Implementacija i poređenje reverznih proksija i metoda za raspodelu opterećenja

Bogdan Bakarec, Nemanja Radosavljević[1]

***Sadržaj* — Ovaj rad istražuje različite aspekte rivers proksija i loud balansera, uključujući njihove osnovne funkcije, podele po nivoima rada (L4 i L7), kao i pregled postojećih rešenja. Dodatno, rad testira i poredi implementaciju sopstvenih rivers proksi i loud balansing rešenja na L4 i L7 nivou sa postojećim tehnologijama na osnovu metrika performansi, skalabilnosti i pouzdanosti.**

***Ključne reči* — arhitektura softvera, bezbednost softvera, dizajn softvera, loud balanser, pouzdanost softvera, rivers proksi, skalabilnost računarskih sistema**

I. UVOD

SAVREMENI digitalni svet zahteva visoko dostupne i skalabilne sisteme koji mogu da podnesu veliki broj korisničkih zahteva. U tom kontekstu, rivers proksiji i loud balanseri igraju ključnu ulogu kao kritične komponente infrastrukture za obradu saobraćaja, optimizaciju performansi i obezbeđivanje pouzdanosti [1]. Njihova primena je posebno značajna u distribuiranim sistemima, mikroservisnim arhitekturama i klaud okruženjima gde su stabilnost i fleksibilnost ključni za uspeh.

Ovaj rad se fokusira na detaljno istraživanje koncepata rivers proksija i loud balansera, obuhvatajući njihovu suštinsku ulogu u optimizaciji saobraćaja, klasifikaciju po nivoima rada (L4 i L7), kao i sveobuhvatnu analizu postojećih tehnologija i rešenja u ovoj oblasti. U radu će se detaljno analizirati kako ove tehnologije doprinose skalabilnosti i performansama savremenih sistema, uz poređenje popularnih open-source rešenja poput NGINX, HAProxy, Traefik i

[1]Bogdan Bakarec Računarski fakultet, Srbija (email: bbakarec120m@raf.rs).

Nemanja Radosavljević, Računarski fakultet, Srbija (email: nradosavljevic@raf.rs)

Envoy, kao i komercijalnih rešenja u oblaku poput AWS Elastic Load Balancing, Google Cloud Load Balancer i Microsoft Azure Load Balancer.
Dodatno, uz ovaj rad su razvijena i prilagođena rivers proksi i loud balansing rešenja na L4 i L7 nivou (dostupna na https://github.com/Baksonator/reverse-proxy). Razvijena rešenja biće podvrgnuta testiranju i upoređivanju sa postojećim tehnologijama, koristeći metrike kao što su performanse, skalabilnost i pouzdanost. Osnovni cilj je da se analiziraju prednosti i ograničenja različitih pristupa, te da se predlože optimalni scenariji primene za svako rešenje.
Na kraju, ovaj rad ima za cilj da pruži sveobuhvatan pregled i analizu trenutnog stanja u oblasti rivers proksija i loud balansera, sa posebnim osvrtom na njihovu ulogu u savremenim velikim sistemima. Ovim se otvaraju mogućnosti za dalje istraživanje i unapređenje ovih tehnologija u skladu sa novim izazovima digitalnog doba.

II. RIVERS PROKSIJI I LOUD BALANSERI

A. Šta su rivers proksiji i loud balanseri

Rivers proksi i loud balanser su ključne komponente u modernim distribuiranim sistemima za obezbeđivanje visoke dostupnosti i optimalnog razpoređivanja resursa. Rivers proksi je posrednik između klijenta i servera koji preusmerava zahteve klijenta prema pravom serveru na osnovu unapred definisanih pravila. Ovo rešenje skriva arhitekturu serverske infrastrukture od korisnika, čime se poboljšava bezbednost i omogućava optimalno korišćenje resursa.
Osnovna uloga rivers proksija je da obrađuje ulazne zahteve i distribuira ih na jedan ili više bekend servera. Pored toga, rivers proksi može da kešira često tražene resurse, smanjujući opterećenje na bekend serverima ubrzavajući vreme odgovora. Ovo ga čini idealnim za primenu u velikim sistemima koji zahtevaju visoku dostupnost.
Sa druge strane, loud balanseri su specijalizovani uređaji ili softverska rešenja koja raspodeljuju saobraćaj ravnomerno između više servera. Njihova primarna svrha je da osiguraju da nijedan server nije preopterećen, čime se unapređuje ukupna stabilnost i performanse sistema. Loud balanseri često koriste različite algoritme za raspodelu opterećenja, kao što su round-robin, least connections, ili hash-based metode. U ovom radu se fokusiramo na sofverske implementacije loud balansera.

B. Osnovne funkcije i role u savremenim sistemima

U arhitekturama modernih sistema često je prisutan koncept velikog broja

servera "radnika" koji svi obavljaju isti posao. Jedan primer takvog sistema jeste Polaris [2]. U takvim ali i sličnim sistemima, često je potrebno implementirati horizontalno skaliranje. Horizontalno skaliranje podrazumeva dodavanje više servera u sistem radi obrade većeg broja zahteva ili upravljanja većim opterećenjem. Ovo je ključno za sisteme sa visokim zahtevima za performansama, jer omogućava povećanje kapaciteta bez promena u pojedinačnim serverima i bez prekida rada sistema, kao i za sisteme sa velikim brojem korisnika i količinom saobraćaja. U takvoj arhitekturi, rivers proksi i loud balanser igraju kritičnu ulogu i znatno olakšavaju život klijentima, budući da je klijentima u ovoj postavci broj servera potpuno nebitan i apstrahovan. Rivers proksi i loud balanser deluju kao posrednici između klijenata i servera ili između samih servera u nekim situacijama. Kombinacija ove dve komponente omogućava efikasno i pouzdano horizontalno skaliranje modernih distribuiranih sistema.

Povećanje pouzdanosti u računarskim sistemima je ključno za osiguranje neprekidnog rada i dostupnosti usluga, naročito u okruženjima sa visokim zahtevima za performansama. Rivers proksi deluje kao posrednik koji omogućava efikasnu obradu zahteva i optimizaciju saobraćaja, dok loud balanser obezbeđuje ravnomernu distribuciju saobraćaja između više servera. Poseban značaj imaju mehanizmi za detekciju otkaza i health-check sistemi, koje loud balanser aktivno koristi da prati status svih servera u klasteru. Ako server postane nedostupan ili ne odgovara na vreme, loud balanser ga uklanja iz rotacije i preusmerava saobraćaj na ispravne servere. Ovi mehanizmi osiguravaju da sistem ostane funkcionalan i otporan na otkaze, čime se poboljšava pouzdanost i održava kvalitet usluge čak i u slučaju nepredviđenih problema.

