UNIVERZITET U BEOGRADU
FAKULTET ORGANIZACIONIH NAUKA

Izabrana poglavlja
iz informacionih sistema

~ Seminarski rad ~

NginX

Mentor: Prof. dr Siniša Nešković

Student: Vojislav Ristivojević 2016/3079

Beograd, 2017.

Sadržaj

Uvod

Osnovna funkcija svakog web servera je izvršavanje aplikacije koja prima HTTP zahteve i slanje adekvatnih odgovora na iste. Ponekad se istim terminom (web server) označava i hardverski uređaj na kome se izvršava softverski sistem koji zapravo pruža uslugu isporučivanja web strana, ali ćemo, u nastavku rada, koristiti primarno značenje koje se odnosi na softver.

Apache, Microsoft Internet Information Services (IIS) i Nginx zajedno pokrivaju oko 85% tržišta web servera. Svaki od njih ima svoje mesto, odnosno korisničku bazu za čije potrebe su adekvatni. Odluka o korišćenju određenog servera treba da se donese veoma rano u projektovanju celokupnog sistema jer su kasnije migracije često komplikovane i skupe. [7]

Dinamični sadržaj nije moguće prezentovati direktno putem NginX servera, već je neophodno preusmeriti (proxy) sadržaj podsistemima koji se bave procesiranjem te vrste podataka. [7] Ipak, ova funkcija, preusmeravanje, u sebi sadrži brojne komponente i može se obavljati na različite načine zavisno od potreba korisnika. Segment administratorskih aktivnosti kojim ćemo se baviti u ovom radu obuhvata pre svega pitanja odabira adekvatnih mogućnosti NginX sistema i njihovo ispravno konfigurisanje. Cilj ovog rada je detaljna naučna deskripcija pojedinih komponenti i mogućnosti softverskog paketa NginX

Osim deskripcije komponenti i komparacije pojedinih delova ovog kompleksnog softverskog sitema sa nekim drugim rešenjima, na nekoliko primera ćemo praktično pokazati upotrebljivost različitih konfiguracionih opcija, kao i relativnu jednostavnost sa kojom se one mogu implementirati. U te svrhe ćemo koristiti naučne metode deskripciju, komparaciju i eksperiment.

1. Karakteristike NginX servera

Razvoj Nginx sistema počeo je 2002. godine, a njegov osnivač je ruski programer Igor Sisojev. Prva verzija je objavljena 2004. godine, a sam proizvod je predstavljen kao slobodan softver otvorenog koda koji vrši funkcije HTTP servera visokih performansi, obrnutog proxy servera, kao i proxy servera za IMAP i POP3 e-mail servise. Neke od osnovnih karakteristika po kojima je odmah postao poznat su stabilnost, dobre performanse, stabilnost, brojne opcije, jednostavna konfiguracija, kao i niska potrošnja resursa. [7]

NginX je veoma brz softver koji raspolaže svim mogućnostima da u potpunosti zameni Apache ili IIS, dok je sasvim pogodan i za upotrebu kao obrnuti proxy server. Od početka njegovog dizajna vođeno je računa o brzini i performansama, što ga čini izuzetno pogodnim za veoma zahtevne zadatke.

Uloga obrnutog proxy servera podrazumeva brzu isporuku statičnih podataka kao što su slike i HTML fajlovi, dok se za dinamične resurse, odnosno web stranice sa promenljivim sadržajem, upit prosleđuje drugom serveru na kome se izvršava aplikacija koja istu kreira. Na ovaj način se postiže značajno ubrzanje aplikacije. [4]

Nginx se veoma često upotrebljava paralelno sa nekim drugim serverom, što je veoma efikasan način poboljšanja performansi celokupnog sistema, s obzirom da je Nginx projektovan sa ovakvim načinom korišćenja u vidu. [7]

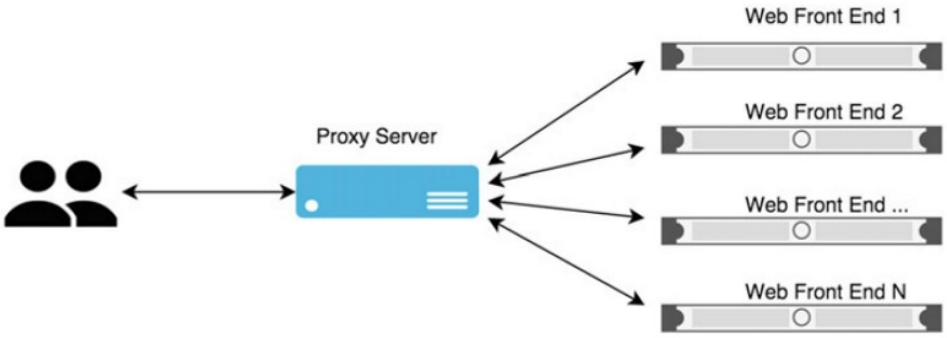
U knjizi *NginX From Beginner to Pro* navodi se sedam razloga zbog kojih je dobro odabrati Nginx:

1) Brzina je veoma značajna, jer osim što će preko 40% posetilaca napustiti sajt ukoliko se isti ne učita za manje od tri sekunde, ovaj faktor odnedavno utiče i na pozicioniranje sajta u google pretrazi.
2) Ubrzavanje rada same aplikacije može se postići i putem inteligentnog preusmeravanja statičnog i dinamičnog sadržaja na samom serveru.
3) Raspoređivanje opterećenja (load balancing) korišćenjem Nginx-a donosi većinu poboljšanja koja pruža hardverski uređaj ove svrhe, uz daleko manje ulaganje i bržu implementaciju.
4) Skaliranje koje Nginx vrši uz minimalne izmene konfiguracije pruža odgovor na tzv. "C10K" problem, odnosno omogućava ostvarivanje deset hiljada istovremenih konekcija, a zahvaljujući specifičnoj arhitekturi samog sistema.

5) Rekonfigurisanje bez prekida funkcionalnosti je veoma važno u produkcionim okruženjima gde korisnici moraju imati neometan pristup, dok istovremeno zahtevi aplikacije traže izmenu konfiguracije servera.
6) Minimalno opterećivanje računarskih resursa, koje je daleko ekonomičnije od Apache-a i IIS-a pri istom opterećenju.
7) Jednostavan korisnički interfejs, koji, iako bez grafičkog okruženja, omogućava lako učenje i snalaženje u brojnim opcijama. [7]

Glavne karakteristike Nginx-a

U suštini, može se reći da je Nginx obrnuti proxy server baziran na događajima (event based), što znači da je njegova funkcija da umesto klijenta pribavlja resurse sa web servera koji se nalaze iza njega. [7]



Slika 1 - Način rada proxy servera

Modularni dizajn omogućava izuzetno proširivu arhitekturu, a dodavanje novih funkcionalnosti se vrši putem priključaka (plugins). Ovakav pristup pomaže u kreiranju instance servera koja je prilagođena konkretnim potrebama sistema. Trenutno postoji preko 60 zvanično podržanih plugin-ova, uz mogućnost neograničenog proširivanja od strane open source zajednice. Sa druge strane, neophodnost kompajliranja modula odnosno samog servera donekle komplikuje ovaj postupak. [7]

Asinhroni način rada je najzaslužniji za odlične performanse Nginx servera. Sama softverska arhitektura na kojoj je zasnovan je bazirana na događajima (event-based), za razliku od Apache-a i IIS-a koji pokreću nove niti (threads) izvršavanja, koje su po prirodi blokirajuće. Minimalna potrošnja resursa postiže se modularnom arhitekturom, koja ipak podrazumeva neophodno kompajliranje modula sa željenom funkcionalnošću. [7]

Obrnuti proxy i balansiranje opterećenja (load balancing) su modovi rada mogući nakon inicijalne analize zahteva tj. URI-ja koji je pristigao na server. Ovakav način donošenja odluka o preusmeravanju zahteva je veoma efikasan, te se može reći da je upravo obrnuti proxy primarna funkcija Nginx servera, odnosno da se Nginx koristi kao front-end koji se oslanja na različite servise iza sebe (npr. Apache). [7]

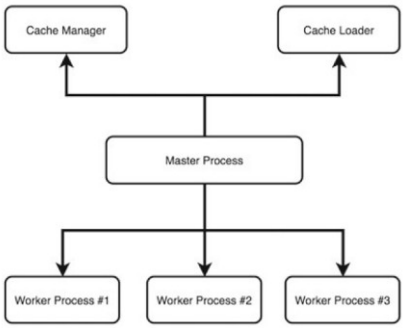
Neke takođe česte upotrebe Nginx-a su isporučivanje statičnih fajlova, keširanje, SSL terminacija, kompresija i sl. Osim standardnih web protokola (HTTP, HTTPS, WebSocket), postoji podrška i za e-mail protokole IMAP, POP3, SMTP kao i za TCP. [7]

Arhitektura NginX sistema

Master proces obavlja operacije za koje je neophodan visi nivo privilegija, kao npr. čitanje konfiguracionih fajlova, otvaranje portva i pokretanje radnih procesa.

