

Pörgésben – Fájrendszer sebességének mérése

Amikor egy számítógép sebességéről beszélünk, akkor általában mindenki a processzor frekvenciájára gondol, ugyanakkor a tényleges teljesítményt nem mindig ez határozza meg. Hiszen a processzor idejének egy részét azzal is tölti, hogy a háttértár ki-beviteli műveletére vár (wait). Ilyenkor a sebesség már sokkal inkább a merevlemez, merevlemez vezérlők gyorsaságán múlik.

A merevlemez egységet nem célozom különösebben bemutatni, hiszen akit érdekel annak működése az az Interneten rendkívül sok anyagot talál róla, ilyen célra például tudom ajánlani a *Wikipédiát*. A merevlemez sebességét lényegében három tényező befolyásolja, az egyik a csatoló felület/szabvány, a másik a lemez körbefordulási sebessége, a harmadik pedig a beépített gyorsító tár mérete. A hagyományos *EIDE/ATA-2* merevlemezek 1996-ban még csak 16 Mbyte/s adatátvitelre voltak maximum képesek, az *ATA* szabvány utolsó leszármazottja 2003-ban jelent meg, *ATAPI-7* néven (*Ultra DMA-133*), mely 133 Mbyte/s átviteli sebességre volt képes, és a legtöbb számítógépben még ilyen merevlemezek vannak a mai napig. Amennyiben új számítógépet vásároltunk a közelmúltban, vagy vásárolunk, akkor mára főleg *SATA* csatolóval ellátott lemezeket találunk benne. Vállalati környezetben a kiszolgálókra ugyanakkor kivétel nélkül csak *SCSI* lemezeket szoktak használni. A *SATA* szabvány elméleti legnagyobb átviteli sebessége 150 Mbyte/s (*8B/10B* kódolást használva a fizikai rétegen), míg a *SATA II* szabvány is már 300 Mbyte/s átviteli sebességre képes. 2007-re várhatóan a *SATA* szabvány képes lesz a 600 Mbyte/s adatvitelre. Ugyanakkor a *SCSI* merevlemezek már jóval régebben támogatják az *Ultra320*-as szabványt, mely 320 Mbyte/s adatátvitelre képes, persze ezek az adatok az elvileg elérhető

maximum értéket jelentik. Az használat közben tapasztalt sebesség akár ennek a töredéke is lehet. A legegyszerűbb eszköz a merevlemezünk sebességének mérésére a *hdparm* (de ez nem alkalmazható *SCSI* merevlemezre), ugyanakkor hamar rá is jövünk, arra, hogy ezzel a programmal inkább csak szép számokat, sem mint használható értéket kapunk. Vegyük például azt az esetet, amikor a merevlemez gyorsítótárának sebességét mérjük a programmal, szinte biztos, hogy az elvileg elérhető értéknél (ami a kábelen adott idő alatt képes átmenni) nagyobb értéket kapunk. Mivel minket a tények érdekelnek nem pedig a szép nagy számok, ezért hívjunk segítségül egy professzionális programot az adott feladatra.

IoZone

Az *IoZone* egy fájlrendszer sebességmérő alkalmazás, tehát nem merevlemezünk tényleges adatátviteli sebességét tudjuk vele megmérni, hanem az adott fájlrendszer teljesítményét. Ugyanakkor mivel fájlrendszer nélkül nem sok mindenre tudjuk használni a merevlemez, ezért felesleges is volna fájlrendszer nélkül mérni a sebességet. Az *IoZone* sokféle fájlművelettel képes megmérni az adott fájlrendszer, illetve számítógép, fájlműveleti teljesítményét, a tesztek nem csak helyi gépeken végezhetjük el, hanem lehetőségünk van megmérni távoli fájlrendszerek (például *NFS*) sebességét is.

A sebesség mérését célszerű nem csak akkor elvégezni, amikor telepítjük és összehangoljuk a rendszert, hanem később is, például amikor a kiszolgáló funkciója bővül, megváltozik. Ha a kiszolgálót eddig adatbázis kiszolgálásra használtuk, és ezentúl mondjuk mentési feladatokat fog ellátni, akkor célszerű lehet a fájlrendszert is megváltoztatni úgy, hogy a szekvenciális olvasást a fájlrendszer jobban támogassa mint a véletlenszerűt.

