvezérlésre, maximális átviteli sebességet és minimális csomagütközést biztosítva. A ***helyi hálózaton (local area network. LAN)*** a csomagok sorrendje nem változik meg, és a hálózati hardver minden csomag sértetlenségét ellenőrző összeggel védi.

Minden ***AoE***-csomag vagy egy ***ATA*** meghajtónak szóló parancsot, vagy egy ***ATA*** meghajtótól érkező választ hordoz. A ***Linux*** rendszermag ***AoE*** illesztőprogramja minden ***AoE***-műveletet elvégez, és a távoli lemezeket normál, blokkos eszközökként teszi elérhetőkké, ilyen például a ***/dev/etherd/e0.0***. Az ***IDE*** illesztőprogram hasonlóan gondoskodik az ***IDE*** kábel végén található helyi meghajtó például ***/dev/hda*** alatti elérhetőségéről. Az illesztőprogram szükség esetén a csomagok újraküldését is elvégzi, vagyis az ***AoE*** eszközök a rendszermag többi része számára tökéletesen egyenértékűek a többi lemezzel.

Az ***ATA*** parancsok továbbítása mellett az ***AoE*** lekérdező csomagokkal a rendelkezésre álló ***AoE*** eszközöket is képes egyszerű módon azonosítani. Ennyi is az egész: ***ATA*** parancscsomagok és lekérdező csomagok.

Aki már dolgozott ***SAN***-nal, vagy legalábbis tanult az ilyen rendszerekről, bizonyára megdöbbenve mered maga elé: ha minden lemez a ***LAN***-ra csatlakozik, hogyan lehet korlátozni az elérésüket? Vagyis hogyan biztosítható, hogy az A gép feltörése után a ***B*** gép lemezei biztonságban maradjanak?

A válasz onnan indul, hogy az ***AoE*** nem irányítható. Azt, hogy mely számítógépek mely lemezeket látják, egyszerűen, alkalmi (ad hoc) ***Ethernet*** hálózatok összeállításával határozhatjuk meg. Mivel az ***AoE*** készülékek nem rendelkeznek ***IP***-címmel, nem okoz nehézséget az elszigetelt ***Ethernet*** hálózatok létrehozása. Egyszerűen adjunk áramot a kapcsolónak, majd csatlakoztassuk hozzá a kívánt rendszerelemeket. Emellett sok újabb kapcsoló kapu alapú ***VLAN*** szolgáltatást is nyújt, amivel a kapcsolót hatékonyan lehet különálló, egymástól elválasztott szórási tartományokra osztani.

Az ***AoE*** protokoll annyira egyszerű, hogy olcsó vassal is használható. Jelenleg a ***Coraid*** az egyetlen gyártó, mely ***AoE*** eszközöket kínál, ám várható, hogy hamarosan további gyártók és alkalmazásfejlesztők is felfedezik majd, hogy az ***AoE*** leírása csupán nyolc oldal hosszú. Érdemes ezt az egyszerűséget szembeállítani az ***iSCSI*** több száz oldalas, többek közt titkosítási szolgáltatásokat, forgalomirányítást és felhasználó alapú hozzáférés-kezelést taglaló leírásával. A bonyolultság mindig költséggel jár, és most már választhatunk, hogy igényeljük ezt a bonyolultságot, vagy inkább megtartjuk az árát.

A legegyszerűbb eszközök is lehetnek hatásosak. A ***Linux***-felhasználók számára aligha újdonság, hogy egyszerűsége ellenére az ***AoE*** lehetőségek elképesztő választékát kínálja, ha az adattárolás egyszer átkerült a hálózatra. Nézzünk tehát egy tényleges példát, majd vizsgáljuk meg a lehetőségeket.

Stan, a levéltáros

A következő példa igaz történetre alapul. ***Stan*** egy képzelt rendszergazda, aki a helyi önkormányzatnak dolgozik. Az új rendelkezések szerint minden hivatalos dokumentumot maradandóan archiválni kell. A város bármely polgára bármikor betekintést kérhet bármelyik dokumentumba. ***Stannek*** tehát hatalmas és korlátok nélkül bővíthető tárolóhelyre van szüksége, ugyanakkor a tárolóeszköz teljesítményének nem kell jobbnak lennie, mint egy átlagos, helyi, ***ATA*** felületű lemezének. Célja az, hogy minden adatot könnyen és gyorsan elő lehessen keresni.

