SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 899

PRIMJENA STROJNOG UČENJA U OPTIMIZACIJI OPERATIVNOG TROŠKA USLUGA U RAČUNALNOM OBLAKU

Antonio Zemunik

Zagreb, lipanj 2015.

Zahvaljujem svojem mentoru prof. dr. sc. Damiru Seršiću, koji me vodio kroz projekte na diplomskom studiju, na stručnom vođenju, savjetima i znanju koje mi je prenio.

Također, želim zahvaliti mag. ing. Ivani Stupar iz kompanije Ericsson Nikola Tesla, Zagreb koja je dala ideju za ovaj rad te mi pomogla u njegovoj implementaciji i izradi.

Sadržaj

Uvod

U današnjem svijetu velikog broja Internet usluga, razmjene informacija i ogromnih količina podataka, razvio se pojam računarstva u oblaku (engl. *cloud computing*). Temelji se na pružanju računalnih resursa korisnicima te bilježi znatan rast na IT (engl. *information technology*) tržištu u posljednjih desetak godina [1]. Usluga je dostavljena krajnjem korisniku putem Interneta, najčešće uz model naplate koji omogućuje plaćanje onoliko resursa koliko je iskorišteno. Resursi mogu biti računalne mreže, poslužitelji, sustavi za pohranu podataka, aplikacije i slične pogodnosti.

Uz širok izbor mogućnosti raspoređivanja (engl. *deployment*) svojih usluga u okruženje računalnog oblaka, kompanije mogu postići znatnu uštedu. Ipak, često puta ušteda nije u potpunosti ostvarena zbog nedostatka informacija ili znanja kojima se raspolaže. Cilj ovog rada je primjenom strojnog učenja i simulacija temeljenih na zadanom modelu usluge dobiti jasniju sliku o utjecaju pojedinih parametara na trošak izvođenja usluge i njezino ponašanje u računalnom oblaku. Uz to, primjenom strojnog učenja želi se ostvariti mogućnost predviđanja ukupnog troška na temelju ulaznih značajki čime bi odluka o konfiguraciji okoline računalnog oblaka, u koju se razmatrana usluga smješta, bila znatno olakšana. Struktura rada opisana je u nastavku.

Prvo poglavlje rada daje jasniju sliku o računalnom oblaku i zašto ga kompanije u IT svijetu koriste. Prikazana je pozadina cjelokupnog koncepta te najvažnije karakteristike i prednosti. Navedeni su i opisani razvojni modeli, tipovi usluga i najčešće naplatne metode računalnog oblaka.

Drugo poglavlje opisuje koncept strojnog učenja. Navodi njegove tipične primjene i osnovnu podjelu. Uvodi se pojam regresije te razlika između linearnog i polinomijalnog regresijskog modela. Objašnjava podjelu na skup za treniranje i ispitni skup te pojam regularizacije.

Treće poglavlje daje opis čitave implementacije projekta i primijenjenih postupaka. Dan je prikaz korištenih alata i njihovih mogućnosti te je opisan cjelokupni pogled na projekt uz podjelu na dvije domene. Navedeni su promatrani ulazni i izlazni parametri, referentna aplikacija namijenjena ispitivanju u okolini

računalnog oblaka i implementacija dodatnih mogućnosti u korištenom simulatoru otvorenog koda. Simulacije ponašanja aplikacijskih usluga unutar okruženja računalnog oblaka čine ulaze u regresijske modele, a primjena strojnog učenja nad dobivenim podacima opisana je u završetku ovog poglavlja.

Četvrto poglavlje sadržava pregled rezultata regresijskih modela i usporedbu među njima. Prikazani su parametri s najvećim značajem za pojedinu simulaciju, odnosno njihov utjecaj na izlazne vrijednosti troška i vremena izvođenja aplikacijske usluge u računalnom oblaku. Uz numeričke rezultate priloženi su grafovi koji daju uvid u sposobnost predikcije dobivenih modela.

U konačnici je izveden zaključak koji sumira sve što je proizašlo iz ovog projekta. Daje komentar dobivenih rezultata i pogled na mogućnosti nastavka istraživanja.

1. Računarstvo u oblaku

Računarstvo u oblaku označava model koji omogućava sveprisutan i prikladan mrežni zahtjev za dijeljenim skupom podesivih računalnih resursa (primjerice, mreže, poslužitelji, skladištenje, aplikacije i usluge). Do takvih resursa moguće je prilično brzo i lako doći uz minimalan kontakt s pružateljem usluge. Koncept računalnog oblaka formiran je od pet esencijalnih karakteristika, tri uslužna modela te četiri razvojna modela. [2]

Navedenih šest karakteristika zajedničke su gotovo svim *cloud* okruženjima:

- korištenje na zahtjev (engl. *on-demand-usage*)

Korisnik računalnog oblaka može jednostrano pristupiti IT resursima oblaka. Jednom uspostavljeno korištenje tih resursa može biti automatizirano, odnosno u budućnosti nije potrebno djelovanje krajnjeg korisnika ili pružatelja usluge (engl. *provider*-a).

- visoka razina pristupa

Usluge računalnog oblaka nude visoku mogućnost pristupa. Takva pogodnost često zahtijeva podršku za različite uređaje, prijenosne protokole, sučelja i sigurnosne tehnologije. Resursi su dostupni preko mreže i putem raznih standardnih mehanizama koji omogućavaju korištenje različitih klijentskih platformi (primjerice, mobiteli, tableti i prijenosna računala).

- povezivanje resursa

Računalni resursi (skladištenje, memorija, CPU, mrežna propusnost itd.) pružatelja usluga povezani su te rade na principu višezakupnog (engl. *multitenant*) modela. Različiti fizički i virtualni resursi dinamički se dodjeljuju većem broju korisnika prema njihovim zahtjevima. Korisnik ne zna točnu lokaciju resursa koje koristi, ali je u mogućnosti dobiti informaciju lokacije na većoj razini apstrakcije (primjerice, država ili podatkovni centar u kojem se nalaze korišteni resursi).

- elastičnost

Elastičnost je automatizirana sposobnost računalnog oblaka da transparentno skalira IT resurse, bilo prema potrebama izvođenja aplikacija (primjerice, uslijed povećanih zahtjeva krajnjih korisnika usluge) ili prema unaprijed postignutom dogovoru korisnika i pružatelja usluge. Ova karakteristika se često

smatra kao srž prihvaćenosti *cloud computing*-a. *Provider*-i, koji imaju ogromne količine IT resursa u svom posjedu, u mogućnosti su ponuditi najveći opseg elastičnosti. S korisničke strane, dobiva se dojam da su količine resursa neograničene i dostupne u svakom trenutku.

- mjerenje korištenja resursa (engl. *measured usage*)

Measured usage karakteristika predstavlja sposobnost *cloud* platforme da prati korištenje svojih IT resursa, primarno od strane korisnika. Vodeći se mjerenjima, pružatelj usluge može naplatiti korisniku samo resurse koji su korišteni i to u vremenskom okviru unutar kojeg je korisnik imao pristup njima, a transparentnost je osigurana za obje strane. U ovom kontekstu, mjerenje korištenja usko je povezano s karakteristikom korištenja na zahtjev. Ovaj mehanizam nije ograničen samo na svrhu naplaćivanja usluge. Također obuhvaća generalni nadzor resursa te je važan za računalne oblake koji ne naplaćuju svoje korištenje (primjerice, privatan oblak o kojem će biti riječi u poglavlju 1.1).

- otpornost

Otpornost u ovom smislu osigurava zaštitu *cloud* sustava od kvara. Problem je riješen redundantnim implementacijama kroz fizičke lokacije. IT resursi su često konfigurirani tako da, ukoliko se na jednom dogodi kvar, proces se automatski nastavlja na drugi, redundantan resurs. U konceptu računalnog oblaka redundantni resursi mogu se nalaziti unutar istog oblaka (ali, na različitim lokacijama) ili u drugom oblaku. Korisnici na ovaj način mogu osigurati pouzdanost i dostupnost svojih aplikacija. [3]

Pojam računalnog oblaka označava koncept korišten u IT infrastrukturi posljednjih desetak godina. Označava aplikacije isporučene kao usluge preko Interneta te, s druge strane, hardver i sistem softver u podatkovnim centrima koji pružaju te usluge. Usluge se još nazivaju i *SaaS* modelom (softver kao usluga, engl. *Software as a Service*), dok se za infrastrukturu podatkovnih centara koristi naziv c*loud*.

Krajnji korisnik, primjerice kompanija koja želi izgraditi veliki informacijski sustav, trebala bi uložiti velika financijska sredstva u potrebno sklopovlje (poslužitelje, usmjernike i sl.), u upravljanje tim sklopovljem (napajanje, hlađenje) i u kadar koji bi se brinuo o sustavu. Korištenjem oblaka ovakva kompanija iznajmljuje potrebnu infrastrukturu te dobiva resurse koji su joj potrebni.

Investicijski troškovi su, na taj način, uvelike smanjeni, a tvrtke diljem svijeta koriste ovaj IT koncept. [1]

Osim već navedenih karakteristika, važne prednosti *cloud computing*-a su: centralizacija, kontroliran korisnički pristup i sigurnost podataka. Svi podaci pohranjeni su na jednom mjestu, odakle su stalno dostupni korisniku, gdje god se on nalazio.

Korisničkom administracijom lako je ograničiti pristupe korisnicima, primjerice pogled na dio informacija, izmjenu, brisanje ili izvoz podataka. Mogućnost gubitka podataka iznimno je mala te postoji čitava paleta sigurnosnih mehanizama. Jedan od njih su sigurnosne kopije podataka koje obično pružaju davatelji ovih usluga. [4]

Ciljevi i dobrobiti *cloud computing* koncepta su:

- smanjena ulaganja i troškovi

Pružatelji usluga baziraju svoj poslovni model na masovnom prikupljanju IT resursa koje nude krajnjim korisnicima po povoljnoj cijeni zakupa. To otvara vrata poduzećima koja imaju pristup snažnoj infrastrukturi bez da investiraju u kupovinu iste. Ovakav koncept, bez da su tvrtke od početka pod velikom financijskom obavezom, omogućava im da krenu s malim zakupom resursa i, prema odgovarajućim potrebama, povećaju svoju potražnju.

- povećana skalabilnost

Zahvaljujući skupu IT resursa koje nudi *provider*, zajedno s alatima i tehnologijama, računalni oblak omogućava *cloud* korisnicima trenutno i dinamičko alociranje IT resursa.

- povećana dostupnost i pouzdanost

Dostupnost i pouzdanost IT resursa su u izravnoj vezi s opipljivim poslovnim dobitima. Kvarovi za vrijeme izvođenja aplikacije mogu ostaviti značajan negativan utjecaj, naročito ako u tim trenucima postoji ogroman broj zahtjeva - kako za aplikacijom, tako i za resursima. Obilježje tipičnog *cloud* okruženja je dodatna podrška resursa kako bi se minimizirala mogućnost zastoja ili kvara. Jako je važno da predstavnici tvrtke pažljivo provjere SLA uvjete (engl. *service-level agreement*) koje im nudi pružatelj usluge prilikom zakupa *cloud* usluga i resursa. Iako brojna okruženja računalnog oblaka nude visoki nivo dostupnosti i pouzdanost, usluga se svodi na garanciju napisanu u SLA uvjetima koji tipično predstavljaju stvarne ugovorne obaveze. [3]

1.1. Razvojni modeli

Koncept računalnog oblaka dijeli se na tri razvojna (implementacijska, engl. *deployment*) modela.

Privatni računalni oblak (engl. *private cloud*) je infrastruktura namijenjena isključivo za korištenje od strane jedne kompanije i njezinih korisnika, primjerice poslovnih jedinica. Može biti u vlasništvu kompanije, neke treće strane ili u kombinaciji. Isto vrijedi i za korištenje, odnosno upravljanje resursima oblaka. Ne mora nužno biti smješten u prostorima tvrtke.

Pojam zajedničkog oblaka (engl. *community cloud*) odnosi se na infrastrukturu namijenjenu korištenju od strane specifičnog društva korisnika. Korisnici obično pripadaju jednoj ili više organizacija te imaju slične zahtjeve (primjerice, slične zadatke, sigurnosne zahtjeve, potrebe korištenje računalnog oblaka itd.). Ovakav tip oblaka može biti u vlasništvu jedne ili više organizacija. Također, postoje slučajevi kada društvenim oblakom upravlja i treća strana.

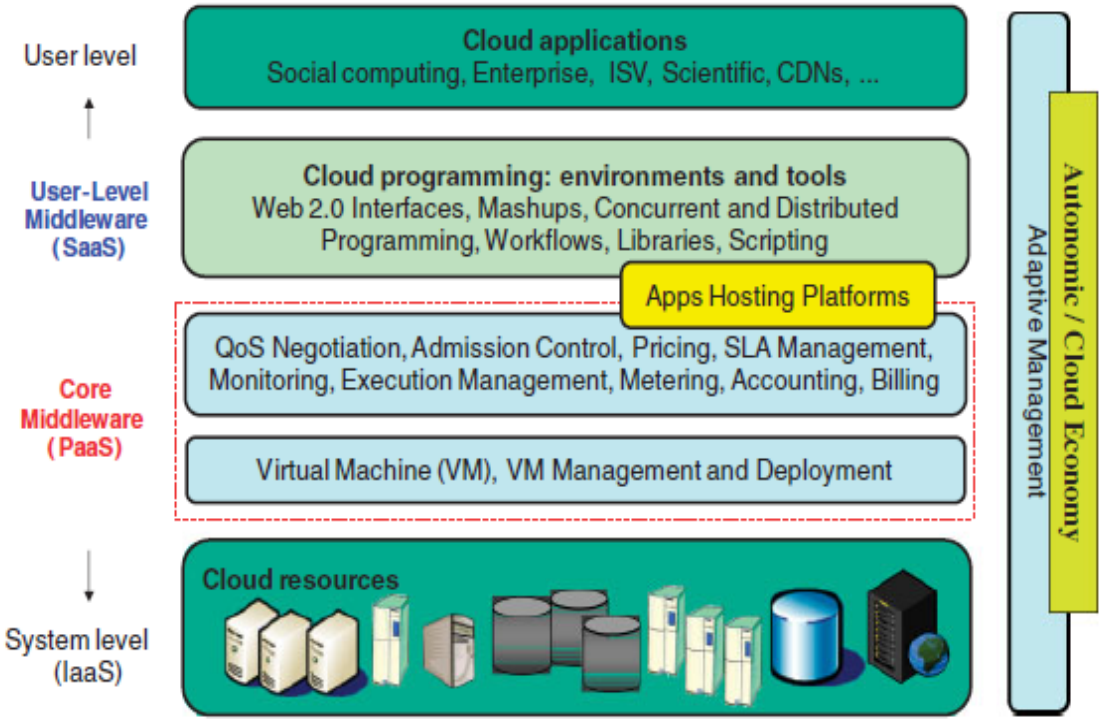
Javni oblak (engl. *public cloud*) je namijenjen otvorenom korištenju, odnosno cjelokupnoj javnosti. Najčešće je u vlasništvu poslovnih, akademskih ili vladinih organizacijskih tijela. Infrastruktura javnog računalnog oblaka nalazi se u okvirima pružatelja usluge.

Hibridni oblak (engl. *hybrid cloud*) označava koncept kompozicije dviju ili više odvojenih *cloud* infrastruktura, bilo privatne, javne ili društvene. Svaka infrastruktura ostaje jedinstveni entitet u ovoj cjelini, ali su međusobno povezane standardiziranim ili vlasnički određenim tehnologijama koje omogućavaju prenosivost podataka i aplikacija. Dobar primjer važnosti zajedničkih tehnologija je balansiranje opterećenja između *cloud* entiteta ove cjeline. [2]

1.2. Modeli usluge u računalnom oblaku

Tri osnovna tipa usluga u konceptu oblaka nazivaju se:

- Infrastruktura kao usluga (engl. *Infrastructure as a service, IaaS*)
- Platforma kao usluga (engl. *Platform as a Service, PaaS*)
- Softver kao usluga (engl. *Software as a Service, SaaS*)



Slika 1. Arhitektura računalnog oblaka po slojevima [5]

Slika 1 prikazuje računalni oblak razložen po slojevima. Fizički *cloud* resursi zajedno s jezgrenim slojem čine *IaaS* i *PaaS* usluge. Korisnički sloj predstavlja *SaaS* model. *PaaS*/*SaaS* usluge često imaju drukčije pružatelje usluga u odnosu na *IaaS* sloj.

