

Upravljanje Web poslužiteljem

(Steve Burnett)

U ovom poglavlju

Kontrola procesa djece (potomaka) na poslužitelju **710**

Uporaba procesne datoteke **711**

Povećavanje uspješnosti softvera za poslužitelj **712**

Automatizacija rotacije prijavne datoteke **713**

O sigurnosti **714**

Ostale teme vezane uz prilagođavanje **717**

Jedna od velikih prednosti Apache Web poslužitelja je visoka prilagodivost. Gotovo svaka osobina koja zahtijeva posebnu vrstu učitavanja poslužitelja je mogućnost, što znači da možete osobine podrediti brzini. To znači da je Apache napravljen tako da bude brz i efikasan. Sve te Apachea osobine rijetko ćete kada morati uporabiti, toliko ih ima.

Apache je napravljen tako da administratorima lokacije omogućuje nadzor nad sigurnosti i funkcionalnosti. Za neke je lokacije, poput ISP-a, vrlo važna mogućnost nadzora vrste funkcija. Također, na Webu je potrebna potpuna fleksibilnost, čak i na račun sigurnosti ili oštećenja koja može nanijeti npr. CGI skripta (puno ljudi smatra da je CGI općenito rupa u sigurnosti).

Kontrola procesa djece (potomaka) na poslužitelju

► Vidi “Pokretanje Apachea”, str. 677

Kao što ste naučili u 35. poglavlju, “Početak u Apacheu”, Apache rabi *demone*, koji se katkad zovu *djeca (potomci)*, istovremeno izvodeći i odgovarajući na upite. Iako je veličina tih demona promjenljiva, postoje ograničenja veličine i brzine rasta. Ta veličina je kritična; jedna od glavnih problema izvedbe starijih poslužitelja koji su izvršavali `fork()` sustavski poziv na svaki zahtjev bio je neograničeni ukupni broj istovremenih demona. Kada bi se glavna memorija zasitila, računalo bi se zablokiralo.

Drugi softver za poslužitelj omogućuje vam određivanje točnog broja procesa, s mogućnosti otkazivanja ako su potomci zauzeti kada stigne novi zahtjev. To nije baš najbolji način – ako se izvodi 30 potomaka, a potrebno je samo 5, to će usporiti izvedbu – ali ovim modelom uklanja se potreba zaštite od blokiranja.

Dakle, Apacheov model je početi s određenim brojem stalnih procesa i uvijek osigurati neki broj rezervnih procesa ukoliko se pojave istovremeni zahtjevi. Tako možete pokrenuti nekoliko procesa kako bi se održao minimalni broj rezervi. Ako imate više poslužitelja u mirovanju od najvećeg broja rezervi, suvišni se mogu onemogućiti. Postoji maksimalni broj procesa kako bi se zaštitilo računalo od blokiranja.

Algoritam za zaštitu od previše procesa treba smjestiti u `/usr/local/apache/httpd.conf`, a izgleda ovako:

```
StartServers 10
MinSpareServers 5
MaxSpareServers 10
MaxClients 150
```

Ti su brojevi podrazumijevani. Upute kažu da se pri pokretanju Apachea automatski pokreće 10 potomaka (StartServer), bez obzira na početno učitavanje. Ako su svi potomci “zauzeti”, ostali se zahtjevi odbijaju. To zahtijeva najmanje pet (MinSpareServers), ali ne više od 10 (MaxSpareServers) slobodnih poslužitelja koji se bave preopterećenjem (tj, kada se pojavi puno zahtjeva). Ta preopterećenja često uzrokuju pretraživači koji otvaraju odvojene TCP veze za svaku sliku na stranici u pokušaju da korisniku poboljšaju izvedbu, često na račun poslužitelja i mreže.