Stan ismeri az ***Ethernet*** hálózatokat és a linuxos rendszereket, ezért az ***ATA over Ethernet*** kipróbálása mellett dönt. Vásárol néhány eszközt; összesen kevesebb mint 6500 dollárt fizetve az alábbiakért:

- Egy darab kétkapus Ethernet kártya, a kiszolgáló régi, 100 Mbps sebességű kártyája helyett
- Egy darab 26 kapus hálózati kapcsoló kettő darab gigabites kapuval
- Egy darab Coraid EtherDrive polc és tíz darab EtherDrive fiók
- Tíz darab 400 GB-os ATA meghajtó

A tíz fiókot befogadó polc három egység magas. Minden ***EtherDrive*** fiók egy-egy kisméretű számítógép, mely az ***AoE*** protokoll kezelésével hatékony módon illeszt a hálózatra egy-egy ***ATA*** lemezt. Az adatokat csíkozással elosztva a polc fiókjai között hasonló teljesítményt kapunk, mint helyi ***ATA*** meghajtókkal, tehát a gigabites összeköttetéssel hatékonyan ki tudjuk használni a rendelkezésünkre álló átviteli kapacitást. Bár ***Stan*** megtehette volna, hogy az ***EtherDrive*** fiókokat ugyanarra a hálózatra csatlakoztatja, amire mindenki más is kapcsolódik, ám biztonsági és teljesítménybeli szempontok miatt inkább úgy döntött, hogy a tárolórendszert külön hálózatra helyezi, és a kiszolgáló második kapujához, az ***eth1*** csatolóhoz kapcsolja.

Stan átolvassa a ***Linux Software RAID HOGYAN***-t (lásd az internetes forrásokat), majd úgy határoz, hogy ***RAID 10*** csíkozást használ, tükrözött lemezpárok felett. Bár ezzel a megoldással kevesebb tárkapacitást kap, mintha ***RAID 5***-öt használna, viszont a ***RAID 10*** a lehető legjobb megbízhatóságot nyújtja, miközben a processzorra a ***RAID*** kezelése miatt háruló többletterhelés minimális marad, valamint a tömb újraépítése is rövidebb ideig tart, ha egy-egy lemez kiesik.

Az ***LVM HOGYAN*** (lásd a forrásokat) átolvasása után ***Stan*** kidolgoz egy tervet, amellyel elkerülheti, hogy valaha is elfogyjon a szabad tárhely. A ***JFS*** fájlrendszer képes dinamikusan méreteződni akár egészen nagyra is, vagyis ***Stan JFS***-t tesz egy logikai kötetre. A logikai kötet ebben az esetben csak egy fizikai kötetre terjed ki, ez pedig a ***RAID 10*** blokkos eszköz lesz. A ***RAID 10*** a ***Coraid*** polcban lévő ***EtherDrive*** fiókokból jön létre, a ***Linux*** szoftveres ***RAID*** megoldásával. Ha később újabb polcra lesz szüksége, akkor majd létrehoz egy másik ***RAID 10*** készletet, abból lesz egy fizikai kötet, amelyre kiterjeszti a ***JFS***-t tartalmazó logikai kötetet.

1. kódrészlet Egy nagyszámú AoE meghajtóból álló szoftveres RAID eszköz létrehozásának első lépése az AoE telepítése, beállítása

```
# A gazdagép beállítása AoE használatára
# az AoE illesztőprogram lefordítása
# és telepítése
tar xvfz aoe-2.6-5.tar.gz
cd aoe-2.6-5
make AOE_PARTITIONS=1 install
# Az AoE nem igényel IP-címet!  :)
ifconfig eth1 up
# A hálózati csatoló felélesztése
sleep 5
# Az ATA over Ethernet illesztőprogram betöltése
modprobe aoe
# A rendelkezésre álló AoE lemezek megtekintése
aoe-stat
```

