Skaliranje web poslužitelja: strojno učenje za predviđanje opterećenja s automatskim skaliranjem okruženja

Ljubojević, Luka

Sveučilište u Rijeci
Fakultet informatike
i digitalnih tehnologija

Sveučilišni diplomski studij Informatika

Luka Ljubojević

Skaliranje web poslužitelja: strojno učenje za predviđanje opterećenja s automatskim skaliranjem okruženja

Diplomski rad

Mentor: prof. dr. sc. Sanda Martinčić-Ipšić

Rijeka, lipanj 2024.

Rijeka, 3.4.2024.

Zadatak za diplomski rad

Pristupnik: Luka Ljubojević

Naziv diplomskog rada: Skaliranje web poslužitelja: strojno učenje za predviđanje opterećenja s automatskim skaliranjem okruženja

Naziv diplomskog rada na eng. jeziku: Adaptive Web Server Scaling: Machine Learning for Load Prediction with Auto-scaling of Environment

Sadržaj zadatka:
Za poslovni sustav kreirati će se klaster web servera na kojem će se izvoditi i model strojnog učenja. Klaster je temeljen na virtualnim strojevima koje će izvoditi hipervizor, a koji simuliraju sustave poput Amazon EC2, DigitalOcean droplet-a i sl. Unutar klastera izvoditi će se: web poslužitelj s aplikacijom, sustav za balansiranje opterećenja prometa, upravljački server koji pogoni model strojnog učenja, pomoćni web poslužitelji, dijeljeni disk s izvornim kodom aplikacije, SQL baza podataka, NoSQL baza podataka, model strojnog učenja, obrada skupa podataka i sustav za vizualizaciju podataka modela. Na postavljenom klasteru će se izvoditi poslovna web aplikacije pri čemu će poslužitelji bilježit podatke o pristupu aplikaciji te opterećenje sustava.
Na klasteru će se postaviti postupak strojnog učenja za predviđanje opterećenja postavljenog sustava. Temeljem predviđanja model automatski će se uključivati i isključivati web poslužitelj. Odabrati će se dovoljno velik skup podataka koji može skalirati na postavljenoj infrastrukturi. Postupak učenja evaluirat će se standardnim metrikama strojnog učenja.

Zadatak na engleskom jeziku:
For the business system, a cluster of web servers will be created, on which a machine learning model will also be executed. The cluster is based on virtual machines that will be run by a hypervisor, simulating systems such as Amazon EC2, DigitalOcean droplet, etc. Within the cluster, the following will be executed: a web server with the application, a load balancing system, a control server running the machine learning model, auxiliary web servers, a shared disk with the application source code, an SQL database, a NoSQL database, the machine learning model, data set processing, and a data visualization system. On the set cluster, a business web application will be executed, where the servers will record data on application access and system load.

The cluster will implement a machine learning procedure for predicting the load on the system. Based on the predictions, the model will automatically turn web servers on and off. A sufficiently large dataset that can scale on the set infrastructure will be selected. The learning procedure will be evaluated using standard machine learning metrics.

Mentor: Prof. dr. sc. Sanda Martinčić-Ipšić

Voditeljica za diplomske radove:
Doc. dr. sc. Lucia Načinović Prskalo

Zadatak preuzet (datum): 3.4.2024.

(potpis pristupnika/ce)

Radmile Matejčić 2,
51000 Rijeka, Hrvatska

T: +385 (0)51/584-700
E: ured@inf.uniri.hr
www.inf.uniri.hr

OIB: 64218323816
IBAN: HR1524020061400006966

Sažetak

Ubrzanim razvojem i napredovanjem web aplikacija, nastala je potreba za rješenjem kojim će se, u što kraćem roku, uspostaviti posluživanje web aplikacija. Kao odgovor na tu potrebu nastalo je računalstvo u oblaku i tehnike kontejnerizacije okoline web aplikacija.

Paralelno s razvojem oblaka, još se većom brzinom razvijaju tehnike strojnog učenja.

Kako raste složenost web aplikacija, tako raste i potreba za povećanjem broja poslužitelja iste aplikacije – skaliranjem. Skalirati web aplikaciju podrazumijeva i pravovremeno otkrivanje potrebe za pokretanjem novih poslužitelja. S obzirom na nedavna visoka povećanja cijena usluga u oblaku, pametno skalirati podrazumijeva i puno uštediti.

Većina pružatelja usluga u oblaku počinje integrirati tehnike strojnog učenja za predikciju skaliranja što dovodi do nastanka automatskog skaliranja računalnih resursa u oblaku.

Ovaj diplomski rad demonstrira princip rada automatskog skaliranja računalnog sustava u simuliranom okruženju. Pokazuje kako uspostaviti „privatni oblak“ na Hetzner pružatelju usluga u oblaku i kako se u taj sustav uključuje XGBoost model strojnog učenja za predikciju opterećenja web poslužitelja.

Rezultati modela prikazuju da je moguće simulirati usluge automatskog skaliranja u oblaku na lokalnim (eng. „*on-premises*“) poslužiteljima. Rezultati modela su dobiveni standardnim evaluacijskim metrikama poput R2 rezultata, srednje i apsolutne kvadratne pogreške.

Skaliranje je testirano na danom sustavu i pokazalo je zadovoljavajuće rezultate – model točno predviđa trend broja zahtjeva, no ne i njihovu vrijednost što je za svrhu demonstracije sustava dovoljno.

Ključne riječi: računalstvo u oblaku; strojno učenje; poslužitelji; skaliranje; opterećenje poslužitelja; XGBoost, R2 rezultat, srednja kvadratna pogreška

Title

Adaptive Web Server Scaling: Machine Learning for Load Prediction with Auto-scaling of Environment

Abstract

With the rapid development and advancement of web applications, there was a need for a solution that would establish the serving of web applications as soon as possible. In response to this need, cloud computing and web application environment containerization techniques were created.

In parallel with the development of the cloud, machine learning techniques are developing at an even greater speed.

As the complexity of web applications grows, so does the need to increase the number of servers of the same application - scaling. Scaling a web application also means timely detection of the need to start new servers. Given the recent high price increases for cloud services, scaling smart also means saving a lot.

Most cloud service providers are beginning to integrate machine learning techniques for predictive scaling leading to the emergence of automatic scaling of cloud computing resources.

This thesis demonstrates the working principle of automatic scaling of a computer system in a simulated environment. It shows how to set up a "private cloud" on one of the cloud service providers and how machine learning techniques are incorporated into that system to predict web server load.

The results of the model show that it is possible to simulate automatic scaling services in the cloud on local ("on-premises") servers.

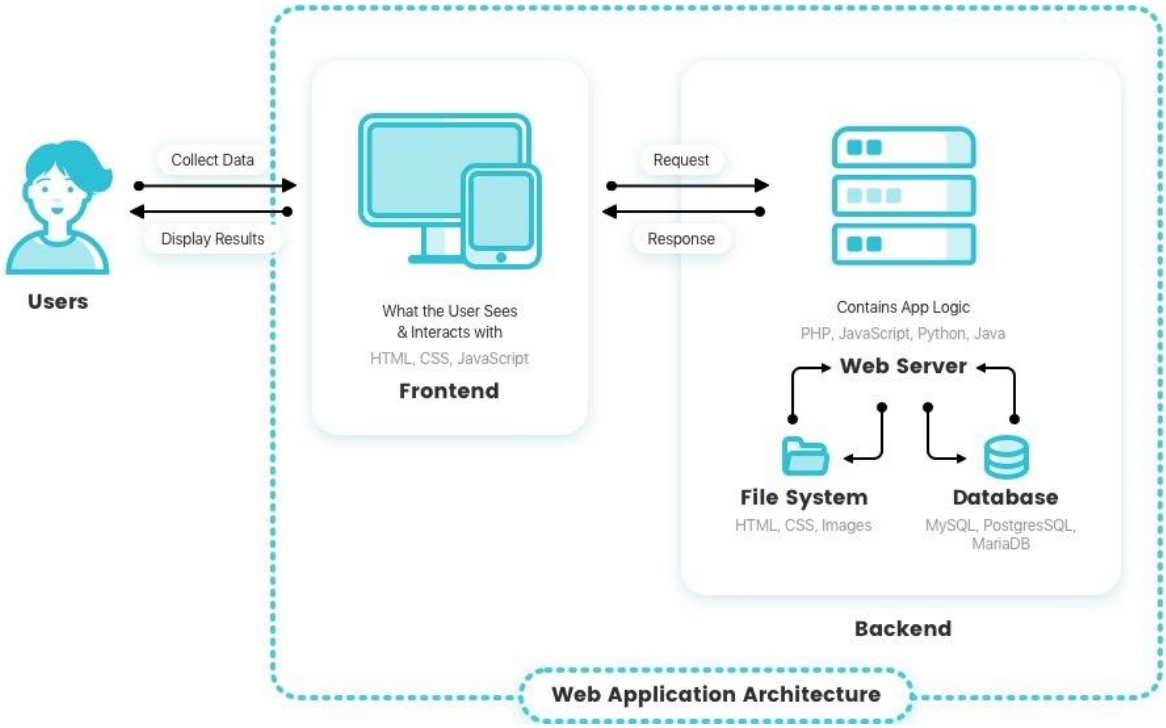
Keywords: cloud computing, machine learning, servers, scaling, server load, XGBoost, R2 score, mean square error

SADRŽAJ

1. Uvod

Aplikacija je računalni program dizajniran za obavljanje specifičnih zadataka njenim korisnicima [1]. Ona obavlja određene funkcije koje su njeni autor programirali s ciljem olakšanja određenih zadataka svojim korisnicima ili zabave. Aplikacija može biti uređivač teksta, društvena mreža, igrica, alat za programiranje, stranica za strujanje medija (eng. „*streaming*“) itd. Sastoji se od izvornog koda koji je preveden u strojni kod i izvodi se na računalu. Današnje aplikacije uglavnom su na internetu, odnosno većinom su to web aplikacije. Web aplikacija je aplikacija koja se izvodi na poslužitelju koji je povezan na Internet.

Slika 1. prikazuje općenitu arhitekturu web aplikacije:



Slika 1. Web stranica i web aplikacija. Izvor: [2]

Poslužitelj je računalo koje se sastoji od sklopovlja i softvera koji se na njemu pokreće. Poslužitelji se razlikuju od osobnih računala, najčešće, po svojoj namjeni i specijalnom sklopovlju koji je dizajniran da radi neprekidno [3]. Poslužitelji su računala sa specifičnom namjenom: posluživanje web aplikacije, posluživanje baze podataka, datotečni poslužitelj, autentifikaciji poslužitelj, poslužitelj virtualnih radnih površina za udaljeni pristup i dr.

Da bi tu namjenu mogli pružati neprekidno, bez gašenja, najčešće se ugrađuju posebni tvrdi diskovi, besprekidna napajanja i memorija (eng. „*Error correction code random access memory*“, ECC RAM).

Poslužitelji su, obično, složeni u stalcima (eng. „*rack*“) i povezani su na mrežu najboljom dostupnom vezom. Slika 2. prikazuje poslužitelje u stalku:

Slika 2. Poslužitelji u stalku. Izvor: [4]

Međutim, na svako računalo može se instalirati poslužiteljski softver, tako da vrijedi relacija svako računalo može biti poslužitelj i svaki poslužitelj može biti računalo.

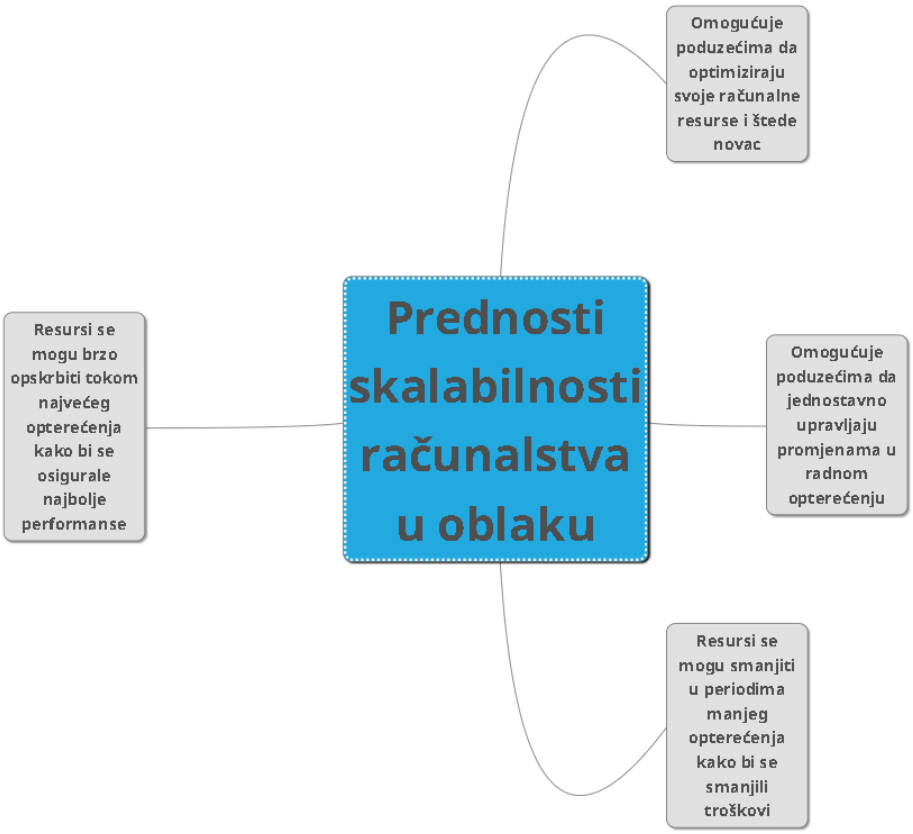
Virtualizacija je tehnologija koja omogućuje kreiranje više računalnih okruženja (hardverskih i softverskih) koristeći resurse računala, poslužitelja, domaćina [5]. Virtualizacijom možemo dobiti potpuno računalo koje se sastoji od virtualnog, emuliranog sklopovlja unutar već postojećeg računala. Virtualna računala dijele resurse s domaćinom, a domaćin na sebi pokreće sustav, ili softver, koji upravlja virtualnim računalima. Hipervizor je softver koji se koristi za kreiranje i upravljanje virtualnim računalima. On može biti direktno pokrenut na domaćinskom hardveru (tip 1, eng. „*bare-metal hypervizor*“) ili softver koji se pokreće na domaćinovom operacijskom sustavu (tip 2, eng. „*hosted hypervizor*“) [5].

Okolina, u kontekstu računalnih aplikacija, predstavlja sav softver, računalne resurse, konfiguracijske datoteke, biblioteke, parametre i postavke sustava koji su potrebni da bi se aplikacija uspješno izvodila.

Pomoću virtualizacije možemo uspostaviti različite okoline za rad i razvoj neke aplikacije, bez promjene postavki domaćinskog računala.

Oblak se odnosi na poslužitelje kojima se pristupa putem Interneta, kao i na sav softver i baze podataka koji se izvode na tim poslužiteljima [6]. Oblak, u kontekstu pružatelja računalnih usluga u oblaku poput Amazona, sastoji se od gotovih usluga poput baze podataka u oblaku, računalnih instanci u oblaku, upravljanje domenama u oblaku, praćenje stanja poslužitelja (eng. „*monitoring*“) u oblaku i brojnih drugih. Sve navedene usluge mogu se rekreirati na lokalnom serveru s jednakim funkcionalnostima i za manji novčani iznos ukoliko postoji tehničko znanje za izradu istog.

Skaliranje u oblaku, odnosi se na mogućnost infrastrukture u oblaku da povećava ili smanjuje resurse kako bi se zadovoljile promjene zahtjeva radnog opterećenja. [7]. Skaliranje u kontekstu posluživanja web aplikacija predstavlja povećanje ili smanjenje broja web poslužitelja ovisno o opterećenju i broju zahtjeva na sadržaj web aplikacije u određenom trenutku. Slika 3. prikazuje neke od prednosti skaliranja u oblaku:



Slika 3. Prednosti skaliranja. Izvor: [7]

Automatizacija u oblaku je skup procesa, dobrih praksi i alata za smanjivanje vremena sistemskih inženjera potrebnog za konfiguriranje, upravljanje i optimizaciju resursa i servisa u oblaku [8]. U situacijama kada je potrebno, u što kraćem roku, osposobiti okolinu za određenu aplikaciju, sistemski inženjeri često koriste alate i skripte za automatizaciju. Automatizacijski

alati i skripte, predstavljaju definiciju okoline, skup uputa i naredbe, koje sustav izvodi umjesto sistemskog inženjera. Alati mogu izvoditi operacije na više poslužitelja istovremeno, što smanjuje ukupno vrijeme uspostave okoline u odnosu na vrijeme koje bi inženjer potrošio ručnim upisivanjem naredbi. Primjeri alata su: Ansible, Chef, Puppet, Jenkins i dr.

1.1. Motivacija

Ovaj rad prikazati će način za uspostavu svih, gore navedenih, koncepata na lokalnom poslužitelju s ciljem simulacije vlastitog oblaka nad kojim imamo potpunu kontrolu.

Cilj diplomskog rada prikazati je da sve gotove funkcije koje pružatelji usluga u oblaku naplaćuju, svaki sistemski inženjer može uspostaviti i lokalno na svojim poslužiteljima. Pri tome poslodavcu, klijentu ili aplikaciji smanjuje troškove a pruža slične, iste ili bolje usluge, u ovisnosti od tehničkog znanja i potrebnih funkcionalnosti.

Prednost oblaka je da upravo ta gotova rješenja na lak, razumljiv i brz način uspostavljaju željenu okolinu ali uz značajne troškove, dok je predloženo rješenje financijski održivije za male obrtnike i mala poduzeća.

U diplomskom radu uspostaviti će se simulirano okruženje oblaka u kojem će se primijeniti metode strojnog učenja za predviđanje skaliranja predložene arhitekture. Budući da stvarni produkcijski sustav, zahtjeva puno dodatnih koraka, alata, konfiguracije i sigurnosnih aspekata, za potrebe diplomskog rad ograničit ćemo se samo na osnovne funkcionalnosti.

1.2. Struktura rada

Diplomski rad podijeljen je u sedam poglavlja. U prvom poglavlju će se objasniti koncepti koje simuliramo i alati i softveri koje koristimo. U drugom poglavlju prikazati će se arhitektura simuliranog oblaka, potom i samo kreiranje istog. Treće poglavlje opisuje razvoj postupaka strojnog učenja za predikciju opterećenja poslužitelja kao i integraciju tehnike sa skriptama za automatsko skaliranje. Na kraju rada dati će se uvid u rezultate odabrane tehnike te će se u posljednjem poglavlju iznijeti zaključci.

2. Koncepti

2.1. Računalstvo u oblaku

Računalstvo u oblaku predstavlja isporuku računalnih servisa: poslužitelja, pohrane, baza podataka, mreže, softvera, analitike i inteligencije putem Interneta [9].

Neke od prednosti korištenja računalnih usluga u oblaku uključuju:

- Automatsku detekciju potrebe za skaliranjem i automatsko skaliranje,
- Veću brzinu usluge radi bolje povezanosti i opreme u podatkovnim centrima,
- Veću sigurnost radi dediciranih timova koji se bave sigurnosnim pitanjima,
- Pouzdanost (garancija dostupnosti 99.999% i veća),
- „Plati kako koristiš“ princip naplaćivanja resursa ,
- i dr.

Oblak se dijeli u tri glavne kategorije: javni, privatni i kombinirani oblak [9]. Javni oblak je usluga računalstva u oblaku dostupna preko interneta, putem određenog davatelja usluge u oblaku (npr. Amazon Web Services [10], Microsoft Azure [11], Cloudflare [12] i dr.). Privatni oblaci su oni oblaci u vlasništvu određenog poduzeća koje isto koristi isključivo za svoje potrebe. Kombinirani su oblaci oni koji koriste privatni oblak unutar nekog poduzeća uz javni oblak kojeg im ustupa davatelj usluge u oblaku.

Računalni servisi, koji se nude putem oblaka, generalno se dijele u tri kategorije [9]:

- Infrastruktura kao servis (eng. „*Infrastructure as a Service*“, IaaS),
- Platforma kao servis (eng.“*Platform as a Service*“, PaaS),
- Softver kao servis (eng. „*Software as a Service*“, SaaS).

Infrastruktura, kao servis, predstavlja pružanje računalnih resursa putem Interneta. Resursi mogu biti: dedicirani poslužitelji, virtualni strojevi, IP adrese, prostori za pohranu velike količine podataka. Primjeri IaaS su Amazon EC2 instance [13], Hetzner VPS [14], Azure VM [15], DigitalOcean droplets [16] itd.

Platforma kao servis predstavlja pružanje i brigu za okoline pomoću kojih se izvode određene aplikacije. Kod takvih usluga, davatelj se brine o računalnim resursima i pratećem softveru, a korisnik se brine o svojoj aplikaciji. Primjeri PaaS su: Amazon Web Services Lambada [17], Oracle Cloud Platform [18], Red Hat OpenShift [19] i dr.

Softver, kao servis, predstavlja korištenje gotove usluge s već spremnom infrastrukturom u pozadini koju krajnji korisnik ne vidi. SaaS je najrašireniji oblik oblaka među korisnicima. Primjeri SaaS su: Netflix [20], Slack [21], Zoom [22], Adobe Creative Cloud [23], Google Suite [24] i dr.

Slika 4. prikazuje primjere SaaS servisa.

Slika 4 Primjeri SaaS. Izvor: [25]

2.2. Skaliranje računalnih resursa u oblaku

Skaliranje se odnosi na mogućnost povećanja ili smanjenja IT resursa po potrebi kako bi se pratila mijenjajuća potražnja za resursima [26].

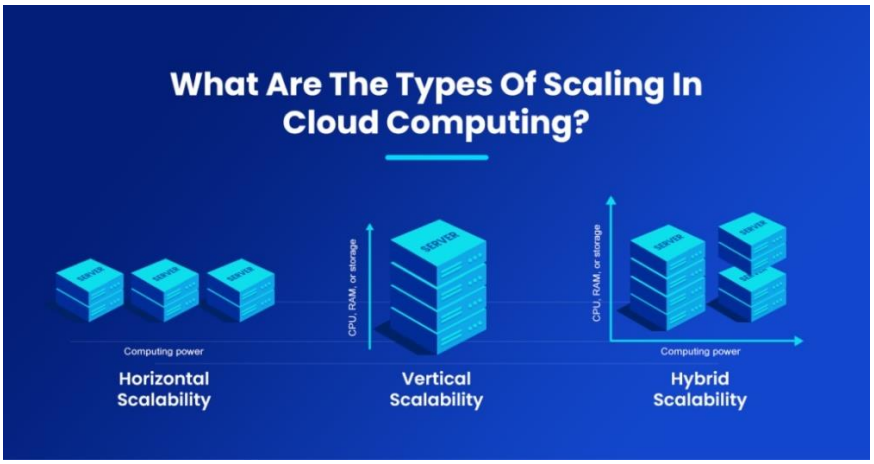
Prednosti skaliranja su:

- Smanjenje opterećenja nad poslužiteljima,
- Povećanje vijeka trajanja poslužitelja (zbog manjeg opterećenja),
- Mogućnosti odgovora na katastrofu (pad poslužitelja),
- Brzina odgovora na povećan broj zahtjeva za određenim resursom.