Radni procesi se pokreću odmah po startovanju ili restartovanju master procesa, a njihov broj je moguće podesiti u kontrolnom fajlu. Preporučljivo je da se njihov broj poklapa sa brojem jezgara procesora sistema na kome je pokrenut master proces, a moguća je i autodetekcija optimalnog broja. Može se reci da je master proces neka vrsta menadžera delova sistema koji zapravo servisiraju zahteve klijenata. [7]

Procesi za upravljanje kešom, ukoliko je ta opcija uključena, vrše dva posebna procesa koji obavljaju učitavanje (cache loader) i upravljanje (cache manager). Loader se pokreće pri inicijalizaciji nginxa, učitava keš sa diska u memoriju i isključuje se radi uštede sistemskih resursa. Menadžer ostaje aktivan i zadužen je za čišćenje odnosno invalidaciju kao i za obnavljanje sadržaja keša. [7]



Slika 2 - Struktura NginX procesa

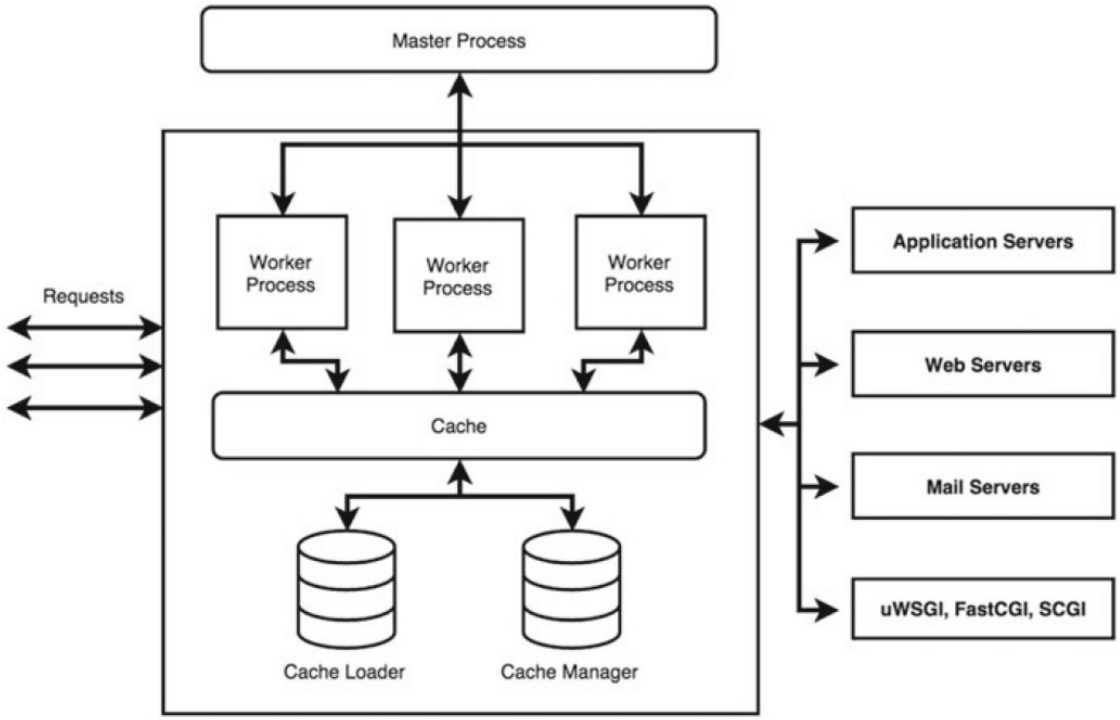
Da bi se bolje razumeo menadžment resursa koje sprovodi nginx, neophodno je napraviti kratak osvrt na unutrašnju arhitekturu os, odnosno način upravljanja memorijom.

Sa stanovišta operativnog sistema, rad koji obavlja procesor vrsi se u okviru procesa korišćenjem jedne ili vise niti (threads). Proces možemo posmatrati kao ograničen memorijski prostor u kome se nalaze niti, dok su niti objekti koji učitavaju instrukcije za čije izvršenje je zadužen procesor. Većina serverskih aplikacija nastoji da pokreće vise paralelnih procesa ili niti kako bi najefikasnije iskoristila sva jezgra procesora.

Softverski paketi kao sto su Apache i IIS koriste način rada sa vise niti (multithreading) pri obradi dolazećih konekcija, sto može da predstavlja problem kada ima vise hiljada istovremenih konekcija, pogotovo ako neke od njih uspostavljaju korisnici sa sporijim vezama. Do ovakvog blokiranja ne dolazi kada se koristi NginX. [7]

Radni proces

Radni proces kod NginX-a se pokreće u jednoj niti i radi nezavisno, a njegov zadatak je da prihvati konekciju i procesira je sto je brze moguće. On takođe čita i piše sadržaj na disk, komunicira sa nadređenim (upstream) serverima, a to postiže koristeći deljenu memoriju za pristup keširanim podacima, podacima o sesijama kao i drugim deljenim resursima. Inače, rad operativnog sistema sa thread-ovima i deljenom memorijom se razlikuje na Windowsu i Linuxu, te su konkretne implementacije adekvatno optimizovane.