Bezbednosna zaštita u računarskim sistemima je od suštinskog značaja za očuvanje poverljivosti, integriteta i dostupnosti podataka i usluga. Rivers proksi deluje kao prva linija odbrane, sakrivajući stvarne IP adrese i strukturu servera iza sebe. On omogućava filtriranje saobraćaja, sprečavanje DDoS napada i primenu SSL/TLS enkripcije, čime se obezbeđuje sigurna komunikacija između klijenata i servera. Loud balanser doprinosi bezbednosti kroz integraciju sa mehanizmima za otkrivanje zlonamernog saobraćaja, ograničavanje brzine zahteva i izolaciju servera koji mogu biti kompromitovani [3]. Zajedno, ove komponente obezbeđuju viši nivo zaštite od uobičajenih pretnji, čime postaju nezamenljive u izgradnji sigurnih i pouzdanih aplikacija.

Optimizacija performansi u računarskim sistemima je ključna za obezbeđivanje brzog i efikasnog odgovora na korisničke zahteve, posebno u sistemima sa velikim opterećenjem. Rivers proksi pomaže u smanjenju latencije keširanjem odgovora, kompresijom podataka i smanjenjem broja

zahteva koji stižu do servera. Takođe, omogućava spajanje više zahteva u jedan, čime se smanjuje opterećenje bekend sistema. Loud balanser obezbeđuje ravnomernu raspodelu saobraćaja na više servera, sprečavajući preopterećenje i poboljšavajući iskorišćenje resursa. Pored toga, loud balanser može dinamički prilagoditi distribuciju saobraćaja u skladu sa trenutnim performansama i opterećenjem servera, čime se postiže optimalna iskorišćenost hardvera i minimizuje vreme obrade.

U mikroservisnoj arhitekturi, gde sistem čine nezavisni i dinamični servisi, rivers proksi i loud balanser igraju ključnu ulogu u servisnom otkrivanju, dinamičkom rutiranju i upravljanju infrastrukturom [5]. Rivers proksi deluje kao centralna tačka za rutiranje zahteva, omogućavajući klijentima da pristupe servisima bez znanja o njihovim tačnim lokacijama ili promenama u topologiji. Loud balanser integriše mehanizme za servisno otkrivanje, dinamički usmeravajući saobraćaj na raspoložive instance servisa na osnovu njihovog statusa i opterećenja [5]. Ovo omogućava lako dodavanje novih servisa u sistem ili uklanjanje postojećih bez prekida u radu, jer se sve promene automatski sinhronizuju kroz ove komponente. Osim toga, ove tehnologije obezbeđuju fleksibilnost u upravljanju saobraćajem, omogućavajući primenu strategija kao što su A/B testiranje ili postepeno uvođenje novih verzija. Ovaj nivo automatizacije i prilagodljivosti čini rivers proksi i loud balanser ključnim za efikasnost i skalabilnost mikroservisnih sistema.

C. Podela po nivoima: L4 vs. L7

Rivers proksiji i loud balanseri mogu raditi na različitim nivoima mrežnog modela, pri čemu se najčešće klasifikuju kao L4 (transportni nivo) i L7 (aplikativni nivo). Ova podela ukazuje na različite pristupe upravljanju saobraćajem, koji se biraju u zavisnosti od zahteva aplikacije i arhitekture sistema.

L4 loud balanseri i rivers proksiji rade na transportnom nivou, gde upravljaju mrežnim saobraćajem na osnovu IP adresa i portova. Ovi balanseri i proksiji ne interpretiraju sadržaj paketa, već se oslanjaju na informacije iz TCP ili UDP zaglavlja. Zbog toga su veoma efikasni u obradi velikog broja zahteva, jer minimalno analiziraju saobraćaj. Budući da nude podršku za TCP i UDP protokole, to ih čini pogodnim za primenu u aplikacijama kao što su VoIP, DNS i VPN. Pored brzine, najveća prednost je manje opterećenje procesora u poređenju sa L7 balanserima i proksijima. Sa druge strane, ovi balanseri i proksiji nemaju mogućnost za rutiranje na osnovu sadržaja ili hedera, te nisu pogodni za aplikacije koje zahtevaju složene logike rutiranja i imaju ograničenu fleksibilnost.

L7 loud balanseri i rivers proksiji rade na aplikativnom nivou i imaju mogućnost interpretacije HTTP/HTTPS saobraćaja, kao i drugih aplikativnih

protokola. Oni analiziraju sadržaj zahteva, uključujući URL, hedere, pa čak i telo poruke, kako bi doneli odluke o rutiranju. Ovi balanseri i proksiji nude SSL/TLS terminaciju, čime se smanjuje opterećenje na bekend serverima i omogućava implementacija sigurnosnih funkcija, kao što su zaštita od DDoS napada i autentikacija korisnika. Takođe, ovi balanseri i proksiji nude podršku za keširanje, kompresiju i manipulaciju zahtevima, kao i podršku za integraciju sa aplikativnim logikama, kao što su API gateway rešenja. Glavna slabost ovih balansera i proksija je da opterećuju procesor i manje su pogodni za velike količine nestrukturiranog saobraćaja kao što je VoIP.

D. AWS Elastic Load Balancing (ELB)

AWS Elastic Load Balancer (ELB) je ključna komponenta AWS ekosistema, dizajnirana da obezbedi visoku dostupnost, skalabilnost i performanse za aplikacije i usluge u oblaku [9]. ELB podržava i L4 i L7 funkcionalnosti, omogućavajući efikasno upravljanje saobraćajem. Njegove mogućnosti uključuju automatsko skaliranje kapaciteta u skladu sa trenutnim opterećenjem, health-check mehanizme za praćenje dostupnosti i ispravnosti servera, kao i SSL/TLS terminaciju radi poboljšanja sigurnosti i optimizacije performansi. ELB je dostupan u tri glavna tipa: Application Load Balancer (ALB), koji je specijalizovan za L7 saobraćaj i omogućava napredno rutiranje zasnovano na HTTP/HTTPS zahtevima; Network Load Balancer (NLB), koji upravlja L4 saobraćajem i idealan je za aplikacije koje zahtevaju nisku latenciju i visok propusni opseg; i Gateway Load Balancer (GLB), koji omogućava laku integraciju sa mrežnim uslugama kao što su zaštitni zidovi ili IDS/IPS sistemi. ELB se savršeno uklapa u AWS okruženja i mikroservisne arhitekture, jer obezbeđuje fleksibilno rutiranje, skalabilnost bez ručne intervencije i besprekornu integraciju sa drugim AWS uslugama kao što su Auto Scaling, Amazon ECS, i Amazon EKS. Ovaj alat je nezaobilazan za izgradnju modernih, pouzdanih i skalabilnih aplikacija u AWS klaudu.

E. Google Cloud Load Balancer

Google Cloud Platform (GCP) Load Balancer predstavlja moćno, globalno distribuirano rešenje za upravljanje saobraćajem koje obezbeđuje visoku dostupnost, nisku latenciju i vrhunske performanse za aplikacije u oblaku [11]. Podržava i L4 i L7 saobraćaj, što omogućava fleksibilno rutiranje i efikasno upravljanje i mrežnim i aplikativnim nivoima saobraćaja. Njegove napredne funkcije uključuju SSL/TLS terminaciju, što poboljšava sigurnost i smanjuje opterećenje bekend servera, kao i keširanje i kompresiju, koji smanjuju latenciju i ubrzavaju isporuku sadržaja. Koristeći Anycast IP arhitekturu, GCP Load Balancer distribuira saobraćaj najbližem serveru ulazne tačke, čime se postiže niska latencija i optimalno korisničko iskustvo, bez obzira na geografsku lokaciju korisnika.

