A gyorstárazás rejtelmei

A Linux futtatására használt géptípusok jelentős eltéréseket mutatnak a gyorstárazás hardveres kezelésének mikéntjében. Tekintsük át, hogyan próbálja meg a rendszermag mindegyik gyorstármegoldásból a legjobbat kihozni.

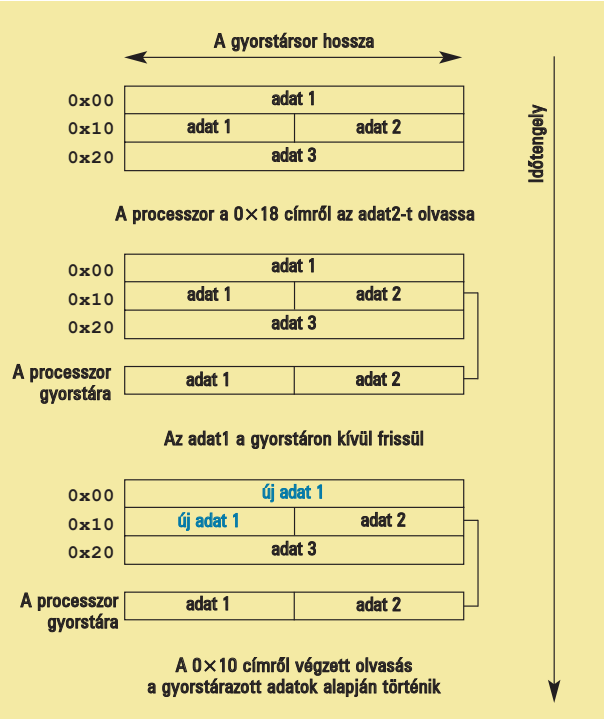
A tervezők számára az első mikroprocesszorok megjelenése óta folyamatosan gondot jelent, hogy a processzorok működési sebessége meghaladja az őket kiszolgáló memória-alrendszer sebességét. El szerették volna kerülni, hogy a processzorok a memóriából lekért adatokra ciklusokon keresztül tétlenül várjanak, ezért általános megoldásként egy gyorsabb – és drágább – memóriaterületet különítettek el a főmemória adatainak gyorstárazására. Így vált lehetővé, hogy a processzor saját képességeit maradéktalanul kihasználva üzemeljen, feltéve, hogy az ehhez szükséges adatok a gyorstárban rendelkezésre állnak.
Írásom célja, hogy a gyorstárazást rendszermag-programozási szempontból tekintsem át. Szó lesz néhány általánosan elterjedt, a gyorstárak kapcsán használt fogalomról, kifejezésről is. Írásom több részre ágazik, ezek egy része szorosabban kapcsolódik a programozás területéhez, míg mások inkább a gyorstárral foglalkoznak, mondanivalójuk a rendszermag működésétől független.

