Memóriából futó adatbázis-kezelő rendszerek

A memóriából futó adatbázis-kezelő rendszerek különösen beágyazott gépek esetében lehetnek hasznosak, amelyeknél minden egyes elhagyott folyamattal csökkenhet a termék ára és nőhet a versenyképessége.

Robbanásszerűen növekszik az intelligens, hálózati kapcsolattal rendelkező eszközök iránti érdeklődés. Akár otthon, a saját zsebünkben, akár ipari, távközlési vagy közlekedési berendezésekbe építve egyre nagyobb teljesítményű processzorokkal és egyre kifinomultabb beágyazott operációs rendszerekkel találkozunk. A hasonló eszközökben egyre gyakrabban látható alkalmazások egyik válfaját az adatbázis-kezelő rendszerek (Database Management System – DBMS) családja képezi. Az adatbázisok megszokott lakók az asztali vagy a kiszolgálógépeken, a beágyazott eszközökön azonban még új jövevényeknek számítanak. Mint minden új környezetbe kerülő „élőlénynek", az adatbázisoknak is fejlődniük kell. A DBMS-ek új fajtája a memóriából futó adatbázis-kezelő rendszerek (In-Memory Database System – IMDS) családja, amelyek a faj fejlődésének új szakaszát jelzik.

Vajon miért fordult a beágyazott rendszerek fejlesztőinek érdeklődése az adatbázisok felé? A piaci verseny kikényszeríti, hogy a set-top-box jellegű készülékek, a hálózati kapcsolók és a fogyasztói eszközök egyre okosabbakká váljanak. A növekvő számú szolgáltatás biztosításához az alkalmazásoknak egyre több, egyre összetettebb adatot kell kezelniük. Emiatt számos fejlesztő szembesült azzal, hogy kinőtte saját adatkezelő megoldását, amelynek fenntartása, illetve vele az alkalmazások igényeinek követése egyre nehézkesebbé válik.

Mindemellett az egységesedő, kereskedelmi, a polcról készen leemelhető beágyazott operációs rendszerek terjedése – és ezzel párhuzamosan az egyedi rendszerek visszaszorulása – szintén az adatbázisok elérhetőségét segíti. A széles körben terjedő operációs rendszerek – például a Linux – köré olyan felhasználói kör épül, amely további lendületet ad az adatbázisok és egyéb eszközök kereskedelmi és más jellegű fejlesztésének.

Az eszközök fejlesztői tehát a kereskedelmi adatbázisok felé fordultak, ám a meglévő beágyazott DBMS-rendszerek nem nyújtottak megfelelő megoldást. Az üzleti rendszerek igényeit követve több mint egy évtizede megjelent beágyazott adatbázisok többek között kifinomult gyorstárazásra és a rendellenes leállások utáni helyreállításra is képesek. Egy set-top-boxban vagy következő nemzedékbeli faxkészülékben azonban nem feltétlenül szükséges ilyen tudás, ami sokszor csak a rendelkezésre álló memória és processzoridő kimerülését okozza. Ráadásul a hagyományos adatbázisok az adatokat lemezen tárolják. A lemezes – mechanikus alkatrészek használatával járó – adatforgalom kezelése nagymértékben ronthatja a rendszer teljesítményét. Emiatt a hagyományos adatbázisok a sokszor valós idejű műveleteket végző beágyazott rendszerekben általában túl rossz teljesítményt nyújtanak.

A memóriából futó adatbázisok kifejezetten a beágyazott rendszerek teljesítményre és erőforrásaik korlátozottságára vonatkozó követelményeit figyelembe véve fejlődtek. Mint nevük is sugallja, az IMDS-ek a memóriából futnak, lemezkezelést nem végeznek. Egy IMDS tehát egy hagyományos adatbázis-kezelő lenne, betöltve a memóriába? Jogos a kérdés, hiszen a lemezkezelés elhagyása az új megoldás leginkább szembetűnő jellegzetessége. A Linux eleve képes ramlemezek és memóriában található fájlrendszerek létrehozására. Vajon nem volna-e érdemes egy hagyományos, jól bevált adatbázis-kezelőt, például egy MySQL-t vagy akár egy Oracle-rendszert ilyen lemezekről futtatni?

Az IMDS-ek ennél sokkal több mindenben különböznek beágyazott DBMS-testvéreiktől. Az IMDS-ek egyszerűbbek, mint a hagyományos adatbázis-kezelők. Nemcsak a lemezkezelés képességét nélkülözik, de kevesebb mozgó résszel vagy felhasználói beavatkozást igénylő folyamattal rendelkeznek. Ennek köszönhetően takarékosabban bánnak a memóriával és a processzoridővel, válaszidejük pedig a hagyományos DBMS-ek memóriából történő futtatásával elérhetőnél jóval kedvezőbb.

Ha el kell döntenünk, hogy egy IMDS megfelel-e egy adott tervezet igényeinek, pontosan meg kell értenünk, mely részek kerültek ki a rendszerből vagy módosultak jelentősebb mértékben. A legfontosabb eltéréseket az alábbiakban ismertetem.