Vršni nivo odnosi se na aplikacije koje koriste usluge (infrastrukturu i platformu) računalnog oblaka. Takve aplikacije dostupne su izravno krajnjim korisnicima, a dobar primjer njihovog predstavnika u današnje vrijeme bile bi društvene mreže. Model plaćanja *cloud* pružatelju usluge, koji je vlasnik aplikacije dužan platiti, baziran je na pretplati ili na *pay-per-use* modelu.

1.2.1. Infrastruktura kao usluga

Iaas model označava skup računalnih, memorijskih i mrežnih resursa računalnog oblaka koji omogućavaju korisniku smještanje i pokretanje proizvoljnog softvera, uključujući operacijske sustave i aplikacije. Pružatelji ove usluge upravljaju čitavom infrastrukturom te korisnik može unajmiti fizička ili, još češće,

virtualna računala i ostale resurse za izvođenje programa, skladištenje i umrežavanje (engl. *networking*).

Računalna snaga u *cloud* okruženjima sastoji se od niza podatkovnih centara (engl. *datacenters*). Centri sadrže nekoliko stotina ili tisuća host (glavnih) računala. Podatkovne centre opskrbljuju memorijski i aplikacijski poslužitelji kojima upravljaju virtualna računala. Takva računala međusobno su izolirana čime je povećana sigurnost njihovog rada.

Kod ovog modela korisnik sam upravlja ravnotežom opterećenja (engl. *load balancing*) sustava te sukladno tome odabire broj virtualnih strojeva koje će koristiti. Virtualni strojevi omogućavaju pokretanje aplikacije, skladištenje podataka i ostale vezane usluge, a korisnik sam brine o eventualnoj nadogradnji i zakupu dodatnih resursa. [5]

Osim opisane skalabilnosti, dodatne karakteristike *IaaS* sloja su automatizacija administrativnih zadataka i usluga bazirana na pravilima (engl. *policy-based service*).

Naplata korisnicima izvršava se prema potrošenoj količini (*pay-as-you-go* model), a kao mjerna jedinica koriste se sat, tjedan ili mjesec. Također, nije rijetkost da korisnici snose trošak prema količini virtualnih jedinica koje njihova aplikacija koristi, a takav način naplate naziva se *on-demand-instance*. [6]

IaaS model naziva se još i sistemski nivo računalnog oblaka. Najčešći primjeri ovog sloja su: *Amazon Web Services (AWS), Microsoft Azure, Google Compute Engine (GCE) i Joyent*. [7]

1.2.2. Platforma kao usluga

Ovaj sloj implementira usluge platformnog nivoa koji nudi okruženje za izvođenje, upravljanje i smještanje (engl. *host*) aplikacija *SaaS* sloja. Programeri kroz ovaj sloj dobivaju razvojni okvir (engl. *framework*) koji koriste za razvoj i prilagodbu svojih aplikacija. *PaaS* omogućava razvoj i testiranje aplikacija te njihovo postavljanje na računalni oblak čini znatno lakšim. Funkcionalnosti platformnih usluga su razmjena poruka među slojevima, otkrivanje novih usluga i mogućnosti te održavanje ravnoteže opterećenja. Stoga ove funkcionalnosti koriste *IaaS* i *SaaS* usluge. Obično su implementirane od strane pružatelja usluge

računalnog oblaka (engl. *cloud providers*). Većina *PaaS* platformi je usmjerena prema razvoju softvera. Uz to, često puta programeri aplikacija imaju mogućnost nadogradnje značajki operacijskog sustava i poboljšanja mehanizama ovog sloja. Primjerice, Amazon nudi razvojnim programerima iz Amazon EC2 odjela mogućnost implementacije mehanizama održavanja ravnoteže i upravljačkih usluga (Cloudwatch) u ovom sloju. Također, Microsoft Azure razvojni inženjeri koriste .NET Service Bus za implementaciju mehanizma razmjene poruka. [5]

Dakle, *PaaS* model krajnjim korisnicima nudi razvojne resurse uključujući razvojne okvire, virtualna računala i operacijski sustav. Pružatelj usluge održava infrastrukturu oblaka, operacijske sustave i softver koji služi kao podrška korisničkim aplikacijama. Platforma u oblaku omogućava korisnicima jednostavan pristup resursima uz model naplate samo onih resursa koje aplikacija stvarno koristi. Također, određeni pružatelji *PaaS* usluga naplaćuju korištenje platforme i postavljanje aplikacija prema fiksnoj mjesečnoj cijeni. Razvojni programeri ne stvaraju izravno virtualne strojeve, nego razvijaju aplikaciju na postojećoj platformi. Sve što korisnik treba napraviti jest na postojećoj platformi stvoriti i popuniti bazu podataka, učitati vlastitu aplikaciju i pokrenuti je, obično kroz Web preglednički sučelje. Za ostale zadatke, poput ravnoteže opterećenja resursa i ažuriranja operacijskog sustava na virtualnim strojevima odgovoran je pružatelj usluge. Zahvaljujući *PaaS* uslugama, korisnik je posvećen isključivo svojoj aplikaciji, a ostali zadaci su u domeni pružatelja usluge. [8]

Glavni nedostatak ovakvog koncepta jest smanjena ovlast i kontrola korisnika. Usluga nameće određena ograničenja kao što je platforma na kojoj korisnik radi koja je unaprijed određena. Dodatni rizik je visoka ovisnost o pružatelju usluge. Moguće je da u jednom trenutku pružatelj *PaaS* usluge prekine s radom ili promijeni razvojnu okolinu. Primjerice, ako pružatelj usluge prekine s podrškom za određeni programski jezik - korisnik je primoran promijeniti pružatelja *PaaS* usluge ili tehnologiju na kojoj je bazirana aplikacija. Oba slučaja su iznimno skupa i nepraktična rješenja.

Najpoznatiji primjeri *PaaS* usluga su *Appear IQ, Mendix, Amazon Web Services (AWS) Elastic Beanstalk, Google App Engine i Heroku*. [9]

1.2.3. Softver kao usluga

Softverski sloj, ili drugim nazivom korisnički sloj (engl. *User-Level middleware*) uključuje softverske radne okvire, kao što su Web 2.0 sučelja (Ajax, IBM Workplace). Takvi *framework* okviri pomažu programerima (*developer*-ima) u stvaranju bogatih korisničkih sučelja za aplikacije vezane uz Internet preglednike (*browser-based applications*). Ovaj model nudi programska okruženja i alate koji omogućavaju jednostavniji razvoj i izvedbu *cloud* aplikacija. Također, ovaj nivo podržava *framework*-e za razvoj višeslojnih aplikacija, a neki od poznatijih su Spring i Hibernate. [5]

SaaS postaje prevladavajući model u svijetu tehnologija koje podržavaju Web servise i servisno-orijentiranu arhitekturu (engl. *service-oriented architecture*, SOA).

Ovaj sloj se povezuje s pružateljem aplikacijskih usluga (ASP, engl. *application service provider*) i *on-demand-computing* softver modelom. Dva su slična pristupa za *SaaS*. Prvi je sličan ASP-u, pružatelj usluge omogućava izvođenje, odnosno *host*-a, komercijalno slobodni softver i postavlja ga na Web. U *on-demand* modelu pružatelj usluge daje krajnjim korisnicima mrežni pristup kopiji aplikacije napravljenoj isključivo za *SaaS* distribuciju.

Najveće prednosti ove usluge su: jednostavna administracija, automatsko ažuriranje i upravljanje nadogradnjom, kompatibilnost i suradnja (jer svi korisnici imaju istu softver verziju) te globalna dostupnost. [10]

SaaS omogućava pristup aplikacijama i uslugama putem Interneta bez prethodne instalacije na računalo. Krajnji korisnici pristupaju aplikaciji iz različitih uređaja putem klijentskih sučelja, kao što je Web preglednik ili programsko sučelje. Pružatelj *SaaS* usluge odgovoran je za održavanje aplikacije. Korisnik nema pristup *cloud* infrastrukturi, koja se nalazi ispod aplikacije, gdje se ubrajaju propusne mreže, poslužitelj, operacijski sustavi, skladištenje i ostale specifikacije.

Korištenjem *SaaS* modela tvrtka se suočava s manjim financijskim rizikom. Najam softvera je izveden po principu plaćanja točno onoliko koliko se koristi (engl. *pay-per-use*) te nema potrebe za nabavljanjem dodatne opreme, licence i lokalne instalacije.

Najveći problem ovakvog pristupa su pravna i regulatorna pitanja. Neke zemlje, uključujući Hrvatsku, imaju zakonom zabranjenu pohranu podataka izvan

državnih granica. Pružatelji usluga u oblaku često puta imaju podatkovne centre smještene u više država, stoga njihovi korisnici koriste resurse koji se nalaze izvan zemlje u kojoj stanuju. [9]

Najpoznatiji primjeri *SaaS* usluga su: *Google Apps*, *Salesforce*, *Workday*, *Concur*, *Citrix GoToMeeting* i *Cisco WebEx*. Među *Google Apps* uslugama svakako se ističu *Gmail, Google Groups, Google Calendar, Google Talk, Google Docs i Google Sites*. Tvrtka *IBM* nudi aplikaciju *Lotus* za međusobnu suradnju zaposlenika unutar poduzeća, dok *Zoho* pruža cijeli set usluga u oblaku: *CRM* (engl. *customer relationship managment*), *Mail*, *Docs*, *Meeting* i brojne druge. *SAP* nudi aplikaciju za planiranje resursa kompanije *ERP* (engl. *Enterprise Resource Planning*), a *NetSuite* ima svoje inačice *ERP*-a i *CRM*-a.

1.3. Modeli naplate u računarstvu u oblaku

Četiri najčešća modela naplate su:

- naplata točno onoliko koliko je potrošeno, gotovina (engl. *cash pay-as-you-go*)
- naplata prema korištenim virtualnim jedinicama (engl. *committed VM*)
- naplata prema udruženim resursima (engl. *resource pooling*)
- naplata točno onoliko koliko je potrošeno, kredit (engl. *credit pay-as-you-go*) [11]

		PAYMENT TERMS	MEDIUM OF EXCHANGE	COMMITMENT	METERING	UNIT PRICE VARIABILITY
CASH PAY-AS-YOU-GO	On-Demand	In-Arrears	Cash	None	Metered	Fixed
	Spot Pricing	In-Arrears	Cash	None	Metered	Variable
	Reserved Instance	Hybrid	Cash	None	Metered	Fixed
COMMITTED VM	Prepaid VM	Up-front	Cash	None	Unmetered	Fixed
	Recurring VM	Up-front	Cash	Recurrent	Unmetered	Fixed
RESOURCE POOLING	Recurring Resource Pooling	Up-front	Resources	Recurrent	Metered	Fixed
CREDIT PAY-AS-YOU-GO	Subscription Credit	Up-front	Credits	Recurrent	Metered	Fixed
	Prepaid Credit	Up-front	Credits	None	Metered	Fixed

Slika 2. Modeli naplate u računalnom oblaku [11]

Na slici 2 prikazane su četiri grupe najčešćih naplatnih modela u današnjem *cloud* svijetu. Metode variraju prema uvjetima plaćanja, korištenim sredstvima razmjene i mjerenjima te varijabilnosti jedinične cijene resursa.

Svaki od navedenih modela ima određen broj naplatnih atributa. U ovakvim naplatnim modelima često dolazi do grupiranja. Primjerice, neki pružatelji usluge nude izbor virtualnih jedinica od kojih korisnik može odabrati onu koja mu odgovara. S druge strane, određeni *IaaS* pružatelji dozvoljavaju kupcima da biraju virtualni stroj točne veličine i specifikacija. Procesor, radna memorija i skladištenje su neke od specifikacija koje čine virtualnu jedinicu. Dakle, u prvom slučaju ti resursi su upakirani (engl. *bundled*) i korisnik plaća fiksnu cijenu VM. U drugom primjeru korisniku je naplaćena svaka od specifikacija po određenoj cijeni. *IaaS* tržište je danas isprofiliralo četiri navedena modela naplate, pri čemu je svaki detaljnije opisan u nastavku.

1.3.1. *Cash pay-as-you-go* model

1.3.1.1. Kupovina na zahtjev (*On-demand*)

Ovakav model naplate izvršava se tako da korisnik plaća potrošnju korištenih jedinica virtualnih mašina prema fiksnoj cijeni jedinice. Navedeni pristup osigurava korisniku veliku fleksibilnost dozvoljavajući kupnju i isporuku resursa po potrebi, bez prethodne najave. Vjerojatno se radi o najskupljoj opciji koju nudi pružatelj usluge, stoga kod dugoročnijih zahtjeva, kao što su osnovna opterećenja, trošak raste jako brzo. Također se radi o metodi koju je najteže kontrolirati. Primjerice, ako su virtualne mašine pokrenute te, ljudskom ili aplikacijskom pogreškom, nisu ugašene (terminirane) na vrijeme, korisnik će platiti visoku cijenu na kraju mjeseca.

Sa strane pružatelja usluge, *on-demand* metoda je najmanje učinkovita jer nailazi na brojne rizike. Pružatelj usluge mora predvidjeti potražnju resursa i izgraditi infrastrukturu prije nego je puštena na tržište. Bez ikakvih prethodnih zahtjeva, ovakvo predviđanje može biti jako veliki poslovni rizik. Ako ponuđeni kapacitet nije iskorišten tijekom određenog perioda, dolazi do financijskog gubitka. S druge strane, ako je ponuđeni kapacitet manji od potreba korisničkih zahtjeva,

posao polako propada i korisnici prekidaju suradnju s takvim pružateljem usluge. Jednostavno nisu u mogućnosti skalirati svoje zahtjeve sukladno zahtjevima aplikacije. To je osnovni problem kod prodaje usluge s "neograničenim resursima".

Dodatan rizik za pružatelja usluge jest u tome da korisnici ne plaćaju potrošnju unaprijed kod ovakve metode naplate. Uvijek postoji mali rizik da će resursi biti korišteni, ali ne i naplaćeni u konačnici. Plaćanje unaprijed za pružatelja usluge bi bilo dobro jer bi mogao uložiti dodatna sredstva u infrastrukturu i stručni kadar.

Usprkos gore navedenim problemima, ova metoda je dosta popularna u *cloud computing* svijetu. Njezina najveća prednost gledana sa strane korisnika jest skalabilnost zahtjeva za resursima, a to je ipak samo središte pozitivnih strana računalnog oblaka.

1.3.1.2. Rezervirana instanca (*Reserved instance)*

Kod *reserved instance* naplate korisnik plaća fiksnu naknadu unaprijed te dobiva popust za potrošnju koja se plaća prema fiksnoj cijeni jedinice. Za unaprijed plaćenu naknadu dobiva popust pri plaćanju virtualnih jedinica u fiksnom razdoblju, najčešće tijekom jedne ili tri godine. Ponude najčešće ovise o razini potrošnje (korištenja). Za veći nivo korištenja, pružatelj usluge zahtjeva veću naknadu plaćenu unaprijed, ali su cijene troška resursa niže.

Jedini pravi rizik za korisnika ovdje je mogućnost da se unaprijed plaćena naknada neće isplatiti, odnosno da virtualni stroj nije efikasno iskorišten.

Ova metoda omogućava pružatelju usluge dobivanje dijela prihoda od virtualnih jedinica unaprijed. Na taj način lakše je planirati, investirati u infrastrukturu i ulaziti u neki novi rizik. Ako pružatelj usluge nudi raspon virtualnih jedinica s različitim razinama korištenja, prodaja veće mašine donosi veći prihod plaćen prije korištenja dotične mašine.

Rezervirane instance su naročito privlačne korisnicima koji imaju raspon korištenja virtualnih strojeva za razna trajanja tijekom godine. Primjerice, aplikacija može imati osnovnu VM za svakodnevne operacije. Ali, uz to može imati i dodatnu VM koju koristi 50% godine za sezonske operacije. Također, može imati i dodatnu VM koja procesira tijekom 75% godine, ali je ugašena tijekom sezonskog zatišja.