Ovo rešenje je u potpunosti integrisano sa ostalim GCP uslugama, uključujući Google Kubernetes Engine (GKE), čineći ga idealnim za mikroservisne arhitekture i kontejnerske aplikacije.. GCP Load Balancer takođe podržava automatsko skaliranje i integraciju sa alatima kao što su Cloud CDN i Cloud Armor za poboljšane performanse i sigurnost. Zahvaljujući ovoj kombinaciji funkcija, GCP Load Balancer je izuzetno rešenje za moderne aplikacije koje zahtevaju brzinu, skalabilnost i globalnu dostupnost.

F. Microsoft Azure Load Balancer

Microsoft Azure nudi dva osnovna tipa rešenja za balansiranje saobraćaja, koja obezbeđuju fleksibilnost i performanse u skladu sa različitim potrebama aplikacija: Azure Load Balancer i Azure Application Gateway. Azure Load Balancer je L4 rešenje koje omogućava brzu i pouzdanu raspodelu TCP i UDP saobraćaja, idealno za osnovne mrežne zahteve kao što su visoko dostupni bekend serveri ili virtuelne mašine [10]. Sa podrškom za automatsko skaliranje i health-check mehanizme, Azure Load Balancer obezbeđuje neprekidan rad i optimalno korišćenje resursa.

S druge strane, Azure Application Gateway je L7 balanser, specijalizovan za složene aplikacije koje zahtevaju napredno rutiranje [14]. Podržava funkcije kao što su rutiranje na osnovu URL putanja, hedera ili čak sadržaja zahteva, što ga čini idealnim za veb aplikacije i mikroservisne arhitekture. Dodatne mogućnosti, poput integrisanog WAF-a (Web Application Firewall), obezbeđuju viši nivo bezbednosti za aplikacije.

Oba balansera nude duboku integraciju sa Azure Kubernetes Service (AKS), omogućavajući automatsko skaliranje i dinamičko rutiranje za mikroservisne aplikacije. Ova rešenja su posebno prilagođena organizacijama koje koriste Azure ekosistem, pružajući besprekornu integraciju sa drugim Azure uslugama, poput Azure Monitor-a za nadzor i Azure Security Center-a za poboljšanu zaštitu. Kombinovanjem performansi, skalabilnosti i bezbednosti, Azure Load Balancer i Application Gateway predstavljaju odličan izbor za izgradnju modernih, visoko dostupnih aplikacija.

G. NGINX

NGINX je jedno od najzastupljenijih i najpouzdanijih rešenja za rivers proksi, loud balansing, i upravljanje saobraćajem u modernim računarskim sistemima [5]. Njegova fleksibilnost i performanse čine ga idealnim izborom za raznovrsne aplikacije, od tradicionalnog veb hostinga do složenih mikroservisnih arhitektura. NGINX podržava i L4 i L7 funkcionalnosti, omogućavajući kako osnovno rutiranje TCP/UDP saobraćaja, tako i napredno rutiranje na nivou aplikacije, zasnovano na HTTP/HTTPS zahtevima, URL-ovima, hederima ili drugim parametrima.

Njegove napredne funkcije, kao što su SSL/TLS terminacija, keširanje statičkog sadržaja i kompresija, značajno smanjuju latenciju i ubrzavaju isporuku podataka. Kao API Gateway, NGINX omogućava efikasno rutiranje API zahteva i upravljanje saobraćajem u mikroservisnim sistemima, dok kao rivers proksi, obezbeđuje dodatni nivo apstrakcije i sigurnosti između klijenata i servera. Pored toga, njegova sposobnost da radi kao visoko optimizovani HTTP server omogućava upotrebu u aplikacijama sa velikim opterećenjem.
NGINX je izuzetno pogodan za mikroservisne arhitekture, zahvaljujući lakoj integraciji sa alatima kao što su Kubernetes i Docker, što omogućava dinamičko rutiranje i automatsko skaliranje. Njegova podrška za modularnost i proširenja, kao što su NGINX Plus, dodaje funkcije poput aktivnog health-check-a, analitike saobraćaja i integracije sa sistemima za nadzor i alatima za bezbednost. Zahvaljujući svojoj brzini, pouzdanosti i skalabilnosti, NGINX je nezaobilazan alat za izgradnju modernih, visoko dostupnih i performantnih sistema.

H. HAProxy

HAProxy je jedno od najpoznatijih open-source rešenja za loud balansing i rivers proksi, široko priznato zbog izuzetnih performansi, stabilnosti i fleksibilnosti [7]. Njegova arhitektura je optimizovana za sisteme koji zahtevaju visoku propusnu moć, minimalnu latenciju i pouzdanost u radu, što ga čini idealnim za primenu u velikim, kritičnim infrastrukturama.
HAProxy podržava i L4 (TCP/UDP) i L7 (HTTP/HTTPS) saobraćaj, omogućavajući preciznu kontrolu nad raspodelom saobraćaja i rutiranje zahteva na osnovu širokog spektra kriterijuma, uključujući URL putanje, hedere, sadržaj zahteva ili metapodatke. Njegovi napredni algoritmi za balansing, kao što su round-robin, least-connections, i hash-based distribucija, obezbeđuju ravnomerno opterećenje servera i optimalno iskorišćenje resursa.
Funkcije kao što su SSL/TLS terminacija olakšavaju upravljanje sertifikatima i smanjuju opterećenje bekend servera, dok ugrađeni health-check mehanizmi kontinuirano prate status i dostupnost servera, automatski uklanjajući neispravne instance iz rotacije. Ovo obezbeđuje stabilnost i visoku dostupnost sistema, čak i u slučaju otkaza pojedinačnih servera.
HAProxy je takođe izuzetno fleksibilan za upotrebu u mikroservisnim arhitekturama, gde se lako integriše sa alatima kao što su Kubernetes i Docker za dinamičko rutiranje i automatsko skaliranje. Njegove napredne mogućnosti konfiguracije omogućavaju kreiranje kompleksnih rutirajućih pravila i prilagođavanje ponašanja na osnovu specifičnih potreba aplikacija.
Pored toga, HAProxy nudi alate za monitoring i analizu saobraćaja, uključujući prikaz metrika u realnom vremenu i podršku za eksterne sisteme za nadzor. Zahvaljujući svojoj skalabilnosti, robusnosti i opsežnim funkcijama, HAProxy

je nezaobilazan izbor za organizacije koje zahtevaju visoke performanse, sigurnost i pouzdanost u upravljanju saobraćajem.