Az *1. kódrészlet* azokat a parancsokat tartalmazza, amelyeket ***Stan*** a kiszolgáló ATA over Ethernet előkészítésére használ. Az ***AoE*** illesztőprogramot az `AOE_PARTITIONS=1` átadott értékkel fordítja le, ugyanis 2.6-os rendszermagot futtató Debian sarge rendszere van. A ***sarge*** jelenleg nem támogatja a nagyobb értékű eszköz-alazonosítókat (lásd az Alazonosítók című széljegyzetet), ezért kikapcsolja a lemezek részekre osztásának támogatását, így ugyanis több lemezt tud használni. Figyelembe véve a ***Debian*** 292070-es számú hibáját, ***Stan*** telepíti a legújabb eszközleképezőt és az ***LVM2*** felhasználói programokat.

Alazonosítók

Ha egy program használni akar egy eszközt, akkor ezt jellemzően az eszközhöz tartozó különleges fájl megnyitásával teszi. Jól ismert példa a ***/dev/hda*** fájl. Ha kiadjuk az `ls -l` parancsot, akkor a ***/dev/hda*** esetében két számot látunk, ezek a 3 és a 0. A ***/dev/hda1*** fájl alazonosítója 1, főazonosítója viszont szintén 3.

A 2.6-os rendszermag megjelenése előtt az alazonosító ábrázolása nyolc biten történt, vagyis értéke 0 és 255 között lehetett. Mivel senkinek nem volt ilyen sok eszköze, a korlát tulajdonképpen senkit nem zavart. Most, hogy a lemezeket le lehet választani a kiszolgálókról, megváltozott a helyzet, ezért a 2.6-os rendszermag már 20 biten ábrázolja az eszközök alazonosítóját.

Így az alazonosító 1048576-féle értéket vehet fel, ami nagy segítség ahhoz, hogy a rendszerekhez nagyszámú eszközt tudjunk csatlakoztatni – csakhogy a változást nem minden program követte. Ha a glibc vagy valamelyik alkalmazás azt hiszi, hogy az alazonosítók még mindig nyolcbitesek, akkor gondjaink lesznek a 255-nél nagyobb alazonosítók használatával. Az átmenetet segíti, hogy az ***AoE*** illesztőprogramot a lemezrészek támogatása nélkül is le lehet fordítani. Ilyenkor egy-egy lemezhez 16 helyett csak egyetlen egy alazonosító tartozik. Nem baj tehát, ha a rendszer nem ismeri még a 2.6-os nagyméretű eszköz-alazonosítóit, akár 256 ***AoE*** lemezt is használhatunk.

2. kódrészlet A szoftveres RAID és az LVM kötetcsoport beállítása

```
# A RAID kezdeti értékadásának felgyorsítása
for f in `find /proc | grep speed`; do
        echo 100000 > $f
done
# Tükrök létrehozása (az mdadm kezeli a forró
# tartalékokat)
mdadm -C /dev/md1 -l 1 -n 2 \
        /dev/etherd/e0.0 /dev/etherd/e0.1
mdadm -C /dev/md2 -l 1 -n 2 \
        /dev/etherd/e0.2 /dev/etherd/e0.3
mdadm -C /dev/md3 -l 1 -n 2 \
        /dev/etherd/e0.4 /dev/etherd/e0.5
mdadm -C /dev/md4 -l 1 -n 2 -x 2 \
        /dev/etherd/e0.6 /dev/etherd/e0.7 \
        /dev/etherd/e0.8 /dev/etherd/e0.9
sleep 1
# A csík létrehozása a tükrök felett
mdadm -C /dev/md0 -l 0 -n 4 \
        /dev/md1 /dev/md2 /dev/md3 /dev/md4
# A RAID 10 megfeleltetése LVM fizikai kötetnek
pvcreate /dev/md0
# Bővíthető LVM kötetcsoport létrehozása
vgcreate ben /dev/md0
# A fizikai kiterjedés vizsgálata
vgdisplay ben | grep -i 'free.*PE'
# A teljes helyre kiterjedő logikai kötet
# létrehozása
lvcreate –extents 88349 –name franklin ben
modprobe jfs
mkfs -t jfs /dev/ben/franklin
mkdir /bf
mount /dev/ben/franklin /bf
```