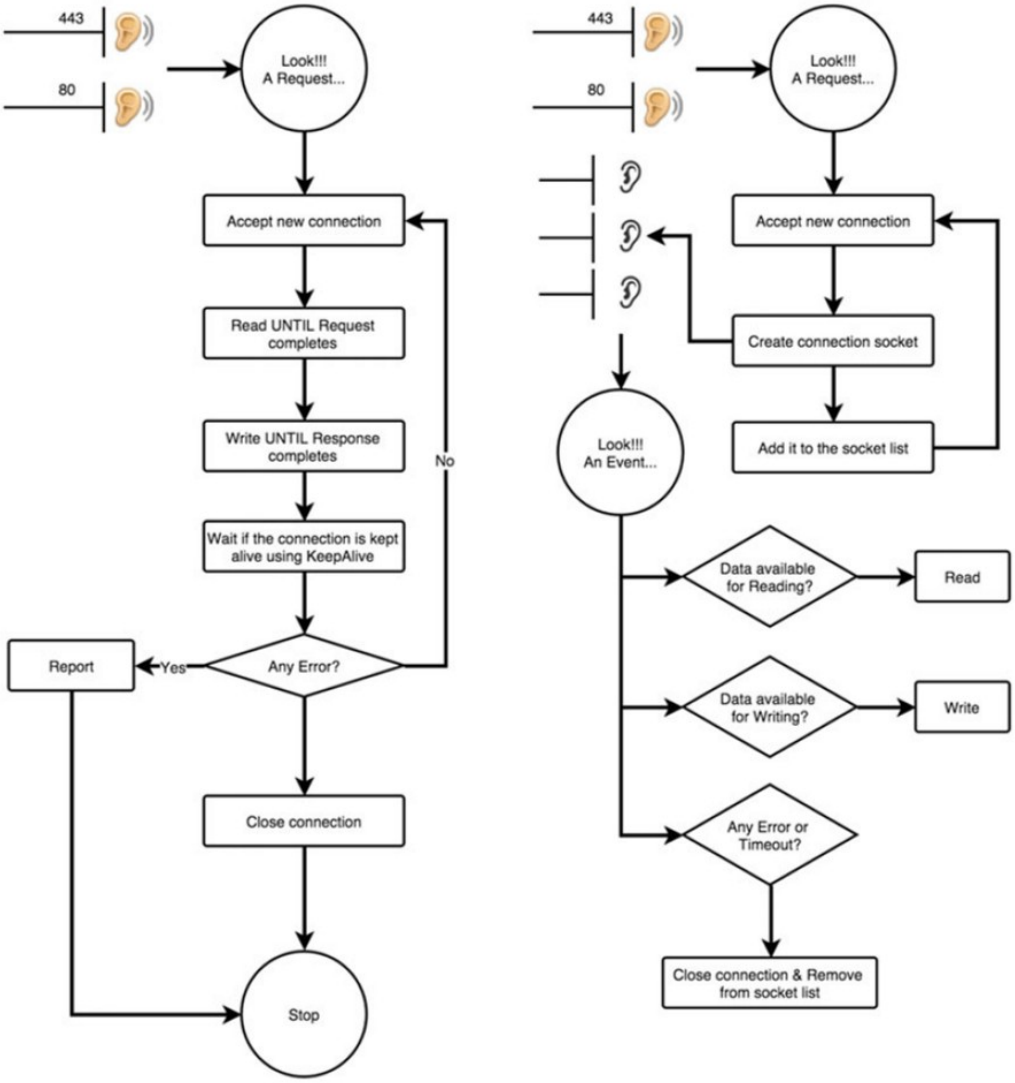
Slika 3 - Povezanost NginX procesa

Kada je uspostavljen efektivan način procesiranja podataka, koji ne dovodi do blokiranja pojedinih procesa, server je u mogućnosti da obradi znatno veći broj upita, a s obzirom da ne zavisi ni od brzine konekcije pojedinih klijenata, može se reći da je postignut asinhroni rad sa dolazećim zahtevima. [7]

Ukoliko ipak dođe do blokirajućeg poziva, npr. čitanja velike datoteke sa diska, usled rada na samo jednoj niti, nije moguće izbeći zauzeće postojeće sesije do kraja izvršenja zadatka. Za zaobilaženje ove pojave moguće je iskoristiti koncept objedinjavanja thrad-ova (thread pooling), a koji je dostupan u novijim verzijama NginX-a. Na taj način blokirani poziv nastavlja da čeka završetak zadatka, dok se umesto njega pokreće nova nit koja prihvata sledeće zahteve. Ipak, potreban je oprez, odnosno detaljna analiza konkretnog slučaja kada se javlja potreba za ovim zaobilaznim rešenjem, jer njegova preterana upotreba može biti loša po ukupne performanse sistema. [7]

Iako radi u jednoj niti, jedan radni proces može da opsluži i do više hiljada zahteva istovremeno, a što je moguće usled rada u petlji koja prati status događaja (event loop) koja je po svojoj prirodi neblokirajuća. Za razliku od Apache-a i IIS-a, NginX ne čeka da se pojedinačni zahtev u potpunosti završi, već odmah po prihvatu jedne, otvara slot (socket) za prijem sledeće konekcije. [7]

Novootvorene konekcije ne čekaju da se isporuče svi traženi podaci, već po potvrdi prijema paketa podataka, kernel obaveštava NginX da može poslati sledeći, dok u međuvremenu server može da radi na servisiranju drugih klijenata.



Slika 4 - Neblokirajuća arhitektura

Postoje značajne razlike u arhitekturi NginX-a i Apache-a, a osavremenjeni pristup ovog prvog se vidi u svim segmentima, od strukture konfiguracionih fajlova do internih mehanizama rada na sistemskom nivou. Osim olakšane upotrebe, uz minimalnu konfiguraciju u većini slučajeva je moguće dobiti značajno veće performanse zamenom postojećeg Apache sistema. [7]

Za kompletno unapređenje, odnosno migraciju na NginX sistem je često potrebno dosta planiranja i taj proces se nažalost ne može automatizovati. Kao srednje rešenje na velikim sistemima gde bi

promena celokupne konfiguracije bila preveliki posao, moguće je relativno lako dodati NginX u ulozi obrnutog proxy servera i značajno ubrzati postojeći sistem.

2. Instalacija i konfiguracija

Instalacija softverskog paketa NginX se može obaviti na više načina, zavisno od tipa i konfiguracije operativnog sistema, kao i konkretnih zahteva implementacije u datom slučaju. Najjednostavnija instalacija podrazumeva download binarne verzije aplikacije ili korišćenje paket menadžera Linux distribucije. Iako najbrži, ovaj pristup je najmanje fleksibilan, jer je korisnik ograničen predefinisanim verzijama i konfiguracijama dodataka.

Poželjan metod instalacije softverskog paketa NginX u produkcionom okruženju podrazumeva kompajliranje osnovnog programa i pratećih modula. Ovaj napredni način ima svoje prednosti u tome što je moguće napraviti optimalnu konfiguraciju neophodnih komponenti na najnižem nivou sistema, čime se rad softvera može značajno ubrzati a potrošnja resursa smanjiti.

Podešavanje pokretanja prilikom podizanja sistema se može izvršiti na više načina, a u zavisnosti od arhitekture samog sistema na kojem se server pokreće. Na Windows sistemima tipičan način je aktivacijom NginX-a kao servisa, što se postiže nekim od integrisanih ili spoljnih alata, dok je na većini Linux distribucija takođe poželjno podesiti neku od dostupnih skripti za pokretanje servisa.

Izmena konfiguracije se vrši jednostavnom komandom *nginx -s reload*, koja pre primene novih pravila proverava da li su ona u skladu sa konvencijom odnosno protokolom. Nakon toga, master proces pokreće nove radne procese sa novom konfiguracijom koji prihvataju nove konekcije, dok stari radni procesi ostaju aktivni dok se konekcije koje opslužuju ne okončaju. Iako ovakav način rada kratkoročno crpi dodatne resurse servera, istovremeno obezbeđuje neprekinutu uslugu korisnicima. [7]

Sličan metod se primenjuje i prilikom izmene verzije celog servera, naime pokreće se nova instanca servera koja deli resurse sa prethodnom pokrećući svoj set radnih procesa. Ukoliko smo zadovoljni radom nove verzije, možemo ugasiti staru, a usluga klijentima će takođe biti neprekinuta.

Da bismo kontrolisali glavni (master) proces NginX-a, neophodno je komandom nginx -s proslediti željeni signal. To može biti gašenje sa čekanjem na zatvaranje započetih konekcija (quit), ponovno

učitavanje konfiguracije (reload), ponovno otvaranje log fajlova (reopen) i gašenje sa prekidanjem konekcija (stop). [7]

Najznačajniji **konfiguracioni** fajl je nginx.conf iz koga se najčešće pozivaju drugi setovi direktiva raspoređeni u fajlove i direktorijume koji sačinjavaju smislenu celinu laku za navigaciju. Direktive su komande odnosno instrukcije kojima se konfiguriše server, a raspoređene su u više konteksta (main, stream, http, server, location…) od čega zavisi i redosled kojim će biti primenjivane. Mogu biti jednostavne ili blok direktive, gde ove druge čine setove jednostavnih okružene zagradama {}.

```
user nginx;
worker_processes 1;
error_log /var/log/nginx/error.log warn;
pid /var/run/nginx.pid;
events {
     worker_connections 1024;
}
http {
     include /etc/nginx/mime.types;
     default_type application/octet-stream;
     access_log /var/log/nginx/access.log main;
     sendfile on;
     #tcp_nopush on;
     keepalive_timeout 65;
     #gzip on;
     include /etc/nginx/conf.d/*.conf;
}
```

Prema definiciji, **moduli** su posebne ali međusobno povezane jedinice koje čine jedan program, odnosno u njih je moguće ugraditi određenu zasebnu aktivnost. Drugim rečima, modul je nezavisan deo koda koji ima sopstvenu funkcionalnost i zahteve, a zajedno sa ostalim delovima koda čini jednu celinu. [7]

NginX moduli se po funkcionalnosti mogu svrstati u pet kategorija, osnovni (core) i moduli za rad sa događajima (event) su uključeni u podrazumevanoj konfiguraciji, dok se HTTP, mail i stream moduli omogućavaju pri kompilaciji. Za pojedine je neophodno obezbediti dodatne biblioteke ukoliko već ne postoje na sistemu na kome se vrši kompajliranje (npr. OpenSSL). [7] Takođe postoje i eksterni

moduli, koje kreiraju pripadnici mnogobrojne NginX open source zajednice, a koji dodaju različite funkcionalnosti.

