Gépköltöztetések és összeomlás-kezelés

Minden kiszolgáló egyedi, következésképpen mindegyikhez egyedi katasztrófatervvel kell rendelkezni. Készítsük el a saját tervünket, és alkossunk néhány olyan szabályt, amelyekkel rendbehozhatjuk az összeomlásokat.

A rendszerfelügyelet világában, bizony, elég sokat számít a tapasztalat. A rendszergazdákat riogató hibás alkatrészek, pusztító programhibák és biztonsági betörések láttán egyre valószínűbb, hogy érdemes valamilyen hatékony mentési stratégiát, biztonsági rendszabályokat és visszaállítási tervet összeállítani. A kérdés nem is az, hogy bekövetkezik-e katasztrófa a gépünkön, hanem sokkal inkább az, hogy mikor és milyen formában fog megtörténni.
Mindezt 2003 augusztusának közepén írom, kevesebb mint egy héttel azután, hogy a saját kiszolgálómat (***lerner.co.il***) új virtuális helyre költöztettem. A sorok nagy része az elsötétült New Yorkban íródott, ahol néhány órát akartam eltölteni egy üzleti megbeszélésen – végül első kézből tapasztalhattam meg, milyen is egy nagyarányú technológiai katasztrófa. Ó igen, és amikor éppen nem a kiszolgálót vittem át vagy ücsörögtem a sötétben, a hét nagy részében internetkapcsolat nélkül maradtam, minthogy éppen az új lakásomba költöztem be Chicagóban. Ezért azután ebben a hónapban egy kicsit szüneteltetjük a Bricolage-ról és az egyéb tartalomkezelőkről szóló sorozatunkat, helyette megnézzük, hogyan költöztessük át kiszolgálóinkat, és miként készítsük el weboldalaink, adatbázisrendszereink katasztrófaterveit. Természetesen minden webhely és kiszolgáló más és más, tehát rászolgálnak a megkülönböztető figyelemre, vagyis a lehető legjobb tervet készítsük el hozzájuk. Némi előretekintéssel azt mondhatom, hogy nem is olyan nehéz kiszolgálónkat egyik helyről a másikra átvinni, az alkatrészhibákat vagy programösszeomlásokat kezelni, illetve elkerülni az olyan nagyszabású katasztrófát, mint amilyet az Egyesült Államok északkeleti része megtapasztalhatott ezen a nyáron.

Kiszolgálóáthelyezés

Az elmúlt néhány évben párszor át kellett helyeznem a kiszolgálómat, és minden költözés olajozottabban zajlott, mint az előző. Az igazat megvallva egy új gépre történő áttérésnek nem kellene nehéznek vagy fáradságosnak lennie, de mindenképpen körültekintően kell megtervezni. Minden lépést csak olyan módon szabad megtenni, hogy közben azt feltételezzük, hogy arra a pontra esetleg még vissza kell térnünk.
A lehető legegyszerűbb kiszolgálófajta, amit egy másik gépre áttehetünk, a statikus weboldalakból álló és csupán CGI programokat használó webhely. Ilyen esetben csak néhány kérdést kell feltennünk magunknak:

- Tartalmazzák-e az Apache-beállítások felhasznált moduljainkat? Ha sokat használjuk a `mod_rewrite` modult, vagy teljesen kiaknáztuk a `mod_speling` (nem elírás, egy l) előnyeit, nem árt, ha kétszer is ellenőrizzük, hogy megvannak-e ezek a modulok. Amennyiben statikusan építettük (compiled) őket a kiszolgálónkba, a `httpd -l` paranccsal kilistázhatjuk őket. Ha dinamikus modul formájában használjuk (DSO-ként), akkor Apache-telepítésünk ***libexec*** alkönyvtárban kell őket keresnünk, ahol az elérhető DSO-fájlok találhatók. Bármely DSO tetszés szerint betölthető, ha az Apache-beállításfájlban kiadjuk a megfelelő `LoadModule` utasítást.
- Melyik felhasználó és csoport neve alatt fut az Apache? A rendszergazdák véleménye gyakran különbözik, ha az kerül szóba, hogy melyik felhasználói és csoportazonosítót kellene használnia az Apache-kiszolgálónak. Sokan az alapértelmezett ***nobody*** felhasználóként futtatják. Mások (mint például én) jobban szeretik, ha az Apache saját felhasználói és csoportnévvel rendelkezik, és szükség esetén új felhasználókat vesznek fel az Apache-csoportba. Megint csak mások az Apache `suexec` képességét használják ki, úgy fordítják le, hogy egy vagy több másik felhasználóként tudjon futni. Bármelyik esetről legyen is szó, győződjünk meg róla, hogy a kiszolgálónk Apache-felhasználói, illetve csoportazonosítói megfelelően be vannak-e állítva az új kiszolgáló */etc/passwd* és */etc/group* állományaiban, illetve az Apache saját beállításfájljaiban.
- Hol van a `DocumentRoot`? Alapértelmezés szerint az Apache azt feltételezi, hogy `DocumentRoot` értéke */usr/local/apache/htdocs*. Ezt az alapértéket a `DocumentRoot` utasítással tudjuk az Apache beállításaiban megváltoztatni, az általunk használt operációs rendszernek vagy terjesztésnek megfelelően. Amennyiben az Apache RPM alapú változatát használjuk, például valamelyik Red Hat stílusú terjesztéshez, a `DocumentRoot` egyaránt lehet a */var/www* alatt vagy egy másik könyvtárban. Ennek semmi hatása nincs az URL-ekre, illetve a programjainkra és dokumentumainkra nézvést, de azért nem árt, ha kétszer is megnézzük, hogy a könyvtár, amibe az állományainkat másoljuk, tényleg a megfelelő hely-e.
- Milyen nyelveken és modulokon alapulnak CGI programjaink? Amennyiben oldalunk CGI programokat is használ, ezek közül legalább egy valószínűleg használ valamilyen külső modult vagy könyvtárat. A ***CGI.pm***, azaz a Perl module for CGI programs (Perl modul CGI programokhoz) több éve része már a Perl-terjesztéseknek, de továbbra is rendszeresen frissítik. Ezért, amennyiben a legfrissebb változat képességeit használjuk, nem árt, ha biztosra megyünk. Ugyanez vonatkozik minden más felhasznált modulra. Az egyik ügyfelem egy régi Perl ***Storable*** modult használt, és végül rájött (nagy nehezen), hogy az új változatra frissítéssel együtt megszűnt a kapcsolattartás az örökös rendszerekkel.

DNS

Minden kiszolgálóáttelepítés sarokköve a DNS-bejegyzések átmentése. Bár az emberek jobban szeretnek neveket használni

(például *www.lerner.co.il*), a hálózati kapcsolatok olyan számalapú IP-címeket használnak, mint például a 69.55.225.93.
A DNS (Domain Name System, vagyis tartománynév-rendszer) feladata, hogy ezeket az emberi neveket számítógépes számokká alakítsa. A kiszolgálóáttelepítés kényes pontja a DNS-bejegyzések átgondolt módosítása.
A gond a DNS-rendszerrel nem is a gép-IP fordítás ténye, hanem az, hogy a DNS-eredmények gyorstárazódnak. Végtére is mi sem szeretnénk, ha minden egyes HTTP-kérés után a DNS-kiszolgálónknak kellene válaszolnia. Az ilyen kérelmek indokolatlanul nagy forgalmat gerjesztenének, és feleslegesen meghosszabbítanák a HTTP-kérelmek kiszolgálását.
Ezért, amikor DNS-kérelmet adunk ki, valójában nem az eredeti, felhatalmazott kiszolgálótól kapjuk a választ, hanem a helyi DNS-kiszolgálótól. Az ugyanis, amennyiben a kért elemet a mostanában kapott eredmények között megtalálja, rögtön visszaadja, anélkül, hogy a fő kiszolgálótól adatot kérne le. Más szavakkal akkor, amikor az `nslookup www.lerner.co.il` DNS-lekérdezést hajt végre ISP-nk DNS-kiszolgálóján. A kiszolgáló vagy a saját gyorstárából ad vissza adatot, vagy a ***lerner.co.il*** tartomány felhatalmazott kiszolgálójához fordul.
Amikor a kiszolgálót az egyik gépről a másikra átvisszük, érdemes kis számra állítanunk a DNS-kiszolgáló TTL (Time To Live, azaz érvényességi idő) értékét, így az azt gyorstárazó DNS-kiszolgálók nem fognak hibás értékeket visszaadni. Úgy találtam, hogy a TTL értékét bőven elegendő 300 másodpercre (öt percre) beállítani. Ha a kiszolgáló teljesen átköltözött, a TTL értékét ismét növelhetjük valamilyen hosszabb időmennyiségre, például hat órára, hogy csökkentsük kiszolgálónk terhelését.
Nézzük meg nagy vonalakban, hogy milyen lépéseket szükséges megtennünk a sikeres áttelepítéshez, amennyiben HTTP-kiszolgálónkat két szolgáltató között költöztetjük át:

- Győződjünk meg róla, hogy új szolgáltatónk DNS-kiszolgálója képes-e (és hajlandó-e) DNS rendszert (előre és vissza) szolgáltatni jelenlegi a IP-címünk és gépneveink számára. Azaz új szolgáltatónk DNS-kiszolgálójának a régi szolgáltatónkhoz kellene irányítania az embereket. A TTL értékét állítsuk öt percre.
- Frissítsük tartományunk ***WHOIS*** bejegyzéseit, jelezve, hogy az új szolgáltatónk a felhatalmazott DNS-kiszolgáló. Akár egy-két napot is igénybe vehet, mire a dolog a teljes DNS-rendszeren keresztülgyűrűzik. Amennyiben új DNS-kiszolgálónk ugyanolyan eredményt ad, mint a régi, az egyetlen módszer, amivel megnézhetjük, hogy működnek-e a dolgok, ha ***WHOIS***, avagy `nslookup -type=ns` *`tartománynév.com`* típusú keresést végzünk.
- A ***WHOIS***-bejegyzések frissítése után kezdjük meg a dolgok átmozgatását. Győződjünk meg róla, hogy az összes programot helyesen állítottuk-e be, az összes modul megvan-e, és hogy frissítettük-e a DNS-kiszolgálót. Amennyiben a DNS-kiszolgáló tartományunk lekéréseire nem válaszol, igen nagy gondban leszünk, amikor a ***WHOIS***-bejegyzések az új kiszolgálóra fognak mutatni.
- Ha minden azonosnak tűnik (ennek megvalósítására jó módszer lehet az `rsync` használata a régi rendszerről az újra való áttéréskor), a DNS-meghatározásokat változtassuk meg úgy, hogy a gépnév a régi helyett az új IP-címre mutasson.

A futtatott kiszolgáló típusától függően esetleg érdemes kikapcsolni a régi HTTP-kiszolgálót, ezzel is csökkentve az átállás okozta zavart. Például ha kikapcsoljuk a régi HTTP-kiszolgálót, még mielőtt átkapcsolnánk a DNS-t, biztosak lehetünk benne, hogy a naplófájlok között nem lesz átfedés, így azokat a ***Webalizer*** vagy az ***Analog*** eszközökkel összefűzhetjük és felhasználhatjuk, ha körül akarunk nézni bennük.
Ezen a ponton az új rendszeren már mindennek megfelelően kell működnie. Nem árt viszont a lehető legtöbb hivatkozást ellenőriznünk, különösen azokat, amelyek CGI programokat, kiszolgálóoldali csatolásokat és nem hagyományos modulokat hívnak meg, vagy amelyeknek szokatlan jogosultságokra van szükségük. Mint mindig, a folyamat során a HTTP-kiszolgáló hibanaplója lesz a legjobb barátunk; ha esetleg a dolgok rosszra fordulnak, a hibanaplóban általában megtalálhatjuk, hogy mi lehet a gond.