A fájlrendszer és a logikai kötet létrehozására használt parancsokat a *2. kódrészlet* tartalmazza. ***Stan*** a kötetcsoportnak a `ben`, a logikai kötetnek pedig a `franklin` nevet adja. Ezután módosítania kell néhány dolgot az ***LVM2*** beállításaiban. Először is, szükség lesz egy `types = [ "aoe", 16 ]` sorra, ami lehetővé teszi, hogy az ***LVM*** felismerje az ***AoE*** lemezeket. Ugyancsak szükség van az `md_component_detection = 1` sorra, amelynek hatására a gép a ***RAID 10*** készleten belüli lemezeket figyelmen kívül hagyja, miután a teljes ***RAID 10*** készlet egyetlen a fizikai kötetet alkot.

Stan rendszerét magam is összeállítottam, ***Debian sarge*** operációs rendszerrel, kettő darab 2,1 GHz-es ***Athlon MP*** processzorral, 1 GB memóriával és egy ***Intel PRO/1000 MT*** kétkapus hálózati kártyával, ma már elavulófélben lévőnek számító 40 GB-os meghajtókkal. Hálózati kapcsolóként egy ***Netgear FS526T*** készüléket használtam. Ha a ***RAID 10*** készletet nyolc darab ***EtherDrive*** fiókra terjesztettem ki, akkor 23,58 MBps folyamatos olvasási és 17,45 MBps folyamatos írási sebességet sikerült elérnem. A mérések elvégzése előtt

3. kódrészlet Leválasztása nélkül úgy tudjuk bővíteni a fájlrendszert, hogy létrehozunk egy második RAID 10 készletet, hozzáadjuk a kötetcsoporthoz, majd kiterjesztjük a fájlrendszert

```
# A második polc RAID 10 készletét
# létrehozása után
# /dev/md5-ként adjuk hozzá a kötetcsoporthoz
vgextend ben /dev/md5
vgdisplay ben | grep -i 'free.*PE'
# A logikai kötet majd a jfs megnövelése
lvextend –extents +88349 /dev/ben/franklin
mount -o remount,resize /bf
```

egy 1 GB-os fájlt a */dev/nullba* másolva kiürítettem a gyorsítótárat, az írási időbe pedig egy `sync` parancs végrehajtását is belevettem.
Ebben az esetben a ***RAID 10*** készletnek négy csíkeleme volt, mindegyik egy pár tükrözött meghajtóból állt. Általános esetben egy ***EtherDrive*** fiókokból álló készlet átviteli sebességét a csíkelemek száma alapján tudjuk megbecsülni. Egy ***RAID 10*** készletben feleannyi csíkelem van, mint lemez, hiszen minden lemez tükrözve van egy másikra.
A ***RAID 5*** esetében egy lemez tárolja a paritásadatokat, a több lemez pedig csíkelemként veendő figyelembe.
A várható olvasási sebesség a csíkelemek száma szorozva 6 MBps-mal. Ha tehát ***Stan*** a nyolc fiókból álló ***RAID 10*** készlet helyett két polcot vásárolt volna, és a készletet 18 meghajtóból állította volna össze, akkor valamivel több, mint kétszer ekkora átviteli sebességet kapott volna. ***Stan*** azonban nem igényel ekkora átviteli sebességet, és egy viszonylag kisméretű, 1,6 TB-os fájlrendszer is megfelel az igényeinek.
A *3. kódrészlet* azt szemlélteti, hogy ***Stan*** milyen könnyen ki tudja bővíteni a fájlrendszert, ha vásárol egy második polcot is. A kódrészletben ***Stan mdadm-aoe.conf*** fájlja és indító és leállító parancsfájljai nem szerepelnek. A figyelő módban futó `mdadm` folyamattal az `mdadm` beállító fájl közli a forró tartalékok kezelési módját, így ezekkel bármikor helyettesíteni lehet bármelyik meghibásodott tükör bármelyik meghajtóját. Lásd még az ***mdadm man*** oldalának ***tartalékcsoportok (spare groups)*** című részét.
Az indító és leállító parancsfájlokat könnyen össze lehet állítani. Az indító parancsfájl egyszerűen összeállítja a tükrözött ***RAID 1*** majd a ***RAID 0*** készleteket, végül elindít egy `mdadm` figyelő folyamatot. A leállító parancsfájl leállítja az `mdadm` figyelőt, melyet követően leállítja először a ***RAID 0***, majd a tükrözött készleteket.