I. Traefik

Traefik je moderan, visoko dinamičan rivers proksi i loud balanser, dizajniran posebno za cloud-native aplikacije i mikroservisne arhitekture [6]. Njegova osnovna prednost je duboka integracija sa savremenim alatima za orkestraciju, kao što su Docker, Kubernetes, Consul, Nomad, i drugi sistemi za otkrivanje servisa. Traefik automatski otkriva nove servise i ažurira svoje konfiguracije u realnom vremenu, što značajno pojednostavljuje upravljanje dinamičnim okruženjima u kojima se servisi često dodaju, uklanjaju ili menjaju.

Jedna od njegovih najistaknutijih karakteristika je podrška za SSL/TLS terminaciju, uključujući automatsko upravljanje sertifikatima putem Let's Encrypt-a. Ovo omogućava sigurnu komunikaciju bez potrebe za ručnim konfigurisanjem sertifikata, čime se značajno smanjuje složenost u održavanju infrastrukture. Traefik takođe nudi dinamičko rutiranje, gde zahtevi mogu biti usmereni na osnovu metapodataka kao što su URL putanje, hederi, i drugi kriterijumi, što ga čini idealnim za napredno upravljanje saobraćajem u mikroservisnim okruženjima.

Dizajniran sa fokusom na performanse i lakoću upotrebe, Traefik pruža moćnu integraciju sa alatima za monitoring i vizuelizaciju, poput Prometheus-a i Grafana-e, omogućavajući pregled metrika saobraćaja u realnom vremenu. Pored toga, njegova sposobnost da funkcioniše kao API Gateway čini ga izuzetno pogodnim za rukovanje API zahtevima u složenim sistemima.

Zahvaljujući svojoj skalabilnosti, lakoći integracije i podršci za cloud-native alate, Traefik je savršeno rešenje za moderne aplikacije koje zahtevaju visoku dostupnost, automatizaciju i fleksibilno upravljanje saobraćajem. Njegova otvorena arhitektura i bogat skup funkcija čine ga jednim od vodećih izbora za organizacije koje usvajaju moderne paradigme u izgradnji i upravljanju aplikacijama.

J. Envoy

Envoy je open-source rešenje za proksi i service mesh, koje je postalo jedan od vodećih standarda u cloud-native okruženjima [8]. Razvijen sa fokusom na performanse, skalabilnost i integraciju, Envoy pruža bogatu funkcionalnost i fleksibilnost za upravljanje saobraćajem u složenim distribuiranim sistemima. Podržava kako L4 tako i L7 funkcionalnosti, omogućavajući rukovanje saobraćajem na nivou mreže i aplikacija, što ga čini izuzetno svestranim.

Jedna od najistaknutijih karakteristika Envoy-a je njegova podrška za naprednu telemetriju. Envoy prikuplja detaljne podatke o saobraćaju, uključujući metrike, logove i tragove (traces), što omogućava dubok uvid u performanse i brzo otkrivanje problema. Ove funkcije ga čine idealnim za upotrebu u

kombinaciji sa sistemima za nadzor, kao što su Prometheus, Grafana, i Jaeger, omogućavajući precizno praćenje u realnom vremenu.
Envoy takođe nudi moćne funkcije za dinamičko rutiranje, što znači da se saobraćaj može usmeravati na osnovu različitih kriterijuma, kao što su URL-ovi, HTTP hederi, metapodaci i status servisa. Njegova podrška za SSL/TLS terminaciju i automatizaciju upravljanja sertifikatima obezbeđuje visok nivo sigurnosti, dok podrška za HTTP/2 i gRPC saobraćaj omogućava korišćenje modernih protokola za komunikaciju između servisa.
Envoy je duboko integrisan sa Istio i drugim service mesh rešenjima, što ga čini osnovnim komponentom za izgradnju mikroservisnih arhitektura. Kao deo service mesh-a, Envoy upravlja mrežnim saobraćajem između servisa, obezbeđuje sigurnost, i automatizuje funkcionalnosti poput circuit breaking-a, rate limiting-a i fault injection-a.
Zahvaljujući svojoj fleksibilnosti, skalabilnosti i naprednim funkcijama, Envoy se koristi u nekim od najzahtevnijih okruženja i ostaje jedan od najboljih izbora za kompanije koje grade složene i visoko dostupne aplikacije u cloud-native ekosistemu.

III. NAPREDNE FUNKCIONALNOSTI

A. Duplikacija soketa

Socket duplication je tehnika koja omogućava da više procesa u operativnom sistemu dele isti mrežni soket. Socket duplication podrazumeva kopiranje deskriptora mrežnog soketa, tako da više procesa ili nitova mogu istovremeno pristupati istom mrežnom resursu. Operativni sistem omogućava ovu funkcionalnost kroz sistemske pozive, poput dup ili dup2 u POSIX sistemima, kao i kroz specifične mehanizme kao što je SO_REUSEPORT opcija soketa.
Socket duplication omogućava više procesa da dele isti mrežni soket bez potrebe za otvaranjem nove TCP veze. Pored ovoga, unapređuje paralelnu obradu zahteva, posebno u slučajevima kada više procesa obrađuje ulazne mrežne pakete. Sve ovo je korisno u arhitekturama gde je potrebno razdvajanje logike obrade na više modula ili procesa.
Rivers proksiji često rade u okruženjima sa visokim opterećenjem, gde istovremeno moraju obrađivati hiljade ili milione mrežnih zahteva. Socket duplication omogućava da više procesa može istovremeno koristiti isti soket, što omogućava distribuciju zahteva na više jezgara procesora. Deljenje soketa između različitih komponenti rivers proksija olakšava implementaciju funkcija kao što su keširanje, autentifikacija i prosleđivanje zahteva. Pored ovoga, podržava povećanje broja procesa bez potrebe za ponovnim otvaranjem mrežnog soketa. Na kraju, jako je bitno napomenuti i da za neke primene gde

će ogromna količina podataka biti poslata od klijenta ka serveru, nije dobra praksa dozvoliti da svi ti podaci moraju da budu slani preko rivers proksija, već je direktno slanje podataka na serverski proces bolje zbog performansi. Što se tiče izazova i ograničenja, implementacija može postati složena, posebno u multiprocesnim i konkurentnim okruženjima. Ne sme se izgubiti iz vida da korišćenje ove tehnike zahteva da proksi i server budu pokrenuti na istoj mašini i istom operativnom sistemu, što nije uvek moguće.

B. NAT port forwarding

NAT (Network Address Translation) port forwarding je tehnika koja omogućava transparentno preposleđivanje saobraćaja sa jednog porta na jedan ili više različitih portova unutar privatne mreže. Ova tehnika se često koristi u sklopu loud balansing sistema, kako bi se zahtevi usmerili na odgovarajuće mašine u klasteru. Kada loud balanser dobije dolazni zahtev na određeni port, on:

1. Analizira port iz dolaznog zahteva.
2. Na osnovu predefinisanih pravila mapiranja, poveže taj port sa određenim "range" (opsegom) portova.
3. Opseg portova ukazuje na konkretan “interni” port na kome mašine u klasteru zaista slušaju, a svaki port iz opsega je napiran na tačno jednu mašinu u klasteru.
4. Zahtev se prosleđuje izabranoj mašini, dok loud balanser obezbeđuje transparentnost prema klijentu.