BILJEŠKA Te upute su dio osnovnog skupa osobina Apachea i trebaju biti dostupne u svakoj inačici Apachea. ■

Obično se postiže stabilan broj procesa potomaka, ali ako dolazi puno zahtjeva, možete doći do MaxClients granice. Zahtjevi će tada čekati na red. Ako i dalje pristižu, korisnici će vidjeti poruku da je veza odbijena. No, i to je bolje od neograničenog broja istovremenih procesa, jer će poslužitelj stalno pokretati potomke, a nitko neće dobiti odgovor od njega.

Preporučuje se da ne prilagođavate MaxClients jer je 150 dovoljan broj za većinu sustava. No, ako želite provjeriti koliko zahtjeva mogu podnijeti vaši najjači procesori, možete postaviti veći broj MaxClients. Ako izvodite Web poslužitelj na računalu s ograničenom memorijom ili CPU-om, postavljanje manje vrijednosti MaxClients ima smisla.

Uporaba procesne datoteke

Višeprocetni model opisan u prethodnom odjeljku zahtijeva određenu komunikaciju između roditeljskog i procesa-potomka, pa je odabrana metoda procesne datoteke gdje svaki potomak ima komadić prostora u datoteci u koji može pisati. Roditeljski httpd proces iz te datoteke dobiva izvješće o statusu i donosi odluke hoće li pokrenuti još procesa potomaka ili zaustaviti mirujuće procese.

U početku je to bila datoteka smještena u /tmp direktoriju. Zbog problema s postavkama Linuxa kada se brišu /tmp direktoriji, procesna je datoteka smještena u /var/log/ direktorij. Uz pomoć instrukcije ScoreBoardFile možete odrediti mjesto te datoteke.

Program httpd_monitor u poddirektoriju support/ u Apacheu može pokrenuti procesnu datoteku kako bi dobio sliku o stanju procesa potomaka, jesu li oni tek na početku, aktivni, miruju ili ne rade. To vam može pokazati jesu li vaše postavke za MaxSpareServers i MinSpareServers u redu. Možete ga smatrati sličnim sustavskoj naredbi iostat.

Povećavanje uspješnosti softvera za poslužitelj

Standardno izvršenje možete poboljšati na puno načina, uključujući pametnije oblikovanje resursa, osobina koje se mogu isključiti, čak i elemente na razini operativnog sustava i hardvera koji se mogu adresirati. Svi ti činitelji pridonose razlici između običnog Web poslužitelja i Web poslužitelja visokih performansi.

Većina poboljšanja koja nisu povezana s hardverom mogu se grupirati u tri kategorije: ona koja smanjuju učitavanje u CPU, ona koja smanjuju iznos ulazno-izlaznih operacija na disku i ona koja smanjuju zahtjeve za memorijom.

SSI pretprocesiranje

Već smo pisali o pretprocesiranju. CPU mora analizirati datoteku tražeći pretprocesorske instrukcije; analiza datoteke je detaljnija od čitanja datoteke.

Problemi u pristupu disku javljaju se zbog dva, tri, četiri ili više odvojenih pristupa disku kako bi se sakupile sve stranice. Na primjer, tipični SSI dokument može imati zaglavlje i podnožje koje treba skupiti zajedno u memoriji. To znači da treba tri puta pristupiti disku. Ako su HTML datoteke velike, razlika nije tako očita. No, HTML datoteke su obično male, pa je kašnjenje zbog pristupa disku relativno veliko. Problem je isti i kod CGI skripti; ako imate SSI stranicu s dvije CGI skripte, vjerojatno ćete dva puta pristupiti disku.

Uporaba .htaccess datoteka

Apache za kontrolu pristupa direktorijima rabi posebne datoteke - .htaccess. Te datoteke rade hijerarhijski. Kada se pojavi zahtjev za /path/path2/dir1/dir2/foo, Apache će potražiti .htaccess datoteku u *svakom* poddirektoriju. U navedenom zahtjevu to je najmanje pet poddirektorija – što predstavlja značajno opterećenje diska koje je najbolje izbjeći.

Želite li izbjeći preveliki broj pristupa disku, trebate sve što kontrolirate preko .htaccess datoteka staviti u access.conf konfiguracijsku datoteku ili čak u srm.conf datoteku.