Gyorstárazás

A fizikai lemezkezelés által okozott teljesítménycsökkenés miatt gyakorlatilag minden DBMS végez gyorstárazást, hogy az adatbázis utoljára használt részeit a memóriában tartsa. A gyorstárazást vezérlő algoritmus végzi a gyorstárnak a lemezen található adatokkal történő egyeztetését, így a gyorstár tartalma egyezni fog a fizikailag a lemezen található adatbáziséval.

A gyorstár tartalmában végzett keresés, amely azt hivatott eldönteni, vajon a kért adatok a gyorstárban vannak-e vagy sem – az utóbbi esetben kezdeményezni kell a szükséges lap betöltését és hozzáadását a gyorstárhoz a további használat érdekében –, szintén fontos feladat.

Ezek a folyamatok lemezalapú DBMS alkalmazásakor akkor is lejátszódnak, ha a rendszer ramlemezek segítségével fut. A gyorstárazási műveletek eltávolításával az IMDS-adatbázisok lényegesen egyszerűbbekké válnak, kevesebb erőforrást – memóriát és processzoridőt – igényelnek, így nagyobb teljesítményt nyújtanak.

Az adatátadásokból fakadó többletterhelés

Vegyük példaként, hogy milyen műveletek szükségesek ahhoz, hogy egy alkalmazás adatokat olvasson ki egy hagyományos lemezalapú adatbázisból, majd módosítsa, végül visszaírja őket az adatbázisba.

1. Az alkalmazás elkéri az adatokat az adatbázis API-n keresztül az adatbázis motorjától.
2. Az adatbázis motorja utasítja a fájlrendszert, hogy keresse elő az adatokat a fizikai adathordozóról.
3. A fájlrendszer az adatokat bemásolja a saját gyorstárába, illetve a motornak is átadja őket.
4. Az adatbázis egy másolatot helyez el a saját gyorstárában, valamint az adatokat továbbadja az alkalmazásnak.

5. Az alkalmazás módosítja az adatokat, majd az adatbázis API-n keresztül átadja őket az adatbázisnak.
6. Az adatbázis motorja a módosított adatokat bemásolja az adatbázis gyorstárába.
7. Az adatbázis gyorstárának tartalma a fájlrendszerbe időnként kiírásra kerül, ahol elsőként a fájlrendszer gyorstárának frissítése történik meg.
8. Végül megtörténik az adatok kiírása a fizikai adathordozóra.

A fenti lépéseket egy hagyományos adatbázis-kezelőben nem lehet kiiktatni, még akkor sem, ha a teljes feldolgozás a memóriában folyik. Nem szabad elfeledkezni a tranzakciók kezeléséhez szükséges másolatok készítéséről és az adatmozgatásokról sem, amelyekkel a fenti példában nem is foglalkoztunk.

A memóriából futó adatbázisok ezzel szemben kevés vagy semennyi adatmozgatást nem végeznek. Az alkalmazás a saját változóiban ugyan az adatokról készíthet másolatokat, ám lényegében erre sincs szükség. Ehelyett az IMDS olyan mutatókat ad át az alkalmazásnak, amelyek közvetlenül az adatbázisban található adatokra mutatnak, így az alkalmazás közvetlenül az adatokkal dolgozhat. Az adatok védelme így sem marad el, hiszen a mutatók kezelése az adatbázis API-n keresztül történik, amely gondoskodik megfelelő használatukról.

A többszörös adatmozgatás kiiktatása növeli a feldolgozás teljesítményét. A többszörös másolatok elhagyása csökkenti a szükséges memória mennyiségét, az egyszerűbb működés pedig nagyobb megbízhatóságot eredményez.

Tranzakciók feldolgozása

Végzetes leállás, például áramkimaradás esetén a lemezalapú adatbázisok a rendszer újraindításakor a naplófájlok alapján a jóváhagyott tranzakciókat újra lejátsszák, a részlegeseket pedig visszagördítik. A lemezalapú adatbázisokba „beledrótozzák” a tranzakciós naplók fenntartását, illetve az ezek és a gyorstár tartalmának kiírását a lemezre, ha egy-egy tranzakció véget ért.

A memóriából futó adatbázisok is képesek a tranzakciók kezelésére. Ennek érdekében az IMDS megőrzi a módosított vagy törölt objektumok másolatát, illetve listába fogja a tranzakció során hozzáadott adatbázislapokat. Amikor az alkalmazás jóváhagyja a tranzakciót, a korábbi állapotot és az új lapokat tároló memóriarész felszabadul – az eljárás gyors és hatékony. Ha egy tranzakciót meg kell szakítani – mert például a bejövő adatfolyam megszakadt –, a korábbi állapot áll vissza, az új adatokat tároló lapok pedig felszabadulnak.

