A PostgreSQL és a teljesítménynövelés

Pörgessük fel a gépet, hogy a legtöbbet hozhassuk ki e nyílt forráskódú programból.

A PostgreSQL objektumközpontú adatbázis-kezelő program, amelyet a Föld minden csücskéből származó fejlesztők készítenek az Interneten keresztül, és tulajdonképpen az Oracle, illetve az Informix kereskedelmi alkalmazások nyílt forrású megfelelője.
A PostgreSQL-t eredetileg a Berkeley-n (University of California) készítették. Az egyetem csapata 1996-ban kezdte meg az Interneten keresztüli fejlesztést: az ötletek cseréjére elektronikus leveleket, a kódfájlok átviteléhez pedig fájlkiszolgálókat használtak. Mára a PostgreSQL képességei teljesítmény és megbízhatóság tekintetében felveszik a versenyt a kereskedelmi alkalmazásokkal. Rendelkezik tranzakciókezeléssel, nézetekkel (views), tárolt függvényekkel és hivatkozási épségkezeléssel (referential integrity constraints) egyaránt. Nagyszámú programozási felületet támogat, többek között az ODBC-t, a Javát (JDBC), a Tcl/Tk-t, a PHP-t, a Perlt és a Pythont. Hála a tehetséges internetes fejlesztőgárdának, a PostgreSQL folyamatosan és hihetetlen ütemben fejlődik.