Első tesztek

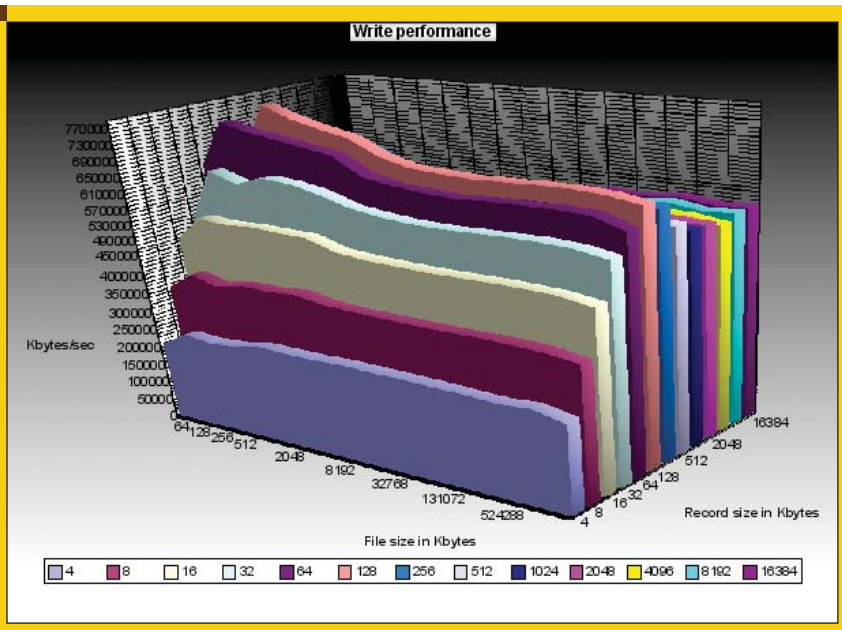
Az *IoZone* jól paraméterezhető, így magunk is definiálhatunk tesztek, de ha nem akarunk ezzel törődni akkor használjuk az *IoZone*-t automatikus módban:

```
iozone -a
```

Amennyiben szeretnénk grafikonok formájában is megtekinteni a kimenetet, akkor használjuk az

```
iozone -Ra
```

parancsot, az elkészült kimenetből pedig *OpenOffice* segítségével készíthetünk grafikonokat. Egy ilyen grafikont láthatunk például az 1. ábrán, melyen a *ReiserFS* fájlrendszer szekvenciális írási sebessége látható egy *IBM x236*-os duál processzoros 3 Gb fizikai memóriával rendelkező kiszolgáló esetén. A különböző színű grafikonok a különböző méretű kiírt rekordok méretét jelentik. Az x tengelyen pedig a kiírt fájl



1. ábra A ReiserFS fájlrendszer teljesítménye szekvenciális írás esetén

tényleges mérete látható, a 64 kByte méretűtől egészen 1 GByte méretű fájllokig. A grafikonról leolvasható például, hogy a jelen konfiguráció esetén a 128 kByte méretű rekord nyújtja a legjobb teljesítményt. A z tengely felosztása lineáris, 0 és 760000 kByte közötti értékek szerepelnek rajta. Lehetőségünk van egyből bináris állományba kimenteni a teszt eredményét, ehhez csak az

`iozone -Rab eredmény.wks`

parancsot kell kiadni. Ahogyan már említettem, az `IoZone` remekül paraméterezhető, így például korlátot szabhatunk annak, hogy mekkora legyen a legnagyobb fájl, amin a tesztet elvégezzük, ha a legnagyobb fájl 1Gb méretűnek engedjük meg, akkor a következő paraméterekkel indítsuk a teljesítménypróbát:

`iozone -Ra -g 1G`

Ha nem szeretnénk az összes tesztet lefuttatni, hanem csak az írás-olvasás eredményére vagyunk kíváncsiak, akkor a következőképpen futtassuk le az alkalmazást:

`iozone -Ra -g 1G -i0 -i1`

Az `NFS` kliens sebességét a következő utasítások segítségével tudjuk megvizsgálni:

`iozone -Rac`

Itt a `c` paraméter azért kell, hogy kiküszöböljük vele az `NFS 3` esetén előforduló kliens oldali gyorsítótár hatását.

Mit tesztlünk

Az `IoZone` lehetőséget biztosít nekünk, hogy minél szélesebb skálán tudjuk megmérni a háttértárunk teljesítményét, viszont fontos tisztában lennünk azzal, hogy melyik teszt pontosan mit csinál, így fussunk át gyorsan az egyes teszteken.