A gyorstár és jellemzői

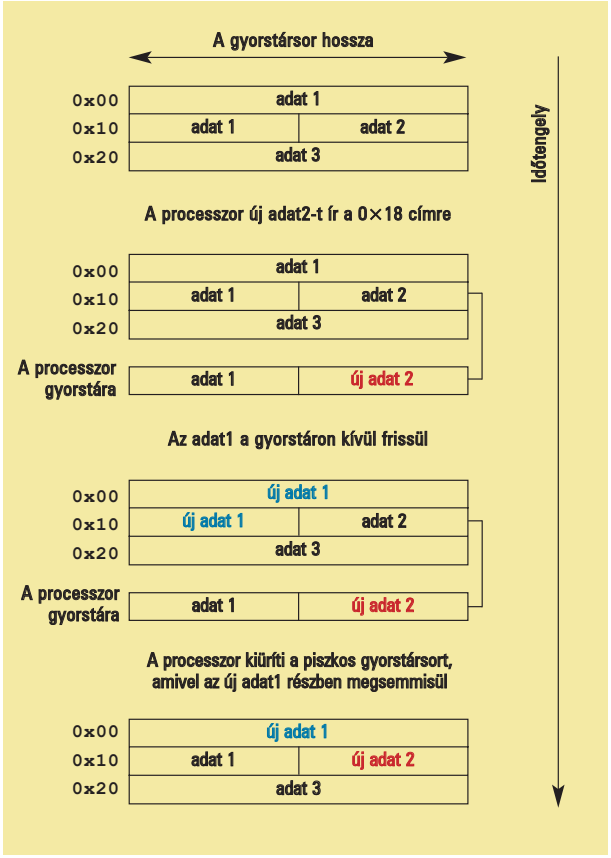
A gyorstár röviden és tömören egy olyan tárhely, amely átmenetitárazza (buffer) a memória-hozzáféréseket és szerencsés esetben rendelkezik a kért adatok másolatával. A gyorstárat, illetve – mivel több is lehet belőlük -tárakat általában úgy képzeljük el, mint egymásra tornyozott rétegeket. Legfelül a processzor, legalul a főmemória található, közöttük pedig egy vagy több gyorstár helyezkedik el. Ebben a rendszerben a gyorstárakat szintjük sorszámával azonosítjuk. A processzorhoz legközelebb lévő gyorstárat első szintűnek nevezzük (angolul Level 1, röviden L1), a további szintszámok a főmemória felé haladva növekednek.
A gyorstárba beírható és belőle kiolvasható legkisebb adategység a gyorstársor (cache line). Egy gyorstár egyik legfontosabb jellemzője az írási és olvasási sorok hossza – ez határozza meg, hogy a főmemóriából vagy az alsóbb szintű gyorstárból hány adatot kell egyszerre kiolvasni, illetve az ellenkező irányba haladva kiírni. A két érték gyakran megegyezik, így a gyorstárakat sokszor egyszerűen csak egyetlen sorhosszal jellemezzük. Ha a két érték eltér, akkor általában a nagyobbat szokás megadni.
A gyorstár másik fontos tulajdonsága a mérete. Ez adja meg, hogy hány adatot lehet benne tárolni. Ökölszabály, hogy minél nagyobb a gyorstár, annál jobb teljesítményre számíthatunk rendszerszinten.
Ha több gyorstárszint is van, ezek befoglalók vagy kizárók lehetnek. A kizárás azt jelenti, hogy egy-egy gyorstársor pontosan csak egy szinten szerepelhet. A befoglaló gyorstár megengedi, hogy egy-egy sorból egynél több szinten is legyen másolat. A sorhossz a különböző gyorstárszinteknél akár eltérő is lehet.
Végül egy gyorstár keresztülírásos (write through) vagy visszaírásos (write back) lehet. A keresztülírás azt jelenti, hogy a gyorstár tárolhatja az adat másolatát, de az írási műveletet az adott szinten csak akkor tekintjük befejezettnek, ha a kiírás az alatta lévő szinten is megtörtént. A visszaírásnál az írási művelet azonnal befejezettnek tekinthető, mihelyt az adatok bekerültek a gyorstárba. A visszaírásos gyorstáraknál a módosított sort addig piszkosnak (dirty) nevezzük, amíg az alsóbb szintű írás meg nem történik – természetesen ezt a kiírást előbb-utóbb mindenképpen meg kell ejteni.

Gyorstárkezelés és egységesség

A gyorstárak kezelése kapcsán az egyik legnagyobb gondot az egységesség, a koherencia biztosítása jelenti. A gyorstár egy sorát egységesnek tekintjük, ha a tartalma megegyezik az általa gyorstárazott főmemóriabeli adatokkal. Ha a két memóriatartalom különbözik, akkor ellentmondásosnak, inkoherensnek nevezzük őket. Az ellentmondásosság kétféle hibához vezethet; az első bármilyen gyorstárnál előállhat, ez az adatok elavulása. Ilyesmi akkor fordul elő, ha a főmemóriában tárolt adatok megváltoztak, de a gyorstár nem frissült, ezért a tartalma nem tükrözi a változásokat. Az eltérés általában hibás olvasást okoz, ennek folyamata látható az *1. ábrán*. A hiba könnyen elhárítható, hiszen a helyes adatok megtalálhatók a főmemóriában, mindössze át kell őket másolni a gyorstárba.
A második hiba csak visszaírásos gyorstáraknál állhat elő, ám ez sokkal súlyosabb, mivel adatvesztéshez vezethet. A folyamat a *2. ábrán* követhető nyomon: az adatok megváltoztak a főmemóriában, majd egy külön írási művelettel a processzor a gyorstár tartalmát is módosította. Mivel a gyorstár csak egy-egy sornyi adatot képes mozgatni, a két