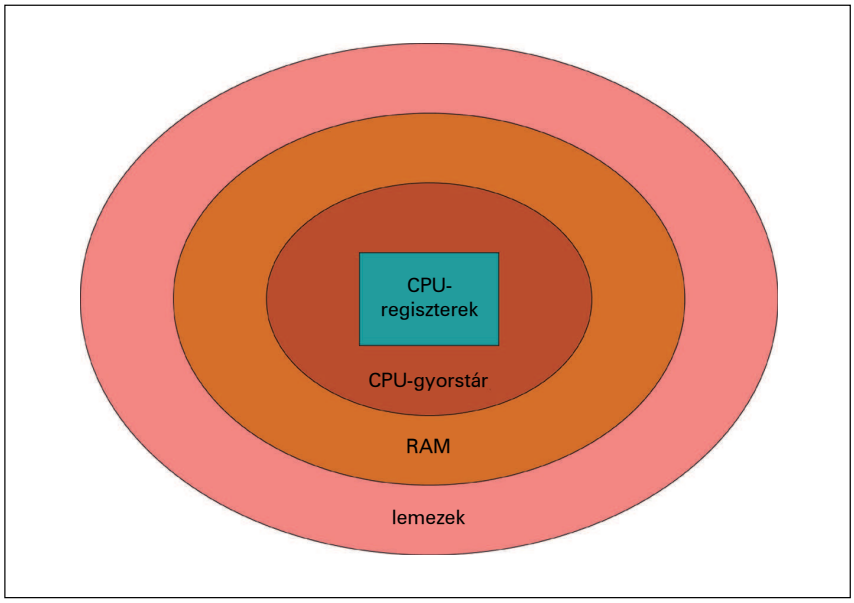
Teljesítményalapfogalmak

Az adatbázis-teljesítmény növelése kétféle módon történhet: az egyik út, hogy a processzor-, a memória- és a merevlemez-terhelést növeljük. A másik lehetőség az adatbázisnak küldött lekérdezések egyszerűsítése. Írásukban a teljesítménynövelés vas felőli oldalát mutatjuk be. A lekérdezések hatékonyabbá tételének eszközei: a `CREATE INDEX`, `VACUUM`, `VACUUM ANALYZE`, `CLUSTER`, `EXPLAIN` és hasonló SQL-parancsok. Ezek bővebb taglalásával könyvemben foglalkozom (PostgreSQL: Introduction and Concepts, elérhető a ➲ http://www.postgresql.org/docs/awbook.html címen is).
A vas teljesítményére vonatkozó dolgok megértéséhez fontos, hogy pontosan lássuk, mi is történik a gép belsejében. Az egyszerűség kedvéért a számítógépre most úgy gondoljunk, mint tárolók körülvette központi egységre (CPU). A processzorral egy lapkán található néhány CPU-regiszter, ezekben számításközi adatok, mutatók és számlálók tárolhatók. Mellettük helyezkedik el a processzor-gyorstár, amely a legutóbb elért adatokat tartalmazza. A gyorstáron kívül még itt található a nagy mennyiségű, véletlen elérésű főmemória (RAM) is, amely a végrehajtandó programot és az adatokat tartalmazza. E főmemória mögött helyezkednek el a merevlemezek, amelyek még nagyobb adatmennyiség tárolására képesek. Egyedül ezek a merevlemezek tekinthetők állandó tárolóhelynek, így mindent, amit meg kell tartani, ide kell helyezni, mielőtt a gépet kikapcsoljuk (lásd a *táblázatot*).
A központi egységet körülvevő tárolóeszközöket az alábbi *ábrán* mutatjuk be.

Hogyan tartsuk a processzor mellett az adatokat?

Az adatok áramlása a különböző tárolóeszközök között önműködően zajlik. A fordítóprogramok meghatározzák, mely adatokat szükséges regiszterekben tárolni. A processzor logikája a gyakran használt adatokat a CPU gyorstárban tartja. Az operációs rendszer pedig meghatározza, melyik adatot kell a RAM-ban tartani és cserélgeti a merevlemezen tároltakkal.
A CPU regisztereket és a processzor-

Jelenleg a PostgreSQL képességei teljesítmény és megbízhatóság tekintetében felveszik a versenyt a kereskedelmi alkalmazásokkal.



Tárolóterületek

Láthatjuk, hogy a processzortól távolodva a tárolóméretek egyre növekednek.
Az eszményi az lenne, ha hatalmas mennyiségű állandó memóriát helyeznénk közvetlenül a központi egység mellé, csakhogy ez igencsak költséges megoldásnak bizonyulna. A gyakorlatban a processzorhoz közel a leggyakrabban használt adatokat tároljuk, a kevéssé használtakat pedig távolabb, és akkor hozzuk közelebb, ha szükség van rájuk.

A tárterületek típusai

Tárterület	*Egység*
CPU regiszterek	bájt
CPU gyorstár	kilobájt
RAM	megabájt
lemezmeghajtók	gigabájt

gyorstárat adatbázis-felügyelőként nemigen lehet hatékonyan egyszerűsíteni. A hatékony adatbázis-gyorsítás a minél nagyobb mennyiségű hasznos adat RAM-ba töltését jelenti, mivel így, ahol csak lehet, megelőzzük a lemezelérést.

Azt hihetnénk, hogy ez könnyű feladat, pedig nem az. A számítógép memóriája számos dolgot tartalmaz, beleértve a végrehajtható programot, a programadatokat és a vermet, valamint a PostgreSQL megosztott és a rendszermag gyorstárát. A helyes egyszerűsítés azt jelenti, hogy annyi adatot tartunk a memóriában, amennyit csak lehetséges, de csak addig, amíg ez nem befolyásolja hátrányosan az operációs rendszer más részeit.

A PostgreSQL megosztott gyorstára

A PostgreSQL az adatokat közvetlenül nem változtatja meg a merevlemezen, hanem megosztott gyorstárába olvastatja be. A PostgreSQL-motor ezután ide írja, illetve innen olvassa a blokkokat, amelyek végül visszaíródnak a merevlemezre. Az a motor, amelynek valamelyik táblát el kell érnie, a szükséges blokkot először ebben a gyorstárban keresi, és ha már úgyis itt van, azon nyomban folytathatja a feldolgozást. Amennyiben a blokkot nem találja, operációs rendszerhívást hajt végre, hogy betöltődjön. A blokkok vagy a rendszermag gyorstárából, vagy a merevlemezről töltődnek be. Ez azonban időigényes művelet lehet.