Korisnik može ostvariti uštedu odabirući nisku, srednju ili visoku razinu potrošnje, naročito ako se radi u duljem vremenskom periodu. Upravo zato, rezervirane instance su praktičan odabir modela naplate za korisnike koji imaju značajnu varijabilnost potražnje virtualnih strojeva tijekom određenog vremena i u mogućnosti su unaprijed predvidjeti minimalne zahtjeve. Dobar primjer takvih korisnika čine velike i afirmirane *SaaS* usluge i e-trgovačke Web stranice.

1.3.1.3. Referentna cijena (*Spot pricing*)

Kod *spot pricing* modela cijena resursa varira s vremenom, a korisnici se nadmeću ponudama (licitiraju) kako bi ostvarili pristup ovim resursima. Naplata korištenja resursa se plaća naknadno. Ovo je još uvijek rijetko korištena metoda naplate.

Glavna točka s korisničke strane kod ove metode jest nova razina kompleksnosti predviđanja i izračuna potrošnje. Kod *on-demand* naplate jedina varijabla jest sama potrošnja - ponekad će više resursa biti potrebno, a ponekad manje, no cijena jedinice je uvijek fiksna. *Spot pricing* sadrži novu varijablu jer su potrošnja i cijena jedinice vremenski promjenjive, stoga su raspodjela proračuna i procjena troška izuzetno otežane. Korisnik postavlja ponuđenu cijenu. Ako je referentna cijena niža od ponuđene, korisnik plaća referentnu cijenu i ima dozvolu pristupa resursu. No, ukoliko je referentna cijena viša, korisnik gubi pravo pristupa resursu i ne plaća nikakvu cijenu.

Ekonomski gledano, ako pružatelj usluge nudi *spot* i *on-demand* naplatne modele, a korisnikova aplikacija je visoke razine otpornosti, ponuđena cijena trebala bi biti postavljena tako da bude jednaka *on-demand* cijeni. U slučaju kada je referentna (*spot*) cijena niža od ponuđene, čista ušteda je ostvarena u odnosu na *on-demand* cijenu resursa. S druge strane, ako je referentna cijena viša, instanca će biti prekinuta i virtualna mašina će automatski započeti na *on-demand* naplatom, stoga nema dodatnog troška.

Spot pricing je potencijalno moćna metoda za pružatelje usluga te u suštini djeluje kao način kojim *IaaS* pružatelj može kontrolirati potražnju potrošača. Za vrijeme niske potražnje pružatelj može sniziti cijene kako bi više resursa bilo prodano i infrastruktura bolje iskorištena. Nasuprot toga, u vrijeme visoke

potražnje, pametno bi bilo podići cijene kako ne bi došlo do prevelike zauzetosti resursa te kako bi bili ispoštovani uvjeti ugovora o razini usluge (SLA).

1.3.2. *Committed VM*

1.3.2.1. Unaprijed plaćena virtualna jedinica (*Prepaid VM*)

Model plaćanja pristupa virtualnom stroju unaprijed nalaže da korisnik treba podnijeti trošak fiksne naknade pristupa jedinici specificirane veličine i karakteristika. Nakon toga korisnik ima pravo neograničenog pristupa virtualnoj jedinici do kraja dogovorenog roka.

Ovo je najmanje fleksibilna metoda za korisnika te je prilično često u uporabi današnjih pružatelja usluga računalnog oblaka. Nakon kupovine virtualne mašine zamjena više nije dozvoljena i korisnik je obično koristi do kraja dogovorenog roka. Međutim, korisnik nije zaštićen od porasta cijene do kojeg može doći. Neki od uzroka poskupljenja bili bi inflacija ili porast cijene energije. Također, ukoliko se *IaaS* pružatelj nađe pred stečajem, trošak virtualne jedinice plaćen unaprijed ne može biti vraćen. Ipak, budući da je VM plaćena unaprijed, ne postoji rizik da će troškovi korisnika drastično porasti. U kombinaciji s *on-demand* naplatom ova metoda može ponuditi jeftino osnovno rješenje za dulji vremenski period.

Situacija za pružatelja usluge je jako dobra jer je usluga potpuno plaćena unaprijed. Omogućeno je planiranje unaprijed budući da korisnik unaprijed određuje specifikacije resursa koje će koristiti. Uz to, *IaaS* pružatelj unaprijed dobiva sredstva za izgradnju potrebne infrastrukture. Ovakav model je najatraktivniji potrošačima s niskom varijabilnošću VM zahtjeva, ali im svejedno treba osnovni kapacitet resursa. Najčešće su to afirmirane kompanije koje su sigurne u svoje buduće zahtjeve i potrebe.

1.3.2.2. Unaprijed plaćena virtualna jedinica uz obročno plaćanje (*Recurring prepaid VM*)

Ova naplatna metoda obvezuje korisnike na obročno plaćanje naknade (najčešće mjesečno) za pristup instanci određenih karakteristika.

S pristupa korisnika, rizici su slični kao kod *prepaid* metode. Ako pružatelj usluge dođe do bankrota, korisnik gubi novac u iznosu naknade jednog mjeseca.

Za pružatelja usluge, rizici su, također, slični kao kod prethodnog modela naplate uz dodatne rizike ukoliko *provider* odluči prekinuti uslugu i ne ispoštuje dogovoreni rok. Kapital za infrastrukturna ulaganje ne dobiva se na početku dogovorenog roka, već u mjesečnim ratama. Metoda je posebno privlačna korisnicima s malom varijacijom u VM zahtjevima tijekom duljeg perioda, ali uz osnovne kapacitete resursa (primjerice manje web stranice).

1.3.3. *Resource pooling*

1.3.3.1. Udruživanje resursa uz obročno plaćanje (*Recurring resource pooling*)

Ovakav model obvezuje korisnike na zakup količine resursa (broja procesorskih jezgri, količine RAM memorije itd.) unaprijed prema ponavljajućoj osnovi. Sljedeća potrošnja resursa je izvučena na temelju prethodnog zakupa. Ova metoda je obično uparena s *on-demand* naplatom dozvoljavajući, na taj način, korisniku da rezervira određen nivo resursa te, uz to, zakupi dodatne resurse prema *on-demand* modelu ukoliko je to potrebno. Rezervirana količina resursa može biti iskorištena za stvaranje virtualnih jedinica bilo kakve veličine, sve dok je u granicama količine zakupljenih resursa.

Po pitanju predviđanja potrošnje, ova metoda ima visoku razinu zrnatosti. Potrošači trebaju imati detaljan pogled na svoje zahtjeve ako ne žele imati prevelike troškove ili nedostatke. Primjerice, ako korisnik predvidi da mu je potrebno 100 GB RAM-a po satu i 1 CPU jedinica po satu, ali se u konačnici ispostavi da je potrošio 1 GB RAM memorije i 100 procesorskih jedinica, 99 GB RAM memorije je uzalud zakupljeno i 99 CPU jedinica je potrebno dodatno kupiti. Ovakva granularnost izlaže krajnjeg korisnika visokom riziku. Velika prednost ovog

modela jest da resursi mogu biti preusmjereni na druge projekte ili odjele u kompaniji bez potrebe za kupovinom dodatnih *on-demand* resursa.

Za pružatelja usluge situacija je ovdje dosta povoljna. Metoda nudi detaljnu i specificiranu prognozu buduće potražnje. Kroz udruživanje resursa *IaaS* pružatelj može napraviti detaljnu procjenu minimalnog zahtijevanog kapaciteta. Uz to, svakog mjeseca dobiva financijski priljev što je odličan preduvjet za planiranje i daljnja ulaganja.

Ovakav model interesantan je korisnicima koji u potpunosti razumiju *cloud* arhitekturu koju koriste i buduće zahtjeve. Tipično, to neće biti potrošači čija će VM potražnja uvelike varirati u vremenskom periodu.

1.3.4. *Credit pay-as-you-go*

1.3.4.1. Unaprijed plaćena potrošnja (*Prepaid consumption*)

Prepaid consumption nalaže korisniku plaćanje na početku roka, a troškovi korištenja resursa su pokriveni s tog unaprijed plaćenog iznosa. Često se taj novac gleda kao kredit pa korisnik koji unaprijed plaća ima veći kredit u odnosu na onog koji je istu količinu novca uplatio sa zaostatkom. Također, moguće je i da korisnik koji unaprijed plaća bude nagrađen nižim cijenama resursa.

Glavni rizik za korisnika jest taj da kupljeni kredit kasnije neće biti potrošen. Rizik je, svakako, najmanji u slučajevima kada korisnici mogu tražiti nadoknadu nepotrošenog kredita, odnosno kada kredit ne može isteći. Tada je uloženi novac uvijek moguće dobiti natrag. Ukoliko kredit može isteći (primjerice na kraju dogovorenog roka) ili ga pružatelj usluge ne može vratiti, korisniku je potrebno pažljivo procijeniti u kakav se trošak upušta.

S pozitivne strane, ova metoda često omogućava popust bez obvezivanja na značajne količine resursa. Korisnik nije pod rizikom da mora predvidjeti točnu potrošnju. Ako procijeni da će koristiti 100 GB RAM memorije po satu i 1 CPU jedinicu po satu, ali u konačnici upotrijebi samo 1 GB RAM-a i 100 procesorskih jedinica, i dalje je moguće sve troškove pokriti unaprijed plaćenim kreditom. Kredit je u suštini prenosiv na niz usluga te je rizik predviđanja potrošnje određenog resursa zapravo nikakav. Radi se o iznimno važnom i korisnom svojstvu za kupca.

Ako je potrebno promijeniti raspodjelu proračuna, lako je preusmjeriti potrošnju na važnije projekte u kompaniji ostvarujući pritom značajnu uštedu.

Za pružatelja usluge *prepaid consumption* je loša opcija u smislu planiranja kapaciteta usluga. Pružatelj unaprijed dobiva uplate koje bi mogle biti investirane u nove tehnologije, ali ostaje pitanje hoće li i kada kredit biti potrošen, koji resursi će biti korišteni i u kojoj mjeri.

Također, *IaaS* pružatelj je pod pritiskom da korisniku mora dostaviti sve što on zahtijeva jer, ipak, posjeduje uplaćena sredstva. Na neki način, pružatelj ima ulogu banke. Ukoliko korisnik pokuša zakupiti neki resurs unaprijed uplaćenim kreditom, a kapaciteta nema slobodnog u tolikoj mjeri, logično je da će korisnik pokušati zatvoriti kredit. Takvim razvojem situacije opada povjerenje u tog *IaaS* pružatelja. Ako je povrat kredita omogućen, problemi mogu nastati jer je teoretski moguće da je investicija već uložena u infrastrukturu.

U slučaju plaćanja kredita unaprijed, ciljani kupci za ovaj model su relativno bogate kompanije s jasnim pogledom na buduće zahtjeve koje imaju plan kako raspodijeliti resurse tijekom vremena. Najčešće se tu radi o velikim afirmiranim poduzećima koja rade na više kratkoročnih projekata. S druge strane, ako su krediti kupljeni na *ad hoc* osnovi, ciljani potrošači mogli bi biti korisnici sa slabim ili nikakvim pogledom na buduće zahtjeve koji će uzeti kredite točno prije potrebe, a uz to vjerojatno imaju ograničena financijska sredstva.

1.3.4.2. Unaprijed plaćena potrošnja uz obročno plaćanje (*Recurring prepaid consumption*)

Model nalaže da su korisnici dužni platiti određen iznos unaprijed, ali u ratama (najčešće mjesečnim). Troškovi potrošnje zatim su podmireni od tog iznosa. Često se unaprijed plaćen novac gleda kao kredit pa korisnik koji unaprijed plaća ima veći kredit u odnosu na onog koji je istu količinu novca uplatio sa zaostatkom (isto kao i kod prethodne metode). Osim toga, moguće je i da korisnik koji unaprijed plaća bude nagrađen nižim cijenama resursa. Ovakav tip obvezivanja najčešće se radi na dulji period, primjerice godinu dana.

Za korisnika su rizici slični kao kod *prepaid consumption* modela, ali nešto manji jer je manja količina sredstava plaćena unaprijed. Stoga će, u slučaju stečaja, biti manji gubici za korisnika. Ukoliko potražnja resursa varira tijekom

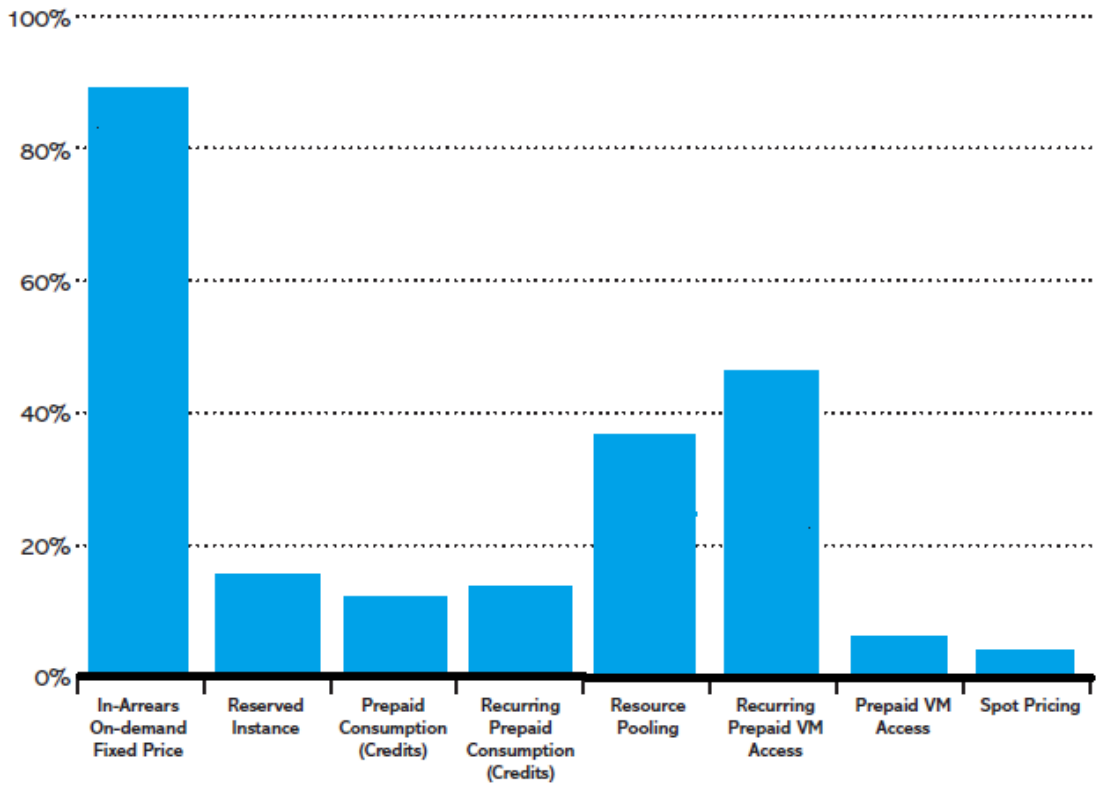
godine, planiranje količine zakupa resursa pravi je izazov. Ako je potreba za *cloud* resursima sezonska (povremena), za određene mjesece kupac bi bio u negativnom financijskom omjeru, dok bi za ostale bio u plusu. Međutim, budući da se kredit može koristiti za bilo koji tip resursa, *prepaid* iznos može biti preusmjeren na pružanje dodatnog kapaciteta drugim virtualnim strojevima. Primjerice, ukoliko mjesečna potrošnja nije u potpunosti iskorištena, može se prenamijeniti na davanje dodatnih mogućnosti Web serverima dok ostali projekti ne zatraže nove troškove.

Velika pogodnost za kupca bilo bi postojanje *on-demand* modela uz ovu metodu. U suprotnom, postojao bi veliki rizik od prevelike i nepotrebne potrošnje ili nedostatak kapaciteta resursa za zahtjeve kompanije. Također, otvoreno je pitanje - prebacuje li se neiskorišteni kredit tekućeg mjeseca na idući ili jednostavno ističe. Većina pružatelja usluga dozvoljava prebacivanje kredita na idući mjesec. Na taj način, kupac može uštediti kredit za mjesece u kojima su potrebni značajni troškovi resursa.