Port forwarding funkcioniše tako što loud balanser prima TCP ili UDP pakete i iz njih parsira zahteve kako bi izvadio informacije o destinacionom portu. Nakon toga se koristi mapiranje porta na odgovarajući opseg, gde se keš tabele ili druge strukture podataka koriste za efikasno usmeravanje paketa. Na osnovu opsega, loud balanser dinamički određuje rutu i port i prosleđuje paket mašini koja pokriva taj port. Pored toga, on obrađuje i povratni saobraćaj, osiguravajući da se odgovori servera vrate odgovarajućem klijentu.

Ovakva realizacija ima brojne prednosti. Najpre, ovakva postavka nam omogućava da napravimo proizvoljnu raspodelu naših servisa i aplikacija, gde možemo da u istom klasteru hostujemo servise za različite namene. Konkretno, svaki tip servisa bi odgovarao jednom opsegu portova, odnosno jednom internom portu na kome taj servis sluša, ali bi bilo moguće više servisa postaviti na istu mašinu dokle god postoje drugi opsezi koji se mapiraju na druge interne portove za ostale servise. Dalje, fleksibilnost se ogleda u tome što je jednostavno promenjivim pravilima mapiranja moguće dodati nove mašine ili portove u sistem. Na kraju, sigurnost se poboljšava kroz NAT, koji omogućava da se unutrašnja struktura mreže sakrije od spoljašnjih klijenata.

Ipak, postoje i određeni izazovi u implementaciji. Konfiguracija je jedan od glavnih izazova jer podešavanje i održavanje mapiranja može biti veoma složeno u velikim sistemima. Greške u raspodeli su takođe čest problem, jer pogrešno konfigurisanje opsega portova može dovesti do konflikta. Dešava se da u velikim sistemima ponestane portova za mapiranje, budući da je maksimalna vrednost porta 65535. Pored toga, performanse mogu biti ugrožene jer saobraćaj kroz loud balanser dodaje određenu latenciju, što utiče na brzinu sistema, naročito ako nije optimizovan.

C. SNI

TLS (Transport Layer Security) je protokol koji obezbeđuje sigurnu komunikaciju između klijenta i servera. Njegova primena je ključna u modernim mrežnim aplikacijama, posebno kada je reč o zaštiti osetljivih podataka tokom prenosa. TLS 1.2 i TLS 1.3 su najzastupljenije verzije ovog protokola, pri čemu TLS 1.3 uvodi brojna poboljšanja u odnosu na prethodne verzije, uključujući bržu inicijalizaciju sesije i veću sigurnost.

SNI (Server Name Indication) je proširenje TLS protokola koje omogućava klijentu da tokom uspostavljanja TLS veze pošalje ime hosta (serverske aplikacije) koji želi da poseti.

Kada klijent inicira TLS sesiju, on šalje "ClientHello" poruku serveru, koja sadrži razne parametre potrebne za uspostavljanje sigurne komunikacije. Ukoliko klijent podržava SNI, u ovoj poruci će biti uključeno i ime domena. Server, na osnovu primljenog SNI, može izabrati odgovarajući sertifikat ili uslugu koja odgovara traženom domenu. SNI je posebno značajan u scenarijima kao što su:

- Višestruki domeni na jednoj IP adresi (virtuelni hosting).
- Usluge koje koriste različite sertifikate za različite domene.
- Primena različitih politika sigurnosti za različite domene.

U loud balanseru ili rivers proksiju, SNI se može iskoristiti za rutiranje saobraćaja na osnovu imena domena. Kada klijent pošalje SNI u "ClientHello" poruci, loud balanser može analizirati ovaj podatak i proslediti zahtev odgovarajućem backend serveru.

SNI je jednostavno, ali veoma moćno proširenje TLS protokola koje omogućava efikasno rutiranje saobraćaja u loud balanserima i rivers proksijima. Njegova primena je od suštinskog značaja u okruženjima sa višestrukim domenima, jer omogućava optimalnu raspodelu resursa i unapređenje sigurnosti. Uz razvoj protokola kao što je TLS 1.3, koji uvodi poboljšanu enkripciju, SNI će postati još važniji alat u mrežnoj infrastrukturi.

IV. METODOLOGIJA ISTRAŽIVANJA

A. *Eksperimentalno okruženje*

Eksperiment će se zasnivati na izvođenju takozvanog "load" testa koji stavlja sistem pod opterećenje. Ovo ćemo izvesti tako što ćemo na lokalnoj mašini podesiti veliki broj TCP/HTTP klijenata i manji broj servera, koji će razmenjivati jednostavne poruke. Klijenti će zapravo poruke na rivers proksi/loud balanser, koji će zatim poruke prosleđivati na servere i obrnuto.
Za ove potrebe, koristimo mašinu sa operativnim sistemom Windows 11 Pro, i sledećim bitnim specifikacijama:

- Procesor: 11th Gen Intel(R) Core(TM) i9-11950H @ 2.60GHz, 2611 Mhz, 8 Core(s), 16 Logical Processor(s)
- Memorija: 63.7 GB

Vredi napomenuti da su specifikacije ove mašine, iako dobre, manje od prosečne virtuelne mašine u data centrima velikih klaud provajdera na kojima bi se zapravo u modernoj aritekturi i sistemu pokretao proksi. Pored ovoga, takva okruženja uglavnom nude određene vrste virtuelizacije kao i kontrole i pravilne raspodele resursa mašine, koje garantuju da se proces koji se pokreće ne takmiči za resurse sa drugim procesima na istoj mašini, što naravno nije slučaj kada se eksperiment izvodi na ličnom računaru.

B. *Definisanje metrika performansi*

U istraživanju i poređenju L4 i L7 rivers proksija i loud balansera, koristimo različite metrike za merenje performansi i efikasnosti sistema. Pored funkcionalnih aspekata, ove metrike omogućavaju kvantitativnu ocenu i poređenje našeg rešenja sa popularnim alatima kao što su NGINX i Traefik.
Metrike koje koristimo su sledeće:

1. Broj zahteva u sekundi - Predstavlja kapacitet sistema u obradi klijentskih zahteva u jedinici vremena. Visok broj zahteva u sekundi ukazuje na sposobnost sistema da obrađuje veliki broj istovremenih zahteva, što je posebno važno za visoko opterećene sisteme.
2. Trajanje obrade zahteva - Meri vreme potrebno za obradu pojedinačnog zahteva, izraženo u milisekundama. Pratićemo percentilne vrednosti (npr. P50, P95) i proseke, što nam pomaže u analizi performansi pod različitim opterećenjem. Niska latencija je ključna za poboljšanje korisničkog iskustva, posebno u aplikacijama sa strimingom ili interaktivnim elementima.
3. Ukupan broj zahteva tokom eksperimenta - Predstavlja ukupan broj uspešno obrađenih zahteva tokom trajanja testa. Omogućava nam procenu ukupnog kapaciteta sistema tokom dužeg vremenskog perioda.
4. Broj istovremenih veza koje sistem može istovremeno da održava - Ova metrika je ključna za aplikacije koje zahtevaju stalne ili dugotrajne konekcije, kao što su VoIP ili veb soketi.