Az alapértelmezett PostgreSQL-beállítás 64 megosztott gyorstárat foglal le, melyek egyenként nyolc kilobájt méretűek. A gyorstárak számának növelésével egyre valószínűbbé válik, hogy a motor a keresett adatot megtalálja a gyorstárban, így takarítva meg az időigényes rendszerhívást.

Mekkora számít túl nagynak?

Úgy vélhetnénk, jó elgondolás az összes memóriát a PostgreSQL megosztott gyorstárának adni. Csakhogy ha ezt tesszük, nem marad elég hely a rendszermag vagy a többi program számára. A PostgreSQL megosztott gyorstárának megfelelő mérete az a legnagyobb hasznos méret, amely az egyéb tevékenységeket még nem befolyásolja hátrányosan.

Ha meg akarjuk érteni a káros folyamatokat, előbb azt kell megértenünk, miképpen kezeli a Unix operációs rendszer a memóriát. Ha elég memória áll rendelkezésre ahhoz, hogy az összes program és adat a memóriában maradhasson, nincs gond. Ha azonban a memóriában már nem fér el minden, a rendszermag a memórialapokat egy lemezterületre kezdi el átirányítani, ezt csereterületnek (swap) nevezzük. A végén használt lapokat kirakjuk (kilapozzuk) a csereterületre. Ezt a műveletet *lapkiolvasásnak* (swap pageout) nevezzük. A lapkidobások nem okoznak bonyodalmat, mivel inaktív időszakban hajtódnak végre. Az viszont nehézséget okoz, ha ezeket a lapokat valamikor vissza kell hozni a csereterületről, azaz a régi lapot, amit már egyszer kimozgattunk a csereterületre, vissza kell másolni a memóriába. Ezt lapbeolvasásnak (swap pagein) nevezzük. Ez kedvezőtlen folyamat, mert amíg a lapbeolvasás a csereterületről véget nem ér, addig a programfutás szünetel.

A lapbeolvasást a vmstat, a sar vagy hasonló eszközökkel figyelhetjük meg, amelyek jelzik, ha a hatékony működéshez már nincs elegendő memória. A cseretár-lapbeolvasásokat ne tévesszük össze a hagyományos lapbeolvasásokkal, amelyekbe a fájlrendszerből olvasó normál lemezműveletek is beletartozhatnak. Amennyiben a csereterület-lapbeolvasásokat nem találjuk, úgy a gyakori lapozások jól mutatják, hogy egyúttal lapbeolvasások is zajlanak.

A gyorstár méretének hatásai

Valószínűleg sokan elcsodálkoznak azon, miért ilyen központi kérdés a gyorstár mérete. Először is képzeljük el, hogy a PostgreSQL megosztott gyorstára elég nagy ahhoz, hogy a teljes táblát tárolni tudja. A táblára irányuló ismételt szekvenciális keresések egyáltalán nem igényelnek lemezelérést, mivel minden adat eleve a gyorstárban helyezkedik el. Most gondoljunk arra, hogy a gyorstár egy blokkal kisebb, mint a tábla. A tábla szekvenciális keresése esetén minden táblablokk a gyorstárba fog töltődni, kivéve az utolsót. Amikor erre a blokkra lesz szükség, a legrégebben betöltött blokk eltávolítódik, mely jelen esetben a tábla legelső blokkja. Amikor a következő szekvenciális keresésre kerül a sor, az első blokk már nincs a tárban, és a betöltéshez ismét a legrégebbi blokkot kell eltávolítani, ez pedig jelen esetben már a tábla második blokkja. És így tovább: az éppen beolvasott blokk mindig kiüti a következő szükségeset, egészen a tábla végéig. Ez természetesen nagyon végletes példa, de jól látható, hogy egyetlen blokk hiánya a gyorstár hatékonyságát száz százalékról nullára csökkentette. Ez azt mutatja, hogy a helyes gyorstárméret megválasztása nagymértékben hat a teljesítményre.