- **Írás (Write)** – Ezzel egy új fájl létrehozásának az idejét tudjuk lemérni. Amikor egy új fájl létrejön, akkor nem csak a fájl tartalmát kell a merevlemezre írunk, hanem azt is, hogy a fájl hol foglal helyet a lemezen, milyen jogosultságok tartoznak hozzá, stb. Ezeket az adatokat nevezzük metaadatoknak, melyek nem az eltárolni kívánt adat tényleges részei. Az első írás pontosan ezért valamivel lassabb, mint a fájl újra írása.
- **Újrairás (Re-write)** – Ez a teszt egy már létező fájlhoz ír hozzá adatot, mivel ilyenkor a metaadatok általában nem módosulnak, így az újra írás természetesen gyorsabb lesz.
- **Olvasás (Read)** – Egy már létező fájlból olvasunk ki adatot.

- **Újraolvasás (Re-read)** – Az újraolvasás sebessége sokkal nagyobb lesz az olvasás sebességénél, hiszen ilyenkor olyan adatokat olvassunk ki, melyeket a közelmúltban olvastunk. Az operációs rendszer (és esetleg a merevlemez belső) gyorsítótára miatt így nem a tényleges merevlemez sebességet mérjük meg.
- **Véletlenszerű olvasás (Random Read)** – Ennél a tesztnél is egy fájlból olvasunk ki adatokat, de a fájl nem az elejétől a végéig olvassuk, hanem a fájl véletlenszerűen kiválasztott rekordjait olvassuk ki. Ezt az olvasási módot jellemzően az adatbázis kiszolgálók teljesítmény mérésénél célszerű figyelembe venni. Ennek a módnak a sebességét nagyon sok tényező befolyásolja, ilyen például a rendszer gyorsítótár mérete, a merevlemez fejmozgások ideje, a rendszer memória, stb.
- **Véletlenszerű írás (Random Write)** – A véletlenszerű olvasáshoz hasonló, csak most nem rekordokat olvasunk az adott fájlból, hanem írunk bele.
- **Véletlenszerű írás-olvasás (Random Mixed)** – Egy azon fájl több folyamat ír és olvas egyszerre. Ez a teszt áll a leginkább közel egy tényleges adatbázis kiszolgáló háttértár használatához. Ez a teszt csak throughput módban futtatható, és egy adott folyamat vagy csak írja, vagy csak olvassa a kérdéses fájl. Az író-olvasó folyamatok közötti ütemezés a round robin algoritmust használja.
- **Hátrafelé olvasás (Backward Reading)** – Ez az olvasási mód elég extrémnek tűnhet, de vannak olyan a gyakorlatban is előforduló alkalmazások, melyek ilyen módon olvassák a fájlkat, mivel a fájlrendszerek általában az előre történő olvasásra vannak felkészítve, ezért így olvasva a fájl sokkal rosszabb eredményeket kaphatunk.
- **Rekord újrairás (Record rewrite)** – Ennél a tesztnél a fájl egy adott darabját írjuk, majd ismételtén írjuk. Itt többszintű értékeket is meg tudunk figyelni, attól függően, hogy az adott fájlrészlet elfér

a CPU gyorsítótárában, vagy csak a fizikai memóriában fér el. Ha a fájlrészlet fizikai memóriacíme megtalálható a *Translation Look-aside Bufferben (TLB)* akkor a gyorsabb virtuális fizikai címle- képezés miatt szintén egy újabb értéket kapunk. Amennyiben a fizikai memóriacím nincsen benne a *TLB*-ben, de a fájl darabja megtalálható az operációs rendszer gyorsítótárában, akkor ez ad egy újabb értéket, és legutoljára kapjuk azt az értéket, amikor a fájl csak a merevlemezen található meg.

- **Lépkedve olvasás (Strided Read)** – Az olvasás egy bizonyos minta alapján történik a fájlból. Ilyenkor kiolvassuk a fájl egy darabját majd bizonyos adatokat átugorva egy ugyanakkora darabot olvassunk ki belőle, és ez így folytatódik a fájl végéig.
- **Fwrite** – A fájlra a *fwrite()* függvényen keresztül írjuk, mely bufferelt és blokkolt írási műveletet valósít meg. A buffer miatt a kisméretű rekordok írási sebessége jelentősen megnőhet. Mivel ennél a módnál is új fájl hozunk létre, ezért itt is számolnunk kell a metaadatok hatásával.
- **Frewrite** – Az *Fwrite* művelethez hasonló, de egy már létező fájlra ír felül az *fwrite()* függvény segítségével.
- **Fread** – Olvasás a fájlból, az *fread()* függvény segítségével, tulajdonságai hasonlóak az *fwrite()* függvényéhez.
- **Freread** – Az *Fread* olvasáshoz hasonló, de a nemrégiben olvasott adatokat olvassuk ki újra.