Pored ovih metrika, za međusobno poređenje sopstevnih rešenja merićemo iskorišćenost procesora i memorije kao i broj aktivnih niti proksija. Ove metrike ne možemo na tačan način izmeriti za open-source rešenja zbog toga što bi zahtevale izmene u samom kodu tih rešenja ili korišćenje nekih alata koje nisu dostupne na operativnom sistemu na kome izvodimo eksperiment.

C. Alati za merenje metrika performansi

Kako bismo na tačan način izmerili metrike koje smo opisali, potrebni su nam određeni alati koji će ispratiti ove metrike i preko kojih ćemo ih prikazati na adekvatan način. Pored ovoga, moramo da se postaramo da naš softver emituje ove metrike prilikom normalnog rada, i da istražimo koje tačno metrike i na koji način emituju open-source rešenja sa kojim se poredimo, i da podesimo odgovarajuće okruženje za integraciju alata za praćenje metrika sa open-source proksijima i loud balanserima.

Za emitovanje i sakupljanje metrika koristimo open-source rešenje pod nazivom Prometheus. Prometheus je moćan sistem za monitoring i sakupljanje metrika, dizajniran da obezbedi dubok uvid u performanse i stanje aplikacija i infrastrukture. Razvijen kao deo Cloud Native Computing Foundation (CNCF), Prometheus je posebno pogodan za mikroservisne arhitekture i cloud-native okruženja zahvaljujući svojoj mogućnosti da prikuplja metrike sa više izvora i da ih skladišti u vremenske serije. Glavni način rada Prometheus-a zasniva se na pull modelu, gde on periodično preuzima metrike sa aplikacija koje ih čine dostupnim preko HTTP endpointa. Podržava prilagođeno definisanje metrika u aplikacijama, što ga čini idealnim za praćenje metrika kao što su broj obrađenih zahteva, latencija, iskorišćenost procesora i memorije. Prometheus koristi svoj jezik za upite, PromQL (Prometheus Query Language), koji omogućava fleksibilnu analizu i vizualizaciju podataka. Dodatno, lako se integriše sa alatima kao što je Grafana za vizuelizaciju metrika.

Grafana je vodeći open-source alat za vizualizaciju i analizu metrika koji se često koristi zajedno sa sistemom za monitoring Prometheus. Dizajnirana je da pruži prilagodljive i interaktivne kontrolne table (dashboards), omogućavajući korisnicima da vizuelizuju i analiziraju podatke u realnom vremenu. Grafana podržava različite vrste grafika, kao što su linijski grafici, tabele i hit mape, što olakšava praćenje ključnih pokazatelja performansi (KPI). Integracija sa Prometheus-om omogućava korišćenje PromQL upita direktno u interfejsu za kreiranje i podešavanje vizuelizacija.

U našem istraživanju, Grafana će služiti za vizuelizaciju metrika kao što su broj zahteva u sekundi, latencija, broj aktivnih konekcija, i iskorišćenost resursa, koje prikuplja Prometheus. Ovaj alat će nam omogućiti da identifikujemo trendove, uska grla i nepravilnosti u performansama. Dodatno,

ukoliko bismo naš sistem plasirali u produkcioni okruženje, sistem za kreiranje alarma koji nudi Grafana omogućava automatsko obaveštavanje o anomalijama, što je ključno za održavanje stabilnosti u proizvodnim okruženjima. U kombinaciji sa Prometheus-om, Grafana predstavlja moćno rešenje za monitoring i optimizaciju našeg sistema.

V. REZULTATI ISTRAŽIVANJA

A. *Poređenje rezultata*

Nakon sprovedenih eksperimenata, analizirani su rezultati za ključne metrike performansi koje uključuju broj zahteva u sekundi, vreme obrade zahteva, kao i kapacitet sistema u pogledu broja aktivnih konekcija. U nastavku su predstavljeni rezultati u tabeli 1, što omogućava jasno poređenje naših implementacija (L4 i L7) sa NGINX-om i Traefik-om.

Posmatrajući prosečan broj zahteva po sekundi (RPS), L4 proksi sa TLS terminacijom ostvaruje najbolji rezultat od 984 zahteva u sekundi, dok L4 proksi bez TLS terminacije ostvaruje 978 zahteva u sekundi, što ukazuje na minimalan trošak uvođenja TLS-a u ovom scenariju. Naše L7 rešenje ima prosečnih 923 zahteva u sekundi, što je konkurentno u poređenju sa NGINX-om, koji beleži 747 zahteva u sekundi, ali ipak primetno niže u odnosu na Traefik, koji sa 982 zahteva u sekundi pokazuje vrhunsku efikasnost.

Kada se analizira broj aktivnih konekcija, Traefik jasno prednjači sa konstantnim brojem od 1000 aktivnih konekcija koje su keširane, dok NGINX postiže samo 12.8 aktivnih konekcija u proseku. Suprotno tome, naše L4 rešenje beleži vrlo nizak prosečan broj aktivnih konekcija od 0.271 i 0.185, što ukazuje na izuzetno brzo rukovanje i zatvaranje konekcija, bez zadržavanja dugotrajnih sesija. L7 proksi ima nešto veći prosek od 3.33 aktivne konekcije, što je očekivano zbog složenijeg rukovanja HTTP zahtevima. U pogledu maksimalnog broja aktivnih konekcija, Traefik dostiže već pomenutih 1000, dok NGINX postiže 456, a naše L4 rešenje dostiže maksimalnih 20 i 25 konekcija, što može da ukazuje na ograničenje u pogledu skalabilnosti, ali je moguće da je samo i posledica brze obrade i načina praćenja metrika. L7 proksi, sa maksimalnih 722 aktivnih konekcija, nadmašuje NGINX ali i dalje zaostaje za Traefik-om.

TABELA 1: REZULTATI EKSPERIMENTA.

	L4 proksi bez TLS terminacije	L4 proksi sa TLS terminacijom	L7 proksi	Nginx	Traefik
Prosečan broj	978	984	923	747	982

zahteva po sekundi (RPS)					
Prosečan broj aktivnih konekcija	0.259	0.185	3.33	12.8	1000
Maksimalni broj aktivnih konekcija	20	25	722	456	1000
Prosečno vreme obrade zahteva (ms)	4.23	4.34	148	76.734	9.25
Medijana vremena obrade zahteva (ms)	3.08	3.25	3	2	6.98
95-i percentil vremena obrade (ms)	9.56	9.64	300+	8	28.8
Ukupan broj zahteva procesiran	586829	590113	553582	448304	590000

Prosečno vreme obrade za L4 proksi bez TLS iznosi 4.23 milisekundi, dok sa TLS terminacijom blago raste na 4.34 milisekundi, što su izuzetno niske vrednosti u poređenju sa NGINX-om koji beleži 76.734 milisekundi zbog velikih ekstremnih vrednosti i Traefik-om koji ostvaruje 9.25 milisekundi. Nasuprot tome, naše L7 rešenje ima prosek od 148 milisekundi, što je značajno veće, što ponovo ukazuje na složeniju obradu zahteva na aplikativnom nivou. Medijana vremena obrade takođe pokazuje stabilnost našeg L4 rešenja, sa vrednostima od 3.08 i 3.25 milisekundi, dok je kod L7 proksija medijana 3 milisekundi, što ukazuje na relativnu stabilnost, ali sa nešto većim ekstremnim vrednostima u latenciji. U 95-om percentilu, L4 proksiji ostaju na 9.56 i 9.64 milisekundi, dok je L7 rešenje daleko iznad, sa 300+ milisekundi, što je posledica pikovitih kašnjenja u obradi zahteva.