Za kompajliranje NginX-a neophodni su određeni preduslovi koji variraju zavisno od operativnog sistema i konkretne instalacije. U ovom radu ćemo se ograničiti na rad pod Linux sistemom baziranom na Debian distribuciji. Uz pretpostavku da je kompajliranje programa iz izvornog koda moguće na datoj mašini, odnosno da su već dostupni sistemski alati i uobičajeni set biblioteka, pažnju ćemo posvetiti dodatnim neophodnim preduslovima odnosno potrebnim bibliotekama.

PCRE je skraćenica od Perl Compatible Regular Expression, a ovaj set biblioteka služi za rad sa uobičajenim zamenskim izrazima (regular expressions) kojima se pretražuju određeni obrasci u znakovnim nizovima. Ova funkcionalnost je neophodna za rad sa URL-ovima. Instalira se sledećim komandama:

```
apt-get install libpcre3 libpcre3-dev
```

Biblioteka zlib služi za kompresiju podataka, kojom se poboljšava brzina transfera i bolje koristi raspoloživi propusni opseg konekcije. Instalira se komandom:

```
apt-get install zlib1g zlib1g-dev
```

OpenSSL se koristi pri uspostavljanju bezbednog komunikacionog kanala između klijenta i servera tehnologijama SSL (secure socket layer), odnosno TLS (transport layer security). Instalira se komandom:

```
apt-get install libssl-dev
```

Zavisno od željene funkcionalnosti, odnosno određenih modula, mogu biti potrebne dodatne biblioteke koje omogućavaju njihovo kompajliranje, a koje se instaliraju na sličan način.

Na primeru ćemo pokazati kompajliranje NginX-a sa modulom za čitanje imena servera (SNI, Server Name Indication) pri uspostavljanju SSL konekcije (SSL preread), kako bismo omogućili da server na istom portu (tcp 443) prima zahteve za HTTPS i OpenVPN, te korisnicima pruža obe usluge

istovremeno, prosleđujući konekcije adekvatnim nadređenim (upstream) serverima. (OpenVPN server je prethodno podešen i prima konekcije na tcp portu 5443).

Komandom nginx -V možemo videti aktuelnu konfiguraciju i prilagoditi je našim potrebama. Trudićemo se da ostvarimo minimalnu razliku u odnosu na tekuću konfiguraciju dodavanjem samo neophodnih novih funkcionalnosti, kao i pratećih biblioteka za odabrane opcije.

```
root@43131nginx-1.13.5$ nginx -V
nginx version: nginx/1.10.3 (Ubuntu)
built with OpenSSL 1.0.2g  1 Mar 2016
TLS SNI support enabled
configure arguments: --with-cc-opt='-g -O2 -fPIE -fstack-protector-
strong -Wformat -Werror=format-security -Wdate-time -
D_FORTIFY_SOURCE=2' --with-ld-opt='-Wl,-Bsymbolic-functions -fPIE -
pie -Wl,-z,relro -Wl,-z,now' --prefix=/usr/share/nginx --conf-
path=/etc/nginx/nginx.conf
--http-log-path=/var/log/nginx/access.log
--error-log-path=/var/log/nginx/error.log
--lock-path=/var/lock/nginx.lock --pid-path=/run/nginx.pid --http-
client-body-temp-path=/var/lib/nginx/body --http-fastcgi-temp-
path=/var/lib/nginx/fastcgi
--http-proxy-temp-path=/var/lib/nginx/proxy --http-scgi-temp-
path=/var/lib/nginx/scgi
--http-uwsgi-temp-path=/var/lib/nginx/uwsgi --with-debug --with-
pcre-jit --with-ipv6 --with-http_ssl_module --with-
http_stub_status_module --with-http_realip_module --with-
http_auth_request_module --with-http_addition_module --with-
http_dav_module --with-http_geoip_module --with-http_gunzip_module
--with-http_gzip_static_module --with-http_image_filter_module --
with-http_v2_module --with-http_sub_module --with-http_xslt_module
--with-stream --with-stream_ssl_module --with-mail --with-
mail_ssl_module --with-threads
```

Za potrebe našeg primera, dodaćemo opciju --with-stream_ssl_preread_module a ostale parametre prilagoditi raspoloživom kompajleru i ostalim sistemskim podešavanjima. Dodatne biblioteke ćemo instalirati komandom:

```
apt-get install libxml2-dev libxslt-dev libgd2-xpm-dev libgeoip-dev
```

Posle uobičajenih komandi *make* i *make install*, dobićemo novu verziju NginX-a koja će koristiti već postojeća podešavanja, odnosno lokacije konfiguracionih fajlova, što dodatno olakšava implementaciju novih mogućnosti u produkcionom okruženju.

3. Komparativna analiza sa alternativnim rešenjima (Node.js)

Node.js je softver zasnovan na platformi otvorenog koda, a koji izvršava javascript kod na serverskoj strani. Samo jezgro sistema je napisano u C/C++ jeziku i njegova funkcija je da kompajlira javascript kod pre izvršenja na mašinskom nivou. [13] Osim HTTP modula, o kome će biti više reči kasnije, Node je moguće proširiti brojnim modulima putem npm menadžera paketa. Asinhrona priroda javascript arhitekture pomaže da se nove funkcionalnosti, odnosno delovi sistema izvršavaju paralelno, te da celokupna aplikacija može biti istovremeno i veoma brza i sadržati brojne module.

NginX je pre svega web server, što znači da isporučuje statičke sadržaje, dok za dinamičke prosleđuje upite određenom interpreteru (npr. php-fpm) od koga, nakon obrade, prihvata rezultate koje šalje klijentu u za njega prihvatljivom formatu. Konkretna putanja na koju se upit prosleđuje, kao i eventualne parametre, NginX iščitava iz URL-a, ako je tako konfigurisan u adekvatnom bloku.

Node.js je značajno drugačiji po ovom pitanju, jer ga je moguće konfigurisati da obavlja sve neophodne poslove, od prijema upita, obrade parametara, komunikacije sa bazom, kreiranja stranice do njene isporuke korisniku. Node.js je u suštini programsko okruženje koje u sebi sadrži HTTP server, te može biti potpuno nezavisno tj. istovremeno vršiti uloge web i aplikacionog servera. [11]

Najčešći slučaj korišćenja ipak podrazumeva samostalnu aplikaciju kao web server, uglavnom NginX, koja vrši prijem i preusmeravanje (rutiranje) zahteva prema određenoj instanci Node.js-a na kojoj je pokrenuta tražena aplikacija. S obzirom da je celokupna HTTP funkcionalnost dodata kao jedan modul, nije realno očekivati isti broj opcija i fleksibilnost koju nude rešenja kao što su Apache i NginX. [12]

Postoji više razloga zašto bismo želeli da koristimo još jedan zaseban web server ispred Node.js-a. S obzirom da samo administrator (root) može otvoriti dolazne portove 80 i 443, a da za tu aktivnost postoje uhodane procedure u naprednim web serverima kao što je NginX, sa bezbednosnog

stanovišta je daleko bolje koristiti takva rešenja nego pokretati Node.js aplikaciju sa integrisanom serverskom komponentom pod administratorskim pravima.

Isporučivanje statičnih sadržaja kao što su slike je daleko jednostavnije i skoro uvek brže korišćenjem NginX-a, nego putem nekog Node.js dodatka. Takođe, rad sa keširanim podacima je daleko jednostavniji i pouzdaniji u NginX-u, [10] a izbegavanjem direktnog izlaganja Node.js aplikacije može se smanjiti mogućnost pojedinih napada koji su posledica bezbednosnih propusta u samom serverskom modulu aplikacije.