Skaliranje u oblaku predstavlja sposobnost računalne infrastrukture u oblaku da dinamički prilagodi resurse za praćenje povećanja potražnje za resursima [27]. Resurse možemo skalirati na tri načina: po horizontali, po vertikali ili hibridno [27].Resurse možemo skalirati na tri načina: po horizontali, po vertikali ili hibridno [27].

Horizontalnim skaliranjem (eng. „*scale-out*“) smatra se dodavanje identičnih poslužitelja jedan uz drugi. **Vertikalnim skaliranjem** (eng. „*scale-up*“) smatra se dodavanje boljeg i jačeg

sklopovlja u postojeću infrastrukturu. A **hibridnim skaliranjem** smatra se dodavanje više poslužitelja, boljih specifikacija (kombinacija). Slika 5. ilustrira tipove skaliranja.

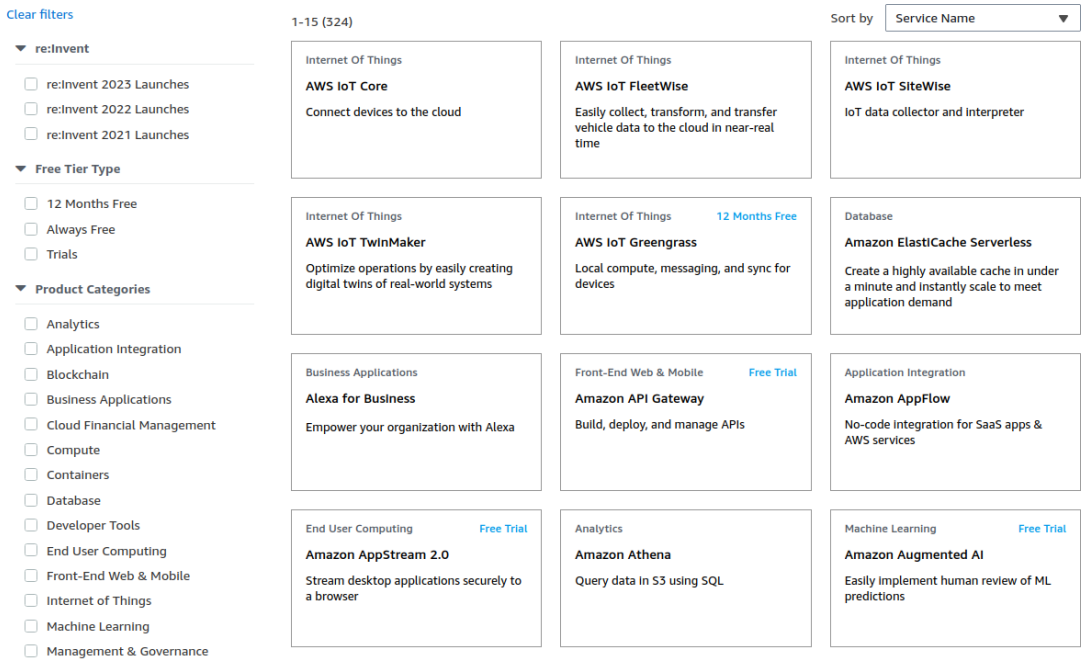


Slika 5. Tipovi skaliranja. Izvor: [27]

2.3. Amazon Web Services

Najpoznatiji i najpopularniji davatelj usluga u oblaku je Amazon, konkretno eng. („*Amazon Web Services*“, AWS) [28]. Dugotrajnim razvojem doveden je do razine da može biti jedna platforma za sve računalne potrebe bilo kojeg poduzeća. Opis svih servisa koje AWS pruža prelazi okvire diplomskog rada.

Neki od servisa su: automatsko skaliranje, AWS Lambda, Amazon Elastic Cache, Amazon virtualni privatni oblak, Amazon Elastic Compute Cloud instance, Amazon servis jednostavne pohrane (S3), Amazon CloudFront i brojni drugi. Slika 5. prikazuje samo mali dio servisa dostupnih na AWS Internetskim stranicama.



Slika 6. AWS servisi. Izvor: [29]

U diplomskome radu simuliraju se neki od Amazonovih servisa: Amazon elastični računalni oblak (eng. „*Amazon Elastic Compute Cloud*“, EC2), Amazon servis relacijske baze podataka (eng. „*Amazon Relation Database Service*“, RDS) i Amazon elastično balansiranje opterećenja (eng. „*Amazon Elastic Load Balancing“*, ELB).

EC2 instance su virtualni strojevi, čije specifikacije možemo birati pri kreiranju, s javnom IP adresom, pripremljenim vatrozidom, mogućnošću sigurnosnog kopiranja i nadzorom rada.

RDS instance su virtualni strojevi koji pokreću jedan od sustava za upravljanje bazama podataka, gdje korisnik ne može pristupiti operacijskom sustavu instance i ne može direktno mijenjati konfiguraciju kroz datoteke, već to radi kroz Amazon sučelja (naredbeno sučelje, web konzola i dr.).

Servis ELB je također poslužitelj, čijem se sustavu i konfiguraciji, ne može pristupiti, već se kroz Amazon sučelja dodaju instance ili servisi na koje se usmjerava promet.

2.4. Ostali pružatelji usluga u oblaku

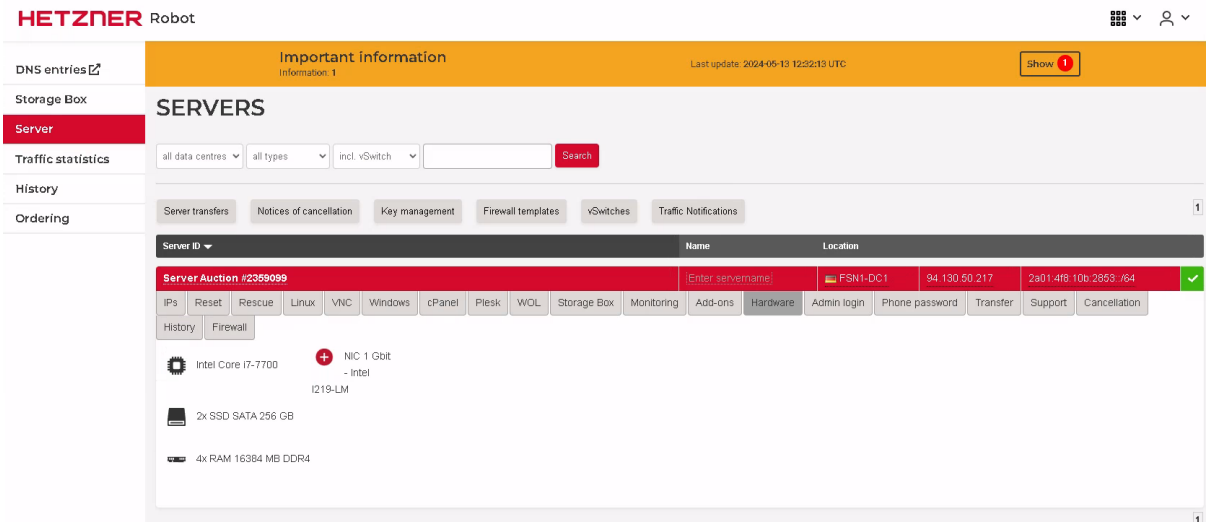
Osim Amazon web servisa, postoji još puno drugih pružatelja računalnih usluga u oblaku. Neki od najpoznatijih su: Microsoft Azure, Google Cloud Platform, Alibaba Cloud, Oracle Cloud, IBM Cloud, DigitalOcean i Akamai Cloud [30]. Većina ih nudi usporedive servise, koji se razlikuju po nazivu servisa, dodatnim mogućnostima, specifikacijama, zonama dostupnosti (regije u svijetu), cijeni i drugom. Primjerice, virtualni stroj se u Amazon terminologiji naziva Amazon elastični računalni oblak, u Azure-u se naziva Azure virtualni stroj, u DigitalOcean-u se naziva kapljica (eng. „*droplet*“) itd. Većina ih nudi usporedive servise, koji se razlikuju po

nazivu servisa, dodatnim mogućnostima, specifikacijama, zonama dostupnosti (regije u svijetu), cijeni i drugom. Primjerice, virtualni stroj se u Amazon terminologiji naziva Amazon elastični računalni oblak, u Azure-u se naziva Azure virtualni stroj, u DigitalOcean-u se naziva kapljica (eng. „*droplet*“) itd.

2.5. Hetzner

Hetzner [31] je jedan od davatelja računalnih usluga u oblaku. Osim oblaka nudi i usluge web hostinga, virtualne servere, domene, certifikate i dr. Poslužitelji su rasprostranjeni po Americi, Njemačkoj i Finskoj s podatkovnim centrima u gradovima: Nuremberg, Falkenstein, Tuusula, Ashburn, Virginia i Hillsboro [31]. Osim resursa, pruža i korisničku podršku te garantira dostupnost svojih usluga 24/7. Poslužitelji su rasprostranjeni po Americi, Njemačkoj i Finskoj s podatkovnim centrima u gradovima: Nuremberg, Falkenstein, Tuusula, Ashburn, Virginia i Hillsboro [31]. Osim resursa, pruža i korisničku podršku te garantira dostupnost svojih usluga 24/7.

Hetzner osim navedenog nudi i mogućnost zakupa rabljenih poslužitelja. Ti su poslužitelji prethodno naručeni za neke od njihovih klijenata, koji su tokom vremena postali nepotrebni istima. Jedan od ponuđenih rabljenih servera zakupljen je za postavljanje sustava koji se koristi u diplomskome radu. Na Slici 7. prikazane su specifikacije zakupljenog poslužitelja:



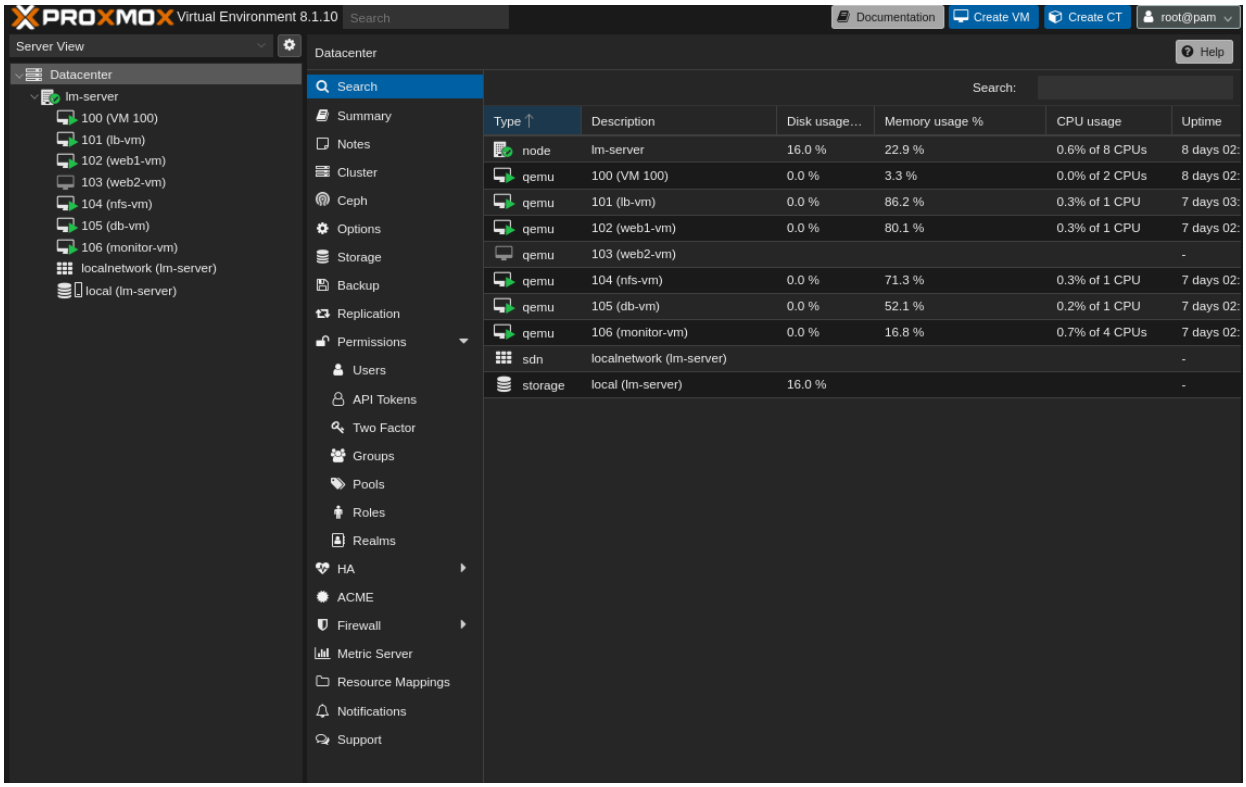
Slika 7. Zakupljeni poslužitelj

2.6. Proxmox virtualna okolina

Proxmox virtualna okolina (eng. „*Proxmox Virtual Environment*“, PVE) je potpuno rješenje, otvorenog koda, za virtualizaciju poduzeća koje integrira KVM hipervizor, Linux kontejnere, softverski definiranu pohranu i mrežu u jednoj platformi [32]. Proxmox VE je operacijski sustav, koji je ujedno i tip 1 hipervizor, koji nam omogućuje kreiranje virtualnih strojeva, virtualne mreže i virtualnih diskova. Pomoću ugrađenog softvera, možemo složiti više-

serversku arhitekturu s pod mrežama i pohranom na jednom poslužitelju – domaćinu. Dolazi s web sučeljem putem kojeg mišem možemo odabrati i kreirati potrebne resurse, a moguće je koristiti i sučelje putem naredbenog retka. Proxmox VE je operacijski sustav, koji je ujedno i tip 1 hipervizor, koji nam omogućuje kreiranje virtualnih strojeva, virtualne mreže i virtualnih diskova. Pomoću ugrađenog softvera, možemo složiti više-serversku arhitekturu s pod mrežama i pohranom na jednom poslužitelju – domaćinu. Dolazi s web sučeljem putem kojeg mišem možemo odabrati i kreirati potrebne resurse, a moguće je koristiti i sučelje putem naredbenog retka.

Proxmox nudi i napredne opcije kreiranja vatrozida, sigurnosnog kopiranja, klasterizacije s drugim Proxmox VE, replikacije, kreiranja klastera visoke dostupnosti (eng. „*High Availability cluster*“), dozvolama, dodatnim korisničkim profilima i dr. vidljivo na Slici 8.



Slika 8. Proxmox web sučelje

Prva stranica na web sučelju daje uvid u pokrenute virtualne strojeve, virtualnu pohranu i kontejnere. Pokazuje nam njihovo stanje, opterećenje memorije, diska i procesora, koliko dugo je pokrenut (eng. „*uptime*“) i koliko memorije domaćinskog poslužitelja svaki virtualni stroj koristi.

Proxmox će se, u kontekstu diplomskog rada, koristiti za kreiranje i serviranje arhitekture virtualnih strojeva koji će biti instalirani na način da simuliranju osnovne funkcionalnosti usluga u oblaku.

2.7. Strojno učenje

Strojno učenje je područje umjetne inteligencije u kojem algoritmi uče modele iz podataka [33, 34].

Postoje tri glavne vrste strojnog učenja:

- Nadzirano učenje,
- Nenadzirano učenje,
- Polu-nadzirano učenje.

Nadzirano učenje uključuje učenja modela s podacima koji sadrže i označene izlazne informacije – npr. klase [33].

Nenadzirano učenje uključuje učenje modela iz ulaznih podatke bez oznaka klase. Najčešće se koriste algoritmi klasteriranja ili grupiranja podataka. Metode uče iz prethodnih neoznačenih, podataka te ih zatim grupiraju prema sličnostima (ili prema nedostatku sličnosti) [33].

Polu-nadzirano učenje smješteno je između nenadziranog i nadziranog učenja. Najprije se uči iz malog skupa označenih podataka (kao u nadziranom učenju). Naučenim modelom označi se dio neoznačenih podataka, eventualno otklone pogreške te se proširenim skupom podatka uči novi model [33]. Postupak učenja nad vremenskim serijama pripada vrsti strojnog učenja [35].Postupak učenja nad vremenskim serijama pripada vrsti strojnog učenja koja se naziva analiza vremenskih serija [35].

2.8. Tehnike strojnog učenja pogodne za problem skaliranja

Analizom i praćenjem opterećenja web poslužitelja nastaje skup podataka koji sadrži sljedeće podatke: vrijeme zahtjeva, broj zahtjeva, postotak opterećenosti računalnih resursa u tom vremenskom trenutku. Nastali skup podataka je vremenska serija koji sadrži događaje (zahtjeve) u računalnom sustavu.

Vremenska serija je skup promatranja x_1, koja se bilježe u specifičnom vremenu t [36]. **Diskretna vremenska serija** je ona u kojoj je skup vremena T_0 koji se promatraju diskretan skup [36]. **Kontinuirane vremenske serije** nastaju kada se promatranja kontinuirano bilježe tokom nekog vremenskog intervala [36]. **Kontinuirane vremenske serije** nastaju kada se promatranja kontinuirano bilježe tokom nekog vremenskog intervala [36].

Skup podataka koji će se prikupljati u diplomskome radu, o opterećenjima poslužitelja je multivarijatna vremenska serija s visokom varijacijom u podacima. Multivarijatna serija je ona u kojoj za određenu vremensku točku u skupu, promatrana vrijednost ovisi o više varijabli [37]. Slika 9. prikazuje najpoznatiji primjer vremenske serije – dionice:Slika 9. prikazuje najpoznatiji primjer vremenske serije – dionice:

Slika 9. Vremenska serija. Izvor: [38]

Analiza vremenskih serija je statistička metoda koja se koristi za analizu i interpretaciju sekvencijalnih podatkovnih točaka u nekom vremenu [39]. Tokom analize vremenskih serija promatramo poredak podatkovnih točaka u skupu podataka kako bi otkrili trend, sezonalnost, uzorke i ovisnosti u podacima. **Trend** promatra rast ili pad vrijednosti u podacima tokom dužeg perioda u skupu. **Sezonalnost** je svojstvo vremenskih serija u kojem podaci doživljavaju redovne i predvidljive promjene koje se događaju svake godine [40] (dan, mjesec, godina, desetljeće; ovisno o promatranom periodu). Neki od algoritama za analizu vremenskih serija su: ARIMA, STL, VAR, GAM, Auto-ARIMA, ETS [41], Prophet [42], Nixtla [43] i dr.Neki od algoritama za analizu vremenskih serija su: ARIMA, STL, VAR, GAM, Auto-ARIMA, ETS [41], Prophet [42], Nixtla [43] i dr.

2.9. XGBoost regresija

XGBoost (Gradijentno pojačanje) je optimizirana i distribuirana biblioteka za povećanje gradijenata postupcima nadziranog učenja [44]. Pojačanje (eng. „*boosting*“) je metoda za kreiranje ansambla [45]. Ansambl (eng. „*Ensambles*“) je kombinacija jednostavnih pojedinačnih modela koji zajedno stvaraju snažniji novi model [45].Pojačanje (eng. „*boosting*“) je metoda za kreiranje ansambla [45]. Ansambl (eng. „*Ensambles*“) je kombinacija jednostavnih pojedinačnih modela koji zajedno stvaraju novi model, običajno boljih perfomansi od početnog modela [45].

Pojačanje započinje prilagođavanjem početnog modela (npr. stabla ili linearne regresije) podacima. Zatim se izrađuje drugi model koji se fokusira na točno predviđanje slučajeva u kojima prvi model ima lošu izvedbu. Očekuje se da će kombinacija ova dva modela biti bolja od svakog pojedinačnog modela [45]. Gradijentno pojačanje je vrsta pojačanja strojnog učenja. Oslanja se na intuiciju da najbolji mogući sljedeći model, u kombinaciji s prethodnim modelima, smanjuje ukupnu pogrešku predviđanja [45].Gradijentno pojačanje je vrsta strojnog učenja, koja se oslanja se na intuiciju da najbolji mogući sljedeći model, u kombinaciji s prethodnim modelima, smanjuje ukupnu pogrešku predviđanja [45].

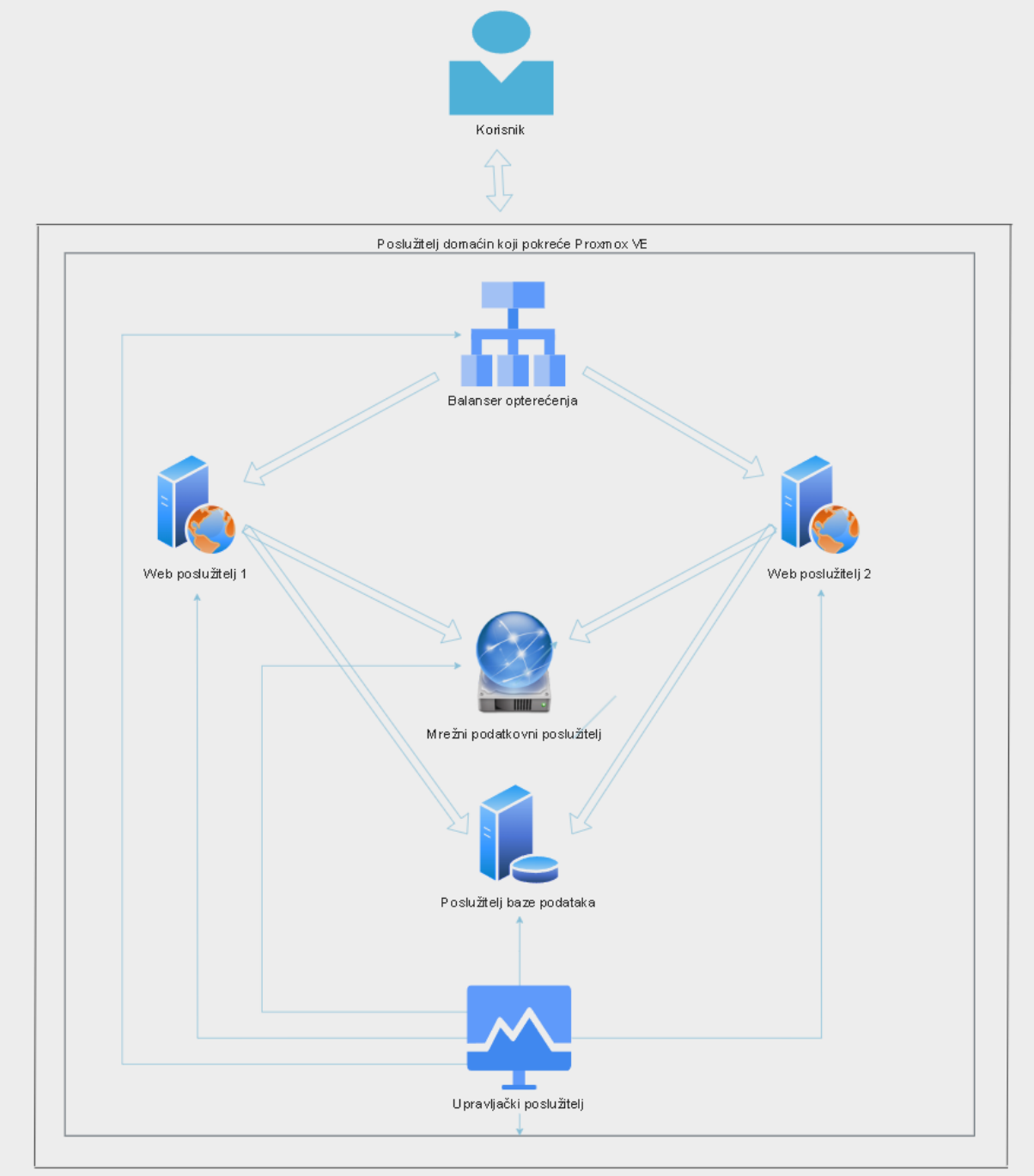
XGBoost omogućava paralelno učenje stabala (poznato i kao GBDT - Gradijentno pojačana stabla odlučivanja (eng. “*Gradient-boosted decision trees*”, GBM – Algoritam gradijentnog pojačanja (eng. “*Gradient boosting algorithm*”)) koje rješava mnoge probleme učenja modela iz poslovnih podataka na brz i točan način [44].