Što se tiče ukupnog broja zahteva procesiranog tokom trajanja testa, Traefik ponovo pokazuje dominaciju sa 590003 zahteva, dok naše L4 rešenje bez TLS terminacije obrađuje 586829, a sa TLS 590113, što je ponovo blizu. L7 proksi ostvaruje ukupan rezultat od 553582 zahteva, što je znatno više od NGINX-a koji beleži 448304 zahteva, čime se pokazuje da naše rešenje ima konkurentan kapacitet u odnosu na popularne alate.

Na osnovu ovih rezultata može se zaključiti da su naša L4 rešenja veoma optimizovana za nisku latenciju i visoku propusnu moć, ali pokazuju ograničenja u rukovanju aktivnim konekcijama, posebno u poređenju sa Traefik-om. Ovo ograničenje prikazaćemo u nastavku dodatnim eksperimentom. Naše L7 rešenje postiže solidne performanse u pogledu ukupnog broja obrađenih zahteva i stabilnosti vremena obrade, ali se i dalje suočava sa izazovima u skalabilnosti i upravljanju latencijom pri većim opterećenjima. Traefik se pokazuje kao najskalabilnije rešenje, dok NGINX i dalje predstavlja stabilan izbor, ali sa značajnim ograničenjima u vremenu obrade i kapacitetu pod velikim opterećenjem. Dalji rad bi se mogao usmeriti na optimizaciju upravljanja konekcijama i TLS obradom kako bi naše rešenje postalo konkurentnije u uslovima većih opterećenja.

B. Dodatno testiranje

Budući da su se naša L4 rešenja pokazala veoma konkurentnim u vidu propusne moći i vremena obrade zahteva, izvodimo i dodatan eksperiment kako bismo analizirali rešenja u ekstremnijim uslovima sa još većim brojem klijenata i konekcja. Na ovaj način, isptitujemo potencijalnu slabost sistema u vidu broja aktivnih konekcija koja može biti ključna u ovakvoj postavci. U ovom eksperimentu, podižemo broj klijenata na 5000, gde će svaki klijent slati poruku na pola sekunde, odnosno dve poruke u sekundi. Sve ostalo ostaje isto kao i u prethodnom testu. Rezultate prikazujemo u tabeli 2.

Primećuju se značajne razlike između performansi naših implementacija sa jedne strane i NGINX i Traefik sa druge. Kada je reč o prosečnom broju zahteva po sekundi, Traefik značajno prednjači sa 3925 zahteva u sekundi, što ga izdvaja kao najbolje rešenje za obradu velikog broja klijenata. NGINX, sa 215 zahteva u sekundi, ostvaruje bolji rezultat u odnosu na naše implementacije. Naše L4 rešenje bez TLS terminacije postiže 104 zahteva u sekundi, dok L4 sa TLS terminacijom blago prednjači sa 106 zahteva u sekundi. Ovi rezultati ukazuju na stabilnost ali i ograničenje L4 proksija u pogledu kapaciteta, posebno u poređenju sa Traefik-om. Nasuprot tome, naše L7 rešenje je postiglo samo 25.8 zahteva u sekundi, što pokazuje značajan pad u performansama usled složenije obrade na aplikativnom nivou. Na kraju, primećujemo da su ovi rezultati daleko gori od onih iz prethodnog eksperimenta, što ukazuje da bi bilo pametno da unapredimo propusnu moć naših sistema, ali i da uvedemo mehanizme ograničenja saobraćaja kako sistem ne bi značajno usporio kao ovde.

TABELA 2: REZULTATI DODATNOG EKSPERIMENTA.

	L4 proksi	L4 proksi sa	L7	Nginx	Traefik

	bez TLS terminacije	TLS terminacijom	proksi		
Prosečan broj zahteva po sekundi (RPS)	978	984	923	747	982
Prosečan broj aktivnih konekcija	0.259	0.185	3.33	12.8	1000
Maksimalni broj aktivnih konekcija	20	25	722	456	1000
Prosečno vreme obrade zahteva (ms)	4.23	4.34	148	76.734	9.25
Medijana vremena obrade zahteva (ms)	3.08	3.25	3	2	6.98
95-i percentil vremena obrade (ms)	9.56	9.64	300+	8	28.8
Ukupan broj zahteva procesiran	586829	590113	553582	448304	590000

Posmatrajući broj aktivnih konekcija, Traefik je takođe najskalabilniji, podržavajući maksimalnih 5000 konekcija, zahvaljujući keširanju, dok NGINX postiže značajno manje, sa 436 konekcija. Naše L4 rešenje bez TLS terminacije podržava 23 aktivne konekcije, dok sa TLS terminacijom taj broj raste na 76, što može da ukazuje na određeno poboljšanje, ali imajući u vidu prethodni eksperiment, vrlo verovatno predstavlja samo jednu ekstremnu vrednost. L7 proksi, sa maksimalno 12 konekcija, pokazuje svoja ograničenja u uslovima velikog opterećenja.

Što se tiče vremena obrade zahteva, naše L4 rešenje bez TLS terminacije postiže prosečno vreme od 75.5 milisekundi, a sa TLS terminacijom 92.9 milisekundi. Ove vrednosti su znatno bolje u odnosu na NGINX, koji ima prosečno vreme obrade od 538.411 milisekundi, i Traefik, koji postiže 785 milisekundi. L7 rešenje ima prosečno vreme od 20.8 milisekundi, što je vrlo konkurentno, ali je i dalje ograničeno u pogledu ukupnog kapaciteta obrade zahteva. Ovo, u skladu i sa rezultatima prvog eksperimenta, pokazuje da su

naši L4 proksiji optimizovani za nisku latenciju. Medijana vremena obrade zahteva je izuzetno niska za sva 3 naša rešenja. Sa druge strane, Traefik ima medijanu od 713 milisekundi, dok je NGINX na 8 milisekundi, što ukazuje na varijabilnost u performansama pri različitim opterećenjima.