Trebate li potražiti .htaccess datoteke u poddirektorijima, a možete to suziti na određeni poddirektorij, uz pomoć AllowOverride možete postići da poslužitelj traži .htaccess datoteke u tom poddirektoriju.

Pretpostavimo da je vaš korijenski direktorij dokumenta u /www/htdocs i da želite isključiti pretraživanje svih .htaccess datoteka osim onih u /www/htdoc/dir1/dir2.

U access.config konfiguracijsku datoteku ćete upisati:

```
<Directory /www/htdocs>
Options All
AllowOverride None
</Directory>
<Directory /www/htdocs/dir1/dir2>
Options All
AllowOverride All
</Directory>
```

Važan je redoslijed direktorija tako da drugi <Directory> ne bude ispred prvog.

Uporaba .asis datoteka

.asis datoteke razlikuju se prema HTTP zaglavljima koja su izravno umetnuta u samu datoteku. Ona su korisna optimizacija određenih vrsta datoteka, kao što su s poslužitelja gurane animacije koje zahtijevaju sposobnost postavljanja vlastitih zaglavlja i obično ih uništavaju CGI skripte. Obična gurana CGI skripta ima dodatne troškove dinamičkog slaganja slika, dok se u .asis datoteci sve može povezati u jednu datoteku, smanjujući ulazno-izlazno opterećenje i zauzeće memorije i CPU-a.

► **Vidi “As-is datoteke”, str 701**

Jedino što gubite uporabom .asis parametra je sposobnost vremenski raspoređeno guranje, gdje postoji vremenski razmak između okvira umetnutih kao sleep() (sustavski poziv gdje program zastane definirani broj sekundi). No kako guranje ima ograničenu širinu pojasa, smatra se da je vremenska uvjetovanost dvosmislena osobina.

Automatizacija rotacije prijavne datoteke

Jedan od ciljeva administratora lokacije treba biti automatizacija rotacije prijavne datoteke i datoteke s podacima o pogreški. Čak i slabo opterećeni poslužitelj generirat će nekoliko megabajta prijava tijekom dana.

Osnovni element ove rotacije je da Web poslužitelj prestane pisati u staru datoteku prijave i počne pisati u drugu bez prekidanja usluge drugim korisnicima. Najbolji je način *neznatno preimenovanje* prijave i slanje SIGHUP signala roditeljskom procesu. To znači preimenovanje u access_log.0 ili slično na istom disku, na istoj podjeli. Zašto? Svaki potomak ima opis datoteke otvoren u datoteci s podacima o prijavi. Kada preimenujete datoteku, opisni blok datoteke i dalje će pokazivati na istu stvarnu prijavu sve dok potomak ne primi “eho” SIGHUP od roditeljskog procesa. Kada se to dogodi, opisni blok datoteke se zatvara, javlja se novi i stvara novi access_log. To je jedini način da se ne izgube izvješća o prometu na poslužitelju dok rotirate prijavu.

Evo primjera skripte:

```
#!/bin/sh
logdir="/usr/local/etc/httpd/logs"      # name of the log directory
accllog="access_log"                    # name of the access log
errlog="error_log"                      # name of the error log
pidfile="$logdir/httpd.pid"            # file that stores the parent's
                                        # process ID

mv $logdir/$accllog $logdir/$accllog.0
mv $logdir/$errlog $logdir/$errlog.0
kill -HUP `cat $pidfile`
```

Ova se skripta treba izvoditi kao i korisnik koji je prvobitno pokrenuo HTTP demon – na primjer “korijenski direktorij”. Možete napisati dodatne skripte kako biste smjestili te .0 datoteke u neku arhivu; ja rabim godine i mjesece kao poddirektorije, npr. prijave za 1. siječanj 1997 idu u datoteku 1997/01/01 izvan direktorija gdje ima puno mjesta. Tako se datoteke s podacima o prijavi micanjem direktorija mogu smjestiti negdje drugdje.