1. ábra Az adatok elavulása



2. ábra Az adatok megsemmisülése a piszkos gyorstársorok miatt

változtatás nem hozható összhangba: vagy kiírás nélkül törölni kell a gyorstársort, ami a processzor által végrehajtott módosítás elvesztését okozza, vagy ki kell írni a gyorstár adatait, amivel viszont a főmemóriában elvégzett módosítás lesz az enyészeté. Minden programozónak törekednie kell az adatvesztéshez vezető helyzetek kialakulásának elkerülésére, ezt pedig a gyorstárkezelő hívások átgondolt alkalmazásával érhetik el.

A gyorstársorok ütközése

Ha két egymástól független adategység található ugyanabban a gyorstársorban, akkor sorütközésről beszélünk. Ilyenkor állhat elő az imént leírt adatvesztés lehetősége. Ha el akarjuk kerülni az ilyen helyzeteket, akkor az adatszerkezetek elhelyezésekor arra kell ügyelnünk, hogy a gyorstárakon kívül is módosítható adatokat soha ne keverjük össze a processzor által normál módon használtakkal. Ha nincs más lehetőség, mint ennek a szabálynak a felrúgása, akkor ügyeljünk arra, hogy az adatszerkezet minden külsőleg módosítható elemének elhelyezését az `L1_CACHE_BYTES` makró segítségével a gyorstár sorhosszához igazítsuk. Vizsgáljuk meg, hogy a kód futtatására kiszemelt processzortípusok közül melyik rendelkezik a legnagyobb gyorstársorméretttel és a makró értékét ezzel megegyezőre állítsuk. A legjobb, ha `kmalloc` segítségével két külön területet foglalunk le. A `kmalloc` soha nem foglal olyan területeket, amelyek azonos gyorstársorba lapolódnak át.

Gyorstárkezelő utasítások

A legalapvetőbb utasítás az `invalidate`, az érvénytelenítés, amely a megadott sort az összes gyorstárból törli. A parancs kiadását követően a törölt adatokra való hivatkozáskor a rendszer azokat újratölti a főmemóriából. Az adatok elavulásával kapcsolatos gondok tehát úgy háríthatók el, hogy a kérdéses gyorstársort az adatolvasás előtt érvénytelenítjük. Linux alatt az érvénytelenítés a következőképpen történik:

```
void
dma_cache_inv(unsigned long cím
              unsigned long méret);
```

A cím az a virtuális cím, amelyen a műveletet el kell kezdeni, a méret pedig az érvénytelenítendő adatrész hosszát adja meg. Megjegyzem, a méretet a gép önműködően a gyorstár-sorméret egész értékű többszörösére kerekíti fel.
A visszaírásos gyorstáraknál a főmemóriába tetszőleges piszkos sor kiírására, más szóval kiürítésére van módunk:

```
void
dma_cache_wback(unsigned long cím,
                unsigned long méret);
```

A kiürítést azelőtt kell elvégezni, hogy bármilyen változtatás történne a piszkos gyorstársorhoz tartozó főmemóriabeli részen. Másként szemlélve a feladatot: a kiürítést azt megelőzően kell végrehajtani, hogy bármilyen külső eszköz, például egy PCI kártya a főmemória tartalmát módosítaná; illetve a kiürítés után egy érvénytelenítési parancsot is ki kell adnunk, mielőtt a processzor valamilyen megváltozott adathoz hozzáférne.

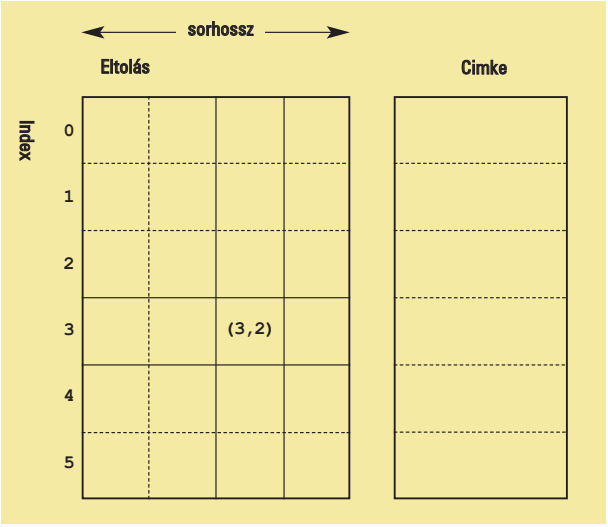
Elméletileg a visszaírásos gyorstáraknál az érvénytelenítés az adatokat a kiírásuk nélkül gyilkolja ki , amelyek így lényegében megsemmisülnek. Ezért biztonságosabb, ha kiürítő és egyben érvénytelenítő utasítást adunk ki:

```
void
dma_cache_wback_inv(unsigned long cím,
                    unsigned long méret);
```