Ipak, vidimo da Traefik i NGINX žrtvuju latenciju za propusnot, jer ukupan broj zahteva procesiran tokom testa takođe jasno ukazuje na dominaciju Traefik-a, koji je obradio 706551 zahtev, sa NGINX-om na drugom mestu koji je obradio 40194 zahteva. Naše L4 rešenje bez TLS terminacije obradilo je 18376 zahteva, a sa TLS terminacijom 19148. L7 rešenje, sa ukupno 3098 zahteva, zaostaje za ostalim rešenjima, što potvrđuje da postoji prostor za optimizaciju.

Na osnovu ovih rezultata, potvrđujemo da naše L4 implementacije postižu stabilne performanse i nisku latenciju, ali su ograničene u pogledu broja aktivnih konekcija i ukupnog kapaciteta obrade zahteva. L7 rešenje je konkurentno u pogledu vremena obrade pojedinačnih zahteva, ali se značajno smanjuje u pogledu skalabilnosti i performansi pod velikim opterećenjem. Traefik se pokazao kao najbolje rešenje za obradu velikog broja klijenata i aktivnih konekcija, dok NGINX ostaje stabilno rešenje ali sa ograničenjima u latenciji. Jasno je da postoji veliki broj unapređenja koja bi nam donela bolje rezultate, a na prvom mestu keširanje konekcija. Ipak, treba napomenuti da su NGINX i Traefik rešenja sa mnogo bogatijim skupom funkcionalnosti u odnosu na naša rešenja, i svaka od tih funkcionalnosti takođe može uvesti i neke neželjene efekte u naš sistem po pitanju performansi.

C. Dalje istraživanje

Pored ovih eksperimenata, dalje istraživanje može uključiti različite vidove testiranja i proširene scenarije koji bi bolje simulirali realne uslove korišćenja rivers proksija i loud balansera.

Jedan od prvih pristupa bi bio testiranje sa postepenim povećanjem opterećenja, gde bi se broj klijenata i zahteva povećavao u kontinuitetu kako bi se utvrdila tačka u kojoj sistem počinje da degradira. Ovaj pristup bi omogućio precizno određivanje breaking point-a, odnosno maksimalnog kapaciteta sistema.

Druga vrsta testiranja podrazumevala bi analizu performansi u uslovima različite složenosti zahteva. Pored jednostavnih tekstualnih poruka, mogli bi se koristiti složeniji zahtevi, poput velikih JSON objekata ili enkriptovanih TLS zahteva, kako bi se ispitala sposobnost sistema da obradi veće poruke i komplikovanije protokole.

Paralelno sa tim, testiranje sa različitim konfiguracijama servera bi otkrilo kako broj servera i njihove individualne performanse utiču na efikasnost

proksija. Ispitivanje bi se izvodilo u okruženjima sa 10, 50 ili 100 servera kako bi se analizirala skalabilnost sistema.

Dalje, nastavak istraživanja bi mogao da uključi i geografski distribuirano testiranje, gde bi klijenti i serveri bili raspoređeni na različitim lokacijama širom sveta. Ovaj pristup bi omogućio ispitivanje uticaja mrežnog kašnjenja i pouzdanosti sistema u uslovima velike udaljenosti. Pored toga, testiranje u uslovima gubitka mrežne povezanosti ili sa varijacijama u mrežnoj latenciji bilo bi korisno za procenu stabilnosti i otpornosti sistema. Dugotrajno testiranje (soak testing) takođe je važan deo budućeg rada, gde bi sistem radio pod umerenim ali konstantnim opterećenjem tokom 24 sata kako bi se utvrdilo postojanje problema poput memory leak-ova ili povećanja latencije tokom vremena.

Na kraju, sigurnosno testiranje bi omogućilo ispitivanje sistema na potencijalne slabosti kao što su DDoS napadi ili neispravno rukovanje SSL/TLS vezama. Takođe, poređenje performansi na različitim platformama, poput lokalnog računara, cloud okruženja (AWS, GCP) ili edge uređaja, moglo bi pokazati kako infrastruktura utiče na performanse. Svi ovi pristupi doprineli bi sveobuhvatnoj evaluaciji sistema, otkrivanju uskih grla i pružanju smernica za dalje optimizacije kako bi rešenje bilo što stabilnije i skalabilnije.

VI. ZAKLJUČAK

Kroz analizu funckinalnosti postojećih sistema pružili smo uvid u dizajn i implementacione detalje neophodne za razvoj skalabilne softverske arhitekture sposobne za procesiranje i rukovanje velikim količinama podataka.

Rezultati dobijeni izvršavanjem testova pokazuju da su L4 proksiji u našem rešenju izuzetno optimizovani za nisku latenciju i brzu obradu zahteva, posebno u jednostavnim scenarijima sa malim zahtevima. Međutim, primećeno je ograničenje u pogledu broja aktivnih konekcija koje naš L4 proksi može da podrži, posebno u poređenju sa Traefik-om koji se pokazao kao najskalabilniji.

Osim detaljnog testiranja, u radu su predloženi i različiti vidovi daljeg testiranja, uključujući stres testove, geografski distribuirane testove, testove sa složenijim porukama i dugotrajnim opterećenjem. Takođe su predložene optimizacije u pogledu poboljšanja skalabilnosti L7 proksija i rukovanja TLS zahtevima kako bi se umanjili pikovi u latenciji i podržao veći broj istovremenih konekcija.

Zaključno, ovaj rad je uspeo da identifikuje ključne prednosti i slabosti različitih rešenja za rivers proksi i loud balansing, uz uspešnu implementaciju i evaluaciju sopstvenih rešenja. Rezultati dobijeni ovim istraživanjem pružaju značajan doprinos razumevanju rada proksija na L4 i L7 nivou i otvaraju

prostor za dalje unapređenje, sa ciljem postizanja veće skalabilnosti, stabilnosti i efikasnosti u realnim scenarijima primene.

LITERATURA

[1] M. Abbott, M. Fisher, "Scalability Rules: 50 Principles for Scaling Web Sites"
[2] J. Aguilar-Saborit, R. Ramakrishnan, K. Srinivansan, "POLARIS: The Distributed SQL Enginer in Azure Synapse"
[3] C. Kopparapu, "Load Balancing Servers, Firewalls, and Caches"
[4] P. Membrey, D. Hows, E. Plugge, "Practical Load Balancing: Ride the Performance Tiger"
[5] M. Kleppmann, "Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems"

ABSTRACT

This paper explores the implementation and comparison of reverse proxies and load balancing methods across Layer 4 (L4) and Layer 7 (L7) of the OSI model. Reverse proxies and load balancers are essential components in modern distributed systems, ensuring scalability, reliability, and performance optimization. The research focuses on evaluating the performance of custom L4 and L7 reverse proxy and load balancing solutions against popular open-source tools such as NGINX and Traefik.

This paper contributes to the understanding of reverse proxy and load balancing mechanisms, presenting insights into their application in high-performance systems. Recommendations for further research include optimizing TLS handling, improving multi-threaded processing, and exploring dynamic load balancing algorithms for distributed architectures. The findings are valuable for designing scalable and efficient network infrastructure in modern software systems.

Implementation and comparison of different reverse proxy and load balancing methods and solutions

Bogdan Bakarec, Nemanja Radosavljević