Ovaj model nudi redovit i predvidljiv novčani tok za pružatelja usluge. Dobiveni prihodi ulažu se u infrastrukturu i dodatne kapacitete. Postojanje *on-demand* modela uz dotičnu metodu omogućava korisniku fleksibilnost i skalabilnost resursa, a to su svakako središnji pojmovi i očekivanja od pružatelja *cloud* usluge. *IaaS* pružatelj može uz dodatna mjesečna ulaganja dozvoliti kupcu još veću fleksibilnost zahtjeva. Sve navedeno vodi k tome da je ova metoda iznimno zanimljiva velikim organizacijama s promjenjivim zahtjevima i višestrukim projektima. Takve organizacije trebaju mogućnost preraspodjele proračuna u ovisnosti o promjeni poslovnih ciljeva.

1.3.5. Današnji trendovi naplatnih modela

Tim 451 Research kontaktirao je *IaaS* pružatelje usluga koji se pojavljuju na trenutnom tržištu tražeći informacije o naplatnim modelima koje nude u sklopu *cloud* usluga. Dobivene informacije korištene su u kombinaciji s javno dostupnim naplatnim podacima na web stranicama pružatelja usluge kako bi se izgradila analiza naplatnih modela koji se trenutno nude na *IaaS* tržištu.



Slika 3. Popularnost modela naplate među pružateljima usluge [11]

Slika 3 prikazuje rezultate klasifikacije naplatnih modela 53 pružatelja *IaaS* usluge. Ne čudi da čak 89% ispitanika nudi *on-demand* naplatni model držeći se toga da korisnik plaća samo onoliko koliko je potrošio. Međutim, zanimljivo je primijetiti da određene metode egzistiraju i danas, a pojavile su se prije *cloud comuputing* koncepta. Primjerice, 45% pružatelja *cloud* usluge omogućava obročno plaćanje virtualne jedinice unaprijed. Radi se o metodi koja je korištena prije desetak godina. Više od trećine *IaaS* pružatelja nudi *resource pooling* model koji je sličan konceptu plaćanja na samim počecima Internetske ere.

S obzirom da većina pružatelja usluga infrastrukture računarstva u oblaku nudi plaćanje usluga prema zahtjevu za resursima (*on-demand-pricing*), tijekom izrade rada korištene su dvije metode naplate prema zahtijevanim resursima pri čemu se naplaćuje čitava VM instanca ili zasebni resursi. Naplatni modeli korišteni u projektu opisani su u poglavlju 3.3.3.

2. Strojno učenje

Strojno učenje (engl. *machine learning*) je ogranak računarske znanosti koji se razvio iz studija raspoznavanja uzoraka i računarske teorije učenja u umjetnoj inteligenciji. Proučava izgradnju i ponašanje algoritama koji se prilagođavaju danim podacima (učenje) i daju predviđanja na novom setu podataka (predikcija). Algoritmi rade na principu formiranja modela na temelju podataka kako bi buduća predviđanja ili odluke bila vođena dobivenim ulazima, a ne programskim instrukcijama. Strojno učenje se ne svodi na učenje napamet, nego na prepoznavanje složenih obrazaca. Dakle, ovaj koncept odgovara na pitanje: *Kako isprogramirati sustav da automatski uči i unaprjeđuje se s iskustvom?*

U nastavku će biti rečeno nešto o povijesti strojnog učenja i povezanosti s drugim granama računarske znanosti. Godine 1946. razvijen je prvi računalni sustav ENIAC. U to vrijeme računalo je označavalo stroj za numeričke izračune. Njime je ručno upravljao čovjek koji je povezivao dijelove stroja kako bi računanje bilo obavljeno. Alan Turing je 1950. napravio testove prema sljedećoj ideji: *Može li se računalo naučiti tako da se ne razlikuje od čovjeka prema nekim sposobnostima?*

Iako niti jedno računalo nije prošlo Turingove testove, novi interesantni sustavi su razvijeni nakon toga.

Oko 1952. godine Arthur Samuel (IBM) razvio je prvu računalnu igru, a idući važan korak bila je ELIZA, sustav razvijen u ranim 60-im godinama. Razvio ga je Joseph Weizenbaum, a sustav je simulirao psihoterapeuta koji slaže rečenice na temelju ključnih riječi. Kod prvog pojavljivanja ELIZA-e, dio ljudi mislio je da se radi o čovjeku.

U isto vrijeme pojavili su se i drugi pristupi strojnom učenju. Frank Rosenblatt je 1957. godine razvio perceptron u Cornell Aeronautical laboratoriju. Prikazao je perceptron kao jednostavan linearan klasifikator te kombinacijom velikog broja perceptrona u mrežu nastaje snažan model (neuronska mreža).

Rane devedesete godine donijele su nove dobitke za ovaj računalni koncept. Naime, na raskrižju su se našle računarska znanost i statistika te se razvio novi način razmišljanja nazvan - statički AI. Ovaj koncept je donio pomak na

podatkovno-vođenom pristupu u odnosu na sustave razvijene ranije koji su bili vođeni isključivo znanjem. Nesigurnost u parametrima je ugrađena u modele i to je glavna značajka ovog probabilističkog načina razmišljanja.

Danas je strojno učenje jako raširenо u svijetu računarstva i algoritamskih rješenja. Stroj potpornih vektora (engl. *support vector machine,* SVM) i Bayesove mreže korištene su u komercijalnim sustavima. Povezano je s Big Data rješenjima, jednim od zanimljivijih pitanja današnjice vezanom uz skladištenje velikih količina podataka. Predviđa se da će strojno učenje igrati važnu ulogu u otkrivanju znanja iz ogromnih količina podataka trenutno dostupnih u raznim područjima primjene. [12]

Strojno učenje je usko povezano i često se preklapa s računalnom statistikom, disciplinom koja se također bavi predikcijom. Dosta presjeka ima s matematičkom optimizacijom čije metode i teoriju koristi. Koristi se u projektima gdje eksplicitno programiranje i *rule-based* algoritmi ne mogu dobro odraditi zadatak. Primjer takvih aplikacija su: filtriranje neželjene pošte, optičko prepoznavanje znakova, tražilice i računalni vid. Ovaj koncept zajedno s raspoznavanjem uzoraka daje dva aspekta istog područja djelovanja.

Strojno učenje često se dovodi u vezu s *data mining*-om. Doslovnog prijevoda dubinska analiza podataka, radi se o mehanizmu sortiranja, organiziranja ili grupiranja velikog broja podataka i izvlačenja relevantnih informacija. Drugim riječima, to je proces pronalaženja korisnog znanja iz velike količine informacija. Oba ova koncepta bave se traženjem uzoraka unutar podataka. Međutim, umjesto ekstrakcije značajki namijenjenih ljudskom shvaćanju, čime se bavi *data mining*, strojno učenje koristi ulaz kako bi program poboljšao vlastito razumijevanje i izvođenje. Dolazi do detekcije uzoraka i prilagodbe programskog djelovanja sukladno tome. Primjerice, *Facebook*-ova sekcija *News Feed* mijenja se prema korisnikovim interakcijama s drugim korisnicima. Ako smo češće u doticaju s određenim korisnikom (lajkovi, komentari, *tagg*-ovi), *News Feed* će više prostora posvetiti aktivnostima tog *Facebook* prijatelja. [13]

Osim navedenog, strojno učenje, kao okosnica umjetne inteligencije, dovodi se u vezu s: robotikom, robotskim vidom, raspoznavanjem govora, obradom prirodnog jezika (parsiranje, rješavanje višeznačnosti, označavanje vrsta riječi...),

pretraživanjem informacija (primjerice, rangiranje), umjetnim neuronskim mrežama o čemu je već nešto rečeno itd.

Dubinska analiza podataka (*data mining*) i otkrivanje znanja u podacima su pokazatelji primjene strojnog učenja na velike baze podataka. Tipične primjene ovog pogleda su:

- trgovina (analiza potrošačke košarice, CRM)
- financije (određivanje kreditne sposobnosti, detekcija zlouporaba kartica)
- proizvodnja (optimizacija, *troubleshooting*)
- medicina (postavljanje dijagnoza)
- telekomunikacije (optimizacija usluga)
- bioinformatika (analiza izražajnosti gena, poravnanje)
- *text mining* (klasifikacija teksta, ekstrakcija informacija)
- računalni vid (prepoznavanje lica, praćenje vozila)

Postoje tri vrste strojnog učenja:

1. Nadzirano učenje (engl. *supervised learning*):

Podaci su oblika *(ulaz, izlaz) = (x,y)*. Potrebno je naći preslikavanje *y' = f(x)*. Ako je izlaz *y* diskretna/nebrojčana vrijednost, radi se o klasifikaciji. Ukoliko je kontinuirana/brojčana vrijednost, takav model zovemo regresija.

2. Nenadzirano učenje (engl. *unsupervised learning*):

Dani su podaci bez ciljne vrijednosti te je potrebno pronaći pravilnost u njima. Nenadzirano učenje dijelimo na: grupiranje (engl. *clustering*), procjenu gustoće (engl. *density estimation*) i smanjenje dimenzionalnosti (engl. *dimensionality reduction*).

3. Podržano/ojačano učenje (engl. *rainforecment learning*):

Radi se o učenju optimalne strategije na temelju pokušaja (serije izlaza) s odgođenom nagradom.

U ovom radu korišteno je nadzirano učenje, točnije regresija. Primjene nadziranog učenja su: predviđanje (na temelju ulaznih vrijednosti predvidjeti buduće), ekstrakcija znanja (učenje lako tumačivih modela), sažimanje (model koji

koncizno objašnjava podatke), otkrivanje ekstremnih vrijednosti (iznimke koje nisu pokrivene modelom) i upravljanje (upravljački ulazi dobiveni kao izlaz regresije). [14]

2.1. Regresija

Regresija je jedna od najšire korištenih metoda analiziranja višeparametarskog skupa podataka. Fokus je na modeliranju odnosa između ulaznih (nezavisnih, prediktorskih) i izlaznih (zavisnih, kriterijskih) varijabli. Prema broju ulaznih varijabli, regresija se dijeli na univarijantnu (jednostavna, jednostruka) i multivarijantnu (višestruka, multipla). Prema broju izlaza podjela je na jednoizlaznu i višeizlaznu. [15]

Cilj regresije je formirati funkciju koja iz ulaznih podataka daje vrijednosti koje odgovaraju dobivenom izlaznom skupu. Dodatno, od interesa je razmotriti varijaciju zavisne varijable oko regresijske funkcije što može biti opisano distribucijom vjerojatnosti.

Analiza regresijom najčešće se koristi za predviđanje budućeg ponašanja varijabli. Osim toga, omogućava razumijevanje poveznice između nezavisnih i zavisnih varijabli te izvođenje zaključaka o uzročnim vezama. Mnoge tehnike su razvijene po pitanju ovog mehanizma, a svakako je jedna od najpoznatijih linearna regresija. Radi se o parametarskoj tehnici s konačnim brojem parametara procijenjenih iz dobivenih podataka. [16]

Ukoliko je riječ o multivarijantnoj jednoizlaznoj regresiji, koja je korištena u ovom radu, hipoteza *h* aproksimira nepoznatu funkciju: *f:* $\mathbb{R}^n \rightarrow \mathbb{R}$

Kao što je već rečeno, ulazni podaci su oblika $D=\{(x^i, y^i)\}$ gdje x označava nezavisnu, a y zavisnu (izlaznu) varijablu.

Rezultati regresijskih modela uvelike ovise o generiranju podataka koje ovaj vid strojnog učenja dobiva za treniranje. Generirane podatke podijeli se na skup za učenje (*training set*) i ispitni skup (*test set*).

Izraz funkcije pogreške glasi:

$$E(h|D) = \frac{1}{2}\sum_{i=1}^{N}\left(y^i - h(x^i)\right)^2 \quad (1)$$

Varijabla y^i u izrazu (1) označava stvarni izlaz za primjer *i*, dok $h(x^i)$ označava izlaz aproksimirane funkcije, odnosno regresijskog modela. [15] Funkcija pogreške na skupu za učenje zove se empirijska pogreška, a na skupu za testiranje generalizacijska pogreška.

Još jedna mjera pogreške modela je srednja kvadratna pogreška (engl. *mean squared error,* MSE) čija formula glasi:

$$E(h|D) = \frac{1}{N}\sum_{i=1}^{N}\left(y^i - h(x^i)\right)^2 \quad (2)$$

Pojam regularizacije korišten je u matematici, statistici i strojnom učenju. Ako model ima mnogo parametara, lako ga je prenaučiti (engl. *overfitting*). Posljedica bi bila da je empirijska pogreška jako mala, a generalizacijska velika što nikako nije dobar rezultat strojnog učenja. Jedan od načina kako to spriječiti jest korištenje regularizacije. Ideja jest ograničiti rast vrijednosti parametara kažnjavanjem hipoteza s visokim vrijednostima istih. Time se ostvaruje kompromis između točnosti i jednostavnosti modela i to već pri samom učenju. Efektivno se ograničava složenost i sprječava prenaučenost. Cilj regularizacije jest što više parametara pritegnuti na nulu. Tako se dobivaju rijetki (engl. *sparse*) modeli. Osim navedene prednosti, računalno su jednostavniji. Složenost modela ugrađena je u funkciju pogreške:

$$E' = empirijska\ pogreška\ +\ \lambda * složenost\ modela \quad (3)$$

Veća vrijednost regularizacijskog faktora λ uzorkuje smanjenje efektivne složenosti modela. Za λ=0 riječ je o neregulariziranoj funkciji pogreške.

Regularizacijski faktor pripada skupu hiperparametara. Radi se o unaprijed definiranim parametrima koji određuju izgled modela, to jest utječu na njegovu konstrukciju. Razlikuju se od parametara modela koji su definirani njegovom izgradnjom.

U ovom radu korištena je L2-regularizacija (engl. *ridge regularization*) koja kažnjava težine proporcionalno njihovom iznosu. Ima rješenje u zatvorenoj formi, no ne rezultira rijetkim modelima, odnosno nije joj karakteristično da pritegne težine modela na nulu. [15]

2.1.1. Linearna regresija

Aproksimacijska funkcija *h* je linearna funkcija parametara **w** = (w_0, w_1,...,w_n).

Izraz koji opisuje mulitvarijantnu jednoizlaznu linearnu regresiju s *n* ulaznih varijabli glasi:

$$h(\boldsymbol{x}|\boldsymbol{w}) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n \quad (4)$$

Dakle, linearna regresija modelira linearnu relaciju između izlazne varijable *y* i ulaznih varijabli ***x***. [15] Ovo je bio prvi tip regresijske analize koji je detaljno priučavan i intenzivno korišten u praktičnim primjenama. Razlog tome leži u činjenici da se modeli koji linearno ovise o parametrima lakše modeliraju u odnosu na one s nelinearnom ovisnošću.

Linearna regresija ima velik broj praktičnih primjena. Većina takvih primjena može se podijeliti u dvije kategorije.

Ako je cilj modela predviđanje, ovaj tip regresije se koristi za podešavanje predikcijskog modela prema promatranom skupu podataka vrijednosti *y* i *x*. Nakon treniranja modela, lako se dobije izlaz *y* za ulazne vrijednosti *x* varijable (očekivano je da vrijednosti *y*-a u tom trenutku nisu poznate).