XGBoost ima nekoliko ključnih prednosti, koje ga čine izvrsnim izborom za predviđanje vremenskih serija [46]:

- Rješavanje nelinearnih zavisnosti,
- Otkriva važnost značajke (eng. „*Feature importance*“),
- Uključuje tehnike regularizacije kako bi se smanjilo prekomjerno prilagođavanje (eng. „*overfitting*“),
- Rješavanje problema oskudnosti podataka, odnosno podatak s vrijednostima koje nedostaju (eng. „*null values*“) te ekstremima,
- Koristi se i za klasifikacijske i regresijske probleme učenja.

3. Arhitektura testnog okruženja

Za diplomski rad predložena je arhitektura virtualnih strojeva koju se može skalirati. Arhitekturu je moguće prilagoditi da jednostavno prikuplja podatke o opterećenju. Za simulaciju opterećenja na arhitekturi izvodi se i aplikacija iz poslovne domene, za koje se prikupljaju podatci o opterećenju potrebni za postupke strojnog učenja. Predložena arhitektura simulacijskog sustava za prikaz automatskog skaliranja web poslužitelja u oblaku putem strojnog učenja prikazana je na Slici 10.



Slika 10. Arhitektura sustava

Sustav se sastoji od:

- Poslužitelja domaćina koji predstavlja davatelja usluga u oblaku,
- Dva web poslužitelja koji predstavljaju virtualne strojeve u oblaku,
- Poslužitelja baze podataka koji predstavlja relacijsku bazu podataka u oblaku,
- Poslužitelja za balansiranje opterećenja koji predstavlja balansiranje opterećenja u oblaku,
- Poslužitelja koji pokreće mrežni datotečni sustav koji predstavlja pohranu u oblaku,
- Poslužitelja za nadzor koji pokreće alat za nadzor poslužitelja i model strojnog učenja koji predstavlja alate za nadzor sustava u oblaku.

Sustav je koncipiran na način da zahtjev kojeg upućuje korisnik prvo dolazi do mrežnog sučelja sa javnom adresom (u terminologiji oblaka Internet vrata (eng. „*Internet Gateway*“). Potom se zahtjev preusmjerava na internu mrežu unutar oblaka (u terminologiji oblaka virtualnu privatnu mrežu) preko koje dolazi do balansera opterećenja. Nakon toga, balanser provjerava koji su web poslužitelji dostupni i rotacijskim algoritmom (eng. „*Round Robin*“) odlučuje na koji će poslužitelj poslati zahtjev.

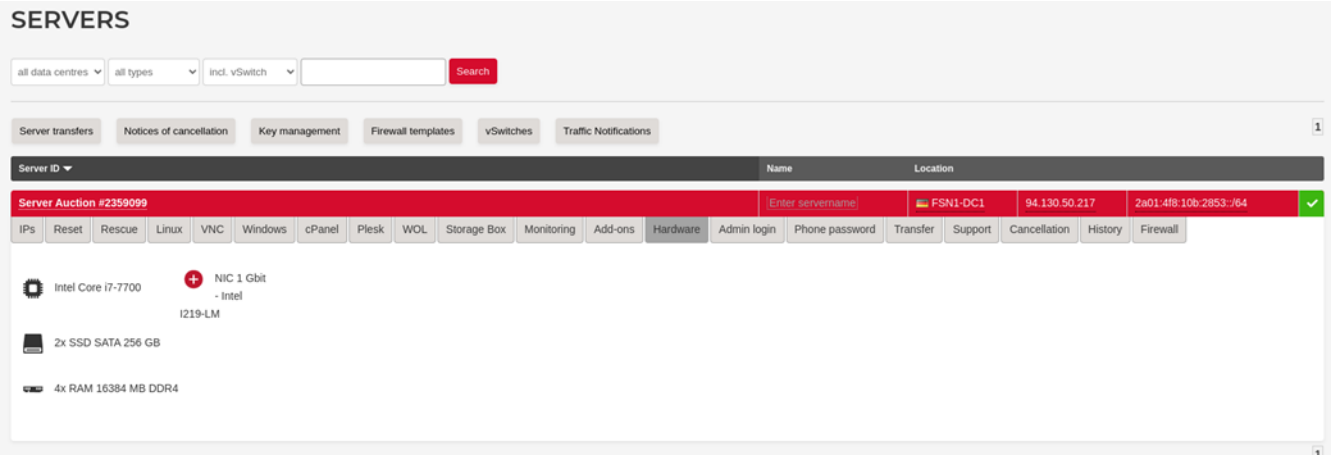
Zahtjev tada dolazi do jednog od web poslužitelja. Web poslužitelj pristupa izvornom kodu aplikacije, koji se nalazi na mrežnom datotečnom sustavu, te s njega učitava sučelje (eng. „*frontend*“) i pozadinski kod (eng. „*backend*“). Kod zatim poziva bazu podataka, dohvaća podatke iz nje i obrađuje podatke. Na kraju se podaci vraćaju natrag korisniku istim putem kako su i došli.

Paralelno s ovim procesom upravljački poslužitelj bilježi podatke o opterećenju web poslužitelja, izvodi model strojnog učenja za predikciju opterećenja poslužitelja, skripte za automatsko skaliranje i alat za nadzor poslužitelja (eng. „*monitoring*“). Predloženi sustav simulira privatni oblak.

4. Implementacija privatnog oblaka

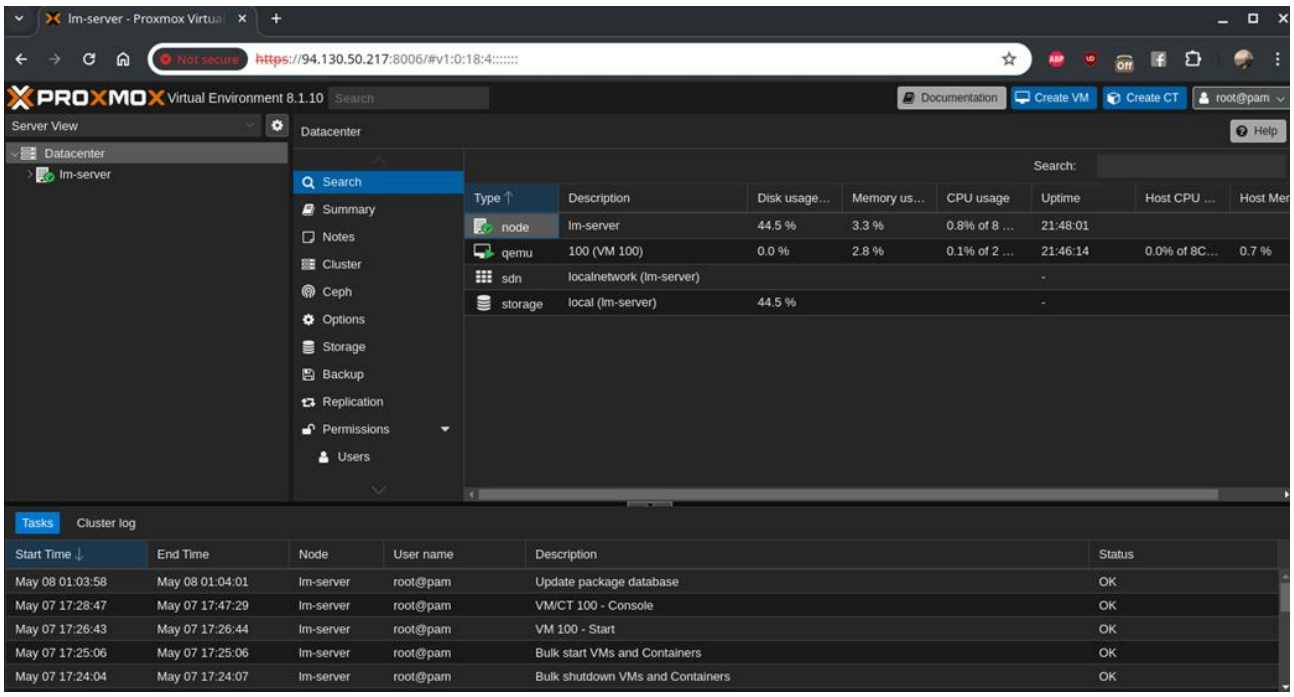
4.1. Domaćinski poslužitelj - sklopovlje

Za potrebe testiranja i demonstracije sustava automatskog skaliranja, zakupljen je rabljeni poslužitelj na pružatelju računalnih usluga u oblaku Hetzner. Hetzner je odabran zbog, u vrijeme pisanja rada, najboljeg omjera cijene i hardverskih specifikacija poslužitelja. Zakupljen je poslužitelj sa statičnom IP adresom, 64 GB računalne memorije, 512GB SSD pogonom i Intel i7 procesorom. Slika 11. prikazuje specifikacije poslužitelja.



Slika 11. Specifikacije poslužitelja

Po zakupu poslužitelja instaliran je operacijski sustav Proxmox Virtual Environment, tip 1 hipervizor temeljen na Debian Linux distribuciji. Slika 12. Prikazuje web sučelje Proxmox-a.

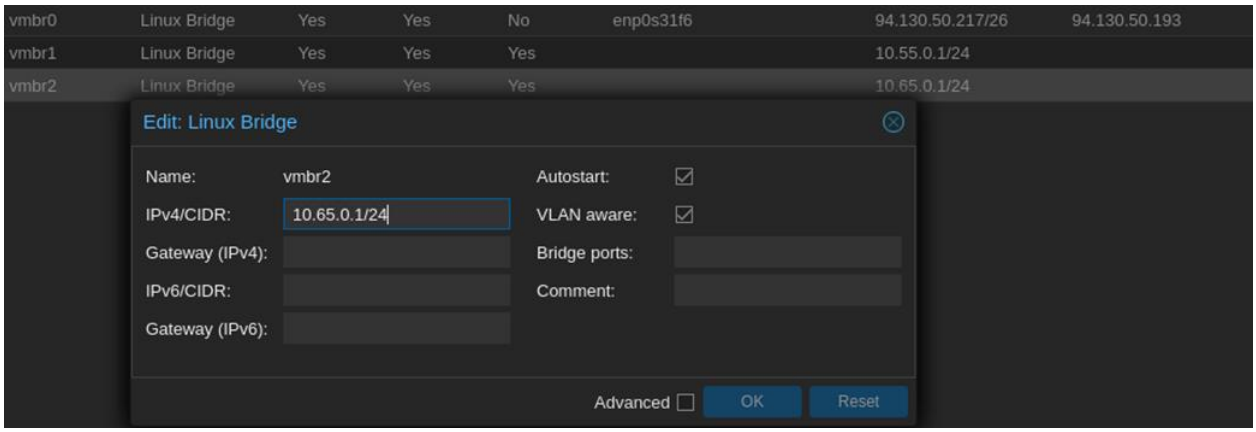


Slika 12. Web sučelje Proxmox-a

Nakon instalacije Proxmox sustava, kreirane su tri virtualne mreže tipa most[1]:

- Prva virtualna mreža "vmbr0" veže se na mrežno sučelje s vanjskom IP adresom,
- Druga virtualna mreža "vmbr1" nije kreirana za potrebe ovog rada,
- Treća virtualna mreža "vmbr2" je podmreža s rasponom adresa 10.65.0.1/24. Unutar te mreže nalazit će se virtualni poslužitelji potrebni za demonstraciju automatskog skaliranja sustava.

Na slici 13. prikazan je izbornik za konfiguraciju virtualne mreže.



Slika 13. Konfiguracija virtualne mreže

Zatim je, na glavnom poslužitelju, koji pokreće Proxmox, uključeno prosljeđivanje IPv4 adresa i usmjeravanje paketa nakon dolaska do vatrozida (eng. „*postrouting*") za sve pakete na sučelju vmbr0 unutar lanca pravila NAT. Mrežno preslikavanje adresa (eng. „*Network Address Translation*", NAT) djeluje kao digitalni plašt za uređaje na privatnoj mreži skrivajući njihove stvarne IP adrese i sprječavajući izravan pristup s interneta [47].

```
root@lm-server ~ # echo 1 > /proc/sys/net/ipv4/ip_forward
root@lm-server ~ # cat /proc/sys/net/ipv4/ip_forward
1
root@lm-server ~ # iptables -t nat -A POSTROUTING -o vmbr0 -j MASQUERADE
```

Iduće su prikazani potrebni koraci za kreiranje virtualnih strojeva i instalacije operacijskih sustava. Za potrebe diplomskog rada odabran je operacijski sustav Debian GNU/Linux.

[1] Mreža tipa most je virtualni preklopnik (eng. „*switch*") koji usmjerava virtualne goste, uređaje, mrežne kartice, do kartice fizičkog mrežnog sučelja domaćina. [67]

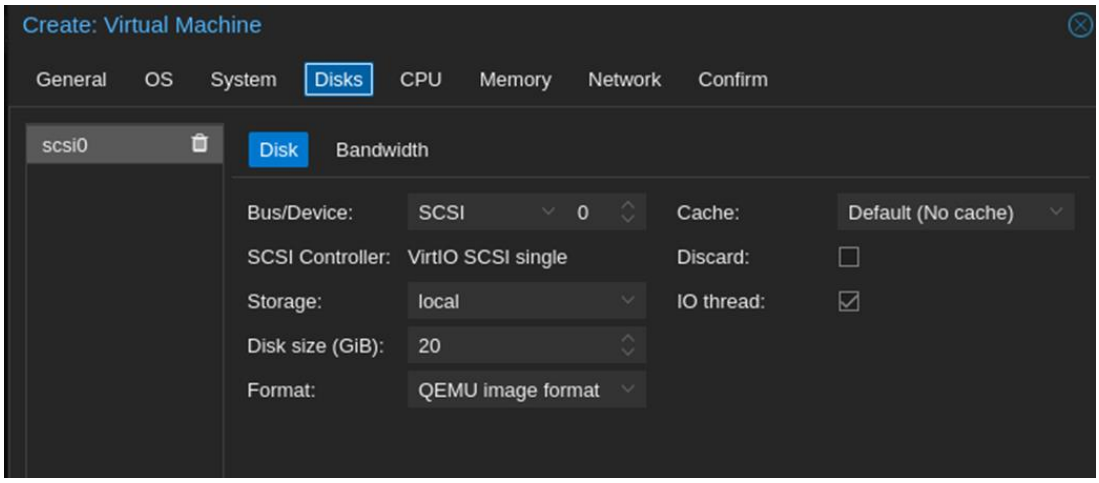
4.2. Kreiranje virtualnih strojeva i instalacija operacijskog sustava

Prvi korak je kreiranje virtualnog stroja u Proxmox sučelju.

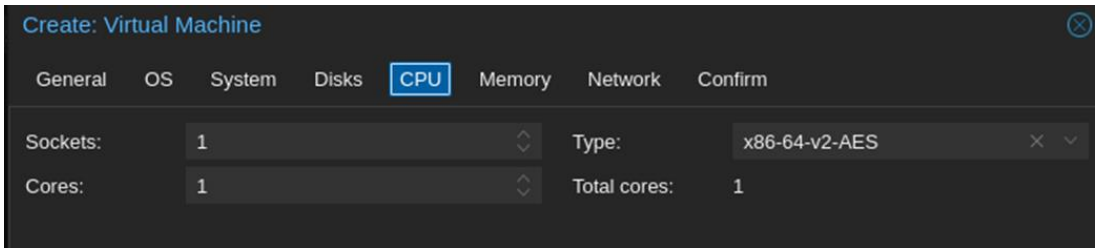
Kreiranje virtualnog stroja prikazano je na slikama, a sastoji se od slijedećih koraka:

- Odabira imena i ID-a stroja,
- Odabira datoteke s slikom instalacijskog CD-a sustava,
- Odabira vrste sustava,
- Odabira vrste sučelja tvrdog diska i njegove veličine,
- Odabira tipa procesora, broja procesora i jezgri,
- Odabira količine memorije,
- Odabira postavki mrežnog sučelja stroja.

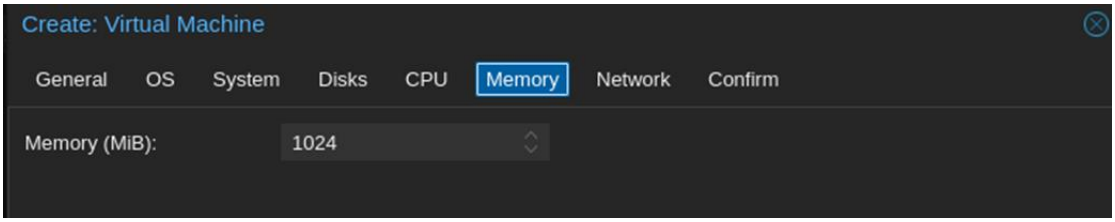
Slike 14., 15., 16. i 17. prikazuju kako kreirati virtualni stroj putem Proxmox web sučelja.



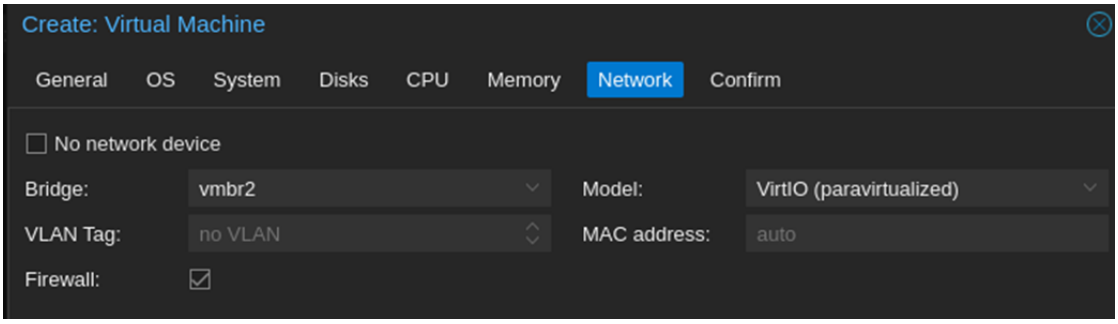
Slika 14. Virtualni stroj – pohrana



Slika 15. Virtualni stroj – procesor



Slika 16. Virtualni stroj – memorija



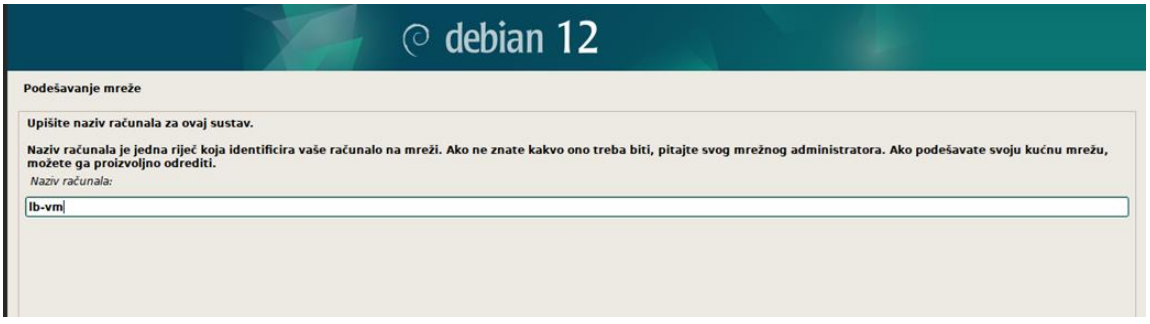
Slika 17. Virtualni stroj - mrežne postavke

Virtualni stroj za balansiranje opterećenja konfiguriran je s jednom procesorskom jezgrom, 1GB memorije, 20GB tvrdog diska i spojen na mrežu "vmbr2". Operacijski sustav instaliran je uz standardne postavke instalacije Debian GNU/Linux sustava, bez grafičkog sučelja. Obzirom da Proxmox nije konfiguriran da pokreće dinamičko dodjeljivanje IP adresa (eng. *„Dymanic Host Configuration Protocol*", DHCP), potrebno je ručno konfigurirati IP adrese i imenski poslužitelj (eng. "*nameserver*").