Oba alata su slična po tome što su zasnovani na asinhronoj arhitekturi baziranoj na događajima (event driven architecture), koja ne blokira protok procesiranja novih zahteva, čime je omogućena obrada daleko većeg broja zahteva nego što je to slučaj kod drugih web servera. Node.js koristi jezik javascript i Google-ov V8 Chrome engine, a nastao je po uzoru na NginX.

Dok je Node.js relativno lako prilagoditi ulozi web servera statičnog sadržaja, dosta teže bi bilo podesiti NginX da direktno procesira dinamičke zahteve. Tako nešto je delimično moguće korišćenjem LUA programskog jezika, ali je njegova upotrebna vrednost u ovakvom kontekstu veoma ograničena.

Node.js dolazi sa modulom HTTP koji je dovoljan za obavljanje velikog broja funkcija web servera. Ipak, svaku od dodatnih funkcionalnosti je neophodno posebno napisati u jvascript-u ili iskoristiti neki od dostupnih modula ili razvojnih okvira (framework) kao sto je npr. Express. Na ovaj način web server postaje integralni deo svake pojedinačne aplikacije. Takođe, moguće je proslediti određene upite drugim interpreterima (npr. php-fpm) čime se još više dobija na fleksibilnosti, dok kompleksnost aplikacije naravno raste.
Svi zadaci koji se u Node.js-u obavljaju vezano za isporučivanje sadržaja izvršavaju se u istoj glavnoj petlji (main loop) koja vrši i sve ostale proračune potrebne za rad glavne aplikacije. Dobra praksa preporučuje da se serverski deo svede na apsolutni minimum, odnosno da se procesorsko vreme Node.js aplikacije maksimalno iskoristi za rad na poslovnoj logici, a da se postavljanjem zaglavlja (headers), isporukom statičnih fajlova, evidentiranjem tipova fajlova i statusa itd. bavi NginX, odnosno druga serverska aplikacija.

Svi ovi zadaci vezani za isporuku sadržaja su razrađeni i optimizovani u NginX-u na takav način da troše najmanje resursa, izvršavaju se na drugom thread-u, i pružaju veliku fleksibilnost i brzinu u radu.

4. Funkcije i mogućnosti

NginX je zreo sistem, nastao na iskustvu mnogih drugih softverskih rešenja, te ga od samog starta prati zaslužena dobra reputacija solidnog, stabilnog i fleksibilnog programskog paketa. Brojne funkcije koje su neophodne u održavanju bezbednog i kontinuiranog pristupa različitim vrstama sajtova i web aplikacija su integrisane u sam sistem, dok postoji praktično neograničena mogućnost proširivanja funkcionalnosti putem modula. U ovom delu ćemo se osvrnuti samo na neke od ovih funkcionalnosti.

S obzirom da je aktuelna dobra praksa činiti sadržaj dostupnim isključivo putem HTTPS protokola, često se javlja potreba za rekonfiguracijom servera koji nije bio prilagođen takvom režimu rada. Takođe, preporučljivo je imati identične, odnosno samo jednu konfiguraciju server bloka po imenu (hostname) za oba protokola, odnosno HTTP (port 80) i HTTPS (443). Najjednostavniji način za postizanje konfiguracije koja će sav sadržaj isporučivati putem HTTPS protokola je stalnom redirekcijom sa neenkriptovanog virtuelnog hosta na enkriptovani. [3]

```
server {
     listen 80;
          server_name test.domen.com;
          rewrite ^/(.*)$ https://test.domen.com/$1 permanent;
}
```

SSL terminacija omogućava da se samo na jednom mestu vrše kriptografske operacije, čime se drastično umanjuje opterećenje na ostalim serverima. Takođe, zavisno od same strukture mreže i servisa koji se nude, ovakvom konfiguracijom je moguće izbeći potrebu za posedovanjem više pojedinačnih serverskih sertifikata, ili kupovinu dosta skupljeg domenskog sertifikata. Kako raste potreba za korišćenjem SSL-a, iz bezbednosnih i SEO razloga, ova funkcionalnost će dodatno doći do izražaja. [7]

Prenos video fajlova u realnom vremenu u formatima MP4/FLV/HDS/HLS je podržan i to na takav način da minimalno opterećuje server, bez obzira na brzinu klijenta odnosno širinu njegovog internet protoka.

Monitoring i beleženje (logging) problema u produkcionom okruženju je veoma važno, ali isto tako veoma komplikovano, pogotovo u arhitekturama koje sadrže veći broj servera pod konstantnim visokim opterećenjem, a koji su istovremeno izloženi napadima zlonamernih korisnika odnosno aplikacija. Logovanje pristupa i grešaka (access i error log) u Nginxu je veoma fleksibilno i može se podesiti kako bi se dobio optimalan balans između detelja i veličine log fajlova. [7] Postoje posebne aplikacije kao što je npr. ngxtop kojima je moguće pratiti logove u realnom vremenu, čime se ova funkcionalnost značajno proširuje.

Resetovanje aktivne instance Nginx servera je izvedeno na takav način da je moguće promeniti aktuelnu konfiguraciju bez gašenja i ponovnog podizanja samog procesa, odnosno bez prekidanja uspostavljenih korisničkih konekcija. Ovim putem je moguće isprobati izmene u konfiguraciji web servera bez značajnijih smetnji u radu aplikacije.

Na sličan način funkcioniše i **izmena sistema** korišćenjem Live Binaries opcije. Naime, Nginx pokreće jedan glavni (master) proces pri podizanju servisa, a njegov glavni zadatak je iščitavanje konfiguracionih fajlova. Osim toga, master proces pokreće jedan ili više radnih (worker) procesa koji prihvataju korisničke konekcije. Pri promeni konfiguracije, paralelno se pokreću novi worker procesi koji će prihvatiti sve nove konekcije, dok stari worker procesi ostaju aktivni sve dok konekcije koje oni servisiraju ne budu prekinute od strane korisnika.

Direktive kojima se uključuje **osnovna autentifikacija** možemo postaviti u bilo kom konfiguracionom bloku (http, server ili location). Takođe, neophodno je da lozinka bude prethodno šifrovana nekim od podržanih algoritama, što možemo učiniti komandom:

```
openssl passwd -apr1 test123
```

Rezultat komande u sledećem formatu unosimo u odgovarajući fajl, koji je prethodno naveden u direktivi:

```
test:$apr1$k.REbOZy$/.pxy5gudv2Zg2P5gHxad.:komentar
```

```
location /basic_auth/ {
     auth_basic "Molimo unesite korisničko ime i lozinku.";
     auth_basic_user_file /etc/nginx/auth.conf;
}
```

Balansiranje opterećenja i visoka dostupnost je nekad bila funkcija rezervisana samo za velike poslovne sisteme, ali je poslednjih godina postala podrazumevana opcija za najveći broj web aplikacija. Na administratorima sistema ostaje isključiva odgovornost da obezbede dostupnost sistema koristeći hardver kojim raspolažu, a kome to možda nije bila primarna namena.

Visoko dostupan je onaj sistem koji funkcioniše u kontinuitetu onoliko koliko je to potrebno. Konkretno određivanje perioda otkaza koji može da se toleriše zavisi od potrebe implementacije, a iskazuje se kroz ugovor (SLA - Service Level Agreement) koji mora da bude ispoštovan radi postizanja navedenog nivoa dostupnosti. [7]

Availability %	Downtime per year	Downtime per day
99% (two nines)	3.65 days	14.4 mins
99.9% (three nines)	8.76 hours	1.44 mins
99.99% (four nines)	52.56 mins	8.66 seconds
99.999% (five nines)	5.26 mins	864.3 milliseconds
99.9999% (six nines)	31.5 seconds	86.4 milliseconds

Slika 5 - Ugovor o dostupnosti

Postoje tri najvažnija aspekta pri dizajniranju visoko dostupnog sistema:

1) Eliminisati jedinstvenu tačku otkaza (single point of failure), odnosno potruditi se da ne postoji jedan element sistema čijim kvarom celokupan mehanizam prestaje da funkcioniše.
2) Pouzdana rezerva (failover) koja u slučaju otkaza nekog od servera preuzima njegove aktivnosti bez ometanja pružanja usluge korisniku.
3) Otkrivanje otkaza u trenutku kada se on dogodi, čime se omogućava blagovremena popravka.