0 sigurnosti

Sigurnost poslužitelja je sigurno najveća briga administratora Web lokacije. Izvođenje Web poslužitelja je sigurnosni rizik, a također i rad na mreži. No, vaš Web poslužitelj može biti siguran od vanjskih utjecaja (ako ljudi pokušaju provaliti na vašu lokaciju) i unutrašnjih utjecaja (korisnici vaše Web lokacije pogreškom ili namjerno otvaraju sigurnosne rupe).

CGI skripte

Većina CGI skripti se temelje na ljski i rabe Perl ili interpretirane programe C ljske, a ne kompilirane programe. Zato se dogodilo puno napada pri korištenju “osobina” u tim ljskama. U ovom odjeljku govorit ćemo o nekim osnovnim stvarima koje se tiču sigurnosti CGI skripti.

CGI skripta izvodi se s korisničkim ID-om. Podrazumijevano je to “nobody”. Želite li se zaštititi, takav korisnik nije povjerljiv, pa on neće imati odobrenje za čitanje vaših privatnih datoteka, kao ni dopuštenje za pisanje u njih. Neke CGI skripte – na primjer, knjiga posjetitelja gdje korisnici mogu pisati komentare o vašoj Web lokaciji – zahtijevat će pristup s mogućnosti pisanja u neke datoteke. Zato je najbolje odrediti direktorij u koji CGI skripte mogu pisati bez ikakvih problema.

Nadalje, administratori lokacije mogu ograničiti uporabu CGI na određene direktorije uporabom upute ScriptAlias. Ako imate .cgi nastavak za CGI datoteke, za bolju kontrolu uporabe CGI datoteka možete uporabiti uputu OptionsExecCGI u access.conf.

U slijedećem primjeru, ispis 37.1, imate kontrolu pristupa uz pomoć ExecCGI, ukoliko želite omogućiti da se CGI rabi svugdje na lokaciji (korijenski direktorij dokumenata /home/htdocs) osim za “korisničke” poddirektorije.

Ispis 37.1 Primjer access.conf datoteke koji prikazuje informacije o oblikovanju direktorija

```
<Directory /home/htdocs/>
Options Indexes FollowSymLinks Includes Multiviews ExecCGI
AllowOverride None
</Directory>

<Directory /home/htdocs/users/>
Options Indexes SymLinksIfOwnerMatch IncludesNOEXEC Multiviews
AllowOverride None
</Directory>
```

Kako ExecCGI nije na popisu Options za drugi direktorij, nitko tu ne može uporabiti CGI skripte.

Na žalost, ne postoji sredina između omogućavanja i onemogućavanja CGI skripti. Većina jezika koja se rabi za CGI programe nemaju ugrađene sigurnosne mjere. Zato pravila poput “ne diraj tvrdi disk” ili “ne šalji /etc/passwd datoteku u e-poštu izvan korisnika” trebaju biti ista kao da imate stvarne korisnike Linuxa na koje trebate primijeniti ista ograničenja. To će se možda u budućnosti promijeniti.

Pretprocesiranje

Iz ispisa 37.1 vidite još jednu promjenu *povjerljivog* dijela poslužitelja i **nesigurnog**: argument Includes u Options promijenjen je u IncludesNOEXEC. Ta uputa omogućuje vašim korisnicima u koje nemate povjerenja uporabu pretprocesorskih instrukcija bez izvođenja #include iz CGI skripte ili #exec naredbe. Naredba #exec je posebno neželjena u nesigurnoj okolini jer autoru HTML-a daje pristup kroz ljsku.

Simboličke veze

U nesigurnoj okolini UNIX-ove *simboličke veze* (omogućuju povezivanje preko granica datotečnog sustava) također mogu zabrinuti administratore Web lokacija. Zlobni korisnici lako mogu napraviti simboličke veze iz direktorija gdje smiju pisati do objekta ili resursa, čak izvan korijenskog direktorija dokumenata. Na primjer, korisnik može napraviti vezu prema /etc/passwd datoteci, a potom je poslati na Web izlažući vašu lokaciju potencijalnim provalama – posebno ako vaš sustav ne rabi rezervne lozinke.