A parancs hatására a rendszer a gyorstársorok tartalmát a főmemóriába másolja, majd a bennük lévő adatokat érvénytelennek nyilvánítja.

Gyorstártípusok

Ebben a részben a gyorstárak tényleges működéséről lesz szó. Mindebből az egyetlen nélkülözhetetlen adat az álnevesítés fogalma, amely azt a jelenséget takarja, amikor a memória adott fizikai címen elérhető tartománya több különböző gyorstársorban is szerepel. Arról, hogy a rendszermag hogyan kezeli az álneveket, a következő szakaszban lesz szó.
A közvetlenül leképezett gyorstáraknál – lásd a *3. ábrát* – a tár meghatározott hosszúságú sorokra bontható (a példában a sorhossz négy). A gyorstár mindegyik sora egyedi indexet kap, vagyis a gyorstárban található bájtok mindegyikét a sorindex és a soron belüli eltolás ismeretében tudjuk megcímezni. Minden indexhez egy rejtett érték is tartozik, ez a címke.
A rendszer minden címe három részre bontható: címke, index és eltolás (*4. ábra*). Amikor egy sor bekerül a gyorstárba, a gép a címből kiemeli a címkét és az indexet. A sor a megfelelő indexszel kerül elhelyezésre a gyorstárban, a címke tárolása pedig rejtetten, a sorban lévő adatokkal együtt történik. Amikor a processzor egy adott címre hivatkozik, a gép a megadott index alapján keres a gyorstárban. Ha a címkék egyeznek, akkor a soron belül az eltolás alapján történik a hivatkozás szerinti rész kikeresése. Ha a címkék nem egyeznek, akkor a meglévő sort ki kell üríteni a főmemóriába, a helyes sort pedig be kell tölteni a gyorstárba. Ilyenkor minden gyorstárazható címhez csak egyetlen indexsor tartozhat, ami sokszor hátrányos. Ha például a processzor olyan címsorozatról olvas, amelynek a tagjai véletlenül azonos gyorstárbeli indexszel rendelkeznek, a gyorstársort minden egyes olvasásnál ki kell üríteni és újra fel kell tölteni. Az ilyen helyzetek nem is olyan ritkák, gondoljunk például egy ciklusra, amely egy olyan adatszerkezet elemeit olvassa be, amelynek a mérete nagyjából a gyorstáréval egyezik. A közvetlenül leképezett gyorstárak indexét időnként gyorstárszínnek (cache-line coloring) is nevezik, a hibajelenséget pedig sorszínezési nehézségnek hívják.
A közvetlenül leképezett gyorstárak színezési nehézségének megoldására a gyorstárak áramköreit sokszor úgy alakítják ki, hogy a gyorstárban végzett keresésnél egyszerre egynél több sor címkéjét is meg lehessen vizsgálni. Egy N utas asszociatív gyorstárnál minden index N darab gyorstársorral – és címkével – mutat egyezést, vagyis egyszerre akár N darab azonos indexű cím is lehet a gyorstárban. A párhuzamos gyorstárbeli keresést lehetővé tévő áramkör természetesen növeli a gyártás költségét, ezért ilyet inkább a felsőbb kategóriás processzorokban találunk.
A csúcs a teljesen asszociatív gyorstár – az ilyen gyorstárakban egyáltalán nincs index, egy adott címke megkeresésekor az összehasonlítás az összes sorral egyszerre történik.