Ukoliko postoji veći broj ulaznih varijabli x_1, x_2,...,x_n povezanih s izlazom *y*, moguće je iskoristiti linearnu regresiju za kvantificiranje jačine relacije između *y* i x_j. Na taj način identificira se podskup ulaznih varijabli koji najviše utječe na vrijednost izlaza. [16]

2.1.2. Polinomijalna regresija

Izraz univarijantne jednoizlazne regresije jest:

$$h(\boldsymbol{x}|\boldsymbol{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_d x^d \quad (5)$$

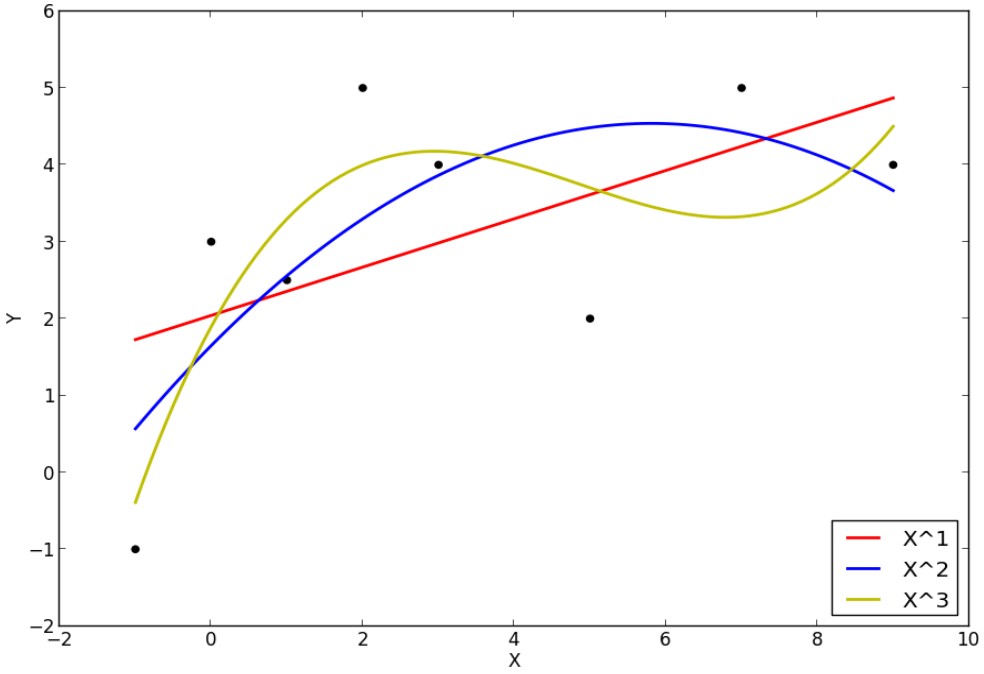
Izraz (5) opisuje model polinomijalne regresije stupnja *d* s jednom značajkom (*n*=1) .

Formula polinomijalne regresije stupnja *d* = 2 i dviju ulaznih značajki (*n* = 2):

$$h(\boldsymbol{x}|\boldsymbol{w}) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2 + w_4 x_1^2 + w_5 x_2^2 \quad (6)$$

Ovaj vid regresije je oblik linearne regresije u kojoj se odnos između nezavisnih varijabli ***x*** i izlazne varijable y modelira kao polinom n-tog stupnja. [15] Naučeni model ima nelinearan oblik.

Oba navedena regresijska modela se najčešće podešavaju (engl. *fit*) uz pomoć metode najmanjih kvadrata. Kod polinomijalne regresije izrazito je važan utjecaj regularizacijskog faktora jer uz visoki stupanj polinoma može vrlo lako doći do prenaučenosti modela.



Slika 4. Usporedba linearne i polinomijalne regresije

Slika 4 najbolje prikazuje usporedbu linearne regresije i dvaju modela polinomijalne. Linearna regresija (crvena linija) se najlošije prilagođava danom setu ulaznih primjera. Plava linija označava model polinomijalne regresije stupnja 2, a žuta polinom 3. stupnja. Jasno je da će veći stupanj polinoma uvijek dati najbolje rezultate na skupu za učenje, no za krajnji produkt bitno je da je generalizacijska pogreška što manja. Stoga se idealan model određuje unakrsnom validacijom (engl.*cross-validation*) gdje model učimo na skupu za učenje, a njegovu generalizacijsku sposobnost provjeravamo na skupu za provjeru.

3. Primjena strojnog učenja na predviđanje troška usluge u računalnom oblaku

S obzirom na složenu i heterogenu okolinu računalnog oblaka, kao i različite modele naplate i cijene resursa koje nude pružatelji usluge *cloud* infrastrukture, cilj rada je primjenom strojnog učenja i simulacija temeljenih na zadanom modelu doći do jasnije slike koji parametri utječu na trošak izvođenja usluge te koliki je njihov utjecaj. Dobiveno znanje posebno je zanimljivo pružateljima *SaaS* usluge, koji svoju aplikaciju žele smjestiti u okruženje računalnog oblaka, kako bi ju u konačnici ponudili krajnjem korisniku. Osim toga, cilj istraživanja je dobiti uvid koje značajke formiraju ponašanje aplikacijske usluge smještene u računalnom oblaku. Tako je, osim aspekta troška, promatrano i vrijeme izvođenja usluge u ovisnosti u ulaznim atributima. Pružatelji *SaaS* usluge često ne znaju kako će se njihova aplikacija ponašati u smislu potrošnje resursa jednom kada je smjeste u okruženje računalnog oblaka, posebice uzimajući u obzir dinamično kretanje korisničkih zahtjeva, niti kolikim troškom će rezultirati rad aplikacije. Također, primjenom strojnog učenja želi se ostvariti predviđanje troška usluge i očekivanog vremena izvođenja na temelju skupa ulaznih parametara koje će pomoći prilikom donošenja odluke o konfiguraciji okoline računalnog oblaka u koju se razmatrana usluga smješta.

Prva faza rada obuhvaćala je generiranje skupa podataka za učenje koristeći model zadane aplikacije i okruženja računalnog oblaka, u koji je aplikacija smještena, te simulacije njezinog izvršavanja u *cloud* okolini. Za navedeno je korišten simulator otvorenog koda CloudSim, koji je prethodno bilo potrebno proširiti, kako bi obuhvaćao sve promatrane značajke računalnog oblaka i referentne aplikacije.

Zatim je u drugoj fazi bilo potrebno primijeniti regresijske modele nad dobivenim skupom podataka kako bi se ustvrdio utjecaj pojedinih značajki na dvije izlazne varijable, trošak i vrijeme izvođenja usluge, te kako bi se, u konačnici, ostvarila njihova predikcija. U nastavku su opisani korišteni alati, model usluge i okruženja računalnog oblaka te obje navedene faze rada.

3.1. Korišteni alati

3.1.1. CloudSim

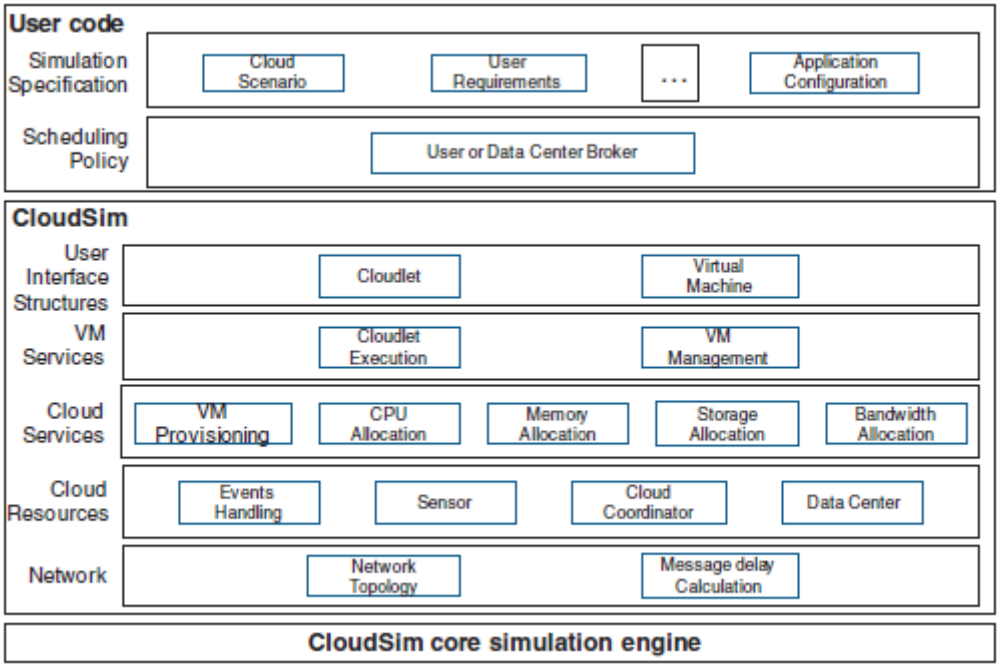
CloudSim [17] je alat za modeliranje i simuliranje okruženja računalnog oblaka te izvođenje algoritama opskrbe resursima. Simuliranje ponašanje *cloud* komponenti kao što su podatkovni centri, virtualne jedinice i pravila rezerviranja resursa. Nadogradnja koda je omogućena pošto se radi o otvorenom (engl. *open source*) simulatoru. U trenutnoj verziji omogućeno je modeliranje *cloud* okruženja koje se sastoji od pojedinačnih i umreženih oblaka (federacija oblaka).

Brojni istraživači i organizacije, kao što je HP Labs u Sjedinjenim Američkim Državama, koriste CloudSim u istraživanjima o učinkovitom upravljanju resursima podatkovnih centara.

Ovim alatom istraživači i programeri mogu efikasno testirati svojstva novorazvijene aplikacijske usluge u kontroliranom okruženju. Prema dobivenim rezultatima simulacije, uslugu je moguće dalje razvijati u željenom smjeru i doraditi potrebne performanse. Najveće prednosti ovakvog testiranja su:

- vremenska učinkovitost
  Potrebno je puno manje vremena za implementaciju *cloud-based* aplikacije i testiranje u takvom okruženju.
- fleksibilnost i izravna primjenjivost
  Programeri modeliraju aplikacije i testiraju njihova svojstva u heterogenim okruženjima računalnog oblaka (primjerice Amazon EC2, Microsoft Azzure) s puno manje posla oko smještanja usluge u takvo okruženje (engl. *service deployment*).

Osim navedenih mogućnosti, ovaj simulator nudi podršku za simulaciju mrežnih veza među elementima sustava. Daje platformu za federaciju oblaka koja povezuje resurse privatnih i javnih domena. [17]



Slika 5. CloudSim arhitektura po slojevima [5]

Na slici 5 prikazan je slojevit dizajn ovog softver radnog okvira (engl. *software framework*) s komponentama arhitekture [5]. Početna izdanja CloudSim-a koristila su SimJava-u kao simulator diskretnih događaja koji podržava nekoliko temeljnih funkcionalnosti kao što je procesiranje događaja i slaganje istih u red izvršavanja, stvaranje *cloud* entiteta (usluge, *host*, podatkovni centar, agent, virtualne jedinice), komunikacija među komponentama te upravljanje simulacijskim satom. Međutim, u trenutno korištenoj verziji ovog alata, SimJava sloj je izbačen kako bi se omogućilo izvođenje nekih naprednijih operacija koje on nije podržavao.

CloudSim simulacijski sloj pruža podršku za modeliranje okruženja podatkovnih centara uključujući upravljačka sučelja za virtualne mašine, memoriju, skladištenje i mrežnu propusnost. Temeljna pitanja, kao što je smještanje virtualnih strojeva na *host* uređaj, upravljanje izvođenjem aplikacija i nadzor dinamičkog stanja sustava su riješena putem ovog sloja. Pružatelj usluge kojeg zanima efikasnost nekog mehanizma u smještanju virtualne mašine na *host* uređaj implementira strategiju upravo u ovom dijelu arhitekturnog koncepta. *Host* može istodobno biti dodijeljen skupu virtualnih jedinica koje izvode aplikacije. Sloj također sadrži funkcionalnosti koje *cloud* aplikacijski programer može proširiti kako bi ispitao situaciju složenog radnog okruženja i ponašanje aplikacije u takvoj situaciji. [18]

Najviša razina u CloudSim stogu je User code. Ona sadrži osnovne entitete *host* uređaja (primjerice broj virtualnih strojeva), aplikacije (broj zadataka i njihovi zahtjevi), virtualne mašine, broj korisnika itd. Nasljeđivanjem osnovnih entiteta ovog sloja aplikacijski razvojni programer ima mogućnosti generiranja različitih zahtjeva opterećenja distribucija (konfiguracija aplikacije). Nadalje, ima slobodu različitih scenarija *cloud* okruženja i ispitivanja robusnih testova na određenim konfiguracijama.

Kako je računarstvo u oblaku još uvijek paradigma u nastajanju, postoje nedostaci u standardizaciji, alatima i metodama koje se bave pitanjem infrastrukture i aplikacijskog dijela. U skoroj budućnosti bit će znatan broj istraživačkih projekata u svijetu znanosti i industrije.

CloudSim je najpoznatiji alat korišten u domeni *cloud computing*-a. Proširenjem njegove trenutne funkcionalnosti, istraživači će imati mogućnost ispitivanja najrazličitijih scenarija i konfiguracija čime se omogućava razvoj i daljnje napredovanje u svim kritičnim aspektima ovog koncepta.

3.1.1.1. Implementacija okoline računalnog oblaka u alatu CloudSim

Usluge infrastrukture računalnog oblaka (*IaaS*) razine mogu biti simulirane u CloudSim alatu [5] koji je korišten u ovom radu. Simulator je korišten za modeliranje *cloud* okruženja. Podatkovni centar upravlja određenim brojem poslužitelja (*host* uređaja). *Host*-u je pridijeljena jedna ili više virtualnih jedinica na kojima se izvodi aplikacija i čije resurse krajnji korisnik može zakupiti po potrebi. Virtualna jedinica pridijeljena je *host*-u prema VM politici dodjele (engl. *VM allocation policy*) kojeg definira pružatelj usluge. U ovom kontekstu riječ politika označava osiguravanje dodjele odgovarajućeg poslužitelja/*host* uređaja virtualnom stroju, stvaranje, uništavanje i migracija VM. Na sličan način jedna ili više aplikacijskih usluga može biti smještena na virtualnu mašinu. Taj dio naziva se opskrba aplikacije u kontekstu računarstva u oblaku (engl. *application provisioning*).

Dakle, podatkovni centar može upravljati s više *host*-ova, a svaki takav uređaj nadzire rad virtualnih strojeva (tijekom njihovog životnog ciklusa) koji su mu pridijeljeni. *Host* je *cloud* komponenta koja predstavlja fizički računalni poslužitelj, a najvažnije njegove karakteristike su: sposobnost procesiranja (izražena u

milijunima instrukcija koje izvodi u sekundi, MIPS), memorija, skladištenje i pravilo za dodjeljivanje procesnih jezgara virtualnim jedinicama. Uz to, implementira sučelja koja podržavaju modeliranje i simulaciju jednojezgrenih i višejezgrenih čvorova.

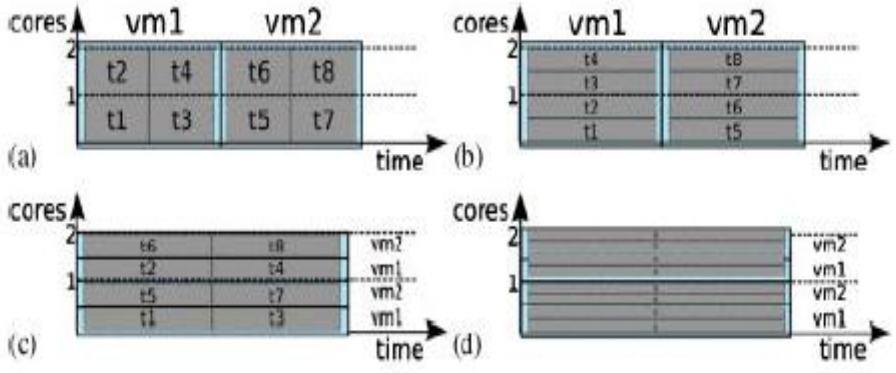
Rezervacija virtualnih strojeva jest proces stvaranja VM instanci na *host*-ovima koji odgovaraju po karakteristikama (skladištenje, memorija), konfiguraciji (softversko okruženje) i zahtjevima *SaaS* pružatelja usluge. Jednom kada je aplikacijska usluga definirana i modelirana, dodjeljuje se jednoj ili više virtualnih jedinica, također preko politike dodjele.

Svaka *host* jedinica ima politiku raspodjele kojom se vodi pri dodjeli procesorskih jezgri virtualnim strojevima koji će se izvoditi na dotičnom *host*-u. Pri tome se u obzir uzima nekoliko hardverskih karakteristika, kao što je broj CPU jezgri i količina memorije, koje će biti dodijeljene VM jedinici. CloudSim simulator podržava iduće scenarije:

- dodjela procesorskih jezgara specifičnoj virtualnoj jedinici, politika prostorne dodjele (engl. *space-shared policy*)
- dinamička distribucija kapaciteta jezgri među virtualnim strojevima, politika vremenske dodjele (engl. *time-shared policy*) [19]

Virtualne jedinice su kontekstualno (fizički i memorijski) izolirane na *host*-u. One dijele procesne jezgre i sabirnice sustava. Stoga je količina hardverskih resursa dostupna svakoj VM određena ukupnom procesnom snagom i propusnošću *host* uređaja. Navedeni kritični faktori uzimaju se u obzir prilikom smještanja virtualnog stroja na *host*. U tom pogledu CloudSim radi na dva nivoa: najprije na nivou *host*-a te zatim na razini virtualne mašine. Na VM razini određena je količina procesne snage dodijeljena svakoj aplikaciji (radnoj jedinici). U daljnjem tekstu radna jedinica (radni zadatak, *cloudlet*) odnosi se na finiju apstrakciju aplikacijske usluge smještene na VM u *cloud* okruženju. Ako je aplikacija jednostavna, pojam *cloudlet* se koristi za čitavu aplikaciju. No, ukoliko je riječ o složenijem tipu usluge, radni zadatak označava jedan njezin dio, primjerice, poslovni tok (engl. *business workflow*) ili dohvat podataka iz baze.