BILJEŠKA Na Alta Vista pretraživačkom servisu (www.altavista.digital.com) pretraživanje po riječima koje imaju datoteke lozinki (*bin*, *root*, *FTP* itd.) preokrenulo je upute na stvarne datoteke lozinki koje su ostavljene u javnosti. Te datoteke lozinki imaju šifrirane lozinke koje je lako razbiti za nekoliko sati. ■

Za zaštitu, administrator lokacije ima dvije mogućnosti: dozvoliti simboličku vezu ako su vlasnik veze i vlasnik resursa s kojim se povezuje isti pomoću `SymLinksOwnerMatch` ili onemogućiti simboličke veze tako da se ne aktivira `FollowSymLinks` ili `SymLinksIfOwnerMatch`.

Uočite da oba `<Directory>` dijela ispisa 37.1 imaju `AllowOverride None`. Onemogućavanje simboličkih veza je najsigurnija postavka; ako ipak želite omogućiti usklađivanje nečeg u tim direktorijima, možete to odrediti uputom `AllowOverride`.

Prostori u koje se može pisati

Posljednja prijetnja sigurnosti za Web poslužitelje je posluživanje prostora gdje se može pisati preko HTTP-a. Na primjer, puno lokacija omogućuje pristup FTP direktoriju izravno preko Weba. To je sigurnosna rupa ukoliko netko tamo stavi zločestu CGI skriptu koja poziva `#exec` da napravi neku štetu. Ako vam treba prostor gdje se može pisati, možete se zaštititi na ove načine:

- Postavku za instrukciju `Options` napravite ovako:

```
Option Indexes
```

Možete uprabit i `None`, ali `Indexes` ne uzrokuju nikakve sigurnosne probleme tako dugo dok vam ne smeta da drugi mogu učitati sve što pošaljete.

- Postavite `AllowOverride None` tako da ljudi ne mogu poslati `.htaccess` datoteku u vaš direktorij i promijeniti vaše postavke.
- Pripazite da FTP demon koji rabite ne dozvoljava postavljanje izvršnog bita. Tako sprečavate izvršenje poslanih CGI skripti. Ako za aktiviranje pretprocesnih instrukcija rabite `XBitHack`, također možete spriječiti njihovo izvođenje. To je uglavnom sigurnosna pohrana podataka za postavljanje `Options` kao na ispisu 37.1, što bi vas trebalo zaštititi od prijetnji.

Ista pravila se primjenjuju ako imate CGI skripte koje proizvode svoje HTML ili CGI datoteke. Na primjer, program `guestbook.cgi` stalno dodaje poslanih osobne informacije u datoteku `guestbook.html`; zato se sadržaj te HTML datoteke na smatra sigurnim. Ako se CGI skripta dvostruko provjerava i otklanja “opasan” kôd, sigurnost je veća.

Ostale teme vezane uz prilagodavanje

Apache Web poslužitelj više je podređen sigurnosti nego brzini; prema izreci “Možeš promašiti brže nego što druga osoba može pucati”. Izvedba je također u redu, jer svaki Web poslužitelj može lako zasititi T1 liniju.

Oni koje zanimaju najnovije mogućnosti poslužitelja mogu ugoditi svoja računala tako da poboljšaju sposobnost odziva. Njajefinije poboljšanje za svaki Web poslužitelj je dodavanje RAM-a. Treba izbjegavati da Web poslužitelj mora tražiti zamjensku memoriju izvan diska. Jedan od načina izbjegavanja je ograničavanje broja MaxClients unutar dostupnog RAM-a. Najbolji izvor informacija o prilagodavanju izvedbe je na Web lokaciji <http://www.apache.org/docs/misc/perf-tuning.html>.

Odavde...

Možete nučiti više o postavkama, oblikovanju i izvođenju Apache Web poslužitelja u 35. poglavlju “Početak u Apacheu”. U tom poglavlju je detaljan uvod u Apache Web poslužitelj.