Balansiranje opterećenja putem hardvera vrši se putem specijalizovanih uređaja koji se nalaze u računskim centrima, a njihov posao je deljenje i preusmeravanje dolaznog saobraćaja po

predefinisanim pravilima. S obzirom da se odluke o protoku donose na hardverskom nivou, to se događa veoma brzo i pouzdano. Ipak, ovi uređaju mogu biti vrlo skupi, a pogotovo je teško pronaći adekvatan odnos cene i zadovoljenja potreba koje mogu biti veoma promenljive u roku eksploatacije. Takođe, njihova adekvatna podešavanja često zahtevaju specijalizovana znanja koja prevazilaze nivo uobičajene mrežne administracije.

Softverski alati za balansiranje opterećenja, koji se ponekad nazivaju kontroleri isporuke aplikacije (Application Delivery Controllers - ADC), mogu imati veći broj namena osim osnovne, čime se povećava njihova fleksibilnost. Takođe, znatno lakše se mogu skalirati i finije podešavati. [7]

Algoritmi za balansiranje protoka igraju značajnu ulogu pri konfigurisanju softvera za balansiranje protoka, a posebno je važno da budu usklađeni sa arhitekturom celokupnog sistema. Tek nakon detaljne analize datog sistema i potreba korisnika moguće je ispravno podesiti sve parametre NginX-a, uključujući i algoritam prema kome će se vršiti balansiranje. [7]

Podrazumevani algoritam je Round Robin, odnosno ravnopravno raspoređivanje opterećenja "u krug". Iako možda deluje suviše jednostavno, ovaj mehanizam je veoma moćan i fleksibilan jer omogućuje dodelu specifičnog razmera (weight) prema kome će saobraćaj biti raspoređivan.

Ukoliko nije moguće unapred pretpostaviti koji će server biti više opterećen, može se koristiti algoritam koji automatski proračunava opterećenje i nastoji da ga održi ravnomernim (least Connected, Optionally Weighted). Ovo je inače najčešće korišćen algoritam jer se najbolje pokazao u praksi.

Ukoliko je neophodno očuvati informaciju o sesiji, odnosno obezbediti da se svaki sledeći zahtev istog korisnika upućuje na isti server, to se postiže algoritmom IP Hash. Ovo je slučaj sa većinom savremenih WEB aplikacija, pisanim u PHP, Node, ASP.NET ili sličnim tehnologijama.

Osim po IP adresi, informacija o sesiji se od strane NginX servera može čuvati po dodatnoj hash vrednosti nekog drugog parametra, korišćenjem Generic Hash algoritma. Ovaj parametar se konfiguriše na serveru i može predstavljati bilo koju vrednost iz svakog zahteva ili kombinaciju više njih. [7]

Bezbednost web sajta sačinjenog od isključivo statičnih komponenti je relativno lako postići, ali većina savremenih web aplikacija koristi veliki broj dinamičkih i interaktivnih komponenti te zahteva posebno planiranje i implementaciju bezbednosne komponente sistema. Autentifikacija i autorizacija korisnika sa jedne, uz neporecivu potvrdu identiteta samog sajta korisniku sa druge strane, praćenu bezbednim međusobnim prenosom podataka, značajno komplikuju postavljanje i održavanje web servera. [7]

Opšteprihvaćena tehnologija zaštite web aplikacija kojom se odgovara na najveći broj postavljenih izazova je SSL enkripcija (Secure Socket Layer), odnosno njena implementacija u vidu HTTPS protokola. Ovaj protokol je samo još jedan sloj dodat na već postojeći HTTP protokol, ali se njime, upotrebom kriptografije bazirane na algoritmu javnog i tajnog ključa, obezbeđuje privatnost i integritet komunikacije, kao i potvrda identiteta servera.

S obzirom da operacije enkripcije i dekripcije sadržaja troše puno procesorskih resursa, potrebno je posvetiti posebnu pažnju podešavanjima koja optimizuju ovaj proces. Najznačajnija je opcija održavanja konekcije (keepalive) koja sprečava da se najzahtevnija operacija otpočinjanja SSL komunikacije (handshake) izvršava po svakom zahtevu.

Slično, SSL sesije se čuvaju u keš memoriji koja se deli između radnih procesa, te je uglavnom preporučljivo povećati njenu vrednost odgovarajućom opcijom (ssl_session) u http bloku konfiguracionog fajla NginX-a. Ipak, konkretni parametri ovakvih optimizacija se mogu odrediti samo u konkretnim uslovima eksploatacije, ali ih ipak pominjemo da bismo ukazali na njihovu važnost. [7]

Iako je uobičajena praksa da same aplikacije vode računa o autentifikaciji i autorizaciji korisnika, odnosno ograničavanju pristupa pojedinim delovima aplikacije, u pojedinim scenarijima se mogu koristiti i mogućnosti samog web servera.

5. Napredne i dodatne funkcije (Nginx+)

Verzija NginX softvera koja se plaća, Nginx Plus ima veći broj funkcija koje mogu biti od koristi pri radu sa veoma posećenim sajtovima. Ovo dodatno ulaganje može biti isplativo, jer opcije kao što su balansiranje opterećenja (load balancing), održavanje sesija (session persistence), kontrola keša (cache control) i provera odziva servera (health checks), a koje su dostupne samo u plaćenoj verziji,

rade uz minimalan napor administratora a zamenjuju veći broj posebnih aplikacija koje se koriste u te svrhe. Takođe je bitno napomenuti da uz plaćeni paket postaje dostupna i profesionalna tehnička podrška. [7] Još jedna značajna razlika je to što Plus verzija dolazi u binarnom obliku, te nije neophodno proširivati joj funkcionalnost naknadno kompajliranim modulima.

Napredno balansiranje opterećenja je dodatno poboljšano u odnosu na besplatnu verziju, pa sada pored osnovna četiri metoda (Round-Robin, Least Connections, Generic Hash i IP Hash) postoji i peti, metod najkraćeg vremenskog intervala (least time method), kao i dodatna optimizacija u slučaju višejezgarnih sistema, koja omogućava dinamičko raspoređivanje opterećenja direktnom komunikacijom između radnih procesa. Nginx PLUS ima dodatni algoritam (Least Time Optionally Weighted) kojim server za balansiranje opterećenja kombinuje dve metrike nadređenih servera: broj aktivnih konekcija i prosečno vreme za koje su bili poslati odgovori na prethodne upite, te proračunava koji server je najmanje opterećen i njemu prosleđuje zahtev. Ovaj način procene je daleko efikasniji jer koristi izvedenu povratnu informaciju o opterećenju servera.

Očuvanje uspostavljenih sesija (session persistence) je neophodno radi održanja određenih funkcionalnosti aplikacije, kao i zadržavanja zadovoljavajućeg nivoa iskustva korisnika. NginX Plus može da identifikuje upite ka određenom serveru (upstream) i da ih uvek njemu i prosleđuje, čime se zadržavaju i funkcionalnost aplikacije i postiže balansiranje resursa. Takođe je moguće određeni upstream server isključiti tako da se on u potpunosti deaktivira tek kada se okončaju sve veze korisnika sa njim (session draining). [7]

Napredno keširanje omogućava prosleđivanje statičkih resursa korisniku bez komunikacije sa upstream serverima, čime se značajno ubrzava odziv i smanjuje opterećenje servera. Invalidacija celog ili delova keša (purging of cache) se ipak mora obaviti posebnim alatom koji će obavestiti NginX o neophodnosti navedene operacije.

Jedna od najznačajnijih razlika besplatne i plaćene verzije NginX-a je opcija za **proveru stanja** nadređenih (upstream) servera (health_check). Automatska provera upstream servera (application health checks) utvrđuje njihovu dostupnost, te uklanja nedostupne sa liste kako bi se izbegao slučaj da korisnici vide poruke o grešci. Moguće je podesiti interval provere, određeni URL koji će biti testiran, broj neuspelih pokušaja nakon kojih će se server smatrati neispravnim, kao i broj naknadno uspelih pokušaja koji će mu vratiti status ispravnog.