Néhány naplózó fájlrendszer összehasonlítása

Ha már az ember összerak egy teszt-környezetet, akkor logikusnak tűnik, hogy amit csak lehet megmér vele. Én is így voltam ezzel, így egy *IBM e-server x236 series* és *SCSI Ultra320*-as háttértárak segítségével a négy leggyakrabban használt fájlrendszer tesztjét mellékelem a cikkhez. Konkrétan az *EXT3*, *JFS*, *ReiserFS* és *XFS* naplózó fájlrendszereket vizsgáltam, a kiszolgálón a *Novell Open Enterprise Server 9sp1* kapott helyet. Mindegyik fájlrendszert alapbeállítással használtam, egy külön erre a célra fenntartott partíción. A tesztet azonos méretű partíción végeztem el, a partícióra előzőleg felmásoltam magát a *IoZone* programot, egyébként a partíció teljesen üres volt. A táblázatot beszúrni ide teljesen felesleges lenne, így inkább a cikkhez csatolom.

Azért néhány fontos dolgot megpróbálok kiemelni belőle. Mivel az első teszt alkalmával csak olyan fájlkon véggeztem a műveletet, melyek elérték az operációs rendszer gyorsítótárában, ezért a fájlrendszerek közötti különbségek itt nem a tényleges lemezműveleti sebességeket mutatják, ugyanakkor az egyes fájlrendszerek sebességviszonya nagyon jól látszik. Szembetűnik az is hogy az adott konfiguráció esetén a legnagyobb adatátvitelt a 128 Kb-os rekord esetén kapjuk (1. ábra).

A 4 vagy 8 Kb-os rekord esetén az átviteli sebesség az elérhető maximum harmada, illetve fele, ez valószínűleg azért van így, mert egy kisebb rekord be-kiviteli pluszköltsége fajlagosan nagyobb, mint egy nagyobb rekordé. Véletlenszerű olvasás esetén az *XFS* fájlrendszer alul marad a másik három fájlrendszerrel szemben, melyek sebessége körülbelül 13%-kal gyorsabb. Véletlenszerű írás esetén az *EXT3* és a *ReiserFS* teljesítménye gyakorlatilag azonos, de a *ReiserFS* egy csöppet gyorsabbnak tűnik, míg az *XFS* szintén gyorsabb a másik kettőnél, de a *JFS* és az *EXT3* közötti különbség eléri a 36%-ot.

A nagy meglepetés a szekvenciális írás sebesség tesztjénél ért engem, amikor is az összes fájlrendszer sokkal rosszabbul teljesített, mint a véletlenszerű írás esetében.

128 kByte méretű rekordok esetében az *EXT3* fájlrendszer átlagos átviteli teljesítménye ~360 Mbyte/s volt, ugyanakkor az *XFS* fájlrendszer sebessége 828 Mbyte/s volt, ami gyakorlatilag a duplája az *EXT3* teljesítményének. Második helyezett lett a *ReiserFS* 676 Mbyte/s-cel és kizárólagosan alapon nem sokkal lemaradva tőle a *JFS* végzett 639 Mbyte/s-es teljesítményével.

A szekvenciális olvasás terén a négy fájlrendszer sebessége között gyakorlatilag nem volt semmilyen különbség, egységesen 2460 Mbyte/s-es teljesítményt nyújtottak. Természetesen ez a magas érték csak azért jöhetett létre, mert az operációs rendszer a memóriából olvasta vissza a fájl.

További összehasonlítás

Az előző összehasonlításoknál a teljes fájl amin az *IoZone* végezte a műveletet elért a rendszer memóriájában, ezért adódtak irreálisan magas értékek. Ugyanakkor azt is megpróbáltam megmérni, hogy mi a különbség az egyes fájlrendszerek teljesítménye között, amikor az aktuális fájl már nem fér el a rendelkezésre álló fizikai memóriában.

Ezeknél a teszteknel természetesen arra számítottam, hogy az elméleti 320 Mbyte/s-es értékeknel sokkal rosszabb eredmények fognak születni. Ugyanakkor ezek az értékek azok melyek valóban megmutatják fájlrendszerek közötti különbségeket. A tesztalany gépen a rendszermemória 3GB volt, tehát egy 4 GB-os fájlra végzett műveletek már biztosan helyes eredményt adnak.