A megosztott gyorstár méretének helyes beállítása

Eszményi esetben a PostgreSQL megosztott gyorstára elég nagy ahhoz, hogy a legtöbbet használt táblákat tárolni tudja, és elég kicsi ahhoz, hogy még ne használjon csereterületet. Ne feledjük, hogy a *Postmaster* (a Postgres főprogram) már indításkor lefoglalja az összes megosztott memóriát. Ez a terület mindig ekkora marad, még akkor is, ha az adatbázist senki sem használja. Néhány operációs rendszer dobja azokat a megosztott memórialapokat, amelyekre semmi sem hivatkozik, míg mások a RAM-ban hagyják. A PostgreSQL 7.2 Administrator's Guide-ban fontos adatokra bukkanhatunk a különféle operációs rendszerekhez tartozó rendszermag-beállításokról: ➲ http://www.postgresql.org/devel-corner/docs/admin/kernel-resources.html.

A rendező memória-köteg (batch) mérete

Egy további szabályozható érték a rendező kötegfájlokhoz használt memória mennyisége. Amikor a PostgreSQL nagyméretű táblákat vagy eredményeket rendez, a műveletet darabokban hajtja végre, a köztes eredményeket ideiglenes fájlokban tárolva. Ezeket a fájlokat később egybeolvassa és újrarendezi, amíg minden sor rendezett nem lesz. A köteg méretének növelése kevesebb ideiglenes fájlt eredményez, és gyakran gyorsabb rendezést tesz lehetővé. Ezzel szemben viszont, ha az adag túl nagy, lapolvasásokat okozhat, mivel a rendező köteg egy része a rendezés során kilapolódik. Ebben az esetben sokkal célszerűbb lenne kisebb adagokat és sok ideiglenes fájlt használni. Tehát még egyszer: a csereterület lapolvasásai határozzák meg, mikor foglaltunk le túl sok memóriát. Ne feledjük azt sem, hogy minden háttérművelet ezt az értéket használja az összes rendezés során, legyen az `ORDER BY`, `CREATE INDEX` vagy `merge join`. Több egyidejű rendezés ennek a memóriának természetesen a többszörösét foglalja le.

Gyorstárméret és rendezőméret

A gyorstárméret és a rendezőméret a memóriahasználatra egyaránt kihat, így egyiket sem növelhetjük anélkül, hogy ne lennénk befolyással a másikra. Ne feledjük, hogy a gyorstárméretet a Postmaster induláskor foglalja le, míg a rendezőterület mérete az éppen futó rendezések számától függően változhat. A gyorstár mérete általában jobban számít, mint a rendezőméret. Emellett bizonyos `ORDER BY`-t, `CREATE INDEX`-et vagy `merge join`-t használó lekérdezések sebessége nőhet, ha nagyobb rendező kötegfájlméretet adunk meg.

Ezenkívül számos operációs rendszer megköti, hogy mennyi megosztott memóriát használhatunk fel. E határ növelése viszont már az operációs rendszer belső ismeretét igényli, hiszen újra be kell állítani és újra kell fordítani a magot. További adatok a PostgreSQL 7.1 Administrator's Guide-ban ➲ http://www.postgresql.org/docs/admin/kernel-resources.html címen olvashatók.