Nginx PLUS poseduje mogućnost **praćenja performansi** i opterećenja sistema u realnom vremenu. Ove podatke čini dostupnim putem JSON interfejsa pa se relativno lako mogu obrađivati različitim alatima.

NginX plus nudi mogućnost nadogradnje plaćenim modulom WAF (**Web Application Firewall**), koji pruža zaštitu od sofisticiranih pretnji na aplikacionom sloju (layer 7). Baziran je na poznatom i proverenom sistemu ModSecurity koji se duže vreme koristi na Apache platformi. Neki od napada koje može da zaustavi pre nego što stignu do same aplikacije su SQL injection, cross-site scripting (XSS), Cross-site request forgery (CSRF), DDoS itd., a takođe ga je moguće podesiti da vodi računa o integritetu HTTP konekcije i reputaciji korisničkih IP adresa. [14] Naravno, poseduje i mogućnost beleženja ovih aktivnosti za kasniju analizu.

Radi postizanja višeg stepena **automatizacije** NginX Plus raspolaže API interfejsom kome se vrlo lako može pristupiti putem HTTP-a i u toku rada izmeniti konfiguracija upstream servera, a postoje i drugi metodi kojima se isto može postići. Takođe, dodavanje ili uklanjanje pojedinih servera moguće je dodatno ubrzati korišćenjem skripte. Promene izvršene ovim putem se odmah manifestuju, bez potrebe za izdavanjem dodatnih komandi, a prava pristupa se definišu u posebnom bloku konfiguracijskih direktiva.

6. Primeri konfiguracije

6.1. Primer 1 – Balansiranje opterećenja

Aplikacija koju ćemo testirati u različitim NginX konfiguracijama se sastoji iz tri povezana dela. MySql baze podataka, php aplikacije bazirane na Symfony 3 razvojnom okviru (framework) koja puža mogućnost manipulacije podacima u bazi preko REST API interfejsa i frontend dela koji putem javascript (jQuery) upita komunicira sa API-jem. Radi jednostavnosti, u primeru konfiguracije sa više servera, odnosno instanci aplikacije, koristićemo samo jednu bazu podataka i njoj prepustiti odgovornost sinhronizacije podataka.

Sama aplikacija ima osnovne CRUD funkcionalnosti za rad korisnika sa člancima i komentarima, odnosno omogućava registrovanim korisnicima da pišu članke i komentare, a njima i svim ostalim posetiocima sajta da ih čitaju. Dužina tekstova je ograničena na 255 karaktera, a frontend deo je

prilagođen svim veličinama ekrana i većini programa za pregledanje internet sadržaja. Da bi se posetilac registrovao i postao korisnik, neophodno je da postavi e-mail adresu i lozinku.

Razvojno okruženje za PHP programski jezik Symfony, koje smo koristili za razvoj test aplikacije raspolaže brojnim naprednim mogućnostima i može se prilagoditi velikom broju potreba odnosno poslovnih aplikacija. Neke posebne funkcionalnosti dolaze u vidu modula tj. kompleta biblioteka koje se lako integrišu u samo okruženje. S obzirom da je backend deo aplikacije projektovan kao API interfejs, iskorišćene su pojedine od ovih biblioteka kako bi se olakšao rad i povećala preglednost koda. (Izvorni kod aplikacije je dostupan na adresi https://github.com/batica81/izb02.git)

Da bi NginX ispravno isporučivao Symfony aplikaciju, neophodno je uneti specifična podešavanja u serverski kontekst konfiguracije:

```
server {
...

    location / {
        try_files $uri /app.php$is_args$args;
    }

    location ~ ^/(app_dev|config)\.php(/|$) {
        fastcgi_pass unix:/run/php/php7.1-fpm.sock;
        fastcgi_split_path_info ^(.+\.php)(/.*)$;
        include fastcgi_params;
                                    fastcgi_param    SCRIPT_FILENAME
$realpath_root$fastcgi_script_name;
        fastcgi_param DOCUMENT_ROOT $realpath_root;
    }

    location ~ ^/app\.php(/|$) {
        fastcgi_pass unix:/run/php/php7.1-fpm.sock;
        fastcgi_split_path_info ^(.+\.php)(/.*)$;
        include fastcgi_params;
                                    fastcgi_param    SCRIPT_FILENAME
$realpath_root$fastcgi_script_name;
        fastcgi_param DOCUMENT_ROOT $realpath_root;
        internal;
    }
```

```
    location ~ \.php$ {
        return 404;
    }
}
```

Na primeru ćemo pokazati jednu od najčešće korišćenih naprednih funkcija NginX-a, raspoređivanje opterećenja (load balancing). Aplikacija je instalirana kao podrazumevana putanja na svim upstream serverima, a imena servera su radi jednostavnosti uneta u hosts fajlove.

```
upstream myapp01 {
     #ip_hash;
     server izb02.chip.dev;
     server izb02.rpi.dev;
     server 127.0.0.1;
}

server {
     listen 80;
     server_name balancer2.dev;

     location / {
          proxy_pass http://myapp01;
          #proxy_set_header Host $host;
          #proxy_set_header X-Forwarded-For $remote_addr;
          #add_header X-Upstream $upstream_addr;
     }
}
```

Inicijalno testiranje je izvršeno na relativno slabom hardveru (laptop, RaspberryPi i Chip) i u nestabilnim mrežnim uslovima (kućna bežična mreža). Laptop je konfigurisan kao load balancer i istovremeno jedan od nodova, dok su druga dva uređaja bili samo instance aplikaije. Za testiranje je korišćen program Apache benchmark koji je sa četvrte, desktop mašine, vršio upite ka početnoj strani aplikacije (ovaj upit ne pravi konekciju ka bazi, već dolazi samo do php procesiranja).

Čak i u ovako ograničenim uslovima više je nego očigledna prednost konfiguracije sa više servera. U prvom slučaju vršeni su upiti direktno na instancu aplikacije na laptopu:

```
C:\>ab -n 100 -c 10 http://192.168.1.105/
This is ApacheBench, Version 2.3 <$Revision: 1706008 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.1.105 (be patient).....done


Server Software:        nginx/1.10.1
Server Hostname:        192.168.1.105
Server Port:            80

Document Path:          /
Document Length:        3421 bytes

Concurrency Level:      10
Time taken for tests:   7.305 seconds
Complete requests:      100
Failed requests:        0
Total transferred:      363900 bytes
HTML transferred:       342100 bytes
Requests per second:    13.69 [#/sec] (mean)
Time per request:       730.542 [ms] (mean)
Time per request:       73.054 [ms] (mean, across all concurrent
requests)
Transfer rate:          48.64 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    2   1.3      1       8
Processing:   189  707 130.2    723     951
Waiting:      189  706 130.4    722     949
Total:        190  709 130.1    724     952

Percentage of the requests served within a certain time (ms)
  50%    724
  66%    766
  75%    791
```

```
  80%    798
  90%    819
  95%    863
  98%    899
  99%    952
 100%    952 (longest request)
```

Dok je u drugom slučaju ispitivana adresa koja, takođe na laptopu, upućuje na blok sa direktivama za balansiranje protoka:

```
C:\>ab -n 100 -c 10 http://balancer2.dev/
This is ApacheBench, Version 2.3 <$Revision: 1706008 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd,
http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking balancer2.dev (be patient).....done


Server Software:        nginx/1.10.1
Server Hostname:        balancer2.dev
Server Port:            80

Document Path:          /
Document Length:        3421 bytes

Concurrency Level:      10
Time taken for tests:   4.810 seconds
Complete requests:      100
Failed requests:        0
Total transferred:      363900 bytes
HTML transferred:       342100 bytes
Requests per second:    20.79 [#/sec] (mean)
Time per request:       481.027 [ms] (mean)
Time per request:         48.103 [ms] (mean, across all concurrent
requests)
Transfer rate:          73.88 [Kbytes/sec] received

Connection Times (ms)
```

```
              min  mean[+/-sd] median   max
Connect:        0    2   2.2      1      16
Processing:    60  438 481.3    165    2001
Waiting:       60  437 481.1    164    2000
Total:         61  440 481.1    166    2002

Percentage of the requests served within a certain time (ms)
  50%    166
  66%    504
  75%    924
  80%    975
  90%   1212
  95%   1333
  98%   1480
  99%   2002
 100%   2002 (longest request)
```

Možemo zaključiti da je došlo do ubrzanja od 50% do 150% zavisno koji parametar posmatramo, uz napomenu da ova metodologija ipak služi samo kao ilustracija i ne može zameniti laboratorijske uslove testiranja.