3. ábra Közvetlen leképezésű gyorstár

4. ábra Címfelosztás

A korszerű processzorok mindegyike címfordítást használ, amelynél a rendszermag vagy az alkalmazások által hivatkozott képzetes címek nem egyeznek meg azzal a fizikai címmel, amely alatt az adatok valójában megtalálhatók. A gyorstárat a címfordító egység elé és mögé is lehet helyezni, a többszintű gyorstáraknál gyakran vegyes megoldásokat találunk. Mindegyik megoldás más és más jellemzőkkel bír, más és más lehetőségeket kínál. Az alábbiakban ezeket tekintjük át.
A fizikai indexelésű, fizikai címkézésű (Physically Indexed, Physically Tagged – PIPT) gyorstáraknál a címke és az index egyaránt a fizikai memóriára utal. Ez a megoldás elegáns és egyszerű, csakhogy a PIPT gyorstárak alkalmazásánál érvényes címfordítási bejegyzésnek kell szerepelnie a processzor fordítási-előretekintő átmeneti tárában (Translation Lookaside Buffer, TLB). Ha a TLB bejegyzését a címfordítás elvégzése előtt kell kikeresni a memóriából, a gyorstár meglétéből fakadó előnyök lényegében elvesznek. Mégha van is megfelelő bejegyzés a TLB-ben, először benne kell keresni, majd utána a gyorstárban is, emiatt az ilyen megoldások lassúak.
A képzetesen indexelt, képzetesen címkézett (virtually indexed, virtually tagged – VIVT) gyorstárakban az index és a címke egyaránt képzetes-címtérbeli. Bár ilyenkor a gyorstárban lényegesen gyorsabban lehet keresni, hiszen ha a keresés sikeres, akkor sem előtte, sem utána nem kell címfordítást végezni, ez a megoldás is számos kérdést vet fel:

1. A képzetes címfordítások a rendszermag normál működése közben általában változnak, így a gyorstárnak figyelnie kell a TLB-bejegyzések, illetve a címtér változásait, és ezeknek megfelelően ki kell ürítenie azokat a sorokat, amelyeknek a fordítása megváltozott.
2. Mégha egyetlen címtér van is, ugyanahhoz a fizikai címhez több képzetes cím is tartozhat. A képzetes címek mindegyike külön is bekerülhet a gyorstárba, függet-

lenül attól, hogy ugyanarra az adatra hivatkoznak. Ezt nevezzük gyorstársor-álnevesítési nehézségnek (cache-line aliasing problem).

A VIVT gyorstárak nagyobb keresési sebességéből fakadó előnyök általában bőven ellensúlyozzák az ilyen memóriák használatából fakadó hátrányokat, így a nem x86 típusú processzoroknál rendszerint ilyen megoldásokkal találkozunk. Létezik egy hibrid gyorstártípus is, amely a VIVT gyorstárak hátrányait próbálja kiküszöbölni úgy, hogy lényeges sebességromlást ne okozzon – ezek a képzetesen indexelt, fizikailag címkézett (Virtually Indexed, Physically Tagged – VIPT) gyorstárak. Ezeknél az index képzetes cím, a címke viszont fizikai, így a címke–eltolás kettősnek kell a teljes fizikai címet megadnia. Emiatt a címkék más gyorstártípusoknál nagyobb méretűek.
A VIPT gyorstárak gyorsabbak, mint a PIPT típusúak, esetükben ugyanis a címfordítás és a gyorstárbeli keresés egyszerre is elvégezhető. Igaz, a processzor a címfordítás elvégzéséig nem tudja, hogy a gyorstársor érvényes-e, vagyis a címkék egyeznek-e.
A VIVT vonatkozású hátrányok eltűnnek, hiszen a címke fizikai, vagyis a VIPT gyorstár az álnevesítést önműködően felismeri, amikor azt látja, hogy a gyorstárban két azonos címke szerepel. A VIPT gyorstár tehát felépíthető úgy, hogy gyorstársor-álnevesítés soha ne történjen. Elméletileg létezik egy negyedik gyorstártípus is, ami fizikai indexelésű, képzetes címkézésű (Physically Indexed, Virtually Tagged – PIVT), ám ez lényegében semmire nem jó, így nem is foglalkozunk vele.