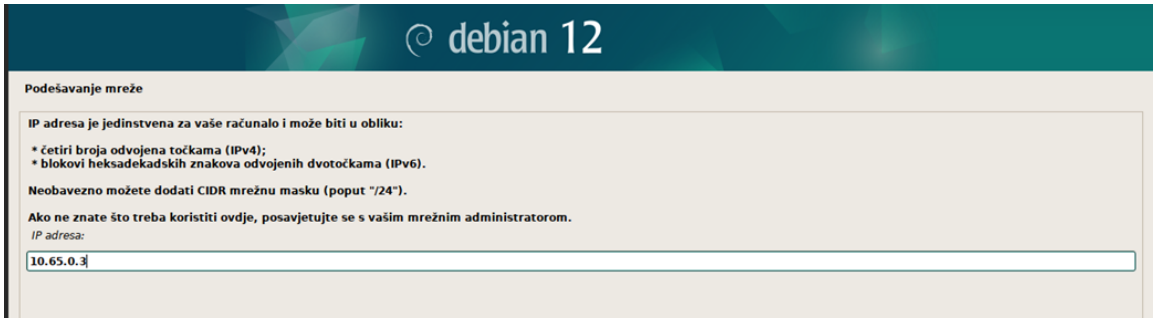
Tokom instalacije prebacujemo se na drugu virtualnu konzolu kombinacijom tipki CTRL + ALT + F3 i uređujemo datoteku /etc/resolv.conf s parametrima:

```
nameserver 10.65.0.1
nameserver 8.8.8.8
```

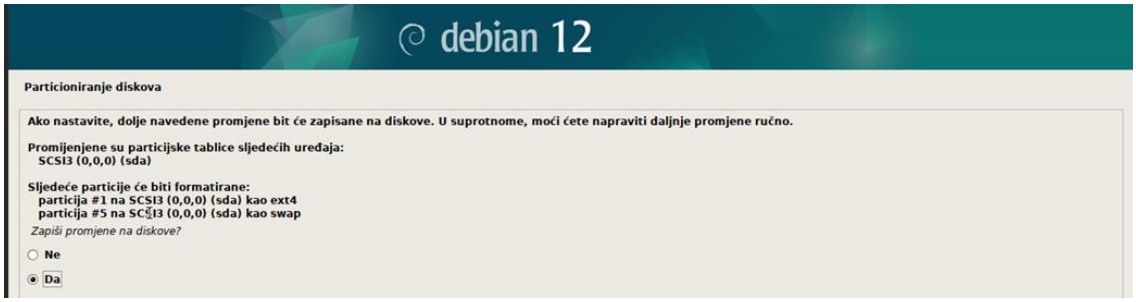
Konfiguracija instalacije operacijskog sustava prikazana je na Slikama 18., 19. i 20.



Slika 18. Konfiguracija sustava - ime poslužitelja



Slika 19. Konfiguracija poslužitelja - Mrežne postavke



Slika 20. Konfiguracija poslužitelja - particije

Nakon instalacije, ponovno konfiguriramo imenski poslužitelj, dodavanjem adresa najpoznatijih imenskih poslužitelja:

```
nameserver 10.65.0.1
nameserver 8.8.8.8
nameserver 1.1.1.1
nameserver 8.8.4.4
```

Zatim instaliramo **openssh** softver za spajanje na virtualni stroj putem SSH protokola i paket **sudo** kako bi mogli pokretati naredbe kao korijenski korisnik putem lokalnog korisnika.

```
root@lb-vm ~ # apt install openssh-server sudo
```

Nakon instalacije paketa dodajemo korisnika, kojeg smo kreirali tokom instalacije, u grupu sudo i omogućujemo korištenje lozinke prilikom spajanja SSH protokolom.

```
root@lb-vm ~ #/usr/sbin/usermod -aG sudo deb
root@lb-vm ~ # echo “PasswordAuthentication yes” > /etc/ssh/sshd_config
```

Sada se možemo spojiti putem SSH protokola na stroj:

```
root@lm-server ~ # ssh deb@10.65.0.3
The authenticity of host '10.65.0.3 (10.65.0.3)' can't be established.
ED25519 key fingerprint is SHA256:H7aroOLQVjCLGX/Y1QfxVokfOfAecCdGzUoaYNaIowE.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.65.0.3' (ED25519) to the list of known hosts.
deb@10.65.0.3's password:
Linux lb-vm 6.1.0-18-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.76-1 (2024-02-01) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
```

```
permitted by applicable law.
Last login: Wed May  8 15:53:30 2024
deb@lb-vm:~$
```

Analogno su postavljeni i ostali virtualni strojevi, koji su predviđeni u arhitekturi sustava (sukladno prikazu na Slika 10) .

4.3. Podešavanje virtualnog stroja za balansiranje opterećenja

Za balansiranje opterećenja koristi se softver HAProxy [48]. HAProxy je besplatan, vrlo brz i pouzdan obrnuti proxy koji pruža visoku dostupnost, balansiranje opterećenja i proxy poslužitelj za TCP i HTTP aplikacije [48].

```
root@lb-vm ~ # apt install haproxy
```

Balanser opterećenja konfiguriran je tako da sluša sve dolazne zahtjeve na vratima (eng. „*port*“) 80 te ih prosljeđuje na virtualne web servere prema shemi arhitekture. Pri čemu koristi algoritam Round-Robin za balansiranje opterećenja.

Konfiguracija je oblika:

```
frontend front
   bind *:80
   default_backend be

backend be
  balance roundrobin
  server be1 10.65.0.4:80 check
  server be2 10.65.0.5:80 check
```

Podešavamo da se HAProxy servis pokreće sa sustavom:

```
root@lb-vm ~ # systemctl enable haproxy
```

Zatim testiramo:

```
$ curl "http://localhost:80"
<html><body><h1>503 Service Unavailable</h1>
No server is available to handle this request.
</body></html>
```

HAProxy trenutno vraća status 503 jer web poslužitelji još nisu konfigurirani.

4.4 Podešavanje virtualnog stroja za pohranu

Virtualni stroj za pohranu služiti će kao mrežni datotečni sustav. Mrežni datotečni sustav (eng. “*Network File System*”, NFS) omogućava dijeljenje direktorija i datoteka s drugim računalima putem mreže [49].

Da bi podesili NFS, potrebno je instalirati paket nfs-kernel-server:

```
deb@nfs:~$ sudo apt install nfs-kernel-server
```

Zatim se, u konfiguracijsku datoteku, s putanjama koje će se dijeliti /etc/exports, dodaju putanje koje se dijele, IP adrese klijenata i prava pristupa putanji.

```
/home/deb/html 10.65.0.4(rw,sync,no_subtree_check)
10.65.0.5(rw,sync,no_subtree_check,no_root_squash)
/home/deb/access_log 10.65.0.4(rw,sync,no_subtree_check,no_root_squash)
10.65.0.5(rw,sync,no_subtree_check,no_root_squash)
/home/deb/sys_usage 10.65.0.4(rw,sync,no_subtree_check,no_root_squash)
10.65.0.5(rw,sync,no_subtree_check,no_root_squash)
```

Odabrana su prava čitanja, pisanja i sinkronizacije.

Dodaje se testna datoteka za testiranje sinkronizacije:

```
deb@nfs:~/html$ touch test
deb@nfs:~/html$ ls
test
```

Na oba web poslužitelja potrebno je instalirati NFS klijent:

```
deb@web1:~$ sudo apt install nfs-common
```

Zatim kreirati direktorij /var/www/html i direktorije /mnt/access_log i /mnt/sys_usage:

```
deb@web1:~$ sudo mkdir /var/www/
deb@web1:~$ sudo mkdir /var/www/html
deb@web1:~$ sudo mkdir /mnt/access_log
deb@web1:~$ sudo mkdir /mnt/sys_usage
```

I podesiti točke montiranja NFS-a na direktorije putem fstab datoteke:

```
10.65.0.6:/home/deb/html    /var/www/html    nfs    defaults    0 0
10.65.0.6:/home/deb/access_log /mnt/access_log nfs  defaults    0 0
10.65.0.6:/home/deb/sys_usage  /mnt/sys_usage  nfs  defaults    0 0
```

Nakon toga se može montirati NFS i testirati:

```
deb@web1:~$ sudo systemctl daemon-reload
deb@web1:~$ sudo mount -a
deb@web1:~$ ls /var/www/html/
test
```

U stvorenom direktoriju instalira se Pimcore Demo e-commerce softver [50]. Pimcore je platforma otvorenog koda koja agregira, obogaćuje i upravlja poslovnim podacima i pruža ažurna, konzistentna i personalizirana iskustva kupcima [51]. Sastoji se od niza alata i funkcionalnosti za vođenje elektroničkog poslovanja kao i svu programsku i administrativnu potporu za vođenje web trgovine. Osim navedenog, pruža već ugrađeno sučelje za web trgovinu, koju korisnik može prilagoditi svojim potrebama.

Pimcore ćemo koristiti kako bi naši web poslužitelji imali konkretnu web aplikaciju za serviranje na koju ćemo slati zahtjeve i putem koje ćemo stvarati opterećenje na poslužiteljima.

Da bi instalirali Pimcore, prvo moramo dodati repozitorij, instalirati PHP8.2, pomoćne pakete, i PHP-composer:

```
deb@web1:~$ sudo apt install software-properties-common ca-certificates lsb-release apt-
transport-https curl
deb@web1:~$ sudo sh -c 'echo "deb https://packages.sury.org/php/ $(lsb_release -sc) main" >
/etc/apt/sources.list.d/php.list'
deb@web1:~$ wget -qO - https://packages.sury.org/php/apt.gpg | sudo apt-key add -
```

```
deb@web1:~$ sudo apt install php8.2 php8.2-fpm php8.2-mysql php8.2-iconv php8.2-dom php8.2-
simplexml php8.2-gd php8.2-exif php8.2-fileinfo php8.2-mbstring php8.2-zip php8.2-intl
php8.2-opcache php8.2-curl php8.2-imagick


deb@web1:~$ php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('sha384', 'composer-setup.php') ===
'dac665fdc30fdd8ec78b38b9800061b4150413ff2e3b6f88543c636f7cd84f6db9189d43a81e5503cda447da73
c7e5b6') { echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('compos
er-setup.php'); } echo PHP_EOL;"
php composer-setup.php
php -r "unlink('composer-setup.php');"


deb@web1:~$ sudo mv composer.phar /usr/local/bin/composer
```

Isti softver treba instalirati i na drugom web poslužitelju.

Zatim se stvara Pimcore Demo projekt putem Composer-a:

```
deb@web1:/var/www/html$ COMPOSER_MEMORY_LIMIT=-1 composer create-project pimcore/demo.
```

Nakon provjere /var/www/html direktorija na drugom web serveru, zaključujemo da sadrži iste datoteke kao i prvi web server, odnosno da dijeljene datoteka putem mreže radi.

4.5. Podešavanje virtualnog stroja baze podataka

Virtualni stroj baze podataka pokreće MariaDB sustav za upravljanje bazama podataka [52]. Na njemu će biti baza podataka za Pimcore softver. MariaDB jedan je od najpopularnijih sustava otvorenog koda za upravljanje relacisjkim bazama podataka koju su razvili inicijalni autori MySQL-a [52].

Da bi se podesio sustav za upravljanje bazama podataka, potrebno je preuzeti paket mariadb-server:

```
deb@db:~$ sudo apt install mariadb-server
```

Zatim se inicijalizira SUBP:

```
deb@db:~$ sudo mariadb-secure-installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
     SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
haven't set the root password yet, you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password or using the unix_socket ensures that nobody
can log into the MariaDB root user without the proper authorisation.

You already have your root account protected, so you can safely answer 'n'.

Switch to unix_socket authentication [Y/n] y
Enabled successfully!
```

```
Reloading privilege tables..
... Success!


You already have your root account protected, so you can safely answer 'n'.

Change the root password? [Y/n] y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
... Success!


By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] y
... Success!

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] y
... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] y
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] y
... Success!

Cleaning up...

All done!  If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
```

Prilikom konfiguracije servisa, odabrani su sljedeći parametari: zabrani vanjsku autentifikaciju kao korijenski korisnik (eng. „*Disallow root login remotely*“) koji onemogućuje prijavu kao korijenski korisnik izvan ovog poslužitelja i omogućena je autentifikacija putem Unix utičnica (eng. „*socket*“) parametrom: prebaciti se na autentifikaciju putem utičnica (eng. „*Switch to unix_socket authentication*“).

Nakon toga ulazimo u SUBP, kreiramo bazu podataka i korisnika te dodajemo prava pristupa korisniku:

```
MariaDB [(none)]> create database pimcore charset=utf8mb4;
```

```
MariaDB [(none)]> create user 'pim'@'10.65.0.%' identified by 'Luka123';
MariaDB [(none)]> grant all privileges on pimcore.* to 'pim'@'10.65.0.%' with grant option;
```

Na kraju, u konfiguracijskoj datoteci /etc/mysql/mariadb.conf.d/50-server.cnf treba promijeniti adresu na kojoj SUBP sluša u:

```
bind-address 10.65.0.7
```

4.6. Podešavanje virtualnih strojeva web poslužitelja

Arhitektura sustava planira dva web poslužitelja, iako ih može biti neograničen broj. Web poslužitelji će posluživati identičan sadržaj, odnosno identični programski kod web aplikacije. Oni će biti softverski i hardverski identični jedan drugom (iste specifikacije, isti softver, iste konfiguracijske datoteke).

Osim prethodnih koraka kreiranja virtualnih strojeva, instalacije sustava i instalacije PHP-a i pripadajućih paketa s PHP-Composer sustavom za upravljanje zavisnostima PHP projekata, potrebno je instalirati web poslužitelj Apache, podesiti konfiguracijske datoteke, isključiti rotaciju log zapisa i podesiti samu sinkronizaciju log zapisa na NFS. Rotaciju log zapisa potrebno je isključiti kako bi svi log zapisi bili u jednoj datoteci. Ukoliko postoji rotacija, svakom rotacijom nastaje nova datoteka u koju se pišu log zapisi zahtjeva.

Prvo je potrebno instalirati Apache web poslužitelj i onemogućiti zadanog virtualnog domaćina:

```
deb@web1:~$ sudo apt install apache2
deb@web1:~$ sudo a2dissite 000-default.conf
```

Zatim je potrebno omogućiti module za PHP-FPM:

```
deb@web1:/var/www/html$ sudo a2enmod actions fcgid rewrite alias proxy_fcgi
```

Nakon toga, treba produžiti interval za rotaciju log zapisa na jednom godišnje, rotiranje svakih 100 godina:

```
deb@web1:/etc/logrotate.d$ cat apache2
/var/log/apache2/*.log {
       yearly
       missingok
       rotate 36500
```

Idući korak je kreirati virtualnog domaćina (eng. „*Virtual host*") na web poslužitelju. Za potrebe diplomskog rada korištena je zadana, gotova konfiguracija dostupna na Pimcore dokumentaciji. U direktoriju /etc/apache2/sites-available kreira se datoteka pimcore.conf.

Datoteka je sadržaja:

```
<VirtualHost *:80>
    ServerName pim.diplomski-rad.online
    DocumentRoot /var/www/html/public
    ErrorLog /var/log/apache2/error.log
    CustomLog /var/log/apache2/access.log combined
    DirectoryIndex index.php index.htm index.html
    <Directory /var/www/html/public>
        Options -Indexes +IncludesNOEXEC +SymLinksIfOwnerMatch +ExecCGI
        allow from all
```

```
        AllowOverride All
Options=ExecCGI,Includes,IncludesNOEXEC,Indexes,MultiViews,SymLinksIfOwnerMatch
        Require all granted
    </Directory>

    <FilesMatch \.php$>
        SetHandler "proxy:unix:/run/php/php8.2-fpm.sock|fcgi://localhost"
    </FilesMatch>

    RewriteEngine on
    RewriteCond %{SERVER_NAME} =pim.diplomski-rad.online
    RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=permanent]
</VirtualHost>
```

Nakon toga, omogućuje se virtualni domaćin:

```
deb@web1:/etc/apache2/sites-available$ sudo a2ensite pimcore.conf
```

I dodaje se .htaccess datoteka u direktorij /var/www/html. Korištena je zadana .htaccess datoteka iz Pimcore dokumentacije.

Prije instalacije Pimcore softvera, potrebno je promijeniti vlasništvo iz trenutnog korisnika u korisnika www-data (korisnik web poslužitelja), taj je korak potrebo napraviti na NFS poslužitelju:

```
deb@nfs:~$ sudo chown -R www-data:www-data html/
```

Iduće se podešava sinkronizacija web log zapisa na NFS.

Prvo se dodaje nova točka monitranja (eng. "*mount*") NFS-a za direktorij u kojem će se nalaziti

Potrebna je skripta oblika:

```
#!/bin/bash
cp /var/log/apache2/pimcore-access.log /mnt/access_log/ws1-access.log
```

Skripta pokreće kopiranje log zapisa web poslužitelja na NFS u direktorij /home/deb/access_log

Tu skriptu potrebno je staviti u cron zadatak koji će ju pokretati svakih sat vremena.

```
0 * * * * /opt/scripts/sync_log.sh
```

Da bi se skripta mogla spojiti na NFS, potrebno je generirati par SSH ključeva:

```
root@web1:/opt/scripts# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

Javni ključ zatim treba dodati u datoteku .ssh/authorized_keys na NFS poslužitelju.

Idući korak je instalirati Pimcore.

Instalacija se izvršava kao korisnik www-data:

```
deb@web1:~$ sudo -u www-data bash
```

Pimcore će se instalirati s zadanim parametrima, na jednom od web poslužitelja, naredbom:

```
www-data@web1:~/html$ ./vendor/bin/pimcore-install --admin-username=admin --admin-
password=Luka123 --mysql-username=pim --mysql-password=Luka123 --mysql-database=pimcore --
mysql-host-socket=10.65.0.7 --mysql-port=3306
```

Na kraju, pokreću se svi servisi i testira se web poslužitelj:

```
deb@web1:~$ sudo systemctl enable --now php8.2-fpm.service
deb@web1:~$ sudo systemctl enable --now apache2

deb@web1:~$ curl -ILX GET "http://localhost:80/"
HTTP/1.1 200 OK
Date: Wed, 08 May 2024 17:15:54 GMT
Server: Apache/2.4.59 (Debian)
Cache-Control: max-age=0, must-revalidate, private
X-Custom-Header: Bazinga
X-Custom-Header3: foo, bar
Pragma: no-cache
Expires: Wed, 08 May 2024 17:15:54 GMT
X-Powered-By: pimcore
Content-Language: en
X-Debug-Token: 51bb23
X-Pimcore-Output-Cache-Disable-Reason: Debug flag DISABLE_FULL_PAGE_CACHE is enabled
X-Debug-Token-Link: http://localhost/_profiler/51bb23
X-Robots-Tag: noindex
Set-Cookie: PHPSESSID=ubsec8sf1ee9b54leolp4cco1s; path=/; httponly; samesite=strict
Vary: Accept-Encoding
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8
```

4.7. Podešavanje upravljačkog virtualnog stroja

Upravljački virtualni stroj služit će za nadzor klastera, izvođenje modela strojnog učenja i pokretanje skripti za automatsko skaliranje. Za nadzor klastera koristi se softver Zabbix [53]. Zabbix je distribuirano rješenje otvorenog koda za nadzor sustava na visokoj poslovnoj razini [53]. Zabbix je softver koji nadzire brojne parametre mreže, zdravlja i integriteta poslužitelja, virtualnih strojeva, aplikacija, servisa, baza podataka, internetskih stranica, računalstva u oblaku i dr. [53].

Da bi se instalirao Zabbix poslužitelj na upravljački virtualni stroj, potrebno je podesiti repozitorij:

```
deb@monitor:~$ wget https://repo.zabbix.com/zabbix/6.4/debian/pool/main/z/zabbix-
release/zabbix-release_6.4-1+debian12_all.deb
deb@monitor:~$ sudo dpkg -i zabbix-release_6.4-1+debian12_all.deb
deb@monitor:~$ sudo apt update
```

Idući korak je instalirati Zabbix poslužitelj, web sučelje, glavnog agenta i bazu podataka:

```
deb@monitor:~$ sudo apt install zabbix-server-mysql zabbix-frontend-php zabbix-apache-conf
zabbix-sql-scripts zabbix-agent mariadb-server
deb@monitor:~$ sudo mysql_secure_installation
```

Zatim kreirati bazu podataka za Zabbix i uvesti u nju inicijalni opis sheme baze podataka:

```
deb@monitor:~$ mysql -u root -p

MariaDB [(none)]> create database zabbix character set utf8mb4 collate utf8mb4_bin;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> create user zabbix@localhost identified by 'password';
Query OK, 0 rows affected (0.004 sec)

MariaDB [(none)]> grant all privileges on zabbix.* to zabbix@localhost;
```

```
Query OK, 0 rows affected (0.004 sec)

MariaDB [(none)]> set global log_bin_trust_function_creators = 1;
Query OK, 0 rows affected (0.000 sec)


deb@monitor:~$ zcat /usr/share/zabbix-sql-scripts/mysql/server.sql.gz | mysql --default-
character-set=utf8mb4 -uzabbix -p zabbix
```

Zatim u konfiguracijsku datoteku /etc/zabbix/zabbix_server.conf dodati parametre za bazu podataka:

```
DBName=zabbix
DBUser=zabbix
DBPassword=password
```

I pokrenuti Zabbix servis:

```
deb@monitor:~$ sudo systemctl enable –now zabbix-server zabbix-agent apache2
```

Nakon podešavanja Zabbix poslužitelja, potrebno je instalirati agente na sve ostale virtualne poslužitelje.

Dodaje se repozitorij Zabbix agenta na sustav i instalira se agent:

```
deb@lb-vm:~$  wget https://repo.zabbix.com/zabbix/6.4/debian/pool/main/z/zabbix-
release/zabbix-release_6.4-1+debian12_all.deb
deb@lb-vm:~$ sudo dpkg -i zabbix-release_6.4-1+debian12_all.deb
deb@lb-vm:~$ sudo apt update
deb@lb-vm:~$ sudo apt install zabbix-agent
```

Zatim se, u konfiguracijsku datoteku /etc/zabbix/zabbix_agentd.conf, dodaje IP adresa glavnog Zabbix poslužitelja i adresa na kojoj agent sluša:

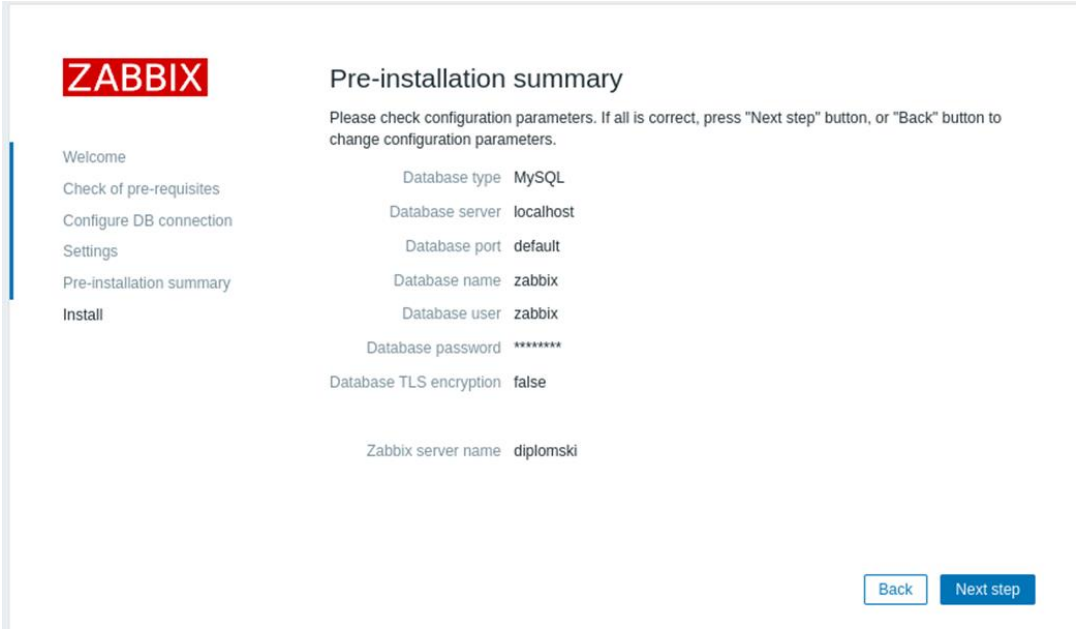
```
Server=10.65.0.8
ServerActive=10.65.0.8
ListenIP=10.65.0.3
Pa onda pokrećemo Zabbix agenta:
deb@lb-vm:~$ sudo systemctl restart zabbix-agent
deb@lb-vm:~$ sudo systemctl enable --now zabbix-agent
```

Analogno se podešavaju i ostali poslužitelji u klasteru.

Proces se može ubrzati kreiranjem skripte oblika:

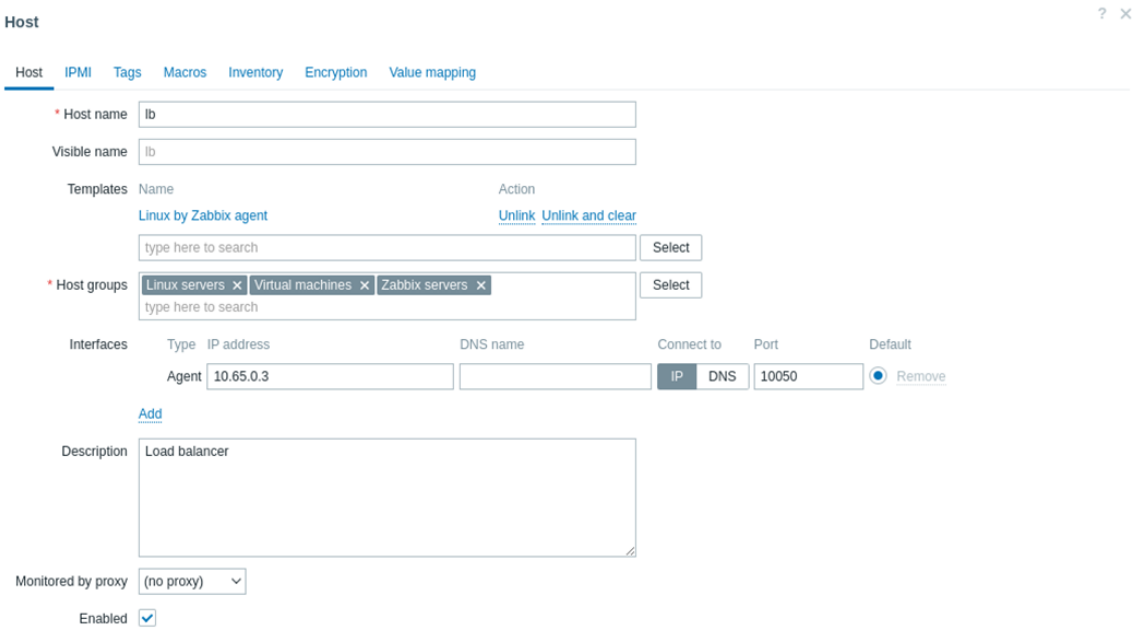
```
#!/bin/bash
wget https://repo.zabbix.com/zabbix/6.4/debian/pool/main/z/zabbix-release/zabbix-
release_6.4-1+debian12_all.deb

dpkg -i zabbix-release_6.4-1+debian12_all.deb

apt update

apt install zabbix-agent -y

sed -i 's/^Server=127\.0\.0\.1$/Server=10.65.0.8/;
s/^ServerActive=127\.0\.0\.1$/ServerActive=10.65.0.8/' /etc/zabbix/zabbix_agentd.conf

ip=$(hostname -i)
echo "ListenIP=$ip" >> /etc/zabbix/zabbix_agentd.conf


systemctl restart zabbix-agent
systemctl enable zabbix-agent
```

Nakon instalacije agenata, u web sučelju Zabbix poslužitelja, pokreće se instalacija, dodaju se virtualni poslužitelji i aktivira se nadzor istih. Slika 21. prikazuje web sučelje Zabbix instalacije.



Slika 21 Instalacija Zabbix-a

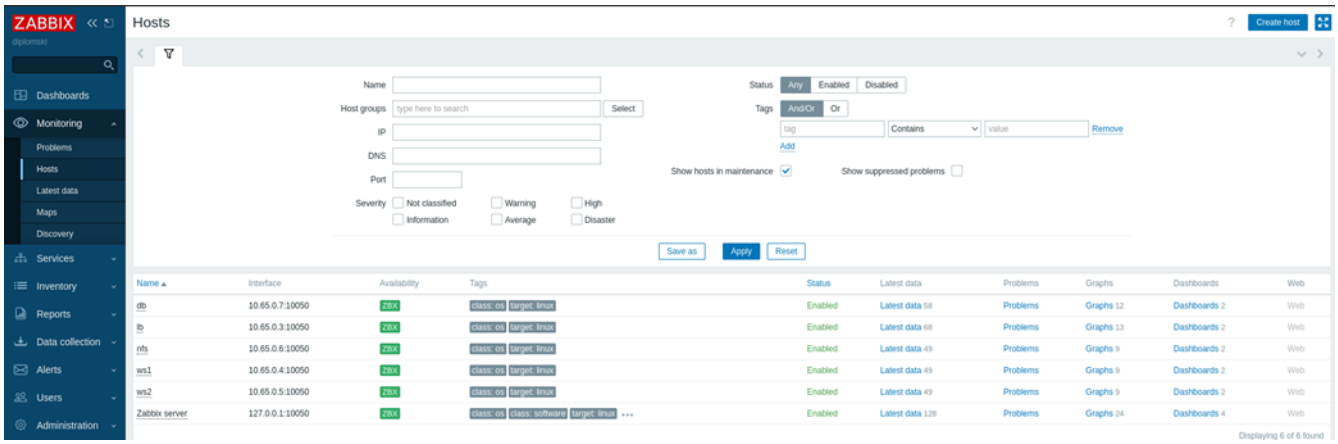
U izborniku „Data Collection – Hosts" odabere se „Create host" i popuni se parametrima od virtualnog stroja. Slika 22. prikazuje formu za dodavanje novog poslužitelja.



Slika 22 Forma za dodavanje poslužitelja

Analogno se popunjavaju parametri za ostale virtualne strojeve.

U konačnici nastaje panel s prikazom svih poslužitelja koje Zabbix prati. Slika 23. prikazuje Zabbix pregledni panel instaliranih poslužitelja.



Slika 23 Panel poslužitelja

Nakon podešavanja nadzora poslužitelja, prelazi se na izvođenje modela strojnog učenja. O modelu i tehnikama će se detaljnije pisati u Poglavlju 5, trenutno će se prikazati potrebni koraci za instalaciju.

Prije podešavanja softvera za izvođenje modela strojnog učenja, treba se vratiti na web poslužitelj. Na njemu podesiti skriptu koja će pratiti i logirati opterećenje hardverskih resursa (procesora i memorije). Ovaj korak je nužan kako bi kasnije bilo moguće konstruirati skup podataka za model.

Potrebno je instalirati supervisor (nadzorni) alat za kontrolu rada procesa:

```
deb@web1:~$ sudo apt install supervisor -y
```

U direktoriju /etc/supervisor/conf.d kreira se konfiguracijska datoteka procesa koji pokreće skriptu za nadzor resursa. Konfiguracija je oblika:

```
[program:sysmon]
command=/usr/bin/python3 usage.py
directory=/opt/scripts/
autostart=true
autorestart=true
stderr_logfile=/var/log/supervisor/sysmon.err.log
stdout_logfile=/var/log/supervisor/sysmon.out.log
process_name=sysmon
user=root
```

Za izvođenje skripte pobrinuti će se alat supervisor. Supervisor je klijent/server sustav koji omogućuje korisnicima nadzor i kontrolu procesa na UNIX sustavima [54]. Supervisor će se instalirati i na upravljački virtualni stroj.

Zatim se, u direktoriju /opt/scripts, kreira Python skripta za praćenje korištenja resursa. Skripta koristi biblioteku Python psutil s ugrađenim funkcijama za praćenje pojedinih resursa. U skripti, definirane su varijable za praćenje korištenja procesora, memorije i swap prostora kao i vremenska oznaka. Izvođenje skripte je neprekidno, a podaci se dohvaćaju svake minute.

Nakon dohvaćanja podataka, isti se zapisuju u csv datoteku u obliku pogodnom za skup podataka vremenske serije.

Skripta je oblika:

```
import psutil
import csv
import time
from datetime import datetime


def collect_system_stats():
  cpu_percent = psutil.cpu_percent()
  ram_percent = psutil.virtual_memory().percent
   swap_percent = psutil.swap_memory().percent
   timestamp = datetime.now().strftime('%d-%m-%Y-%H-%M')
   with open('system_usage.csv', mode='a', newline='') as file:
       writer = csv.writer(file)
       if file.tell() == 0:
           writer.writerow(['timestamp','CPU (%)', 'RAM (%)', 'Swap (%)'])
       writer.writerow([timestamp,cpu_percent, ram_percent, swap_percent])

if __name__ == "__main__":
   while True:
       collect_system_stats()
       time.sleep(60)
```

Nakon toga se pokreće supervisor proces i provjerava status skripte:

```
deb@web1:~$ sudo supervisorctl status
sysmon                           RUNNING   pid 23938, uptime 0:00:11
deb@web1:/opt/scripts$ cat system_usage.csv
timestamp,CPU (%),RAM (%),Swap (%)
09-05-2024-11-14,0.0,22.7,0.1
09-05-2024-11-15,0.0,20.5,0.1
```

Skripta se uredno izvršava.

Datoteku system_usage.csv potrebno je sinkronizirati na NFS kako bi bila dostupna upravljačkom serveru. Za to se koristi skripta i cron zadatak za kopiranje web log zapisa:

```
#!/bin/bash
cp /opt/scripts/system_usage.csv /mnt/sys_usage/ws1-usage.csv

0 * * * * /opt/scripts/sync_usage.sh
```

Za izvođenje modela strojnog učenja potreban nam je Python najnovije verzije, pip upravitelj paketa kao i biblioteke za strojno učenje.

Paketi i biblioteke instaliraju se naredbama:

```
deb@monitor:~$ sudo apt install python3-pip
deb@monitor:~$ pip3 install pandas numpy statsmodels scikit-learn prophet xgboost
matplotlib --break-system-packages
```

Za kraj, potrebno je montirati NFS direktorije i na upravljački server.

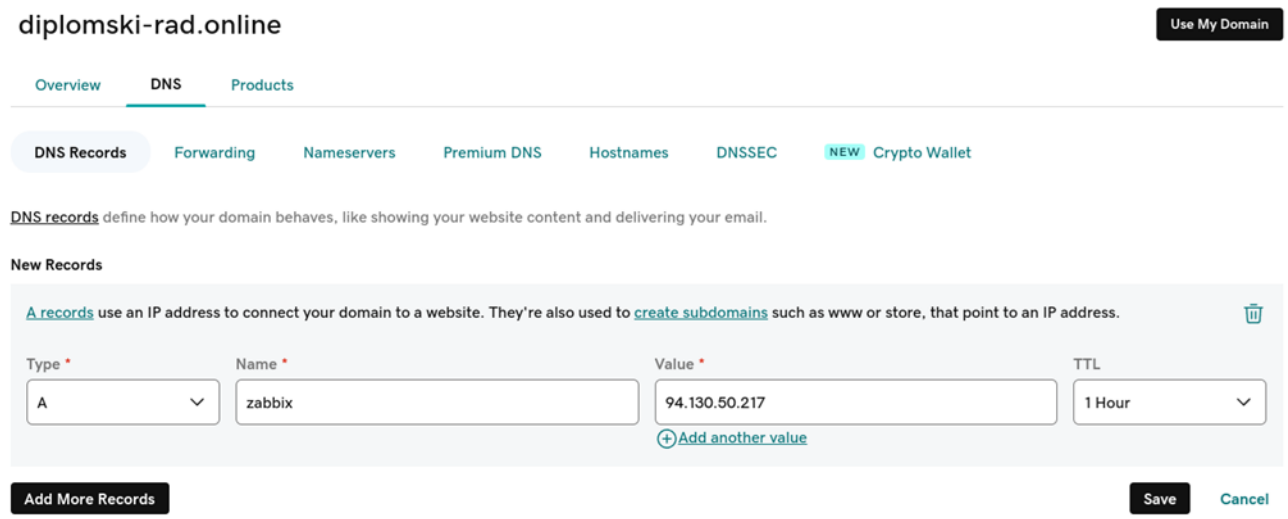
4.8. Podešavanje domene i Proxy zahtjeva

Radi jednostavnijeg pristupa resursima zakupiti će se domena na GoDaddy DNS poslužitelju [55]. Zakupljena domena je d**iplomski-rad.online.** Zakupljena domena je diplomski-rad.online

Na platformi GoDaddy DNS dodati ćemo slijedeće poddomene:

- zabbix.diplomski-rad.online i
- pim.diplomski-rad.online.

Slika 24. prikazuje formu za dodavanje domene.



Slika 24 Forma za dodavanje pod domene

Zatim se omogućuje Proxy način rada Apache web poslužitelja:

```
root@lm-server /etc/apache2/sites-available # a2enmod proxy
```

Nakon toga, na glavnom poslužitelju, dodaju se dva virtualna domaćina. Prvi (zabbix) će preusmjerit zahtjev na upravljački virtualni poslužitelj, drugi (pim) će preusmjeriti zahtjev na balanser opterećenja.

Konfiguracija virtualnih domaćina je oblika:

```
root@lm-server /etc/apache2/sites-available # ls
pim.conf  zabbix.conf
root@lm-server /etc/apache2/sites-available # cat *.conf
<VirtualHost *:80>
   ServerName pim.diplomski-rad.online
   ProxyPass "/" "http://10.65.0.3:80/"
   ProxyPassReverse "/" "http://10.65.0.8:80/"
</VirtualHost>

<VirtualHost *:80>
   ServerName zabbix.diplomski-rad.online
   ProxyPass "/" "http://10.65.0.8:80/"
   ProxyPassReverse "/" "http://10.65.0.8:80/"
</VirtualHost>
```

Omogućuju se virtualne domaćine i servis se ponovno pokreće:

```
root@lm-server /etc/apache2/sites-available # a2ensite pim.conf zabbix.conf; systemctl
restart apache2.service
```

Zadnji korak je kreirati SSL certifikate. Za SSL certifikate koristiti će se alat Certbot koji generira besplatne Let'sEncrpyt certifikate [56].

Certbot se instalira naredbom:

```
sudo apt install certbot python3-certbot-apache
```

Certifikat se generira naredbom:

```
root@lm-server /etc/apache2/sites-available # sudo certbot –apache
Which names would you like to activate HTTPS for?
We recommend selecting either all domains, or all domains in a VirtualHost/server block.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1: pim.diplomski-rad.online
2: pve.diplomski-rad.online
3: zabbix.diplomski-rad.online
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Select the appropriate numbers separated by commas and/or spaces, or leave input
blank to select all options shown (Enter 'c' to cancel): 1
Requesting a certificate for pim.diplomski-rad.online

Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/pim.diplomski-rad.online/fullchain.pem
Key is saved at:         /etc/letsencrypt/live/pim.diplomski-rad.online/privkey.pem
This certificate expires on 2024-08-07.
These files will be updated when the certificate renews.
Certbot has set up a scheduled task to automatically renew this certificate in the
background.

Deploying certificate
Successfully deployed certificate for pim.diplomski-rad.online to /etc/apache2/sites-
available/pim-le-ssl.conf
Congratulations! You have successfully enabled HTTPS on https://pim.diplomski-rad.online
```

4.9. Podešavanje skripti za generiranje zahtjeva

Zadnji korak pripreme je uspostavljanje skripti koja će kreirati zahtjeve na Pimcore stranicu. Obzirom da je ovaj rad simulacija stvarnog okruženja u oblaku, zahtjevi na stranicu će također biti simulirani u ciklusima.

Ciklusi će biti:

- Radnim danom od 9h do 16h, svaki puni sat, 500 zahtjeva na veće datoteku (npr. sliku),
- Radnim danom, od 9h do 16h, svakih 5 minuta, 500 zahtjeva na male datoteke (npr. tekstualni članak),
- Radnim danom od 17h do 23h, svakih 5 minuta, 1000 zahtjeva na male datoteke,
- Radnim danom, od 17h do 23h, svakih 20 minuta, 2500 zahtjeva na veće datoteke,
- Vikendom, svakih 15 minuta, 5000 zahtjeva na male datoteke,
- Vikendom, svakih 30 minuta, 6000 zahtjeva na veće datoteke.

Ovi ciklusi simuliraju ponašanje korisnika. U danoj simulaciji, radnim danom, tokom radnog vremena, povremeno se čitaju članci ili se gledaju edukativna videa. Van radnog vremena, radnim danom, gledaju se slike i video zapisi, čitaju se članci – korisnici pretražuju internet nakon posla. Vikendom je većina radnika slobodna pa pretražuju internet. Promet će se generirati oko 45 dana, od 1.5.2024. do 20.6.2024.

Skripte koje simuliraju promet generiraju sekvencu brojeva od 1 do zadanog, zatim tu sekvencu prosljeđuje kao argument u iduću naredbu koja izvodi n procesa u paraleli (-p), s jednim argumentom po pozivu (-n 1) pri čemu će svaki argument biti zamijenjen „-I {}" s naredbom curl koja generira zahtjev na web stranicu i taj zahtjev ispisuje u virtualni uređaj koji odbacuje sve podatke koji su u njega upisani (/dev/null) – tzv. „virtualna crna rupa" u Linux sustavu.

Skripte za generiranje zahtjeva su oblika:

```
deb@monitor:/opt/scripts$ cat *.sh
#!/bin/bash
seq 1000 | xargs -P 500 -n 1 -I {} curl -s -o /dev/null https://pim.diplomski-
rad.online/en/More-Stuff/Terms-and-Conditions

#!/bin/bash
seq 2500 | xargs -P 750 -n 1 -I {} curl -s -o /dev/null https://pim.diplomski-
rad.online/Car%20Images/bmw/252/image-thumb__252__galleryLightbox/BMW_507.b51d2bea.jpg
#!/bin/bash
seq 5000 | xargs -P 1000 -n 1 -I {} curl -s -o /dev/null https://pim.diplomski-
rad.online/en/More-Stuff/Terms-and-Conditions

#!/bin/bash
seq 500 | xargs -P 500 -n 1 -I {} curl -s -o /dev/null https://pim.diplomski-
rad.online/en/More-Stuff/Terms-and-Conditions

#!/bin/bash
seq 500 | xargs -P 250 -n 1 -I {} curl -s -o /dev/null https://pim.diplomski-
rad.online/Car%20Images/bmw/252/image-thumb__252__galleryLightbox/BMW_507.b51d2bea.jpg
#!/bin/bash
seq 6000 | xargs -P 950 -n 1 -I {} curl -s -o /dev/null https://pim.diplomski-
rad.online/Car%20Images/bmw/252/image-thumb__252__galleryLightbox/BMW_507.b51d2bea.jpg
```

Stvara se šest takvih skripti koje pokrivaju broj zahtjeva prema danim specifikacija simulacije.

```
Zatim se kreira cron zadatak koji te skripte izvodi, on je oblika:
0 9-16 * * 1-5 /opt/scripts/500-veliki.sh
*/5 9-16 * * 1-5 /opt/scripts/500-mali.sh
*/5 17-23 * * 1-5 /opt/scripts/1000-mali.sh
*/20 17-23 * * 1-5 /opt/scripts/2500-veliki.sh
*/15 * * * 6-7 /opt/scripts/5000-mali.sh
*/30 * * * 6-7 /opt/scripts/6000-veliki.sh
```

Skripte će se izvoditi na upravljačkom virtualnom stroju.

Kada su skripte pokrenute, privremeno se onemogućuje drugi web poslužitelj u HAProxy konfiguraciji, kako bi prikupilo dovoljno podataka za modeliranje:

```
backend be
   balance roundrobin
   server be1 10.65.0.4:80/ check
   #server be2 10.65.0.5:80/ check
```

5. Razvoj modela za predikciju opterećenja

5.1. Priprema podataka

Prikupljeni podaci su u formatu dvije datoteke log zapisa. Prva datoteka sadrži log zapise svih zahtjeva na web poslužitelj, dok druga datoteka sadrži podatke o opterećenju web poslužitelja svake minute.

Prva log datoteka-zahtjevi je oblika:

```
deb@nfs:~/access_log$ tail -n 5 ws1-access.log
10.65.0.3 - - [06/Jun/2024:10:00:16 -0400] "GET /en/More-Stuff/Terms-and-Conditions
HTTP/1.1" 200 29692 "-" "curl/7.88.1"
10.65.0.3 - - [06/Jun/2024:10:00:16 -0400] "GET /en/More-Stuff/Terms-and-Conditions
HTTP/1.1" 200 29711 "-" "curl/7.88.1"
10.65.0.3 - - [06/Jun/2024:10:00:16 -0400] "GET /en/More-Stuff/Terms-and-Conditions
HTTP/1.1" 200 29711 "-" "curl/7.88.1"
10.65.0.3 - - [06/Jun/2024:10:00:16 -0400] "GET /en/More-Stuff/Terms-and-Conditions
HTTP/1.1" 200 29692 "-" "curl/7.88.1"
10.65.0.3 - - [06/Jun/2024:10:00:16 -0400] "GET /en/More-Stuff/Terms-and-Conditions
HTTP/1.1" 200 29711 "-" "curl/7.88.1"
```

Druga log datoteka - opterećenja je oblika:

```
deb@nfs:~/sys_usage$ tail -n 5 ws1-usage.csv
06-06-2024-09-55,23.0,23.3,11.3
06-06-2024-09-56,0.5,18.2,11.3
06-06-2024-09-57,0.4,18.2,11.3
06-06-2024-09-58,0.1,18.2,11.3
06-06-2024-09-59,0.1,18.2,11.3
```

Kako bi se pripremili podaci za učenje modela strojnog učenja, ova dva izvora moraju se spojiti u jedan skup podataka iz kojeg će se pripremiti značajke za učenje modela.

Podaci će se spajati na slijedeći način:

- Najprije je potrebno parsirati log datoteku web zahtjeva na način da se zbroji količina zahtjeva u jednoj jedinici vremena (za potrebe rada je definirana jedinica vremena minuta) i zapiše u obliku: vrijeme, broj zahtjeva. Zatim je potrebno spojiti broj zahtjeva iz novonastalog skupa s podacima o opterećenju poslužitelja po vremenskom trenutku (eng. „*timestamp*“), pri čemu će se, ako nedostaje podatak iz broja web zahtjeva za određeni trenutak, upisati broj 0, a ako nedostaju podaci o opterećenju poslužitelja, broj zahtjeva će se odbaciti.
- Kao rezultat nastaje pripremljen skup podataka oblika: vrijeme, opterećenje procesora (%), opterećenje memorije (%), opterećenje swap prostora (%), broj zahtjeva

Skripta za parsiranje log zapisa broja zahtjeva je oblika:

```
import re
from collections import defaultdict
from datetime import datetime, timedelta

log_file_path = 'access_log/ws1-access.log'
output_file_path = 'requesti.csv'
```

```
log_pattern = re.compile(r'\[(\d{2}/\w{3}/\d{4}:\d{2}:\d{2}):\d{2} -\d{4}\]')

request_counts = defaultdict(int)

first_timestamp = None
last_timestamp = None

with open(log_file_path, 'r') as log_file:
    for line in log_file:
        match = log_pattern.search(line)
        if match:
            timestamp = datetime.strptime(match.group(1), '%d/%b/%Y:%H:%M')
            formatted_timestamp = timestamp.strftime('%d-%m-%Y-%H-%M')
            request_counts[formatted_timestamp] += 1
            if first_timestamp is None or timestamp < first_timestamp:
                first_timestamp = timestamp
            if last_timestamp is None or timestamp > last_timestamp:
                last_timestamp = timestamp

if first_timestamp and last_timestamp:
    with open(output_file_path, 'w') as output_file:
        current_timestamp = first_timestamp
        while current_timestamp <= last_timestamp:
            formatted_timestamp = current_timestamp.strftime('%d-%m-%Y-%H-%M')
            count = request_counts.get(formatted_timestamp, 0)
            output_file.write(f'{formatted_timestamp}, {count}\n')
            current_timestamp += timedelta(minutes=1)
else:
    print("Error")
```

Python skripta koristi regularne izraze i paket datuma i vremena. Putem regularnih izraza, definiran je obrazac datuma i vremena u log datoteci i definiran je rječnik (eng. „*Dictionary*“) koji će spremati sumu zahtjeva po minuti. Zatim se, u petlji, čita log datoteka, traži se izraz koji odgovara obrascu i za svako podudaranje, oblik zapisa vremena pretvara se u onaj kompatibilan s drugom skriptom i prema tom formatu očitava sve zahtjeve u određenoj minuti. Svaki put kada očita zahtjev, uvećava vrijednost rječnika za 1. Ukoliko za određenu minutu nema podataka o broju zahtjeva, ispisuje taj trenutak s vrijednosti broja zahtjeva 0. Kada se svi zahtjevi pravilno parsiraju, zapisuju se u novu CSV datoteku.

Pokretanjem skripte naredbom:

```
deb@nfs:~$ python3 parse_access_log.py
```

nastaje skup podataka oblika:

```
deb@nfs:~$ tail -n 5 requesti.csv
31-05-2024-23-42, 18
31-05-2024-23-45, 1005
31-05-2024-23-50, 1000
31-05-2024-23-52, 1
31-05-2024-23-55, 998
```

Skripta koja će spojiti broj zahtjeva i podatke o opterećenju je oblika:

```
import csv

requests_file_path = 'requesti.csv'
sys_usage_file_path = 'sys_usage/ws1-usage.csv'
output_file_path = 'dataset.csv'
```

```
requests_data = {}

with open(requests_file_path, 'r') as requests_file:
    reader = csv.reader(requests_file)
    for row in reader:
        timestamp = row[0]
        num_requests = row[1]
        requests_data[timestamp] = num_requests

with open(sys_usage_file_path, 'r') as sys_usage_file, open(output_file_path, 'w',
newline='') as output_file:
    reader = csv.reader(sys_usage_file)
    writer = csv.writer(output_file)

    headers = next(reader)
    writer.writerow(headers + ['Requests(n)'])

    for row in reader:
        timestamp = row[0]
        if timestamp in requests_data:
            row.append(requests_data[timestamp])
            writer.writerow(row)
```

Skripta učitava parsiranu datoteku log zapisa i datoteku s podacima o opterećenju. U prvoj petlji čita datoteku log zapisa i spaja broj zahtjeva i vremenski žig u polje, gdje je indeks varijabli polja vremenski žig. Nakon toga, otvara se datoteka s podacima o opterećenju, a podaci se prepisuju u novu datoteku. Kada se prepišu, petlja uspoređuje indeks polja s vremenskim žigom u novoj datoteci i kada utvrdi podudaranja, upisuje podatke o broju zahtjeva u tom trenutku.

Pokretanjem skripte naredbom:

```
deb@nfs:~$ python3 combine_data.py
```

Nastaje skup podataka oblika:

```
deb@nfs:~$ tail -n 5 dataset.csv
06-06-2024-09-55,23.0,23.3,11.3, 490
06-06-2024-09-56,0.5,18.2,11.3, 1
06-06-2024-09-57,0.4,18.2,11.3, 0
06-06-2024-09-58,0.1,18.2,11.3, 0
06-06-2024-09-59,0.1,18.2,11.3, 0
```

Konačni skup podataka sadrži 60 467 zapisa:

```
deb@nfs:~$ cat dataset.csv |wc -l
60467
```

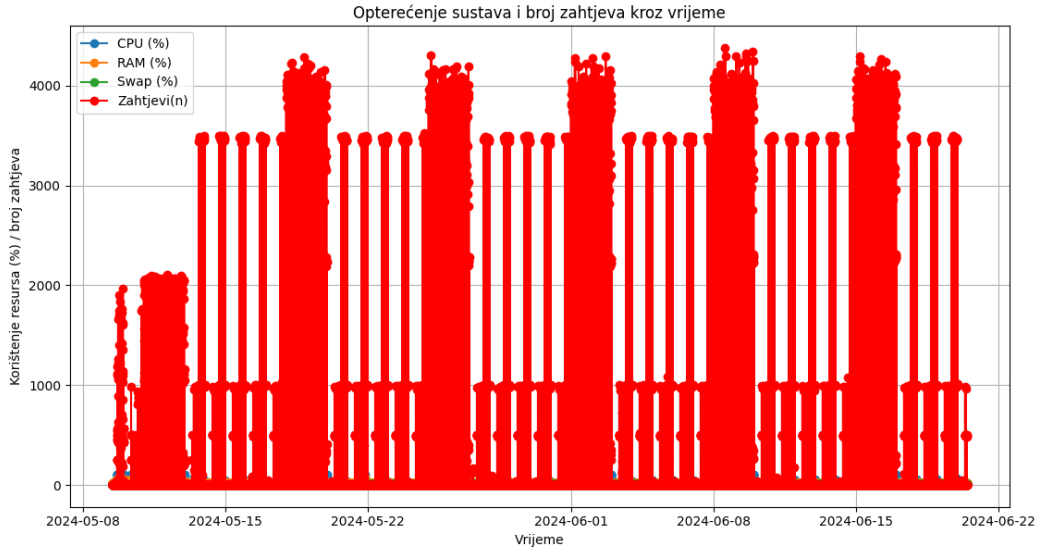
Idući korak je vizualizacija trenutnog skupa podataka kako bi se lakše prepoznale značajke koje se trebaju ispraviti u procesu čišćenja.

Koristi se skripta oblika:

```
import pandas as pd
import matplotlib.pyplot as plt

input_file_path = 'dataset.csv'

df = pd.read_csv(input_file_path)

df['timestamp'] = pd.to_datetime(df['timestamp'], format='%d-%m-%Y-%H-%M')

df.set_index('timestamp', inplace=True)
```

```
plt.figure(figsize=(12, 6))

plt.plot(df.index, df['CPU (%)'], marker='o', linestyle='-', label='CPU (%)')

plt.plot(df.index, df['RAM (%)'], marker='o', linestyle='-', label='RAM (%)')

plt.plot(df.index, df['Swap (%)'], marker='o', linestyle='-', label='Swap (%)')

plt.plot(df.index, df['Requests(n)'], marker='o', linestyle='-', label='Zahtjevi(n)',
color='red')

plt.title('Opterećenje sustava i broj zahtjeva kroz vrijeme')
plt.xlabel('Vrijeme')
plt.ylabel('Korištenje resursa (%) / broj zahtjeva')
plt.legend()
plt.grid(True)

plt.savefig('plot.png')

plt.show()
```

Skripta učitava skup podataka i iscrtava sve varijable skupa podataka u obliku linijskog grafikona. Pokretanjem skripte nastaje grafikon sa Slike 25:



Slika 25. Vizualizacija nastalog skupa podataka

Vidljivo je da skup podataka sadrži i netipične vrijednosti (eng. „*outlier*“) koje odstupaju od ostatka grafa.

5.2. Čišćenje podataka

U daljnjim postupcima pripreme podataka počišćene su netipične vrijednosti te je napravljena transformacija vrijednosti podataka.

Provedeno je grupiranje podataka po razredima (eng. „*Categorical data bining“*):

- Ostaviti sve vrijednosti 0,
- Sve vrijednosti u intervalu od 1 do 200 transformirati na 100,
- Sve vrijednosti u intervalu od 200 do 750 transformirati na 500,
- Sve vrijednosti u intervalu od 750 do 1350 transformirati na 1000,
- Sve vrijednosti u intervalu od 1350 do 2350 transformirati na 2000,
- Sve vrijednosti u intervalu od 2350 do 3250 transformirati na 2000,
- Sve vrijednosti od 2950 na dalje transformirati na 4000.

Podaci su grupirani s ciljem smanjena kategorija podataka. Velik broj kategorija podataka u skupu utječe na kvalitetu učenja modela i njegovu točnost. Broj kategorija može se smanjiti grupiranjem više kategorija podataka u jednu, odnosno klasificiranjem podataka u predefinirane razrede kao u gornjem primjeru.

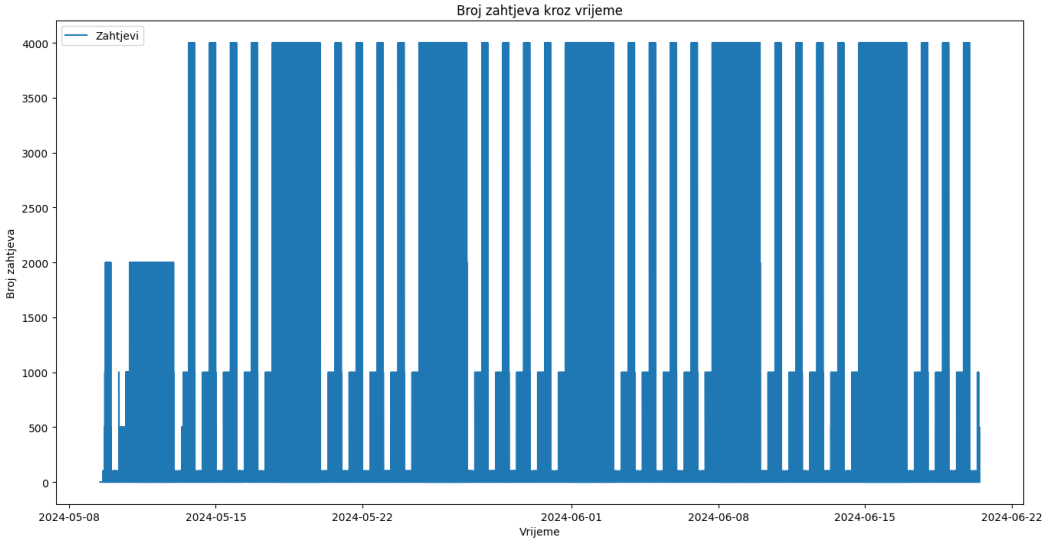
Skripta za provođenje grupiranja podataka je oblika:

```
import pandas as pd

df = pd.read_csv('dataset.csv')

df.loc[(df['Requests(n)'] >= 1) & (df['Requests(n)'] <= 200), 'Requests(n)'] = 100
df.loc[(df['Requests(n)'] >= 200) & (df['Requests(n)'] <= 750), 'Requests(n)'] = 500
df.loc[(df['Requests(n)'] >= 750) & (df['Requests(n)'] <= 1350), 'Requests(n)'] = 1000
df.loc[(df['Requests(n)'] >= 1350) & (df['Requests(n)'] <= 2350), 'Requests(n)'] = 2000
df.loc[(df['Requests(n)'] >= 2350) & (df['Requests(n)'] <= 3250), 'Requests(n)'] = 3000
df.loc[(df['Requests(n)'] >= 3250), 'Requests(n)'] = 4000

df['timestamp_dt'] = pd.to_datetime(df['timestamp'], format='%d-%m-%Y-%H-%M',
errors='coerce')

df.drop(columns=['timestamp_dt'], inplace=True)

df.to_csv('new_dataset.csv', index=False, date_format='%d-%m-%Y-%H-%M')
```

Skripta prolazi kroz sve stupce broja zahtjeva za svaki vremenski žig i vrši filtriranje i transformaciju kako je gore navedeno.

5.3. Vizualizacija podataka

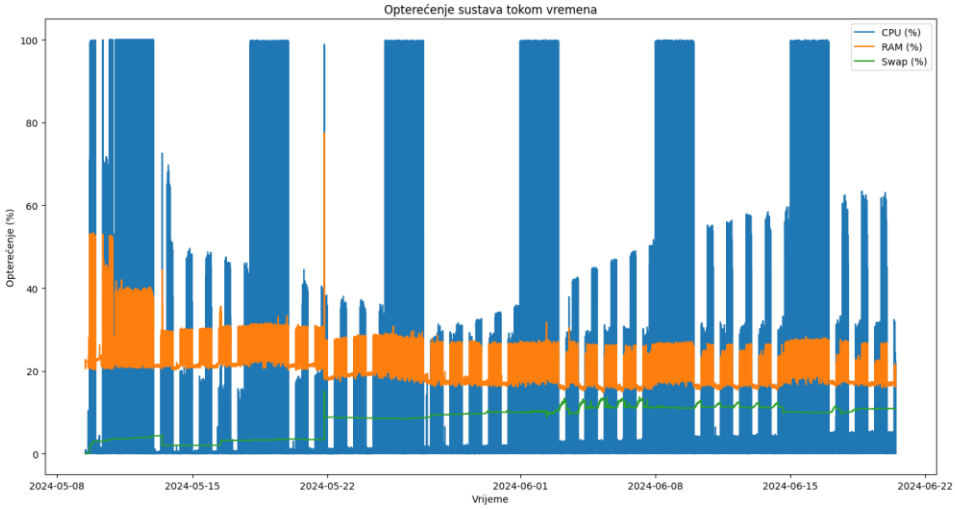
Prvi aspekt vizualizacije obuhvaća broj zahtjeva kroz vrijeme. Graf broja zahtjeva u vremenu prikazuje generalno ponašanje korisnika na postavljenoj web aplikaciji. Slika 26. prikazuje broj zahtjeva (iz očišćenog skupa) u vremenu od početka do kraja eksperimentalnog mjerenja.



Slika 26. Broj zahtjeva u vremenu

Iz slike je vidljivo podudaranje s ciklusima zadanim prilikom kreiranja skripte za generiranje prometa. Vidljiv je ciklus aktivnosti korisnika te rast i pad zahtjeva ovisno o trenutku u danu i tjednu.

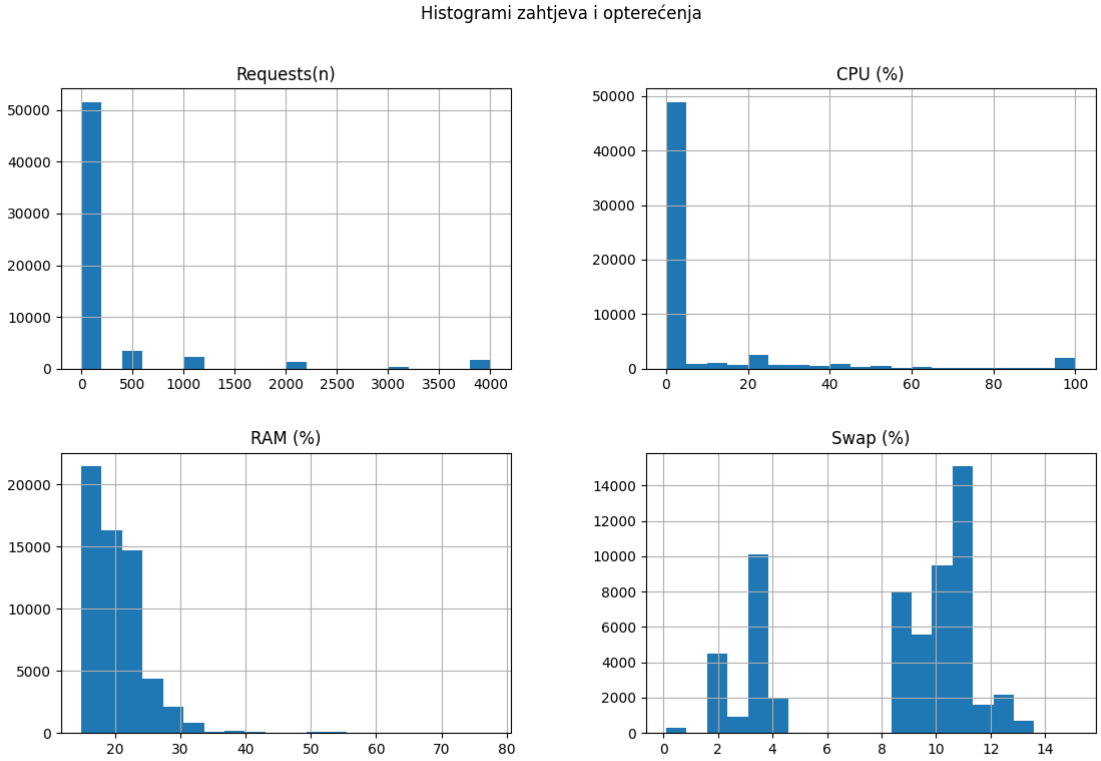
Drugi aspekt promatranja je opterećenje sustava kroz vrijeme. Graf opterećenja sustava u vremenu bitan je kako bi se uvidjela ovisnost broja zahtjeva i opterećenja sustava. Slika 27. prikazuje opterećenje poslužitelja u vremenu od početka do kraja prikupljanja podataka.



Slika 27. Graf opterećenja

Iz grafa se uočava ovisnost procesora poslužitelja i broja zahtjeva. Isto tako, vidljivo je, da je i korištenje memorije u ovisnosti s brojem zahtjeva, ali ne u tolikoj mjeri kao procesor poslužitelja.

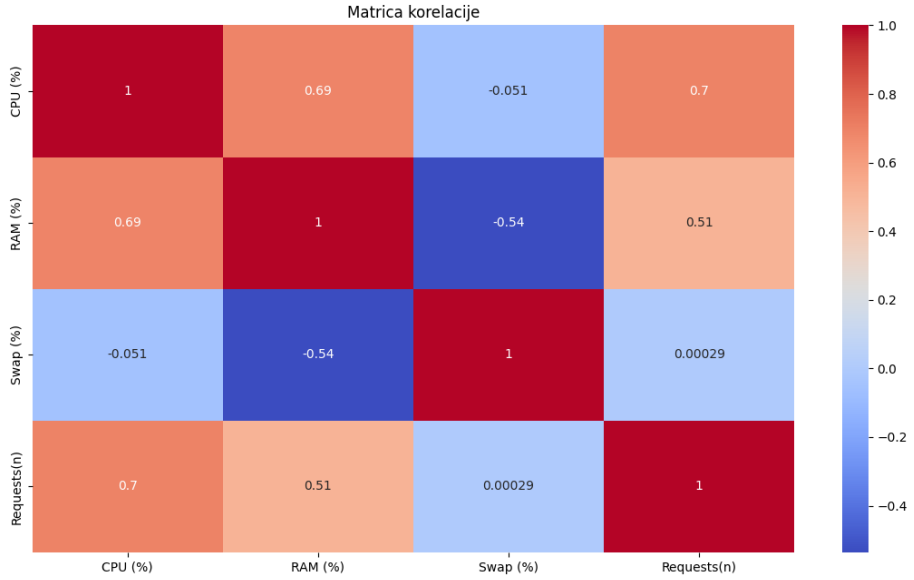
Treći aspekt promatranja je histogram podataka o opterećenju i broju zahtjeva. Slika 28. prikazuje histograme broja zahtjeva i opterećenja.



Slika 28. Histogrami opterećenja i zahtjeva

Iz histograma je razvidno da većina zahtjeva ima vrijednost 0 što odgovara postavljenom intervalu generiranja zahtjeva. Vidljivo je da je korištenje procesora, uglavnom, do 20%, a memorije između 20% i 30%. Može se uočiti da, kada na poslužitelj dođe velik broj zahtjeva, korištenje procesora skoči na 100%, odnosno da dolazi do preopterećenja kada se dogodi nalet prometa (eng. „*burst"*).

Peti aspekt promatranja je matrica korelacije. Prikazati će međusobne ovisnosti između zauzeća procesora, memorije i swap prostora i broja zahtjeva. Korištena je metoda izračuna Pearsonovog korelacijskog koeficijenta (eng. „*Pearson correlation coefficient*“) [57]. Slika 31. prikazuje matricu korelacije.



Slika 29. Matrica korelacije

Iz matrice je vidljivo da najveću korelaciju imaju varijable broj zahtjeva i zauzeće procesora (0.7), odnosno da se zauzeće procesora povećava rastom broja web zahtjeva u trenutku. Vidljivo je da i zauzeće memorije ima određenu korelaciju s povećanjem broja zahtjeva (0.5), odnosno da raste povećanjem broja zahtjeva u trenutku. Navedeno se potvrđuje i na poslužitelju, gdje Apache i PHP servisi-radnici (eng. „*worker process*“) naglo povećavaju korištenje procesora i memorije kako raste broj zahtjeva na koje moraju odgovoriti.

5.4. Implementacija modela strojnog učenja za problem skaliranja

Za izabrani problem predviđanja skaliranja web poslužitelja modelima strojnog učenja, izabran je algoritam strojnog učenja nad multivarijatnim vremenskim serijama XGBoost. XGBoost je odabran jer podržava vremenske serije s visokim varijacijama vrijednosti u određenom ciklusu, gdje ostali modeli često nisu efikasni. Model strojnog učenja će koristiti XGBoost algoritam za predviđanje buduće vrijednosti broja zahtjeva. Skripta za predviđanje buduće vrijednosti broja zahtjeva je oblika:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from xgboost import XGBRegressor
from datetime import timedelta

df = pd.read_csv('new_dataset.csv')
df['timestamp'] = pd.to_datetime(df['timestamp'], format='%d-%m-%Y-%H-%M')
df = df.rename(columns={'timestamp': 'ds', 'Requests(n)': 'y', 'CPU (%)': 'cpu', 'RAM (%)':
'ram'})

df['hour'] = df['ds'].dt.hour
df['dayofweek'] = df['ds'].dt.dayofweek
df['minute'] = df['ds'].dt.minute

X = df[['cpu', 'ram', 'hour', 'dayofweek', 'minute']]
y = df['y']

X_train, X_test, y_train, y_test, ds_train, ds_test = train_test_split(X, y, df['ds'],
test_size=0.2, random_state=42)

xgb_model = XGBRegressor(n_estimators=100, random_state=42)
xgb_model.fit(X_train, y_train)
xgb_y_pred = xgb_model.predict(X_test)

last_timestamp = df['ds'].max()
future_timestamps = [last_timestamp + timedelta(minutes=i) for i in range(1, 169999)]
future_df = pd.DataFrame(future_timestamps, columns=['ds'])

future_df['hour'] = future_df['ds'].dt.hour
future_df['dayofweek'] = future_df['ds'].dt.dayofweek
future_df['minute'] = future_df['ds'].dt.minute

last_known_values = df[['cpu', 'ram']].iloc[-1]
future_df = future_df.assign(cpu=last_known_values['cpu'], ram=last_known_values['ram'])

future_df['y'] = xgb_model.predict(future_df[['cpu', 'ram', 'hour', 'dayofweek',
'minute']])
future_df['y'] = future_df['y'].round()

inicijalna_vr = future_df['y'].iloc[0]
srednja_vr = future_df['y'].mean()
zadnja_vr = future_df['y'].iloc[-1]

if zadnja_vr > inicijalna_vr and srednja_vr > inicijalna_vr:
    stanje_zahtjeva = "Raste"
elif srednja_vr > inicijalna_vr and zadnja_vr > srednja_vr:
    stanje_zahtjeva = "Stagnira visoko"
elif zadnja_vr < inicijalna_vr and srednja_vr < inicijalna_vr:
    stanje_zahtjeva = "Pada"
else:
    stanje_zahtjeva = "Stagnira nisko"
```

```
print(f'Stanje zahtjeva: {stanje_zahtjeva}')
print(f'Inicijalna vrijednost: {inicijalna_vr}')
print(f'Srednja vrijednost: {srednja_vr}')
print(f'Zadnja vrijednost: {zadnja_vr}')
```

Algoritam XGBoost definiran je u strojnom jeziku Python putem biblioteke XGBoost [57]. Skripta prvo uključuje potrebne biblioteke, učitava skup podataka i transformira vremenski zapis u oblik pogodan za model. Također dodaje dodatne značajke poput sata, dana u tjednu, mjeseca i minute za bolju analizu podataka.