Az 1. Táblázat alapján láthatjuk, hogy a gyakorlati adatátviteli sebesség közel a harmada annak amit a merevlemez, és a kábel elméletileg lehetővé

1. táblázat *Fájlrendszerek összehasonlítása*

| 128 Kb rekord és 4GB fájl méret | ReiserFS MB/s | EXT3 MB/s | JFS MB/s | XFS MB/s |
|---------------------------------|---------------|-----------|----------|----------|
| Szekvenciális írás | 113 | 105,3 | 112,8 | 116,8 |
| Újraírás | 113,7 | 106,3 | 118,5 | 114 |
| Olvasás | 78,9 | 83,3 | 84,8 | 83,6 |
| Újraolvasás | 71,5 | 83,4 | 84,8 | 83,6 |
| Véletlenszerű olvasás | 65,8 | 66 | 68 | 64,9 |
| Véletlenszerű írás | 132,7 | 55,8 | 127,2 | 103,4 |

tesz. Furcsa anomáliát tapasztalunk ugyanakkor, ha megnézzük, azt, hogy a véletlenszerű olvasás sokkal lassabb, mint a véletlenszerű írás. Valószínűleg ez azért van, mert az írást az operációs rendszer nem írja ki rögtön a merevlemezre, hanem csak a memóriában tárolja ideiglenesen. Ugyanakkor az olvasásnál nem tud a gyorsítótárból olvasni, mert a fájl nincsen benne.

Konklúzió

Ahhoz, hogy megtaláljuk a számunkra leginkább megfelelő fájlrendszert mindenképpen ajánlom, hogy magunk végezzük el a tesztek a saját gépünkön, és vegyük figyelembe azt is mindig, hogy milyen alkalmazásokat akarunk használni. Ahogy telik az idő a fájlrendszerünk teljesítménye esetleg rosszabb lehet, mint kezdetben, mert időközben fellép a fájl töredezettsége, mely teljesítményromláshoz vezet. Ez a jelenség ugyanakkor manapság nem olyan szembe-tűnő, mint régebben a nem linuxos FAT fájlrendszer esetén volt. Akik idáig eljutottak a cikk olvasásában, azok joggal tehetik fel a kérdést,

hogy miért mértem meg a fájlrendszerek teljesítményét akkor is, amikor a rendszer tulajdonképpen a memória gyorsítótárból használja az adott fájlt. A válaszom pedig erre igen egyszerűen az, hogy jó, ha ismerjük a fájlrendszerünk ezen tulajdonságát is, hiszen az operációs rendszerek mindig megpróbálnak lehetőleg arra törekedni, hogy az általunk leginkább használt fájlokat a memóriában tartsák. Ezért nagy esélyünk van arra, hogy rendszerünk főleg így üzemeljen, ekkor pedig nem hagyhatjuk figyelmen kívül ezeket az értékeket sem.

A fenti statisztikák alapján én azt javaslom, hogy ha lehet akkor a *ReiserFS* vagy a *JFS* fájlrendszert részesítsük előnyben, amikor naplózó fájlrendszert szeretnénk használni linuxos munkakörnyezetben. Az *EXT3* fájlrendszert pedig semmiképpen se használjuk például adatbázis kiszolgálón, ahol véletlenszerű íráskor nyújtott teljesítménye igen gyenge.

Az egyes fájlrendszerek sebessége még tovább javítható, például ha úgy csatlakoztatjuk fel a fájl-

rendszert, hogy beállítjuk a *noatime* opciót, akkor a rendszer nem fogja naplózni, hogy mikor történt az utolsó hozzáférés az adott fájlhoz. Hiszen ha rögzítjük az utolsó hozzáférés idejét, akkor minden egyes olvasási műveletet szükségképpen egy írási művelet is követ. Ezt az opciót főleg flash memóriák esetén érdemes mindig bekapcsolni. De például a *ReiserFS* fájlrendszer esetében megadhatjuk azt is, hogy a naplót ne az adott partíción tárolja a rendszer, hanem egy másik lemezen. Mindenkinek sikeres finomhangolást, és lemezfeldörgetés kívánok!



Horváth Ernő

ernohorvath@gmail.com
24 éves, műszaki informatikus. Három évvel ezelőtt ismerkedett meg komolyabban

a Linux rendszerekkel és emellett érdeklődik még a robotika és a biztonságtechnika iránt is. Ha lenne szabadideje sokat kirándulna, biciklizne és filmeket nézne.