Adatbázisok

Természetesen a fentiek során feltételeztük, hogy egy viszonylag egyszerű oldallal dolgozunk. A legtöbb korszerű weboldal azonban ilyen vagy olyan okból kifolyólag relációs adatbázisokat is tartalmaz. Használatuk egyaránt népszerű egységességük és rugalmasságuk, valamint a gyors fejlesztés lehetősége és az általánosan használt paradigmák beépíthetősége folytán, ezeket könnyű használni és a hibáikat elhárítani.
A relációs adatbázisok egy vagy több táblában tárolják az adatokat, amelyeket általában adatbázisba szerveznek. (Igen, némileg zavaró, hogy egyetlen adatbázis-kiszolgáló több adatbázist is tartalmazhat, amelyek mindegyike egy vagy több táblát is tárolhat, de pontosan ez a helyzet.) Ha adatbázisunkat az egyik rendszerről a másikra szeretnénk átvinni, egyaránt át kell juttatnunk az adatbázissémát (táblák, nézetek és függvénymeghatározások), valamint magát az adatot. Természetesen az adatbázis tulajdonosa valamilyen adatbázis-felhasználó (aki általában nem azonos a Unix-felhasználóval), és egyedi jogosultságkészlettel rendelkezik rajta.
Amennyiben honlapunk adatbázist is tartalmaz, ezt is át kell vinnünk a régi rendszerünkről az új alá, ideértve a tulajdonosokkal és a jogosultságokkal kapcsolatos adatokat is. Hogy ezt miképpen kell megtennünk, attól függ, hogy milyen adatbázist használunk, és van-e valamilyen buktatója e folyamatnak.
Az ISAM/MyISAM táblákat (az alapértelmezett és máig legnépszerűbb lehetőség) használó MySQL-adatbázisokat az egyik MySQL rendszerről egyszerűen átmásolhatjuk a másikra. Általában az adatbázissal kapcsolatos valamennyi állományt megtaláljuk a */var/lib/mysql* könyvtárban, az adatbázis nevével azonos könyvtárban. Így, ha a *foo* adatbázist szeretnénk átvinni, másoljuk a */var/lib/mysql/foo* könyvtárat és a benne lévő állományokat az új gépünk */var/lib/mysql/foo* könyvtárába. (Mielőtt ezt megtennénk, ne felejtsük el lezárni az adatbázist.) Indítsuk el az új rendszer kiszolgálóját, és mindennek jól kell működnie.
PostgreSQL alatt, amely az adatbázissémákat és az adatokat alacsony szintű bináris formában tárolja, már korántsem ilyen egyszerűek a dolgok. Amennyiben `tar` vagy `rsync` programmal próbálnánk átmásolni PostgreSQL adatbázisunkat, igen valószínűtlen, hogy működne; és ha mégis, valószínűleg komoly adatkárosodást fog szenvedni, sőt akár a háttéradatbázis-kiszolgálót is lefagyaszthatja. Ehelyett használjuk inkább a `pg_dump` eszközt, amellyel a PostgreSQL adatbázist `CREATE`, `INSERT` és `COPY` parancsok sorozataként, egyszerű szöveges formátumban kimenthetjük. Például:

```
pg_dump -U mydb mydb > /tmp/mydb-dump.txt
```

A `-U mydb` szakasz azt jelzi, hogy a `mydb` adatbázis-felhasználót szeretnénk használni. Ide a megfelelő nevet kell beírni.

A kimentett adatokat a következő paranccsal lehet visszatölteni egy működő adatbázisba:

```
$ createdb -U mydb mydb2
$ psql -U mydb mydb2 < /tmp/mydb-dump.txt
```

A parancsokat követve most két adatbázissal rendelkezünk (***mydb*** és ***mydb2***), amelyeket egyaránt a `mydb` felhasználó birtokol. MySQL alatt igen könnyű az ilyen helyzetek kezelése, hiszen egy beépített elsődleges, illetve másodlagos rendszerrel rendelkezik. Az egyik adatbázis lehet az elsődleges, ez fogadja majd az összes SQL-parancsot és lekérdezést, míg a másik csendben követi, lehetővé téve, hogy összeomlás esetén felváltsa az elsődlegest.

Áramszünet esetén...

Mint korábban említettem, augusztusban New York városában volt „szerencsém” megtapasztalni azt a komoly áramkimaradást, amelynek emberek milliói érezhették hatását az Egyesült Államokban és Kanadában. Ironikus, de a kimaradás körülbelül egy órával az után történt, hogy az egyik potenciális ügyfelünk kiszolgálótelepét meglátogattam, ahol bemutatta, miképpen menti cégük az összes adatukat egy connecticuti távoli csomópontra. (Végtére is mennyi az esélye annak, hogy valami egyszerre hasson New York városban és Connecticutban? Legalábbis erre gondoltam, miközben helyeslően bólogattam egy órával a kimaradás előtt.)