Blokkos tárolóeszköz megosztása

Láttuk, hogyan működik az ***ATA over Ethernet***, és az olvasóban bizonyára felmerül a kérdés: vajon mi történik, ha egy másik állomás is hozzá szeretne férni a tárolóhálózathoz? Van arra lehetőség, hogy a második állomás is befűzze a ***JFS*** fájlrendszert, és hozzáférjen ugyanazokhoz az adatokhoz? Nos, biztonságosan erre nincs mód. A ***JFS***-t az ***ext3***-hoz és a többi fájlrendszer túlnyomó részéhez hasonlóan egyetlen állomással való használatra tervezték. Az ilyen „egygépes” fájlrendszerek megsérülhetnek, ha ugyanazt a blokkos tárolóeszközt több állomás is befűzi. Ennek oka a puffer gyorsítótárban keresendő, ami a 2.6-os rendszermagokban egyesítve van a lapgyorsítótárral.
A ***Linux*** meglehetősen rámenős módon, minden lehetőséget megragadva gyorsítótárazza a fájlrendszer adatait a memóriában, kerülve a lassú blokkos tárolóeszközök használatát, és ezzel javítva a teljesítményt.
A gyorsítótárazás hatékonyságáról bárki meggyőződhet, ha kétszer egymás után lefuttat egy `find` parancsot ugyanazon a könyvtáron.
Bizonyos fájlrendszereket úgy terveztek, hogy több állomás is használhassa őket. Az úgynevezett fürtfájlrendszerek például megfelelő eljárásokat tartalmaznak annak biztosítására, hogy az összes állomás gyorsítótára összhangban maradjon a fájlrendszerrel. Kiváló példa erre a nyílt forrású ***GFS***.
A ***GFS*** külön fürtkezelő programot használ annak követésére, hogy ki szerepel a fájlrendszert használó állomások csoportjában. A fájlrendszert elérő állomások együttműködését zárak segítségével biztosítja.
Fürtfájlrendszer, például ***GFS*** használatával megoldható, hogy az ***Ethernet*** hálózat több állomása is használhassa ***ATA over Ethernet*** felett ugyanazt a blokkos tárolóeszközt. Ilyenkor nincs szükség például ***NFS***-kiszolgálóra, hiszen mindegyik állomás közvetlenül, a be- és kiviteleket szépen elosztva éri el a tárolóeszközt. Van azonban egy kis bökkenő. Minél nagyobb számú lemezt használunk, annál nagyobb a valószínűsége annak, hogy az egyik meghibásodik. Ezt a problémát általában ***RAID*** használatával szokták orvosolni, némi redundanciát hozva a rendszerbe. Sajnos a ***Linux*** szoftveres ***RAID***-je nem képes a fürtök kezelésére. Nincs tehát mód arra, hogy a hálózat összes állomása ***RAID 10*** készletet és `mdadm`-et használjon.
A ***Linux*** fürtkezelése ugyanakkor elképesztő tempóban fejlődik. Biztos vagyok abban, hogy egy-két év múlva kiváló, fürtképes ***RAID*** áll majd rendelkezésünkre. Addig is azonban más megoldásokat kell keresnünk az ***AoE*** alapú fürtök megosztására. Az alapötlet az, hogy központosítsuk a ***RAID*** szolgáltatást. Vásárolhatunk például egy vagy két ***Coraid RAIDBlade*** vezérlőt, majd a fürtcsomópontokkal az ezeken keresztül exportált tárolóeszközt érhetjük el. A ***RAIDBlade*** vezérlők minden mögöttük lévő ***EtherDrive*** fiókot képesek kezelni. Ha szeretünk barkácsolni, akkor ugyanezt a feladatot egy linuxos, szoftveres ***RAID***-et futtató géppel is megoldhatjuk, mely ***ATA over Etherneten*** keresztül maga teszi elérhetővé a lemezhibáktól mentesített blokkos tárolóeszközt. Például a `vblade` program (lásd a forrásokat) képes tetszőleges tárolóeszközt elérhetővé tenni ***ATA over Etherneten*** keresztül.