Lemezkezelés

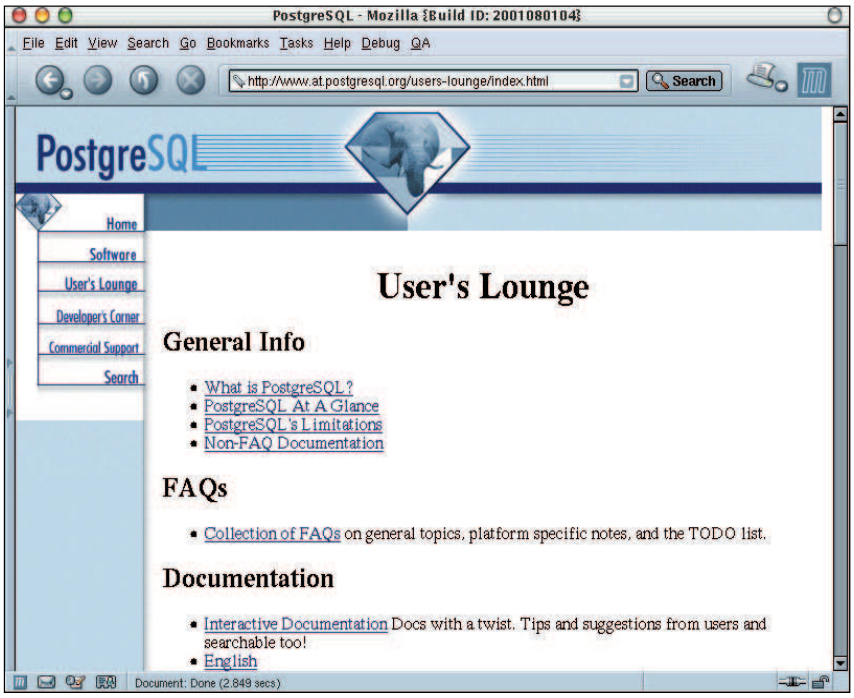
A lemezmeghajtók fizikai felépítése ezeket az eszközöket a fentebb említett egyéb tárolóeszközöktől a teljesítmény szempontjából különbözteti meg. A többi tárolóeszköz bármely bájtot ugyanolyan sebességgel képes elérni. A lemezmeghajtó – pörgő lemezei és mozgó feje miatt – a fej pillanatnyi helyzetéhez közeli adatokat sokkal gyorsabban éri el, mint a távolabb lévőket. A fej átmozgatása a lemez másik cilinderére elég sok időt vesz igénybe és ezt a Unix-rendszermag fejlesztői természetesen tudják. Amikor nagyméretű fájlokat tárolunk a lemezen, a fájldarabkákat egymáshoz minél közelebb próbálják meg elhelyezni. Tegyük fel, hogy a fájl tíz blokkot foglal el a lemezen. Az operációs rendszer az 1–5. blokkokat az egyik cilinderen, a 6–10. blokkokat a másik cilinderen helyezi el. Ha a fájlt elejétől a végéig beolvassuk, mindössze két fejmozgás szükséges – az első, hogy elérjük a 1–5. blokkokat tároló cilindert, valamint még egy, hogy átlépjünk a 6–10. blokkok cilinderére. Ha viszont a fájladatokat nem sorban olvassuk be, hanem például 1., 6., 2., 7., 3., 8., 4., 9., 5., 10. sorrendben; máris tíz fejmozgás szükséges. Amint láthatjuk, lemezek esetében a szekvenciális elérés sokkal gyorsabb, mint a véletlen elérés, ezért ha a tábla jelentős részét be kell olvasni, a PostgreSQL az indexekre a szekvenciális keresést részesíti előnyben. Az előbbiek fényében már a gyorstár értékét is láthatjuk.