Definiraju se značajke modela (zauzeće procesora i memorije, dan u tjednu, mjesec i minuta) i ciljna varijabla (broj zahtjeva). Skup podataka se zatim dijeli na skup za učenje i skup za testiranje pri čemu se 20% podataka koristi za testiranje, a 80% podataka za učenje modela. Za predviđanje budućih vrijednosti, koristi se posljednji vremenski zapis u skupu podataka iz kojeg se generira broj zahtjeva u idućih 15 minuta. Za usporedbu vrijednosti, uzima se prva vrijednost iz 15-minutne prognoze, srednja vrijednost i zadnja vrijednost zahtjeva iz 15-minutnog intervala.

Trend zahtjeva definiran je sljedećim uvjetima:

- Broj zahtjeva raste ako su zadnja vrijednost i srednja vrijednost veće od inicijalne vrijednosti,
- Broj zahtjeva stagnira pri visokim vrijednostima ako su srednja vrijednost i zadnja vrijednost veće od inicijalne vrijednosti,
- Broj zahtjeva pada ako su zadnja vrijednost i srednja vrijednost manje od inicijalne vrijednosti,
- Broj zahtjeva stagnira nisko ako nijedan od prethodna tri uvjeta nije ispunjen.

Skripta provjerava početnu, srednju i zadnju vrijednost uvjetnim grananjem i ovisno o zadovoljenju uvjeta, popunjava varijablu stanje zahtjeva.

5.5 Rezultati modela strojnog učenja

Rezultati modela strojnog učenja biti će validirani putem 4 metrike:

- Srednja apsolutna pogreška (eng. „*Mean Absolute Error*“, MAE) [58]
- Srednja kvadratna pogreška (eng. „*Mean Squared Error*“, MSE) [59]
- Korijen srednje kvadratne pogreške (eng. „*Root Mean Squared Error*“, RMSE) [60]
- R2 rezultat (eng. „*R2 score*“, R2)

Srednja apsolutna pogreška MAE računa se kao prosjek apsolutnih razlika između predviđenih i stvarnih vrijednosti, formulom: $\frac{1}{n}\sum_{i=1}^{n}|y_i - y_{ip}|$.
Srednja kvadratna pogreška MSE računa se kao prosjek kvadrata razlika između predviđenih vrijednosti i stvarnih vrijednosti, formulom: $\frac{1}{n}\sum_{i=1}^{n}(y_i - y_{ip})^2$.
Korijen srednje kvadratne pogreške RMSE računa se kao korijen od srednje kvadratne pogreške podijeljen s brojem uzoraka, formulom: $\sqrt{\frac{\sum_{i=1}^{N}(y_i - y_{ip})^2}{N}}$
R2 rezultat mjeri koliko dobro predviđene vrijednosti objašnjavanju varijabilnost stvarnih vrijednosti, formulom: $1 - \frac{\sum_{i=1}^{n}(y_i - y_{ip})^2}{\sum_{i=1}^{n}(y_i - y_i)^2}$.

U skriptu s modelom podataka dodane su slijedeće linije za evaluaciju modela:

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

def evaluate_model(y_true, y_pred):
    r2 = r2_score(y_true, y_pred)
    mae = mean_absolute_error(y_true, y_pred)
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    return r2, mae, mse, rmse

r2, mae, mse, rmse = evaluate_model(y_test, xgb_y_pred)

print(f'R2 Score: {r2:.2f}')
print(f'MAE: {mae:.2f}')
print(f'MSE: {mse:.2f}')
print(f'RMSE: {rmse:.2f}')

plt.figure(figsize=(14, 7))
plt.plot(df['ds'], df['y'], label='Stvarni podaci (zahtjevi)')
plt.plot(future_df['ds'], future_df['y'], 'r--', label='Predikcija zahtjeva')
plt.xlabel('Vremenski žig')
plt.ylabel('Broj zahtjeva')
plt.title(f'Predikcija modela')
plt.legend()
plt.show()
```

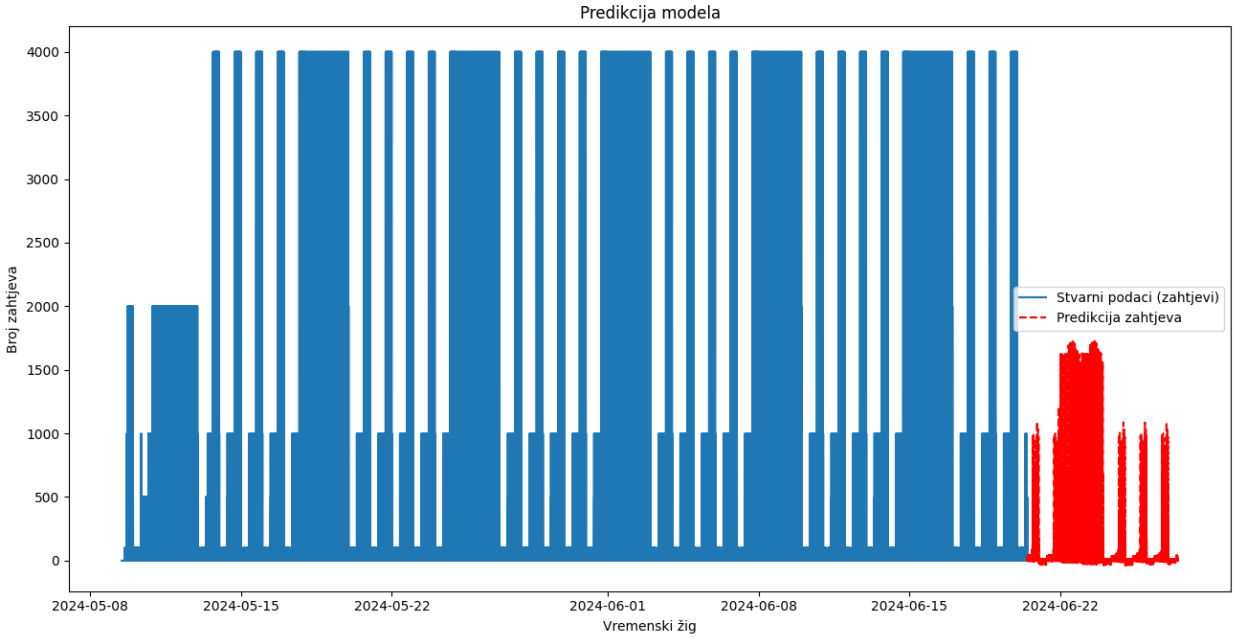
Definirana je funkcija koja poziva metode za izračune MAE, MSE, RMSE i R2 iz ugrađenih biblioteka. Ulazni parametri su stvarne vrijednosti i vrijednosti predviđanja, nad kojima se vrši evaluacija. Po evaluaciji se ispisuje rezultat i crta graf koji prikazuje predviđene vrijednosti za idućih 7 dana.

Model je evaluiran na skupu podataka prikupljanom u periodu od 43 dana (09.05.2024. – 20.06.2024.). Rezultati modela su:

```
R2 Score:      0.98
```

```
MAE:            25.80
MSE:            9874.24
RMSE:           99.37
```

Gdje je graf predikcije prikazan na slici 31.



Slika 30. Predikcija modela

6. Implementacija automatskog skaliranja uz strojno učenje i automatizacijske skripte

6.1. Popratne skripte za automatsko pokretanje i gašenje web poslužitelja

Poslužitelje će se automatski skalirati skriptama koje će pokretati i gasiti web poslužitelje. Ovim načinom može se skalirati neograničen broj poslužitelja, ovisno o dostupnim resursima na glavnom poslužitelju.

U svrhu demonstracije rada sustava, kreirana su dva web poslužitelja. Na tim poslužiteljima će aplikacija skalirati.

Prvi korak je postaviti procese Apache i PHP-fpm, koji su potrebni za posluživanje web aplikacije, na automatsko pokretanje sa sustavom:

```
deb@web2:~$ sudo systemctl enable php8.2-fpm.service
deb@web2:~$ sudo systemctl enable apache2
```

Zatim, na upravljačkom poslužitelju, generirati par SSH ključeva. Javi ključ se dodaje u datoteku .ssh/authorized_keys na poslužitelju balanser opterećenja i glavnom poslužitelju.

Nakon toga, na upravljačkom poslužitelju, kreiraju se dvije skripte. Prva skripta pokreće poslužitelj i omogućuje ga u HAProxy balanseru opterećenja kad on postane dostupan. Druga skripta isključuje poslužitelja i onemogućuje ga u balanseru opterećenja.

Skripta za pokretanje poslužitelja je oblika:

```
#!/bin/bash
URL="http://10.65.0.4/admin/login"
ssh root@10.65.0.1 "qm start 103"
sleep 60
STATUS_CODE=$(curl -s -o /dev/null -w "%{http_code}" "$URL")
if [ "$STATUS_CODE" -eq 200 ]; then
        ssh root@10.65.0.3 "sed -i 's/\#server/server/g' /etc/haproxy/haproxy.cfg;
systemctl reload haproxy"
else
        echo "Error starting"
fi
```

Skripta definira testni URL, varijablu statusni kod koji pokreće naredbu curl i dohvaća status poslužitelja. Ako je status 200, omogućuje poslužitelja u balanseru opterećenja i osvježava mu konfiguraciju.

Skripta za gašenje poslužitelja je oblika:

```
#!/bin/bash
URL="http://10.65.0.4/admin/login"
ssh root@10.65.0.1 "qm shutdown 103"
sleep 60
STATUS_CODE=$(curl -s -o /dev/null -w "%{http_code}" "$URL")
if [ "$STATUS_CODE" -ne 200 ]; then
        ssh root@10.65.0.3 "sed -i 's/\#server/server/g' /etc/haproxy/haproxy.cfg;
systemctl reload haproxy"
else
        echo "Error stopping"
fi
```

Skripta definira testni URL, varijablu, statusni kod, koja dohvaća status poslužitelja. Ako status nije jednak 200, onemogućuje poslužitelja u balanseru opterećenja i osvježava mu konfiguraciju.

6.2. Povezivanje popratnih skripti i modela strojnog učenja

Predviđeno je da će se naučeni XGBoost model neprekidno izvoditi. Kako bi se povezao model i popratne skripte za pokretanje i gašenje poslužitelja, potrebno je definirati pravila kada se drugi poslužitelj uključuje a kada isključuje. Poslužitelj će se uključivati kada model predvidi rast (slučaj „**Raste**“) u idućih 15 minuta. Ako model predvidi **stagnaciju pri visokom prometu**, poslužitelj ostaje uključen. Ako predvidi “**pad”** ili s**tagnaciju pri niskom opterećenju** gasi poslužitelj.

Postojeći model, sekcija s uvjetnim grananjem, se mijenja s programskim kodom oblika:

```
log = "/var/log/prediction.log"

def logiraj(prediction):
    with open(log, 'a') as f:
        f.write(f"{prediction}\n")

def citaj_log():
    if os.path.exists(log):
        with open(log, 'r') as f:
            lines = f.readlines()
            if lines:
                return lines[-1].strip()
    return None

prethodno_stanje = citaj_log()

if zadnja_vr > inicijalna_vr and srednja_vr > inicijalna_vr:
    stanje_zahtjeva = "Raste"
    if prethodno_stanje in ["Pada", "Stagnira nisko"]:
       os.system('/opt/scripts/start_server.sh')
    logiraj("Raste")
elif srednja_vr > inicijalna_vr and zadnja_vr > srednja_vr:
    stanje_zahtjeva = "Stagnira visoko"
    if prethodno_stanje in ["Pada", "Stagnira nisko"]:
        os.system('/opt/scripts/start_server.sh')
    logiraj("Stagnira visoko")
elif zadnja_vr < inicijalna_vr and srednja_vr < inicijalna_vr:
    stanje_zahtjeva = "Pada"
    if prethodno_stanje in ["Raste", "Stagnira visoko"]:
        os.system('/opt/scripts/stop_server.sh')
    logiraj("Pada")
else:
    stanje_zahtjeva = "Stagnira nisko"
    if prethodno_stanje in ["Raste", "Stagnira visoko"]:
        os.system('/opt/scripts/stop_server.sh')
    logiraj("Stagnira nisko")
```

Pri čemu su definirane dvije nove funkcije. Prva služi kako bi se pročitala zadnja vrijednost stanja predikcije, dok druga upisuje novo stanje predikcije u log datoteku.
Sekcija uvjetnog grananja dopunjuje se s pokretanjem skripti za paljenje i gašenje poslužitelja putem Python biblioteke „os“ te se uvodi upis stanja predikcije putem definirane funkcije. Dodano je novo uvjetno grananje koje ograničava nepotrebno izvođenje skripte za gašenje

poslužitelja, gdje se izvođenje ograničava samo na slučajeve kad je prethodno stanje ili rast ili stagnacija pri visokom opterećenju i obrnuto za skriptu pokretanja.

Kako bi rezultati bili točniji, u model se uvodi sekcija programskog koda koja će dohvatiti nove skupove podataka opterećenja i zahtjeva, predprocesirati i očistiti ih, te zatim s njima učiti novu verziju XGBoost modela. Ovom metodom osigurava se da su podaci nad kojima model uči relevantni za vremenski period predviđanja, odnosno da se model douči na najnovijim vrijednostima podataka. Ubačen je programski kod iz skripti za procesiranje zapisa te čišćenje podataka iz prethodnih poglavlja. Za neprekidno izvođenje modela dodaje se u programski kod biblioteka time, program se stavlja u beskonačnu petlju i na kraju svakog ciklusa čeka 15 minuta putem direktive spavaj (eng. "*sleep*"). Dodani programski kod je oblika:

```
import time

while True:
       # ostatak programskog koda
       time.sleep(900)
```

Kako bi se uspostavio nadzor nad izvođenjem modela, kao i oporavak u slučaju pogreške, koristi se supervisor alat. Model se definira kao servis unutar supervisor sustava putem konfiguracije:

```
[program:ml_model]
command=/usr/bin/python3 /opt/scripts/ml_model.py
directory=/opt/scripts
user=deb
autostart=true
autorestart=true
startretries=3
stderr_logfile=/var/log/supervisor/ml_model.err.log
stdout_logfile=/var/log/supervisor/ml_model.out.log
```

6.3. Testiranje sustava

Model se neprekidno izvodi:

```
deb@monitor:~$ sudo supervisorctl status
ml_model                         RUNNING   pid 3108157, uptime 4 days, 21:33:25
```

Model bilježi rezultate treniranja i trenutke kada uključuje drugi web poslužitelj:

```
deb@monitor:~$ cat /var/log/supervisor/ml_model.out.log
...
Ukljucivanje posluzitelja
R2 Score: 0.99
MAE: 22.22
MSE: 5288.22
RMSE: 72.72
R2 Score: 0.99
MAE: 22.22
MSE: 5288.22
RMSE: 72.72
R2 Score: 0.99
MAE: 22.22
MSE: 5288.22
RMSE: 72.72
```

```
Iskljucivanje posluzitelja
Stanje zahtjeva: Stagnira nisko
Inicijalna vrijednost: 31.0
Srednja vrijednost: 49.09931564331055
Zadnja vrijednost: 27.0
R2 Score: 0.99
MAE: 23.04
MSE: 5948.38
RMSE: 77.13
...
```

Vidljivo je kako model automatski uključuje i isključuje web poslužitelj ovisno o opterećenju.

Model sprema trend predikcije u log datoteku:

```
deb@monitor:~$ tail -n 5 /var/log/prediction.log
Pada
Pada
Raste
Raste
Raste
```

Model po detekciji rasta prometa uključuje server:

```
deb@monitor:/var/log/supervisor$ tail -n 5 ml_model.out.log
Ukljucivanje posluzitelja
R2 Score: 0.99
MAE: 23.41
MSE: 5373.27
RMSE: 73.30

 deb@web2:~$ uptime
```

10:51:09 up 0 min, 1 user, load average: 0.00, 0.00, 0.00

Model po detekciji pada prometa isključuje poslužitelj:

```
deb@monitor:~$ tail -n 5 /var/log/prediction.log
Raste
Raste
Raste
Raste
Pada
root@lm-server ~ # ssh deb@10.65.0.5
ssh: connect to host 10.65.0.5 port 22: No route to host
```

Dio log zapisa pokretanja poslužitelja na Proxmox domaćinu:

```
root@lm-server /var/log/pve/tasks # tail -n 50 index |grep qm
UPID:lm-server:00086F8E:017FE193:663E1E85:qmstart:103:root@pam: 663E1E85 VM 103 already
running
UPID:lm-server:0008705C:01800607:663E1EE2:qmstart:103:root@pam: 663E1EE2 VM 103 already
running
UPID:lm-server:00115B7C:03120C29:66422411:qmstop:103:root@pam: 66422411 OK
UPID:lm-server:00115C5E:0312341F:66422477:qmstart:103:root@pam: 66422478 OK
UPID:lm-server:00116764:0313DE80:664228BA:qmshutdown:103:root@pam: 664228BC OK
```

7. Zaključak

Ovaj diplomski rad prikazao je metode za stvaranje privatne infrastrukture u oblaku na Hetzner platformi. Prikazan je način stvaranja osnovne arhitekture sustava u oblaku koja se sastoji od mrežnog datotečnog sustava, dva web poslužitelja, balansera opterećenja, upravljačkog poslužitelja i poslužitelja baze podataka. Odnosno, arhitektura simulira pokretanje LAMP stoga (eng. „*Linux Apache MySQL PHP*“) u oblaku uz dva poslužitelja i balanser opterećenja. Na infrastrukturi je postavljena gotova PHP web aplikacija s ciljem prikupljanja podataka o web i sistemskom opterećenju u trajanju izrade rada.

Rad detaljno obrađuje ponašanje, jednostavnost i učinkovitost ovog rješenja za infrastrukturu u oblaku simuliranjem stvarnog okruženja (opterećenja). Isto tako, pokazano je kako takva infrastruktura može uspješno skalirati uz model strojnog učenja. Za istraživanje, korišten je model strojnog učenja temeljen na algoritmu XGBoost. Model je korišten za predviđanje budućeg opterećenja poslužitelja radi bolje optimizacije i performansi infrastrukture (i aplikacije). Uveden je i nadzorni poslužitelj kako bi se osiguralo proaktivno rješavanje potencijalnih problema.

Simulirana infrastruktura u oblaku, na Hetzner platformi – poslužitelj i Proxmox hipervizor, pruža stabilno i skalabilno okruženje za Web aplikaciju. Jednostavni model strojnog učenja, uz minimalne ulazne varijable i transformacije podataka, uspio je identificirati porast i pad opterećenja te adekvatno skalirati broj web poslužitelja.

Za daljnji rad i produkcijsko okruženje, preporuča se pravilna konfiguracija vatrozida, dorada sigurnosti zakupljenog poslužitelja, virtualna privatna mreža i ostalo u skladu s preporukama dobrih praksi konfiguracije produkcijskih servera. Postavljeni sustav je simulacija osnovne infrastrukture za pokretanje aplikacije iz LAMP (eng. “*Linux Apache MySQL PHP*”) stoga. Pokazano je kako implementirati strojno učenje za skaliranje aplikacije. Model točno predviđa trend broja zahtjeva (rast/pad), ali ne i vrijednost broja zahtjeva - što je za potvrđivanje uspješnosti razvijenog rješenja skaliranja dostatno. Učenjem regresijskih modela mogli bise u budućem radu proširiti s točnijim predviđanjem broja zahtjeva. Model strojnog učenja može se unaprijediti i uključivanjem novih značajki i tekućih podataka, kako bi se osiguralo kontinuirano učenje i povećavala točnost modela.

Literatura

[1] E. o. Encyclopaedia, »Britannica - Application software,« Britannica, 11 siječanj 2024.. [Mrežno]. Available: https://www.britannica.com/technology/application-software. [Pokušaj pristupa svibanj 2024.].

[2] M. Dabbs, »The Fundamentals of Web Application Architecture,« Reinvently, 29 srpanj 2019.. [Mrežno]. Available: https://reinvently.com/blog/fundamentals-web-application-architecture/. [Pokušaj pristupa svibanj 2024.].

[3] A. J. Montecarlo, »What is a server? | Definition, Types, and Features,« ServerWatch, 25. rujan 2023.. [Mrežno]. Available: https://www.serverwatch.com/guides/what-is-a-server/. [Pokušaj pristupa svibanj 2024.].

[4] R. Electronics, »Beginners Guide To Server Racks,« 24 prosinac 2019.. [Mrežno]. Available: https://risingracks.com/blog/beginners-guide-to-server-racks/. [Pokušaj pristupa svibanj 2024.].

[5] J. K. A. Belle Wong, »What Is Virtualization? Definition, Benefits & Examples,« Forbes Advisor, 14. rujan 2023.. [Mrežno]. Available: https://www.forbes.com/advisor/business/software/what-is-virtualization/. [Pokušaj pristupa svibanj 2024.].

[6] Cloudflare, »What is the cloud?,« [Mrežno]. Available: https://www.cloudflare.com/learning/cloud/what-is-the-cloud/. [Pokušaj pristupa svibanj 2024.].

[7] P. Estrach, »Mega.io - Scalability in Cloud Computing: A Deep Dive,« 18. kolovoz 2023.. [Mrežno]. Available: https://www.mega.com/blog/what-is-scalability-in-cloud-computing. [Pokušaj pristupa svibanj 2024.].

[8] L. Gil, »Cast AI - Cloud Automation In 2024: The New Normal In The Tech Industry,« 13. prosinac 2023. [Mrežno]. Available: https://cast.ai/blog/cloud-automation-the-new-normal-in-the-tech-industry/. [Pokušaj pristupa svibanj 2024.].

[9] Microsoft Azure team, »What is cloud computing?,« Microsoft, 2024.. [Mrežno]. Available: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-cloud-computing. [Pokušaj pristupa svibanj 2024.].

[10] Amazon, »Cloud Computing Services,« [Mrežno]. Available: https://aws.amazon.com/. [Pokušaj pristupa svibanj 2024.].

[11] Microsoft, »Cloud Computing Services,« [Mrežno]. Available: https://azure.microsoft.com/en-us. [Pokušaj pristupa svibanj 2024.].

[1 2] Cloudflare, »Get started with Cloudflare,« [Mrežno]. Available: https://developers.cloudflare.com/learning-paths/get-started/?_gl=1*10eu6uy*_ga*MTE1NjYwNjkyOS4xNzE2OTA3NTMx*_ga_SQCRB0TXZW*MTcxNjkwNzUzMS4xLjEuMTcxNjkwNzU4MC4wLjAuMA... [Pokušaj pristupa svibanj 2024.].

[1 3] Amazon, »Amazon EC2,« [Mrežno]. Available: https://aws.amazon.com/ec2/. [Pokušaj pristupa svibanj 2024.].

[1 4] Hetzner, »HETZNER CLOUD,« [Mrežno]. Available: https://www.hetzner.com/cloud/. [Pokušaj pristupa svibanj 2024.].

[1 5] Microsoft, »Azure Virtual Machines,« [Mrežno]. Available: https://azure.microsoft.com/en-us/products/virtual-machines. [Pokušaj pristupa svibanj 2024.].

[1 6] DigitalOcean, »Virtual machines anyone can set up in seconds,« [Mrežno]. Available: https://www.digitalocean.com/products/droplets. [Pokušaj pristupa svibanj 2024.].

[1 7] Amazon, »AWS Lambda,« [Mrežno]. Available: https://aws.amazon.com/lambda/. [Pokušaj pristupa svibanj 2024.].

[1 8] Oracle, »Oracle Cloud Infrastructure (OCI),« [Mrežno]. Available: https://www.oracle.com/cloud/. [Pokušaj pristupa svibanj 2024.].

[1 9] R. Hat, »Red Hat OpenShift,« [Mrežno]. Available: https://www.redhat.com/en/technologies/cloud-computing/openshift. [Pokušaj pristupa svibanj 2024.].

[2 0] Netflix, »Netflix,« [Mrežno]. Available: https://www.netflix.com/browse. [Pokušaj pristupa svibanj 2024.].

[2 1] Slack, »Made for people. Built for productivity.,« [Mrežno]. Available: https://slack.com/. [Pokušaj pristupa svibanj 2024.].

[2 2] Zoom, »Zoom: One platform to connect,« [Mrežno]. Available: https://zoom.us/. [Pokušaj pristupa svibanj 2024.].

[2 3] Adobe, »Start your Creative Cloud free trial.,« [Mrežno]. Available: https://www.adobe.com/creativecloud.html. [Pokušaj pristupa svibanj 2024.].

[2 4] Google, »Google Workspace,« [Mrežno]. Available: https://workspace.google.com/. [Pokušaj pristupa svibanj 2024.].

[2 5] T. Infotech, »Top 14 SaaS Application Examples [Successful Strategies],« Tagline Infotech, 25 studeni 2022.. [Mrežno]. Available: https://taglineinfotech.com/saas-examples/. [Pokušaj pristupa svibanj 2024.].

[26] VMware, »What is Cloud Scalability?,« VMware, [Mrežno]. Available: https://www.vmware.com/topics/glossary/content/cloud-scalability.html. [Pokušaj pristupa svibanj 2024.].

[27] nOps , »What is Scalability in Cloud Computing? Types, Benefits, and Practical Advice,« 30. travanj 2024.. [Mrežno]. Available: https://www.nops.io/blog/cloud-scalability/. [Pokušaj pristupa svibanj 2024.].

[28] Statista, »Cloud infrastructure services vendor market share worldwide from fourth quarter 2017 to first quarter 2024,« [Mrežno]. Available: https://www.statista.com/statistics/967365/worldwide-cloud-infrastructure-services-market-share-vendor/. [Pokušaj pristupa svibanj 2024.].

[29] Amazon, »AWS Cloud Products,« [Mrežno]. Available: https://aws.amazon.com/products/?aws-products-all.sort-by=item.additionalFields.productNameLowercase&aws-products-all.sort-order=asc&awsf.re%3AInvent=*all&awsf.Free%20Tier%20Type=*all&awsf.tech-category=*all. [Pokušaj pristupa svibanj 2024.].

[30] M. Zhang, »Top 10 Cloud Service Providers Globally in 2024,« Dgtl Infra, 18 travanj 2024.. [Mrežno]. Available: https://dgtlinfra.com/top-cloud-service-providers/. [Pokušaj pristupa svibanj 2024.].

[31] Hetzner, »About Us,« Hetzner, [Mrežno]. Available: https://www.hetzner.com/unternehmen/ueber-uns/. [Pokušaj pristupa svibanj 2024.].

[32] Proxmox, »Datasheet: Proxmox Virutal Environment,« 6. svibanj 2024. [Mrežno]. Available: https://www.proxmox.com/en/downloads/proxmox-virtual-environment/documentation/proxmox-ve-datasheet. [Pokušaj pristupa svibanj 2024.].

[33] S. Daley, »Machine Learning Technology,« Buitin, 8 studeni 2022.. [Mrežno]. Available: https://builtin.com/machine-learning. [Pokušaj pristupa svibanj 2024.].

[34] StatLearning, »An Introduction to Statistical Learning,« [Mrežno]. Available: https://www.statlearning.com/. [Pokušaj pristupa svibanj 2024.].

[35] Oracle, »13 Time Series,« [Mrežno]. Available: https://docs.oracle.com/en/database/oracle/machine-learning/oml4sql/21/dmcon/time-series.html. [Pokušaj pristupa svibanj 2024.].

[36] R. A. D. Peter J. Brockwell, Introduction to Time Series and Forecasting, Second Edition, Springer, 2002.

[37] N. Katardjiev, »High-variance multivariate time series forecasting using machine learning,« 4 lipanj 2018.. [Mrežno]. Available: https://uu.diva-portal.org/smash/get/diva2:1219571/FULLTEXT01.pdf. [Pokušaj pristupa svibanj. 2024.].

[38] P. Dix, »Time Series Data Analysis: Definitions & Best Techniques in 2024,« InfluxData, 2024.. [Mrežno]. Available: https://www.influxdata.com/what-is-time-series-data/. [Pokušaj pristupa svibanj. 2024.].

[39] L. Yen, »What is Time Series Analysis? Definition, Types, and Examples,« Datamation, 13 studeni 2023. [Mrežno]. Available: https://www.datamation.com/big-data/what-is-time-series-analysis/. [Pokušaj pristupa svibanj 2024.].

[40] W. Kenton, »Seasonality: What It Means in Business and Economics, Examples,« Investopedia, 30 studeni 2020. [Mrežno]. Available: https://www.investopedia.com/terms/s/seasonality.asp. [Pokušaj pristupa svibanj 2024.].

[41] statsmodels, »ETS models,« [Mrežno]. Available: https://www.statsmodels.org/dev/examples/notebooks/generated/ets.html. [Pokušaj pristupa svibanj 2024.].

[42] B. L. Sean J. Taylor, »Forecasting at scale,« Meta, 2023.. [Mrežno]. Available: https://facebook.github.io/prophet/. [Pokušaj pristupa svibanj 2024.].

[43] Nixtla, »About,« [Mrežno]. Available: https://www.nixtla.io/about. [Pokušaj pristupa svibanj 2024.].

[44] DMLC, »XGBoost Documentation,« 2022.. [Mrežno]. Available: https://xgboost.readthedocs.io/en/stable/. [Pokušaj pristupa svibanj 2024.].

[45] J. Hoare, »Gradient Boosting Explained – The Coolest Kid on The Machine Learning Block,« DisplayR, [Mrežno]. Available: https://www.displayr.com/gradient-boosting-the-coolest-kid-on-the-machine-learning-block/. [Pokušaj pristupa svibanj 2024.].

[46] N. Sharma, »How to Use XGBoost for Time-Series Forecasting?,« Analytics Vidhya, 4 siječanj 2024.. [Mrežno]. Available: https://www.analyticsvidhya.com/blog/2024/01/xgboost-for-time-series-forecasting/. [Pokušaj pristupa svibanj 2024.].

[47] P. Painter, »Firewalls 101: Network Address Translation (NAT),« HorizonIQ BeyondIT, 28 ožujak 2024.. [Mrežno]. Available: https://www.horizoniq.com/blog/firewalls-nat/. [Pokušaj pristupa svibanj 2024.].

[48] »HAProxy Community Edition,« HAProxy, 2024.. [Mrežno]. Available: https://www.haproxy.org/. [Pokušaj pristupa svibanj 2024.].

[49] Ubuntu, »Network File System,« [Mrežno]. Available: https://ubuntu.com/server/docs/network-file-system-nfs. [Pokušaj pristupa svibanj 2024.].

[50] Pimcore, »Pimcore Demo,« 18 travanj 2024. [Mrežno]. Available: https://github.com/pimcore/demo. [Pokušaj pristupa svibanj 2024.].

[5 Pimcore, »What is Pimcore?,« [Mrežno]. Available: https://pimcore.com/en/what-is-
1] pimcore. [Pokušaj pristupa svibanj 2024.].

[5 M. Foundation, »MariaDB Server: the innovative open source database,« [Mrežno].
2] Available: https://mariadb.org/. [Pokušaj pristupa svibanj 2024.].

[5 Zabbix, »What is Zabbix,« 2024. [Mrežno]. Available:
3] https://www.zabbix.com/documentation/6.2/en/manual/introduction/about. [Pokušaj pristupa svibanj 2024].

[5 Supervisor, »Supervisor: A Process Control System,« 19 svibanj 2024. [Mrežno].
4] Available: http://supervisord.org/. [Pokušaj pristupa svibanj 2024].

[5 GoDaddy, »Welcome to your launch pad.,« [Mrežno]. Available:
5] https://www.godaddy.com/en-uk. [Pokušaj pristupa svibanj 2024.].

[5 Let'sEncrypt, »A nonprofit Certificate Authority providing TLS certificates to 363 million
6] websites.,« [Mrežno]. Available: https://letsencrypt.org/. [Pokušaj pristupa svibanj 2024.].

[5 x. developers, »xgboost,« 2022.. [Mrežno]. Available:
7] https://xgboost.readthedocs.io/en/stable/python/python_intro.html#install-xgboost.

[5 M. W. Ahmed, »Understanding Mean Absolute Error (MAE) in Regression: A Practical
8] Guide,« Medium, 2023. kolovoz 24. [Mrežno]. Available: https://medium.com/@m.waqar.ahmed/understanding-mean-absolute-error-mae-in-regression-a-practical-guide-26e80ebb97df. [Pokušaj pristupa lipanj 2024.].

[5 Mahesh, »MSE (Mean Squared Error),« Medium, 13 svibanj 2023.. [Mrežno]. Available:
9] https://medium.com/@maheshhkanagavell/mse-mean-squared-error-e2c06c02ab3f. [Pokušaj pristupa lipanj 2024.].

[6 V. Rastogi, »RMSE and MAE,« Medium, 13 kolovoz 2023.. [Mrežno]. Available:
0] https://medium.com/@vaibhav1403/rmse-and-mae-415470f52b58. [Pokušaj pristupa lipanj 2024.].

[6 V. Cheng, »Examples of PaaS Compaines [Updated 2024],« SaasCEO, 2024.. [Mrežno].
1] Available: https://www.saasceo.com/examples-of-paas/. [Pokušaj pristupa svibanj 2024.].

[6 S. R. P. T. S. M. S. K. V. Akshay B R, Machine Learning: A Comprehensive Beginner's
2] Guide, CRC Press, 2024.

[6 GeeksForGeeks, »Time Series Analysis and Forecasting,« GeeksForGeeks, 1 travanj 2024.
3] [Mrežno]. Available: https://www.geeksforgeeks.org/time-series-analysis-and-forecasting/. [Pokušaj pristupa svibanj 2024.].

[6 K. Braun, »How Machine Learning can be used for AI-powered Autoscaling,« Medium,
4] 20 kolovoz 2021.. [Mrežno]. Available: https://medium.com/codex/how-machine-

learning-can-be-used-for-ai-powered-autoscaling-9d03385ff8a. [Pokušaj pristupa svibanj 2024.].

[65] M. Zhang, »Top 10 Cloud Service Providers Globally in 2024,« 18 travanj 2024.. [Mrežno]. Available: https://dgtlinfra.com/top-cloud-service-providers/. [Pokušaj pristupa svibanj 2024.].

[66] J. Fernando, »What Are Autoregressive Models? How They Work and Example,« Investopedia, 6 travanj 2022.. [Mrežno]. Available: https://www.investopedia.com/terms/a/autoregressive.asp. [Pokušaj pristupa svibanj 2024.].

[67] V. Sokolenko, »How to set up a network bridge for virtual machine communication,« Red Hat, 10 ožujak 2022. [Mrežno]. Available: https://www.redhat.com/sysadmin/setup-network-bridge-VM. [Pokušaj pristupa svibanj 2024.].

[68] DSAML, »Model Accuracy in ML,« Medium, 30 listopad 2023.. [Mrežno]. Available: https://dsaml.medium.com/model-accuracy-in-ml-ca8be7bf57e1. [Pokušaj pristupa lipanj 2024.].

[69] J. J. Berman, »Pearson's Correlation,« ScienceDirect, 2016. [Mrežno]. Available: https://www.sciencedirect.com/topics/computer-science/pearson-correlation. [Pokušaj pristupa lipanj 2024.].

[70] M. Steele, »Feature Engineering Examples: Binning Categorical Features,« Medium, 31 ožujak 2021.. [Mrežno]. Available: https://towardsdatascience.com/feature-engineering-examples-binning-categorical-features-9f8d582455da. [Pokušaj pristupa lipanj 2024.].

Popis slika

Popis priloga

Prilog 1

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from xgboost import XGBRegressor
from datetime import timedelta
import pandas as pd
import re
from collections import defaultdict
from datetime import datetime, timedelta
import time
import os
while True:

    log_file_path = '/mnt/access_log/ws1-access.log'
    output_file_path = '/home/deb/dataset/requesti.csv'


    log_pattern = re.compile(r'\[(\d{2}/\w{3}/\d{4}:\d{2}:\d{2}):\d{2} -\d{4}\]')

    request_counts = defaultdict(int)

    first_timestamp = None
    last_timestamp = None

    with open(log_file_path, 'r') as log_file:
        for line in log_file:
            match = log_pattern.search(line)
            if match:
                timestamp = datetime.strptime(match.group(1), '%d/%b/%Y:%H:%M')
                formatted_timestamp = timestamp.strftime('%d-%m-%Y-%H-%M')
                request_counts[formatted_timestamp] += 1
                if first_timestamp is None or timestamp < first_timestamp:
                    first_timestamp = timestamp
                if last_timestamp is None or timestamp > last_timestamp:
                    last_timestamp = timestamp

    if first_timestamp and last_timestamp:
        with open(output_file_path, 'w') as output_file:
            current_timestamp = first_timestamp
            while current_timestamp <= last_timestamp:
                formatted_timestamp = current_timestamp.strftime('%d-%m-%Y-%H-%M')
                count = request_counts.get(formatted_timestamp, 0)
                output_file.write(f'{formatted_timestamp}, {count}\n')
                current_timestamp += timedelta(minutes=1)
    else:
        print("No valid timestamps found in the log file.")

    import csv

    requests_file_path = '/home/deb/dataset/requesti.csv'
    sys_usage_file_path = '/mnt/sys_usage/ws1-usage.csv'
    output_file_path = '/home/deb/dataset/dataset.csv'

    requests_data = {}

    with open(requests_file_path, 'r') as requests_file:
        reader = csv.reader(requests_file)
        for row in reader:
            timestamp = row[0]
            num_requests = row[1]
            requests_data[timestamp] = num_requests
```

```
    with open(sys_usage_file_path, 'r') as sys_usage_file, open(output_file_path, 'w',
newline='') as output_file:
        reader = csv.reader(sys_usage_file)
        writer = csv.writer(output_file)

        headers = next(reader)
        writer.writerow(headers + ['Requests(n)'])

        for row in reader:
            timestamp = row[0]
            if timestamp in requests_data:
                row.append(requests_data[timestamp])
                writer.writerow(row)


    df = pd.read_csv('/home/deb/dataset/dataset.csv')

    df.loc[(df['Requests(n)'] >= 1) & (df['Requests(n)'] <= 200), 'Requests(n)'] = 100
    df.loc[(df['Requests(n)'] >= 200) & (df['Requests(n)'] <= 750), 'Requests(n)'] = 500
    df.loc[(df['Requests(n)'] >= 750) & (df['Requests(n)'] <= 1350), 'Requests(n)'] = 1000
    df.loc[(df['Requests(n)'] >= 1350) & (df['Requests(n)'] <= 2350), 'Requests(n)'] = 2000
    df.loc[(df['Requests(n)'] >= 2350) & (df['Requests(n)'] <= 3250), 'Requests(n)'] = 3000
    df.loc[(df['Requests(n)'] >= 3250), 'Requests(n)'] = 4000

    df['timestamp_dt'] = pd.to_datetime(df['timestamp'], format='%d-%m-%Y-%H-%M',
errors='coerce')

    df.drop(columns=['timestamp_dt'], inplace=True)

    df.to_csv('/home/deb/dataset/cleaned_dataset.csv', index=False, date_format='%d-%m-%Y-
%H-%M')

    df = pd.read_csv('/home/deb/dataset/cleaned_dataset.csv')
    df['timestamp'] = pd.to_datetime(df['timestamp'], format='%d-%m-%Y-%H-%M')
    df = df.rename(columns={'timestamp': 'ds', 'Requests(n)': 'y', 'CPU (%)': 'cpu', 'RAM
(%)': 'ram'})

    df['hour'] = df['ds'].dt.hour
    df['dayofweek'] = df['ds'].dt.dayofweek
    df['minute'] = df['ds'].dt.minute

    X = df[['cpu', 'ram', 'hour', 'dayofweek', 'minute']]
    y = df['y']

    X_train, X_test, y_train, y_test, ds_train, ds_test = train_test_split(X, y, df['ds'],
test_size=0.2, random_state=42)

    xgb_model = XGBRegressor(n_estimators=100, random_state=42)
    xgb_model.fit(X_train, y_train)
    xgb_y_pred = xgb_model.predict(X_test)

    last_timestamp = df['ds'].max()
    future_timestamps = [last_timestamp + timedelta(minutes=i) for i in range(1, 10080)]
    future_df = pd.DataFrame(future_timestamps, columns=['ds'])

    future_df['hour'] = future_df['ds'].dt.hour
    future_df['dayofweek'] = future_df['ds'].dt.dayofweek
    future_df['minute'] = future_df['ds'].dt.minute

    last_known_values = df[['cpu', 'ram']].iloc[-1]
    future_df = future_df.assign(cpu=last_known_values['cpu'],
ram=last_known_values['ram'])
```

```
    future_df['y'] = xgb_model.predict(future_df[['cpu', 'ram', 'hour', 'dayofweek',
'minute']])
    future_df['y'] = future_df['y'].round()

    inicijalna_vr = future_df['y'].iloc[0]
    srednja_vr = future_df['y'].mean()
    zadnja_vr = future_df['y'].iloc[-1]

    log = "/var/log/prediction.log"

    def logiraj(prediction):
        with open(log, 'a') as f:
            f.write(f"{prediction}\n")

    def citaj_log():
        if os.path.exists(log):
            with open(log, 'r') as f:
                lines = f.readlines()
                if lines:
                    return lines[-1].strip()
        return None

    prethodno_stanje = citaj_log()

    if zadnja_vr > inicijalna_vr and srednja_vr > inicijalna_vr:
        stanje_zahtjeva = "Raste"
        if prethodno_stanje in ["Pada", "Stagnira nisko"]:
            print("Ukljucivanje posluzitelja")
            os.system('/opt/scripts/start_server.sh')
        logiraj("Raste")
    elif srednja_vr > inicijalna_vr and zadnja_vr > srednja_vr:
        stanje_zahtjeva = "Stagnira visoko"
        if prethodno_stanje in ["Pada", "Stagnira nisko"]:
            print("Ukljucivanje posluzitelja")
            os.system('/opt/scripts/start_server.sh')
        logiraj("Stagnira visoko")
    elif zadnja_vr < inicijalna_vr and srednja_vr < inicijalna_vr:
        stanje_zahtjeva = "Pada"
        if prethodno_stanje in ["Raste", "Stagnira visoko"]:
            print("Iskljucivanje posluzitelja")
            os.system('/opt/scripts/stop_server.sh')
        logiraj("Pada")
    else:
        stanje_zahtjeva = "Stagnira nisko"
        if prethodno_stanje in ["Raste", "Stagnira visoko"]:
            print("Iskljucivanje posluzitelja")
            os.system('/opt/scripts/stop_server.sh')
        logiraj("Stagnira nisko")


        print(f'Stanje zahtjeva: {stanje_zahtjeva}')
        print(f'Inicijalna vrijednost: {inicijalna_vr}')
        print(f'Srednja vrijednost: {srednja_vr}')
        print(f'Zadnja vrijednost: {zadnja_vr}')

    from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

    def evaluate_model(y_true, y_pred):
        r2 = r2_score(y_true, y_pred)
        mae = mean_absolute_error(y_true, y_pred)
        mse = mean_squared_error(y_true, y_pred)
        rmse = np.sqrt(mse)
        return r2, mae, mse, rmse

    r2, mae, mse, rmse = evaluate_model(y_test, xgb_y_pred)
```

```
print(f'R2 Score: {r2:.2f}')
print(f'MAE: {mae:.2f}')
print(f'MSE: {mse:.2f}')
print(f'RMSE: {rmse:.2f}')

time.sleep(900)
```