Takođe napominjemo da je kod prosleđivanja upita nekom od nadređenih (upstream) servera, u realnom scenariju, najčešće potrebno poslati većinu informacija koje su stigle u zaglavlju (header) upita, a što nije podrazumevano ponašanje NginX-a. Naime, NginX će obrisati prazna polja u zaglavlju, ignorisaće polja koja sadrže donju crtu (_), izmeniće IP adresu dolazećeg upita i automatski dodati zaglavlje o zatvaranju konekcije (connection header close). Da bi se održalo željeno ponašanje aplikacije, najčešće je neophodno izmeniti ova podešavanja NginX-a određenim adekvatnim direktivama. [7]

```
location / {
     proxy_set_header HOST $proxy_host;
     proxy_set_header X-Real-IP $remote_addr;
     proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
     proxy_pass http://backend;
     proxy_set_header HOST $host;
     underscores_in_headers on ;
     proxy_next_upstream     error timeout invalid_header http_500;
     proxy_connect_timeout   2;
```

```
}
```

6.2. Primer 2 – Iščitavanje SSL zaglavlja

Pošto smo prethodno iskompajlirali verziju NginX-a koja podržava SNI stream pre-read opciju, možemo je isprobati u praksi. U ovom scenariju imamo podešen NginX server sa jednom ip adresom i tri različita domena (cinamontest.duckdns.org, outerspace.duckdns.org i cinnamonvpn.duckdns.org), a osim test aplikacije izb02, podešen je OpenVPN server, firewall, SSL sertifikati i druge sistemske aplikacije.

Funkcionalnost koju želimo da postignemo se ogleda u sledećem: ka Internetu su otvoreni samo portovi 80 i 443 a potrebno je imati dva različita web sajta (različiti domeni) kojima se pristupa isključivo putem HTTPS-a kao i OpenVPN kome se takođe pristupa putem porta 443.

Jedan od načina na koji je ovo moguće postići je konfigurisati NginX tako da pre prosleđivanja konekcije nekom od upstream servera utvrdi da li se radi o HTTPS saobraćaju ili o drugoj vrsti TCP konekcije. Upravo iz ovog drugog razloga je neophodno korišćenje dodatnog modula, tj. izmeštanje čitanja informacije iz početne TLS konekcije u stream blok (u podrazumevanoj konfiguraciji, to se dešava u HTTP bloku).

Shodno navedenom, u konkretnoj konfiguraciji ćemo imati sledeći blok direktiva koji će nam omogućiti željeno ponašanje NginX-a.

```
stream {

    map $ssl_preread_server_name $name {
        #cinnamonvpn.duckdns.org vpn1_backend;

        cinamontest.duckdns.org https1_backend;
        outerspace.duckdns.org https2_backend;
        default stream_default_backend;
    }

    upstream https1_backend {
        server 127.0.0.1:6443;
    }
```

```
    upstream https2_backend {
        server 127.0.0.1:7443;
    }

    upstream stream_default_backend {
        server 127.0.0.1:5443;
    }

    server {
        listen 443 so_keepalive=on;
          proxy_connect_timeout 300s;
          proxy_timeout 300s;
          proxy_pass $name;
          ssl_preread on;
    }
}
```

6.3. Primer 3 - GeoIp

NginX i NginX Plus mogu da razlikuju korisnike na osnovu njihove geografske lokacije, što se može iskoristiti za pružanje različitog sadržaja, preusmeravanje upita na fizički bliže servere ili blokiranje pristupa. Da bi se postigla ova funkcionalnost, koristi se spoljna biblioteka, odnosno baza podataka MaxMind koja sadrži ažurne informacije o geografskoj lokaciji IP adresa, a sam modul se dodaje pri kompajliranju NginX-a.

Nakon instalacije svih neophodnih biblioteka, dodaćemo odgovarajuće parametre, odnosno napraviti varijable koje ćemo kasnije koristiti u konfiguracionim blokovima.

```
apt-get install geoip-database libgeoip1
```

```
http {
    geoip_country usr/share/GeoIP/GeoIP.dat;
    geoip_city    usr/share/GeoIP/GeoIPCity.dat;
}
```

```
Server {
        map $geoip_country_code $allow_visit {
           default yes;
           US no;
        }

     location /country_block/ {
          if ($allow_visit = no) {
               return 403;
          }
     }
}
```

Zaključak

U ovom radu prikazali smo neke od osnovnih karakteristika NginX sistema i relativnu jednostavnost njihove konfiguracije i upotrebe. Iako je projektovanje mrežne infrastrukture dugotrajan, temeljan i dobro promišljen proces, često ažuriranje pa i izmene pojedinih elemenata osnovne strukture sistema su administratorska svakodnevica. Upravo je fleksibilnost u toj oblasti jedna od najznačajnijih karakteristika NginX-a i može se reći, njegova osnovna komparativna prednost.

NginX web server ima veliki značaj za internet zajednicu u celini. Njegova široka upotreba učinila je administratorske poslove lakšim, resurse bolje iskorišćenim a Internet bezbednijim i udobnijim. Specifična kombinacija oslanjanja na entuzijazam i znanje open source zajednice, koji je dopunjen zdravim poslovnim modelom Nginx korporacije, učinila je da se sistem kontinuirano razvija prateći potrebe prakse, a istovremeno izbegavajući uobičajene probleme isključivo besplatnih odnosno potpuno komercijalnih rešenja.

Nginx korporacija svojim novim proizvodima *Unit*, *Amplify* i *Controller* već pokazuje spremnost za osvajanje drugih delova tržišta, pružajući rešenja za isporuku distribuiranih aplikacija, praćenje stanja infrastrukture u realnom vremenu i centralizovano upravljanje instancama NginX servera. Budući razvoj internet tehnologija će pokazati neke nove trendove i stvoriti nove potrebe na koje će softverske kompanije morati da pronađu adekvatne odgovore. Sa velikom sigurnošću možemo reći da će NginX nastaviti da se razvija i preuzima sve veći udeo tržišta uz nesmanjen kvalitet i performanse.

Literatura

[1] Aivaliotis, D., *Mastering Nginx*, Packt Publishing, 2013.

[2] Dar, U., *Nginx Module Extension*, Packt Publishing, 2013.

[3] Kholodkov, V., *Nginx Essentials*, Packt Publishing, 2015.

[4] Ling, A., Apache, *PHP-FPM & Nginx Reverse-Proxy*, Adrian Ling, 2015.

[5] Nedelcu, C., *Nginx HTTP Server*, 3rd Ed., Packt Publishing, 2015.

[6] Sarkar, D., *Nginx 1 Web Server Implementation Cookbook*, Packt Publishing, 2011.

[7] Soni, R., *Nginx: From Beginner to Pro*, Apress, 2016.

[8] Sharma, R., *Nginx High Performance*, Packt Publishing, 2015.

[9] DeJonghe, D., *Complete NGINX Cookbook*, O'Reilly Media Inc., 2017.

[10] https://stackoverflow.com/, septembar 2017. godine

[11] https://iwf1.com/apache-vs-nginx-vs-node-js-and-what-it-means-about-the-performance-of-wordpress-vs-ghost/, septembar 2017. godine

[12] Powers, S., *Learning Node*, 2nd ed., O'Reilly Media Inc., 2016.

[13] Krause, J., *Programming Web Applications with Node, Express and Pug*, Apress, 2017.

[14] https://www.nginx.com, septembar 2017.