Na ove dvije razine CloudSim definira politiku prostorne i vremenske podjele objašnjene nekoliko redova iznad. Slika 6 prikazuje svaki od navedenih scenarija zasebno.



Slika 6. Različiti scenariji smještanja virtualnih jedinica na *host* i radnih jedinica na VM [5]

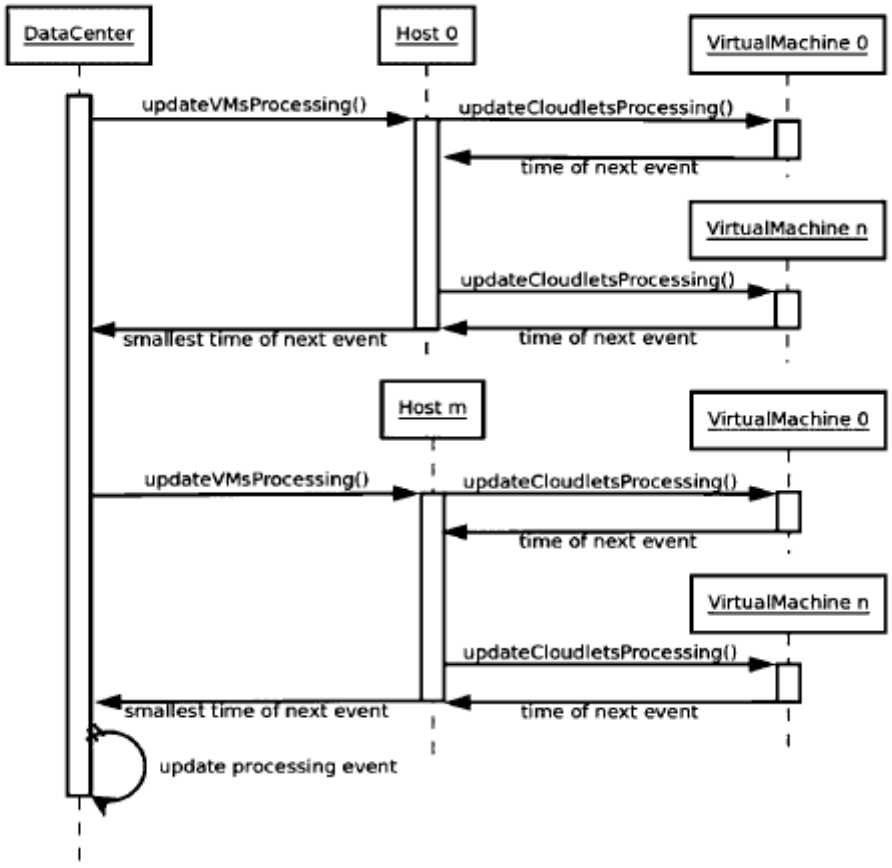
Navedeni primjer koristi *host* uređaj s dvije CPU jezgre i dvije virtualne jedinice. VM1 i VM2 zahtijevaju po dvije jezgre. Radne jedinice t1, t2, t3 i t4 su smještene na VM1, a t5, t6, t7 i t8 na VM2.

Odjeljak 6(a) prikazuje scenarij *space-shared* politike podjele na obje razine (*host* i VM razina). Kako svaka virtualna jedinica traži dvije jezgre za svoj rad, tako se u jednom trenutku može izvoditi samo jedna od njih. Stoga, VM2 može dobiti jezgre tek kada VM1 završi s izvođenjem svojih zadataka. Slično se događa i na razini ispod. Svaki zadatak zahtijeva jednu jezgru za svoje izvođenje, stoga se u jednom trenutku mogu izvoditi dva zadatka. Do njihovog završetka preostale dvije radne jedinice dotične virtualne mašine smještene su u red čekanja.

Slika 6(b) sugerira *space-shared* dodjelu za alociranje virtualnih strojeva na *host*-u i *time-shared* za alociranje radnih zadataka na VM. Tijekom vijeka virtualne jedinice svi njezini zadaci se izvode paralelno. Točnije, dinamički se izmjenjuju dok ne završe s izvođenjem.

Na 6(c) je prikazan *time-shared* način dodjele za virtualne mašine te *space-shared* za radne zadatke. Svaka VM dobiva vremenski period na svakoj procesnoj jezgri. Uz to, u jednom trenutku samo jedan zadatak aktivno koristi procesnu jezgru dodijeljenu virtualnom stroju (*space-shared* politika).

Odjeljak 6(d) demonstrira *time-shared* način rada na obje razine. Procesna snaga *host*-a je konkurentno podijeljena među virtualnim strojevima, a isto vrijedi i za raspored zadataka. U ovom slučaju ne postoji red čekanja među radnim jedinicama. [5]



Slika 7. Sekvencijski dijagram izvođenja procesa na virtualnim strojevima [5]

Sekvencijski dijagram sa slike 7 prikazuje način izvođenja procesa na virtualnim jedinicama, odnosno ažuriranje stanja cjelokupnog podatkovnog centra. VM kontinuirano nadgleda i ažurira stanje radnih zadataka koji su joj dodijeljeni. U CloudSim-u interni događaj je generiran te informira podatkovni centar o skorom završetku izvođenja određene radne jedinice. Podatkovni centar ažurira stanje svih *host*-ova kojima upravlja, a svaki *host* isto radi za virtualne strojeve koji mu pripadaju. Virtualni stroj izvršava zadatke te uzima nove ukoliko red čekanja nije prazan. [5]

3.1.2. SciPy

Nakon generiranja simulacija u CloudSim-u idući korak projekta bio je napraviti regresiju nad danim podacima. Od alata za strojno učenje korišten je SciPy.

Radi se o Python biblioteci (engl. *library*) otvorenog koda. Koriste ga znanstvenici, analitičari i inženjeri koji se bave računarskom znanošću. Sadrži module za optimizaciju, linearnu algebru, integraciju, interpolaciju, posebne funkcije (kao što je brza Fourierova transformacija, FFT), obradu signala i slike te ostale module zajedničke znanosti i inženjerstvu.

SciPy je izgrađen nad NumPy poljem objekata i dio je NumPy stoga koji uključuje alate kao što je Matplotlib, pandas, Sympy i nose. Postoji širok spektar biblioteka povezanih s računarskom znanošću koje se dodaju NumPy stogu svakog dana. Srodni alati ovom konceptu mogu se pronaći kao dio Matlab-a, GNU Octave-a i Scilab-a. Često se koristi i naziv SciPy stog. [20]

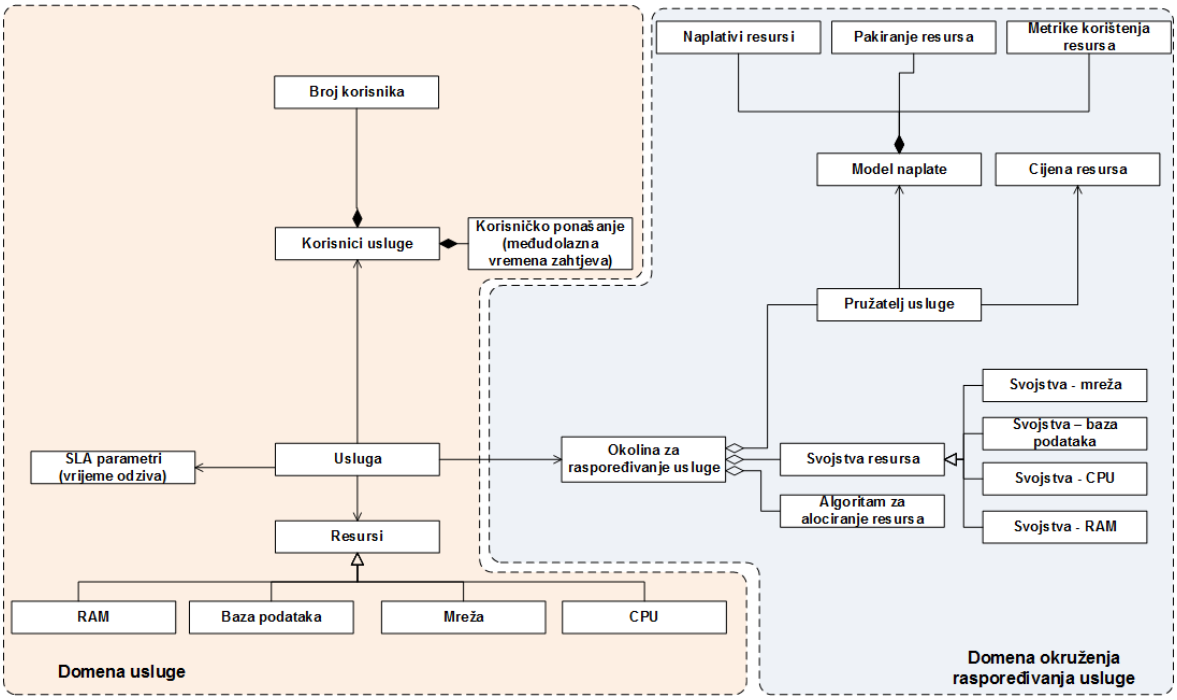
90-ih godina prošlog stoljeća Python je proširen tako da je uključivao tip polja za numeričko računanje nazvan Numeric. Taj paket zamijenjen je NumPy-om 2006. koji je napisao Travis Oliphant. Ista osoba je 1999. godine formirala veliku kolekciju ekstenzijskih modula namijenjenih znanstvenom računarstvu u Pythonu. U suradnji s Pearuom Petersonom 2001. godine napisali su paket nazvan SciPy. To je bio rezultat nastavka projekta kojeg je započeo Eric Jones. Novostvoreni paket sadržavao je standardnu kolekciju određenih numeričkih operacija koristeći Numeric strukturu podatkovnih polja. Ubrzo potom Fernando Pérez razvio je IPython, proširenu interaktivnu ljusku široko korištenu u cjelokupnom računarstvu, uključujući i ovaj istraživački projekt. John Hunter nakon toga je objavio prvu verziju Matplotlib-a, biblioteku korištenu tada za 2D iscrtavanje. Od tada se SciPy okruženje poprilično raširilo uz brojne pakete i alate korištene u svijetu računarstva.

Osnovna podatkovna struktura SciPy alata je višedimenzionalno polje koje nudi NumPy modul. Osim toga, NumPy nudi sofisticirane funkcije (primjerice, vezane uz linearnu algebru, Fourierove transformacije i generiranje slučajnih brojeva), integriranje C/C++ i Fortran koda. Može se koristiti kao efikasan višedimenzionalni spremnik podataka proizvoljnog tipa. NumPy je licenciran pod BSD licencom. [21]

SciPy paketi ključnih algoritama i funkcija uključuju:

- konstante: fizičke konstante i pretvorbene faktore
- grupiranje (engl. *clustering*): hijerarhijsko grupiranje, vektorska kvantizacija, K-means
- FFT paket: algoritmi diskretne Fourierove transformacije
- interpolacijske metode
- lib: Python-ove omotače (engl. *wrapper*-e) za eksterne biblioteke
- ndimage: brojne funkcije za višedimenzionalnu obradu slike
- optimizacijske algoritme
- alate za obradu signala
- metode linearne algebre
- algoritme povezane s rijetkim (engl. *sparse*) matricama
- prostorne metode najbližih susjeda i funkcije udaljenosti
- statističke metode
- weave: alat za pisanje C/C++ koda

3.2. Model



Slika 8. Cjelokupni model projekta

Na slici 8 prikazan je model usluge i okruženja računalnog oblaka u kojemu je usluga smještena. Promatrani parametri podijeljeni su na dvije domene, domena usluge (engl. *service domain*) i okruženja (engl. *deployment domain*).

Parametri u domeni usluge odnose se na resurse potrebne za njezino izvođenje (potrošnja CPU kapaciteta, RAM memorija, skladišna memorija i mrežna propusnost) te parametre koji opisuju ponašanje korisnika usluge (broj istovremenih korisnika, međudolazna vremena korisničkih zahtjeva). SLA atributi [3] označavaju ugovorom definirane vrijednosti kvalitete usluge, primjerice vrijeme odziva usluge ili postotak vremena za koje pružatelj jamči da će usluga biti dostupna krajnjim korisnicima.

Domena okruženja raspoređivanja usluge u prvom redu sadrži parametre virtualnih jedinica (broj procesorskih jedinica, RAM memoriju, memoriju skladištenja podataka, mrežnu propusnost). Algoritmom za alociranje resursa smještaju se usluge na pojedince virtualne jedinice. Korištena su tri algoritma: vremenska dodjela resursa, prostorna dodjela te vremenska s automatskim skaliranjem. Domena okruženja sadrži i parametar naplatnog modela, a za potrebe ovog projekta korišteni su: *on-demand-instance* i *pay-as-you-go.* Prvi

model naplaćuje prema broju korištenih virtualnih jedinica, a drugi prema količini iskorištenih resursa (RAM memorije, procesorskih jedinica, skladišne memorije i mrežne propusnosti). Naplatni model i cijene resursa postavlja pružatelj *cloud* usluge.

3.3. Generiranje podataka za učenje

Ulazni podaci za strojno učenje generirani su CloudSim simulacijom izvođenja aplikacije u računalnom oblaku te istodobnog ponašanja domene okruženja raspoređivanja usluge.

3.3.1. Ulazni i izlazni podaci simulacija

Po pitanju izlaza regresijskih modela u ovom istraživanju, interesantne su vrijednosti prosječnog troška i prosječnog vremena izvođenja *cloudlet*-a (engl. *response time*).

Po pitanju ulaznih varijabli, podaci su definirani na tri razine. Prva razina odnosi se na *host* uređaj. Svaki takav poslužitelj ima: određen broj procesorskih jedinica, RAM memoriju, mrežnu propusnost i skladišnu memoriju. Procesorske jedinice definirane su MIPS parametrom (engl. *million instructions per second*) koji označava brzinu procesora. Radi se o broju instrukcija računalnog programa, izraženom u milijunima, koje procesor može obraditi u jednoj sekundi. Ostale mjerne jedinice koje se inače koriste u ovom kontekstu su kIPS (engl. *thousand instructions per second*) i GIPS (engl. *Giga instructions per second*). Pored toga, potrebno je odrediti politiku dodjele procesorskih jezgara virtualnim jedinicama smještenim na dotični *host* uređaj. Na ovoj razini korišten je *space-shared* algoritam u istraživanju.

Druga razina tiče se resursa virtualnih jedinica. Slično kao i *host*, virtualni stroj ima atribute: broj procesorskih jezgri, RAM memoriju, mrežnu propusnost i memoriju skladištenja, a procesorske jezgre definirane su MIPS atributom. Jedan od ključnih parametara svakako je politika dodjele resursa virtualne jedinice radnim zadacima (*cloudlet*-ima). Korištena su tri tipa rasporeda: vremenska dodjela resursa, prostorna dodjela te vremenska s automatskim skaliranjem (engl.

autoscaling-om). O implementaciji rasporeda s automatskim skaliranjem bit će rečeno više u nastavku rada (poglavlje 3.3.3).

U trećoj razini definiraju se parametri *cloudlet*-a. Tu se ubrajaju: MI vrijednost (broj instrukcija radnog zadatka, izražen u milijunima), broj procesorskih jedinica potreban za izvođenje, iskorištenje (utilizacija) procesora, RAM memorija, skladišna memorija koju zauzima i utilizacija mrežne propusnosti. Stupanj utilizacije izražen je u postotku. Postotak iskorištenja mrežne propusnosti definira omjer izlazne veličine datoteke *cloudlet*-a i mrežne propusnosti virtualnog stroja na kojem je *cloudlet* postavljen. Stupanj RAM iskorištenja *cloudlet*-a definiran je omjerom RAM memorija *cloudlet*-a / RAM memorija virtualnog stroja.