Ha valaki az enyémhez hasonló helyzetben volt, New Yorkon kívül található kiszolgálóit nemigen zavarta a kimaradás. Ráadásul a legtöbb szolgáltatóhely háttérgenerátorral is rendelkezik, amely szükség esetén részben vagy egészben fedezni tudja az épület áramigényét.

De ha kiszolgálóink az irodában állnak, vagy egy egyszerű szünetmentes tápegységre (UPS) bíztuk futásuk állandóságát, a kiszolgáló valószínűleg elérhetetlen lesz egy olyan áramszünet esetén, mint amilyet augusztus közepén megtapasztalhattunk, s amely egyes északkeleti részeken 48 óránál is tovább tartott. Amennyiben kiszolgálónk üzletünk létfontosságú része, komolyan érdemes elgondolkodnunk azon, nem lenne-e érdemes egy társszolgáltatónál elhelyezni.

De időnként még a társszolgáltatónál elhelyezett kiszolgálók is leállhatnak és lekapcsolódhatnak – ezt többéves, saját tapasztalataim alapján állíthatom. Ez azt jelenti, hogyha nagymértékben függünk a kiszolgálótól, rendszeresen mentenünk kell. Továbbá folyamatosan másolnunk kell egy másik fizikai helyen (és lehetőleg egy másik cégnél) elhelyezett kiszolgálóra.

A különbségeknek ezzel vége is – a programbeállítások jobb, ha azonosak maradnak. Amennyiben kiszolgálónkon a HTML-oldalakhoz, sablonokhoz és programkönyvtárakhoz, valamint a CGI programokhoz az `rsync` eszközt használjuk, és hasonlóan automatizáljuk az adatbázismentések áttöltését a második kiszolgálóra, a második kiszolgáló közel azonos másolata lesz az elsőnek, és a megfelelő pillanatban szolgálatba lesz állítható. Ennél is továbbléphetünk és akár egyszerre is használhatjuk a két kiszolgálót. Természetesen ez már jóval nehezebb feladat, mivel megköveteli, hogy vagy egyetlen adatbázis-kiszolgálót használjunk (s így egyetlen helyre bízzuk rá az adatokat), avagy az adatbázisokat sűrűn össze kell hangolnunk. Ennek ellenére ez mindenképpen lehetséges megoldás, különösen azokon a helyeken, amelyek nagy, statikus oldalakat tartalmaznak – ez jól látszik az Akamai sikeréből, amelynek számos redundáns kiszolgálója van szerte a világon. Minél statikusabb egy oldal, annál könnyebb lemásolni és a futását folyamatossá tenni.

Az üzleti adatbázisprogramoknak (például Oracle) még mindig van egy előnyük a PostgreSQL és MySQL rendszerekhez képest. Igaz ugyan, hogy az utóbbi bizonyos mértékben képes elsődleges, másodlagos másolatkészítésre, de az összehangolás egyáltalán nem olyan kifinomult és nem is annyira hibatűrő, mint az Oracle nyújtotta változata. Idővel várhatóan ez a helyzet is megváltozik, ahogy az ilyen megoldások egyre gyakoribbakká válnak.

Összegzés

Új helyre költözni nehéz, kiszolgálónkat új helyre vagy gépre költöztetni nemkülönben. Egy jó áttelepítési tervvel felvértezve, lépésenként haladva és munkánkat minden ponton ellenőrizve (olyan segédeszközökkel, mint az `nslookup`, `dig`, `telnet`, valamint a Perl LWP, vagy a hasonló műveleteket végző `curl` eszközkészlettel szállított HEAD és GET programok) átállásunk zökkenőmentes lehet.

Linux Journal 2003. november, 115. szám

Reuven M. Lerner (➲ http:/www.lerner.co.il/atf) Nyílt forrású programokra, valamint web- és adatbázis-alkalmazásokra szakosodott tanácsadó. Könyve, a Core Perl, 2002 januárjában jelent meg a Prentice Hall gondozásában. Reuven feleségével és lányaival Izraelben, Modi'inben él.