Végzetes leállás esetén a memóriában tárolt adatbázis elvész – ez az egyik legfontosabb különbség a lemezalapú adatbázisokkal szemben. A készülék bekapcsolása után az IMDS-t újra fel kell tölteni. Nyilvánvaló, hogy nincs értelme tranzakciós naplókat készíteni, amivel újabb összetett, nagymennyiségű memóriát igénylő folyamatot iktattunk ki az IMDS-ből.

Természetesen ezzel a korlátozással nem felelhetünk meg minden környezetben, de a beágyazott rendszerek világában bőségesen találni olyan adattároló és -kezelő alkalmazásokat, amelyek adattárát újraindítás után valós időben gond nélkül újra fel lehet tölteni. Ilyen példa a műsornézegető set-top-box, ami műholdról tölti le az adatokat, vagy egy vezeték nélküli hozzáférési pont, amelyet az útválasztási protokollok hálózati elrendezést felderítő folyamatai révén lehet újra feltölteni adatokkal. A hasonló eszközök fejlesztői boldogan korlátozzák a tranzakciós szolgáltatások kiterjedtségét, ha cserébe növelhetik a rendszer teljesítményét, amely így ráadásul kisebb teljesítményű vassal is beéri.

Természetesen az adatok helyi tárolásáról sem kell teljesen lemondani. Az IMDS segítségével az alkalmazás megnyithat egy folyamot – ez lehet foglalat, csővezeték vagy fájlmutató –, majd utasíthatja az adatbázis-kezelő motorját egy adatbázis-lenyomat beolvasására vagy kiírására a folyamon keresztül. Így készíthető például olyan alapértelmezett lenyomat, amely a rendszer indítása után a működéshez egyfajta kiindulópontot jelent. A folyam másik végén egy program lehet, vagy valamilyen fájl egy tetszőleges fájlrendszen (mágneses, flash- vagy optikai tárolón stb.).

Alkalmazási példa: IP-útválasztók

Hol érdemes IMDS-megoldást választani? A memóriából futó adatbázisok iránti igény számos helyen felmerült, az alábbi környezet talán a leginkább jellemző, az IMDS-megoldás képességeinek kihasználására a leginkább alkalmas: az IP-útválasztó elterjedt, beágyazott operációs rendszert futtató internetes eszköz.

A korszerű IP-útválasztók útválasztási táblakezelő (RTM) programot futtatnak, amely a készülék legfontosabb feladatát, a hálózaton keresztül befutó adatcsomagok következő állomásának meghatározását végzi. Az útválasztási protokollok folyamatosan figyelik a használható útvonalakat és a többi útválasztóeszköz állapotát, majd az eszköz útválasztó tábláját frissítik a megfelelő adatokkal.

Az útválasztó táblák jellemzően az RTM program futásának eredményeként alakulnak ki. Ez a megoldás hozza magával a következő nemzedékbeli útválasztók fejlesztésének egyik legnagyobb kihívását. Ahogy sokasodnak a készülék által ellátott feladatok, az útválasztási tábla kezelése egyre összetettebb munkát jelent. A saját fejlesztésű, az útválasztási adatok kezelését végző programok – az adatbázis-kezelőkkel ellentétben – általában nem képesek az összetett adategyüttesek kezelésére vagy a többszörös hozzáférés biztosítására.

Emellett – ahogyan az alkalmazásba beleépített adatkezelő megoldásoknál is általában – az útválasztó táblák bővíthetősége és megbízhatósága korlátozott. Az adatkezelő eljárások módosítása visszahat a teljes RTM felépítésére, ennek következményeként kellemetlen meglepetések és minőségbiztosítási gondok jelentkezhetnek. Hasonlóan gondot okozó pont a méretezhetőség: az egy-egy feladatra jól használható, saját fejlesztésű adatkezelők általában összeomlanak, ha növekvő terhelés éri őket. Az Internet növekedése eközben gyors előrelépést igényelne az útválasztási megoldások terén, ám az eszközök fejlesztését hátráltatják az elavult megközelítés szerint készített alkalmazások.

A memóriából futó adatbázisok fejlődésével a DBMS-megoldások számos beágyazott rendszerben elérhetővé válnak.

A beágyazott rendszerek fejlesztői számára a már bizonyított adatbázis-kezelő megoldások olyan előnyöket kínálnak, mint hatékony adatelhelyezés és hozzáférési módok, egyszerű és szabványos adatkezelési eljárások, többszörös hozzáférési lehetőség, az adatok épségének védelme, megnövelt rugalmasság és hibatűrés. A DBMS-ek új fajtájának alkalmazásával egyszerűsödhet a beágyazott termékek fejlesztése, miközben nem kell lemondani az összetett alkalmazásokról, a kiváló rendelkezésre állásról és a megbízhatóságról.

Linux Journal 2002. szeptember, 101. szám

Steve Graves
a McObject, az eXtremeDB nevű memóriából futó adatbáziskezelő rendszert fejlesztő cég elnöke és társalapítója. A Raima Corporation elnökeként úttörő szerepet vállalt a DBMS-megoldásoknak a beágyazott rendszereken történő terjesztésében.