Kao što je već rečeno, u istraživanju su korištena dva naplatna modela, *on-demand-instance* i *pay-as-you-go*. Pri korištenju prvog modela, potrebno je definirati cijenu virtualne jedinice temeljenu na korištenju tijekom jednog sata. Za drugi tip naplate, definiraju se cijene resursa (RAM memorije, CPU jedinice, memorije skladištenja i mrežne propusnosti). RAM memorija i CPU jedinice se obično naplaćuju prema broju sati korištenja. Memorija skladištenja i mrežna propusnost se naplaćuju prema fiksnoj cijeni, primjerice prvih 5 GB memorije po određenoj cijeni, a idućih 5 GB po nešto višoj itd. Cijene resursa postavlja pružatelj usluge koji u svom vlasništvu ima podatkovni centar na kojem se izvodi aplikacijska usluga.

3.3.2. Promatrana aplikacija

Referentna aplikacija u ovom studijskom slučaju je zadatak koji računa faktorijel broja 10 000.

Tijekom izvršavanja aplikacije na poslužitelju izmjereni su podaci koji se odnose na utilizaciju procesora, RAM memoriju aplikacije i vrijeme odziva. Pomoću tih atributa i parametara virtualne mašine moguće je izračunati MI vrijednost i RAM utilizaciju *cloudlet*-a. Kako se radi o jednom korisničkom zahtjevu, MI vrijednost dobivena je umnoškom MIPS atributa VM i vremena odziva. Osim parametara aplikacije, atributi vezani uz virtualnu mašinu i *host* uređaj, također su postavljeni kao referentne vrijednosti u simulacijama.

Host uređaj na kojem je smještena promatrana aplikacija ima sljedeće specifikacije:

- 8 procesorskih jedinica, svaka vrijednosti MIPS = 8000
- RAM memorija = 8 GB
- mrežna propusnost = 10 GBps
- skladišna memorija = 512 GB

Virtualna jedinica koja obavlja izvođenje aplikacije:

- 1 procesorska jedinica, MIPS parametar = 1000
- RAM memorija = 1 GB
- mrežna propusnost = 2 GBps
- memorijski spremnik = 12 GB

Specifikacije promatrane aplikacijske usluge, odnosno *cloudlet*-a:

- MI atribut = 13 262 milijuna instrukcija
- 1 procesorska jedinica potrebna za izvođenje
- iskorištenje procesora = 6.346%
- RAM memorija = 66 MB
- veličina podataka dotične aplikacijske usluge (skladišna memorija) = 16 KB
- izlazna veličina datoteke *cloudlet*-a = 4 KB
- mrežna utilizacija = $1.9*10^{-6}$%

Korišten aplikacijski primjer ima jako malu potrošnju mrežne propusnosti jer za rad aplikacije nije potrebno prenositi podatke putem mreže, već krajnji korisnik u konačnici kao rezultat dobiva ispis dobivene vrijednosti. Također, memorija skladištenja je zanemariva u odnosu na kapacitet virtualne jedinice.

Kako bi se dobio osjećaj što će se dogoditi s drukčijim aplikacijama, parametri su modificirani tijekom simulacija. Na taj način, svaki atribut, osim vrijednosti vezanih uz referentnu aplikaciju, imao je pridruženih još nekoliko vrijednosti.

3.3.3. Proširivanje alata CloudSim

Za potrebe projekta simulator otvorenog koda CloudSim proširen je dodatnim mogućnostima.

U prvom redu, implementirana su dva naplatna modela, *on-demand-instance* za naplatu prema broju korištenih virtualnih jedinica te *pay-as-you-go* za naplatu prema zasebnim korištenim IT resursima koje nudi pružatelj usluge.

Za *on-demand-instance* promatrana je cijena zakupljene virtualne mašine koja izravno utječe na ukupan trošak. Naplata se vrši prema broju sati upotrebe.

Kod *pay-as-you-go* modela na trošak utječu četiri cijene:

- cijena po GB RAM memorije
- cijena skladištenja podataka (baza podataka)
- cijena CPU jedinice te
- cijena mrežne propusnosti.

Navedene četiri cijene postavljene su na razini podatkovnog centra. RAM memorija i CPU jedinice se obično naplaćuju prema broju sati korištenja. Memorija skladištenja i mrežna propusnost naplaćuju se prema fiksnoj cijeni, primjerice prvih 5 GB memorije po određenoj cijeni, idućih 5 GB po nešto višoj itd.

Za navedena dva tipa naplatnih modela cijene resursa su preuzete sa Web stranica najpoznatijih pružatelja *cloud* usluge: Google [22], Amazon EC2 [23], Microsoft Azure [24] i vCloud Air [25].

Za potrebe *on-demand-instance* metode, dodan je atribut cijene virtualne jedinice. Također, u svrhu izračunavanja troška prema *pay-as-you-go* modelu, za svaku virtualnu jedinicu potrebno je pamtiti najveću zabilježenu razinu utilizacije RAM memorije i mrežne propusnosti te iskorišteni kapacitet memorijskog spremnika.

Kako bi se simuliralo korisničko ponašanje, implementirana su međudolazna vremena korisničkih zahtjeva za izvođenjem radnih zadataka. CloudSim uz svaki *cloudlet* zahtjev postavlja vrijeme kada počinje s izvođenjem. Vrijeme je simulirano satom koji radi na razini čitave simulacije. *Cloudlet* zahtjevi definirani su na samom početku programa te im CloudSim, u principu, svim dodijeli isto vrijeme početka izvođenja. Međudolazna vremena simulirana su tako

da je svakom zahtjevu dodano određeno kašnjenje, odnosno početak izvođenja pomaknut za neki vremenski interval.

Osim navedenih prostorne i vremenske podjele resursa virtualne jedinice, za potrebe projekta implementirana je i vremenska podjela s automatskim skaliranjem (engl. *autoscaling*). Implementacijom *autoscaling* algoritma omogućeno je simuliranje automatiziranog horizontalnog skaliranja. Označava scenarij u kojem utilizacija radnih zadataka na virtualnoj jedinici prijeđe neki unaprijed postavljeni prag. Automatskim skaliranjem instancira se novi virtualni stroj identičan trenutnom po specifikacijama te se novopridošli radni zadatak (koji je uvjetovao *autoscaling*) izvršava na tom stroju. Pritom je osigurano to da VM-e neće biti preopterećene, a radni zadaci će se prije izvršiti čime se osigurava razina kvalitete usluge definirana SLA ugovorom. U ovom projektu utilizacijski prag postavljen je na 85%, a promatra se zauzetost RAM memorije i CPU iskorištenja virtualne mašine. Dakle, ukoliko zadatak pristigao na VM pridonosi preopterećenosti, instancira se nova mašina i taj zadatak se izvodi na njoj. Preopterećenost se događa ukoliko postojeći radni zadaci dotične virtualne jedinice zbrojeni s novopridošlim *cloudlet*-om prelaze 85% kapaciteta VM, bilo po RAM memoriji ili po postotku iskoristivosti procesora. [26]

3.3.4. Pokretanje simulacija i generiranje podataka za strojno učenje

Za pokretanje simulacija implementirane su skripte koje upisuju rezultate izvođenja, zajedno sa svim potrebnim parametrima, u datoteke. Takve datoteke čine ulazni skup primjera za strojno učenje.

Pojedini atributi fiksirani su u svim primjerima svih simulacija na jednu vrijednost (dobivenu iz promatrane aplikacije). Razlog tome jest da se radi o vrlo malom, odnosno nikakvom, utjecaju na izlaznu vrijednost s obzirom na tip aplikacija koje su korištene u ovom istraživanju. S obzirom je istraživanje bazirano na stranu aplikacijske usluge, ono što se događa na strani *IaaS* poslužitelja manje je interesantno (primjerice raspodjela instanciranih virtualnih strojeva na *host* uređaje). Stoga su specifikacije *host* uređaja fiksirane i ne mijenjaju se kroz simulacije. Naravno, one ne utječu na izlazne vrijednosti prosječnog troška i vremena izvođenja *cloudlet*-a.

Što se tiče parametara virtualnih jedinica, broj procesorskih jezgri, mrežna propusnost i memorijski spremnik u svim su primjerima postavljeni na vrijednosti korištene u referentnoj aplikaciji.

Također, aplikacijska usluga ne mijenja određene parametre kroz iteracije primjera, a radi se o: broju procesorskih jedinica potrebnih za izvođenje, korištenoj skladišnoj memoriji, izlaznoj veličini datoteke i mrežnoj iskoristivosti.

Napravljeno je pet zasebnih simulacija koje variraju prema korištenim ulaznim parametrima i promatranoj izlaznoj varijabli.

Prve dvije simulacije odnose se na slučajeve koji u sebi sadrže primjere svih triju rasporeda. Dakle, napravljena je kombinacija simulacija *space-shared, time-shared* i *time-shared* s *autoscaling*-om rasporeda dodjele resursa virtualnih mašina. Kao izlazna varijabla promatrana je prosječna vrijednost troška (engl. *cost*). Nakon inicijalnih simulacija s kombinacijom algoritama dodjele resursa, generirani su primjeri kod kojih je korišten samo jedan od navedenih algoritama kako bi se provjerilo hoće li podaci izolirani prema zasebnom rasporedu dodjele resursa doprinijeti točnosti predikcije troška.

On-demand-instance, kombiniran raspored dodjele resursa

On-demand-instance naplatni model ima izgeneriranih 750 000 primjera. Vrijednosti ulaznih varijabli unesene su u tablici 1, a primjeri su generirani tako da obuhvaćaju sve kombinacije tabličnih vrijednosti. Uz svaku kombinaciju stoje po tri ulazna primjera, za svaki raspored po jedan, a uz svaki primjer dobivena je izlazna vrijednost troška izračunatog po dotičnom naplatnom modelu.

Tablica 1. Vrijednosti ulaznih parametara *on-demand-instance* naplatnog modela s kombinacijom sva tri načina dodjele resursa

Broj korisničkih zahtjeva	5, 25, 40, 50, 65
Međudolazna vremena zahtjeva (sec)	0.0, 2.0, 5.0, 10.0, 15.0
Broj instrukcija *cloudlet*-a (izražen u milijunima, MI vrijednost)	3631, 6631, 13262, 19893, 21595
Iskorištenje procesora VM (%)	1.5, 6.346, 9.849, 11.412, 14.211
RAM memorija *cloudlet*-a (MB)	15, 66, 90, 120, 140
Brzina procesora VM (MIPS vrijednost)	600, 800, 1000, 2000, 4000
RAM memorija VM (MB)	512, 1024, 2048, 4096
Cijena virtualne jedinice ($/h)	0.0104, 0.013, 0.0156, 0.0179

Potrebno je naglasiti da, osim navedenih atributa, svaki primjer ove simulacije sadrži još tri dodatna parametra. Ti parametri imaju binarnu (0-1) vrijednost koja govori o kojem *scheduler* algoritmu je riječ u dotičnom primjeru. Primjerice, ako trenutni primjer radi na principu *time-shared* politike dodjele resursa, onda je značajka tog algoritma postavljena u 1, dok su značajke *space-shared* i *time-shared autoscaling* algoritma postavljene u 0.

Pay-as-you-go, kombiniran raspored dodjele resursa

Druga simulacija odnosi se na *pay-as-you-go* naplatni model uz, kao i u prošlom primjeru, promatrana sva tri tipa rasporeda dodjele resursa. Za potrebe regresijskih modela izgenerirano je 157 464 ulazna primjera. Vrijednosti atributa koji sačinjavaju navedeni broj kominacija navedene su u tablici 2.

Tablica 2. Vrijednosti ulaznih parametara *pay-as-you-go* naplatnog modela s kombinacijom sva tri načina dodjele resursa

Broj korisničkih zahtjeva	5, 25, 50
Međudolazna vremena zahtjeva (sec)	0.0, 5.0, 10.0
Broj instrukcija *cloudlet*-a (izražen u milijunima, MI vrijednost)	6631, 13262, 19893
Iskorištenje procesora VM (%)	1.5, 6.346, 11.412
RAM memorija *cloudlet*-a (MB)	15, 66, 120
Brzina procesora VM (MIPS vrijednost)	800, 1000, 2000
RAM memorija VM (MB)	1024, 2048
Cijena RAM memorije ($/h)	0.0168, 0.021, 0.0252
Cijena CPU jedinice ($/h)	0.0112, 0.014, 0.0168
Cijena memorije skladištenja ($)	0.0016, 0.002
Cijena mrežne propusnosti ($)	0.016, 0.02

Zahtjevi aplikacija, korištenih u istraživanju, za resursima skladištenja i mrežne propusnosti su minorni, stoga su smješteni u najniži razred naplatnih modela za dotična sredstva.

Isto kao i u prethodnom, *on-demand-instance*, slučaju, i ova simulacija sadrži tri dodatna atributa koji daju informaciju o trenutno korištenom *scheduler* algorimu.

On-demand-instance, odvojen raspored dodjele resursa u 3 skupa primjera

Nakon opisanih dviju simulacija i dobivenih odgovarajućih regresijskih modela, istraživanje je nastavljeno simuliranjem odvojenih pravila dodjele resursa. Dakle, napravljene su tri simulacije za *on-demand-instance* naplatnu metodu (jedna za *time-shared scheduler*, druga za *space-shared* i treća za *time-shared* s *autoscaling*-om). Tablica 3 prikazuje kombinacije parametara korištene u generiranju primjera za *time-shared* i *space-shared* raspored dodjele resursa. Potrebno je naglasiti da su tri atributa postavljena na fiksne vrijednosti za razliku od dosad opisanih simulacija. Radi se o parametrima:

- RAM memorije virtualnog stroja (postavljen na 1024 MB)
- RAM memorija *cloudlet*-a (postavljen na 66 MB)
- iskorištenje procesora virtualne jedinice od strane *cloudlet*-a (6.346%).

Navedeni parametri ne utječu na izlaz, to jest na trošak aplikacijskog zahtjeva izračunat *on-demand-instance* metodom.

Tablica 3. Vrijednosti ulaznih parametara *on-demand-instance* naplatnog modela za *time-shared* i *space-shared* raspored dodjele resursa

Broj korisničkih zahtjeva	5, 25, 50, 75
Međudolazna vremena zahtjeva (sec)	0.0, 5.0, 10.0, 15.0
Broj instrukcija *cloudlet*-a (izražen u milijunima, MI vrijednost)	3631, 6631, 13262, 19893
Brzina procesora VM (MIPS vrijednost)	800, 1000, 2000
Cijena virtualne jedinice ($/h)	0.0104, 0.013, 0.0156, 0.0179

Kod vremenskog rasporeda dodjele resursa koji koristi automatsko skaliranje, potrebni su atributi zanemareni u prethodnoj tablici. RAM memorija virtualne jedinice te isti taj atribut kod aplikacijskog zadatka (*cloudlet*-a) određuju trenutnu razinu iskorištene RAM memorije VM. Ukoliko s novim *clodulet*-om ona prelazi 85% kapaciteta virtualnog stroja, automatski se instancira nova mašina istih specifikacija. Isto vrijedi i za razinu iskorištenog procesorskog kapaciteta. Stoga su vrijednosti tih triju atributa varijabilne kod *time-shared* simulacije s automatskim skaliranjem. Njihove vrijednosti navedene su u tablici 4. Ostali atributi su pri generiranju primjera ostali isti.

Tablica 4. Vrijednosti triju preostalih ulaznih parametara *on-demand-instance* naplatnog modela za *time-shared* metodu s automatskim skaliranjem

Iskorištenje procesora VM (%)	1.5, 6.346, 11.412, 14.211
RAM memorija *cloudlet*-a (MB)	15, 66, 120
RAM memorija VM (MB)	512, 1024, 2048

Potrebno je naglasiti da *space-shared* i *time-shared* simulacije imaju po 768 primjera, a verzija vremenske podjele s automatskim skaliranjem ima 27648 izgeneriranih primjera. Veći broj atributa, koji nisu fiksirani na jednu vrijednost, uzrokuje drastično povećanje za potrebnim brojem kombinacija, odnosno ulaznih primjera za strojno učenje.