Többszörös lemezhasználat

A lemezfej jó néhányszor körbefordul az adatbázis-tevékenység alatt. Ha túl sok olvasási, illetve írási kérelem érkezik, a meghajtó telítődhet, ez pedig gyenge teljesítményt eredményez (a vmstat és a sar az egyes lemezek használatának mértékéről ugyancsak szolgáltat adatokat).
A lemeztúlterheltség elkerülésére az egyik megoldás az, ha a PostgreSQL néhány adatfájlját egy másik meghajtóra helyezzük. Ne feledjük, itt a fájlok azonos meghajtónak másik fájlrendszerére mozgatása jelent segítséget, ugyanis a meghajtó összes fájlrendszere ugyanazokat a fejeket használja. Az adatbázis-elérés a lemezek között többféleképpen is szétoszthető:

- *Adatbázisok mozgatása* – a initlocation segítségével más lemezeken is készíthetünk adatbázisokat.
- *Táblák mozgatása* – közvetett hivatkozások segítségével a táblákat vagy indexeket másik lemezre is áthelyezhetjük. Ezt a mozgatást csak akkor szabad elvégezni, ha a PostgreSQL le van állítva. Továbbá a PostgreSQL semmit sem tud a közvetett hivatkozásokról (symbolic link), így ha letöröljük, majd újra létrehozzuk a táblákat, akkor azok az adatbázishoz rendelt alapértelmezett helyen fognak létrejönni. A 7.1 változat alatt a pg_database.oid és a pg_class.relfilenode rendeli az adatbázis-, a tábla- és az indexneveket a fájlnevekhez.
- *Indexek átmozgatása* – a közvetett hivatkozások lehetővé teszik, hogy az indexeket adatszerkezet-táblájukból más meghajtókra helyezzük át. Ezáltal megvalósítható, hogy míg az egyik lemezen indexkeresést hajthatunk végre, addig a másik adatszerkezet-keresést (heap lookups) végezhet.
- *Joinok mozgatása* – közvetett hivatkozások használatával az összekapcsolt táblákat is külön lemezre helyezhetjük. Ha az A és a B táblát összekapcsoljuk, az A tábla keresései az egyik meghajtón, a B tábla keresései a másik meghajtón hajtódhatnak végre.
- *Naplók átmozgatása* – közvetett hivatkozásokkal a thepg_xlog könyvtárat is átmozgathatjuk másik lemezre. (A Pg_xlog csak a PostgreSQL 7.1-es vagy későbbi kiadásokban létezik.) Más írásszolgáltatásokkal ellentétben a PostgreSQL-naplózást mindig a tranzakció befejezése előtt kell a lemezre kiírni. A gyorstárat ezekhez az írásokhoz nem használhatjuk. Ha a naplózáshoz külön lemez áll rendelkezésre, akkor a lemezfej folyamatosan a napló pillanatnyi cilinderében maradhat, és nem kell kivárni a fejmozgatási időt. Használhatjuk a Postgres `-F` kapcsolót is, ami a naplóírások azonnali ürítését megakadályozza, de egy rendszerösszeomlás esetén mindent mentésekből kell helyreállítani.

További lehetőség a RAID-képességek használata, amikor egyetlen fájlrendszert osztunk szét számos meghajtó közt.



➲ http://www.at.postgresql.org/users-lounge/index.html

Összegzés

Szerencsére a PostgreSQL-nek nincs szüksége túl sok gyorsításra. A legtöbb érték a hatékony teljesítmény eléréséhez önműködően beállítódik. A felügyelő befolyásolhatja a gyorstár méretét és a rendezőméretet a rendelkezésre álló memória minél jobb kihasználása céljából. A lemezműveleteket több lemez közt is megoszthatjuk. További értékeket állíthatunk be a share/etc/postgresq/postgresql.conf fájlban. Amennyiben a PostgreSQL-t különlegesebb értékekkel is ki szeretnénk próbálni, akkor ezt a fájlt másoljuk a data/postgresql.conf helyre.

Bruce Momjian a PostgreSQL Global Development Team társalapítója és a Great Bridge adatbázis-fejlesztés LLC (➲ http://www.greatbridge.com/) alelnökeként tevékenykedik. Ő a szerzője az Addison-Wesley kiadásában megjelent PostgreSQL: Introduction and Concepts című könyvnek.