Biztonsági mentés

Mivel az ***ATA over Ethernet*** olcsó merevlemezeket csatlakoztat az ***Ethernet*** hálózatra, akár biztonsági mentések készítésére is alkalmazható. A mentési stratégia sokszor másodvonalbeli adattárolást is magába foglal, vagyis olyan tárolóeszközt, amely ugyan nem olyan gyors, mint az éles rendszer, de könnyebben hozzáférhető, mint a szalagos.
Az ***ATA over Ethernet*** segítségével olcsó ***ATA*** meghajtókat használhatunk másodvonalbeli adattároló eszközökként.

Sőt! Ha ilyen olcsó merevlemezeink vannak, és ennyire üzembiztos szoftveres ***RAID***-et tudunk alkalmazni, miért is ne használhatnánk mentési adathordozóként a merevlemezeket? A szalagokkal ellentétben ezek azonnali hozzáférést biztosítanak bármely archivált fájlhoz.
Sok új biztonsági mentési alkalmazás a fájlrendszerek mentési szolgáltatásait is képes kihasználni. Közvetlen hivatkozásokkal több teljes mentést is képesek elvégezni, a növekményes mentések hatékonyságával. További tudnivalók az internetes források között szereplő ***Backup PC*** és ***rsync*** hivatkozások révén érhetők el.

Összefoglalás

Olcsó merevlemezek közvetlenül a hálózaton – jogos a kérdés, hogyhogy korábban senkinek nem jutott eszébe? Az adattárolás a kiszolgálóktól való elválasztását érdemes lehet valamilyen egyszerű hálózati protokoll segítségével végezni, drága eszközök nélkül – még ha az egyszerű protokoll használhatósága a helyi ***Ethernet*** hálózatra korlátozódik is. A helyi hálózaton ugyanakkor nincs is szükség a mindenre kiterjedő szolgáltatásokat nyújtó internetes protokollok, például a ***TCP/IP*** bonyolultságára és többletterhelésére. Ha az adattárolást a helyi hálózatra akarjuk korlátozni, és az ***Ethernet*** hálózatok kiépítése révén kapott hozzáférésvezérlés kielégíti az igényeinket, akkor az ***ATA over Ethernet*** számunkra a megfelelő választás. Ha a tárolóprotokolltól titkosítási, irányíthatósági és felhasználó alapú hozzáférés-vezérlési szolgáltatásokat várunk, akkor az ***iSCSI***-val érdemes barátkoznunk.

Az ***ATA over Ethernet*** olyan egyszerű alternatíva, amely eddig egyszerűen hiányzott a linuxos adattárolási lehetőségek közül. Az egyszerűség egyben lehetőségeket is jelent. Az ***AoE*** által biztosított alapra tetszőleges adattároló megoldást felépíthetünk. Mindenkit szeretnék arra buzdítani, hogy engedje szabadjára a képzeletét, majd írja meg nekem, hogy milyen rendszert sikerült felépítenie.

Köszönetnyilvánítás

Értékes visszajelzéseiért köszönettel tartozom ***Peter Andersonnak***, ***Brantley Coile***-nak és ***Al Dixonnak***. Szintén szeretném megköszönni ***Brantley***-nek és ***Sam Hopkinsnak***, hogy ilyen kiváló tárolóprotokollt fejlesztettek ki.

Linux Journal 2005. június, 134. szám

A cikkhez tartozó források elérhetősége:
➲ ***www.linuxjournal.com/article/8201***

Ed L. Cashin 1997 óta különféle tudományos és linuxos szakmai területekkel foglalkozott, volt már webalkalmazás-fejlesztő, rendszergazda és rendszermag-programozó egyaránt. Jelenleg a Coraid munkatársa, itt történt az ATA over Ethernet kifejlesztése. Az ecashin@coraid.com címen érhető el. Miközben harcművészeti óráira tart, legszívesebben zenét és hangos könyveket hallgat.