Az álnevesítési nehézség

Minden alkalommal, amikor a rendszermag a megadott fizikai laphoz egynél több képzetes leképezést hoz létre, gyorstársor-álnevesítés történik. Szerencsére a rendszermag az ilyen helyzeteket gondosan kerüli, így álnevesítés csak egyetlen esetben történik, amikor a felhasználó `mmap` segítségével használ egy fájlt. Ekkor a rendszermag egy képzetes címmel rendelkezik a fájlnak a lapgyorstárban lévő lapjaihoz, és a felhasználó is rendelkezik egy vagy több különböző képzetes címmel. Mindez azért lehetséges, mert a felhasználót semmi nem akadályozza meg abban, hogy a fájlt az `mmap` segítségével több helyen is elérhetővé tegye. Az `mmap` használatakor a rendszermag a fájlleíró listák egyikéhez adja hozzá a leképezést; ha az adatok módosítása nem megengedett, akkor az `i_mmap` listát, ha pedig az adatmódosítás is lehetséges, akkor az `i_mmap_shared` listát bővíti. Megadott lap gyorstárálneveit a következő hívással hozhatjuk összhangba:

```
void flush_dcache_page(struct page *page);
```

A hívást minden alkalommal meg kell ejteni, amikor egy lap tartalmát a rendszermag módosítja, és erre a hívásra a lap adatainak olvasása előtt kell sort keríteni, feltéve, hogy a `page->mapping->i_mmap_shared` nem üres. A géptípusoktól függő kódoknál a `flush_dcache_page` végighalad az `i_mmap_shared` listán és a gyorstárból kiüríti az adatokat. Ezt követően az `i_mmap` listán is végigmegy és érvényteleníti – ezzel minden álnevet egységesít.

Elkülönülő utasítás- és adatgyorstárak

A lehető legnagyobb hatékonyságra törekedve a processzorok gyakran külön gyorstárat alkalmaznak a végrehajtott utasítások és az általuk feldolgozott adatok tárolására. A két gyorstár sokszor teljesen elkülönül, és például az adatírásokat az utasításgyorstár nem is látja. Mindez akkor okoz gondot, ha olyan utasításokat próbálunk végrehajtani, amelyeket csak az előző pillanatban írtunk be a memóriába – ilyen például modul betöltésekor vagy ugródeszka (trampoline) használatakor fordul elő. A következő hívást kell használnunk:

```
void
flush_icache_range(unsigned long kezdet,
                   unsigned long vég);
```

A hívással biztosíthatjuk, hogy a végrehajtás megkezdése előtt az utasítások az utasításgyorstár számára is láthatóvá váljanak. A kezdet és a vég a módosított, az utasításokat tartalmazó memóriablokk kezdő és a záró címét adja meg.

A gyorstár általános kiürítése

A processzorok gyorstárainak általános kiürítésére két hívás használható:

```
void flush_cache_all(void);
```
és
```
void flush_cache_mm(struct mm_struct *mm);
```

Az első hívás a rendszer összes gyorstárát kiüríti, a második viszont csak azokat a sorokat, amelyek az `mm` mutatóval megadott folyamat címteréhez tartoznak. Mindkét hívás végrehajtása rendkívül költséges, ezért csak akkor alkalmazzuk őket, ha erre mindenképp szükség van.

Gyorstárazás SMP környezetben

Ha egynél több processzor van a rendszerben, akkor valamilyen szintig saját gyorstárral rendelkeznek.
A géptípustól függően lehetséges, hogy a rendszermag feladata annak biztosítása, hogy az egyik processzor által a saját gyorstárában végrehajtott változások a többi processzor számára is láthatók legyenek. Szerencsére a legtöbb processzor az ilyen egységességi kérdéseket hardveresen kezeli. Ha az adott processzor nem is ilyen, az itt ismertetett hívások használatakor nem kell attól tartanunk, hogy a processzorok közötti egységességgel gondjaink lennének.

Összegzés

Remélem, rövid összefoglalóm alapján mindenki át tudta tekinteni a gyorstárak működését és azt, hogy a rendszermag hogyan kezeli őket.

Linux Journal 2004. január, 117. szám

James Bottomley
A SteelEye programtervezője. A SCSI alrendszer, a Linux Voyager-átültetés és az 53c700 illesztőprogram karbantartásáért felel. DMA/eszközmodellezési-elvonatkoztatási munkájával a PA-RISC Linux fejlesztéséhez is hozzájárult.