Pay-as-you-go, odvojen raspored dodjele resursa u 3 skupa primjera

Slična situacija je i s *pay-as-you-go* naplatnom metodom. Nakon što je napravljena simulacija u kojoj su primjeri sadržavali kombinacije sva tri rasporeda dodjele resursa te nakon što su dobiveni regresijski modeli u dijelu sa strojnim učenjem, idući pokušaj je odvojiti algoritme dodjele resursa u zasebne simulacije. Na taj način formiraju se tri reda ulaznih parametara te se očekuje da će se regresijski modeli bolje prilagoditi takvim jednostavnijim podacima.

Najprije su generirani podaci za prostornu i vremensku dodjelu resursa (bez automatskog skaliranja). Za razliku od *on-demand-instance* naplate, u ovom slučaju vrijednosti RAM memorije kod virtualne jedinice i *cloudlet*-a imaju puno veću ulogu. Razlog tome jest što je to jedan od resursa koji se naplaćuje prema *pay-as-you-go* modelu. No, utilizacija procesora virtualne jedinice i dalje ne igra neku ulogu, stoga je njezina vrijednost fiksirana na 6.346% kod trenutno promatrana dva *scheduler*-a. Ostali ulazni parametri dani su u tablici 5.

Tablica 5. Vrijednosti ulaznih parametara *pay-as-you-go* naplatnog modela za *time-shared* i *space-shared* raspored dodjele resursa

Broj korisničkih zahtjeva	5, 25, 50
Međudolazna vremena zahtjeva (sec)	0.0, 5.0, 10.0
Broj instrukcija *cloudlet*-a (izražen u milijunima, MI vrijednost)	6631, 13262, 19893
RAM memorija *cloudlet*-a (MB)	15, 66, 120
Brzina procesora VM (MIPS vrijednost)	800, 1000, 2000
RAM memorija VM (MB)	1024, 2048
Cijena RAM memorije ($/h)	0.0168, 0.021, 0.0252
Cijena CPU jedinice ($/h)	0.0112, 0.014, 0.0168
Cijena memorije skladištenja ($)	0.0016, 0.002
Cijena mrežne propusnosti ($)	0.016, 0.02

Kod rasporeda vremenske dodjele resursa s automatskim skaliranjem, važnu ulogu imaju svi atributi. Stoga, CPU utilizacija cloudlet-a neće imati fiksnu vrijednost, nego njezine vrijednosti iznose: 1.5%, 6.346% i 11.412%. Ostali atributi kreću se u rasponima navedenim u prethodnoj tablici.

Ove tri simulacije imaju nešto veći broj primjera u odnosu na one s *on-demand-instance* naplatom jer izlaz formira veći broj atributa. Prostorna i vremenska podjela resursa generiraju po 17 496, dok raspored s *autoscaling*-om sadrži 52 488 primjera.

Vrijeme odziva*, odvojen raspored dodjele resursa u 3 skupa primjera*

Osim vrijednosti potrošnje, u regresijskim modelima je prosječno vrijeme izvođenja aplikacijskog zadatka, također, promatrano kao izlazna varijabla. Napravljene su tri simulacije, odnosno tri reda ulaznih primjera. Prvi red generiranih primjera odnosi se na raspored vremenske dodjele resursa, drugi red na *space-shared* i, u konačnici, treći skup na raspored s automatskim skaliranjem. Naravno, u ovim slučajevima cijene pojedinačnih resursa, odnosno instanci virtualnih jedinica, nisu potrebni atributi jer ne utječu na vrijeme izvođenja aplikacijskog zadatka.

Za *time-shared* i *space-shared* raspored, korišteni su atributi navedeni u tablici 6. Vrijednosti RAM memorije *cloudlet*-a (66 MB), RAM virtualnog stroja

(1024 MB) i *cloudlet*-ova CPU utilizacija (6.346%) su fiksirane jer nam one nisu važne za ove rezultate.

Tablica 6. Vrijednosti ulaznih parametara *response time* modela za *time-shared* i *space-shared* raspored dodjele resursa

Broj korisničkih zahtjeva	5, 25, 50, 75, 100
Međudolazna vremena zahtjeva (sec)	0.0, 2.0, 5.0, 10.0, 15.0
Broj instrukcija *cloudlet*-a (izražen u milijunima, MI vrijednost)	3631, 6631, 13262, 19893, 21595
Brzina procesora VM (MIPS vrijednost)	600, 800, 1000, 2000, 4000

Kao i u prethodnim simulacijama, *time-shared scheduler* s *autoscaling*-om koristi više parametara, odnosno navedena tri nisu fiksirana na jednu vrijednost. Kroz primjere se kreću po vrijednostima navedenim u tablici 7. Ostali atributi ostaju isti.

Tablica 7. Preostale vrijednosti ulaznih parametara *response time* modela za *time-shared* raspored s automatskim skaliranjem

Iskorištenje procesora VM (%)	1.5, 6.346, 9.849, 11.412, 14.211
RAM memorija *cloudlet*-a (MB)	15, 66, 90, 120, 140
RAM memorija VM (MB)	512, 1024, 2048, 4096

Prva dvije simulacije (vremenska i prostorna dodjela resursa) imaju 625 izgeneriranih ulaznih primjera za regresijske modele. Treća simulacija (*autoscaling*) sadrži 62 500 primjera.

3.4. Primjena regresijskih modela nad generiranim podacima

Nakon generiranja ulaznih primjera nad svakim skupom podataka konstruiran je model linearne i polinomijalne regresije.

Rezultati simulacija zapisani su u datotekama koje je najprije trebalo isparsirati. Parser, koji je implementiran u Pythonu, radi tako da vrijednosti ulaznih

atributa uzima iz datoteke i slaže u dvodimenzionalno polje. Ulazni atributi formiraju polje m*n gdje je m broj ulaznih primjera, a n broj atributa koji utječu na izlaz (nezavisne varijable). Na sličan način konstruirano je jednodimenzionalno polje s vrijednostima izlazne, zavisne varijable, veličine m koja označava broj dobivenih primjera. Parametri koji su u simulaciji bili fiksirani na jednu vrijednost i koji nemaj utjecaj na izlaznu varijablu nisu uzeti u obzir pri konstruiranju modela.

Odabir modela, odnosno optimizacija hiperparametara, napravljena je unakrsnom provjerom na tri disjunktna skupa:

- skup za učenje (engl. *training set*)
- skup za provjeru (engl. *validation set*)
- skup za ispitivanje (engl. *test set*)

Podjela ulaznih primjera napravljena je tako da 50% primjera čini skup za učenje, 25% skup za provjeru te, u konačnici, 25% testni skup. Algoritam odabira modela napravljen je tako da se najprije model nauči na skupu za učenje. Zatim se izračuna pogreška na skupu za provjeru. Ta dva koraka se ponavljaju za sve kombinacije hiperparametara. U konačnici se odabiru oni hiperparametri koji daju najmanju pogrešku na validacijskom skupu. Nakon toga, za odabrane hiperparametre, se primjenjuje postupak učenja (linearnom ili polinomijalnom regresijom). Dobivena empirijska pogreška odnosi se na taj ujedinjeni skup primjera (skup za učenje + skup za provjeru), a generalizacijska pogreška je izračunata nad ispitnim skupom. Osim generalizacijske pogreške, nad istim skupom je dobivena i srednja kvadratna pogreška koja daje vjeran prikaz pogreške tog modela.

Linearna regresija, kod svih ulaznih podataka, ima isti hiperparametar. Radi se o regularizacijskom faktoru λ. Optimum tog faktora tražio se između sljedećih vrijednosti: 0, 1, 2, 3, 4, 5, 10, 15, 20, 30, 40, 50, 60, 70, 80, 90, 100.

Kod polinomijalne regresije, osim optimalnog regularizacijskog faktora, potrebno je pronaći i najpovoljniji stupanj polinoma. Stoga su optimalni hiperparametri ispitivani kroz dvije *for* petlje. Faktor λ je, kod svih simulacijskih primjera, ispitivan između ovih vrijednosti: 0, 5, 10, 15, 20, 30, 40, 50, 60, 70, 80, 90, 100. Stupanj polinomijalne regresije varira u ovisnosti o ulaznom setu podataka, odnosno o broju ulaznih značajki i dobivenih primjera.

Python skripte polinomijalne regresije izvodile su se na poslužitelju zbog prevelike računalne složenosti algoritma. Model korištenog poslužitelja jest Dell PowerEdge r710 te ima 64 GB RAM memorije i 2 procesora s ukupno 24 jezgre. Ispitivane vrijednosti stupnja polinomijalne regresije prilagođene su pojedinom simulacijskom slučaju, a vrijednosti za svaki od modela navedene su u nastavku.

On-demand-instance, kombinirana sva tri rasporeda dodjele resursa

Optimalni stupanj polinomijalne regresije tražio se između vrijednosti: 2, 3

Isto ispitivanje odnosi se i na *pay-as-you-go* naplatni model koji kombinira sva tri rasporeda dodjele.

Simulirani primjeri s odvojenim rasporedima dodjele resursa:

***On-demand-instance, time-shared i space-shared* raspored**

Optimalni stupanj polinomijalne regresije tražio se između vrijednosti: 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

***On-demand-instance, time-shared* raspored s automatskim skaliranjem**

Optimalni stupanj polinomijalne regresije tražio se između vrijednosti: 2, 3, 4, 5

Pay-as-you-go, time-shared, space-shared i time-shared s automatskim skaliranjem

Optimalni stupanj polinomijalne regresije tražio se između vrijednosti: 2, 3, 4

Vrijeme izvođenja, time-shared i space-shared raspored

Optimalni stupanj polinomijalne regresije tražio se između vrijednosti: 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

Vrijeme izvođenja, time-shared raspored s automatskim skaliranjem

Optimalni stupanj polinomijalne regresije tražio se između vrijednosti: 2, 3, 4, 5

4. Rezultati

Ovo poglavlje prikazuje rezultate strojnog učenja te daje usporedbu među pojedinim regresijskim modelima. Rezultati su složeni prema ulaznim skupovima podataka. Slike prikazuju usporedbu stvarne vrijednosti izlazne varijable i izlazne vrijednosti modela na 100 primjera ispitnog skupa. U opisu svake slike napisan je korišten regularizacijski model te optimalni hiperparametri. Uz to su za svaki model navedene vrijednosti pogrešaka na skupu za treniranje (skup za učenje + skup za provjeru) i na ispitnom skupu te, najvjerodostojnija, srednja kvadratna pogreška na ispitnom skupu.

Uz svaku linearnu regresiju navedene su težine, koje su konstrukcijom modela pridijeljene svakom atributu, u formi atribut: težinski koeficijent. Na taj način vidljiv je utjecaj pojedinih parametara na izlaznu vrijednost što je, uz predikciju, glavni cilj ovog istraživanja.

Usporedba između regresijskih modela s različitim ispitnim skupom napravljena je pomoću relativne pogreške. Ona je određena omjerom:

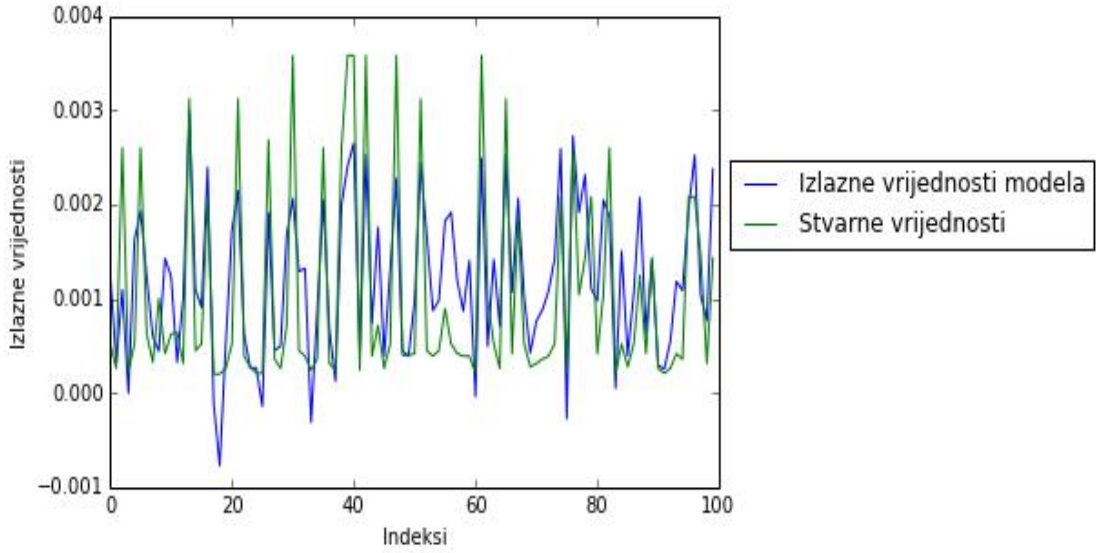
$$\frac{MSE\ ispitnog\ skupa}{varijanca\ izlazne\ varijable\ ispitnog\ skupa}\ . \tag{7}$$

Varijanca izlazne varijable određena je izrazom:

$$E[(y-\bar{y})^2] = \frac{1}{N-1}\sum_{i=1}^{N}(y_i-\bar{y})^2 \tag{8}$$

gdje $\bar{y}$ označava srednju izlaznu vrijednost. Korištena je nepristrana, odnosno korigirana varijanca. Relativna pogreška također je naznačena ispod svakog regresijskog modela.

1. *On-demand-instance naplatni model, kombinirana sva tri rasporeda dodjele resursa*



Slika 9. Linearna regresija, reg. faktor = 0

Empirijska pogreška = 0.159265037177

Generalizacijska pogreška = 0.0531760421522

Srednja kvadratna pogreška na ispitnom skupu= $5.67211116291*10^{-7}$

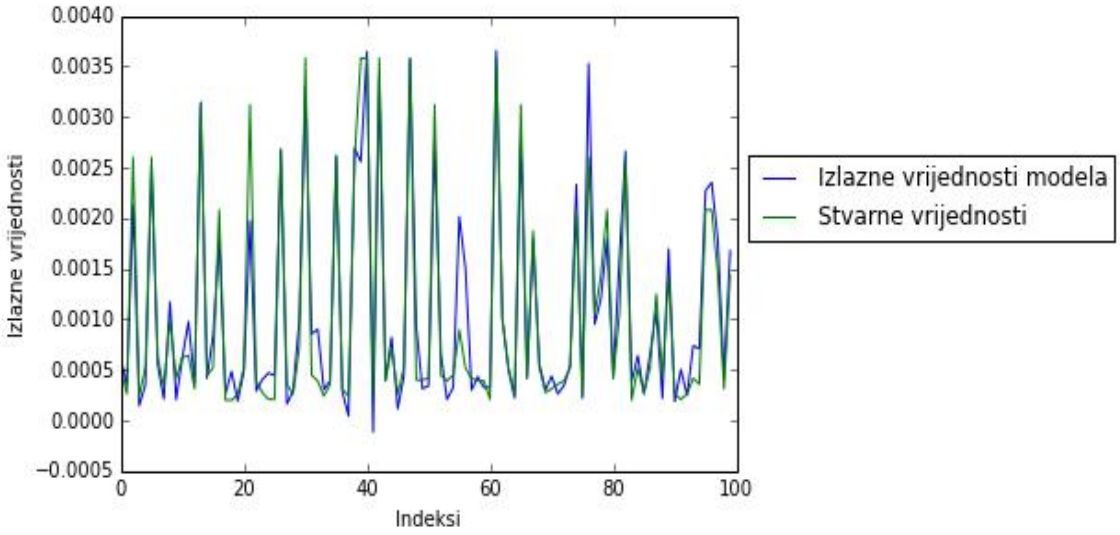
Relativna pogreška: 0.410078612827

Atribut: težinski koeficijent

broj korisničkih zahtjeva: $-3.86505528*10^{-5}$; međudolazno vrijeme zahtjeva: $-3.04298272*10^{-5}$; RAM memorija *cloudlet*-a: $1.01202792*10^{-6}$; utilizacija procesora kod *clodulet*-a: $1.14107252*10^{-5}$; broj instrukcija *cloudlet*-a: $8.70140298*10^{-9}$; RAM memorija VM: $-4.28357276*10^{-8}$; brzina procesora VM: $-4.59170383*10^{-8}$; cijena virtualne jedinice: $7.54888675*10^{-2}$; *time-shared* algoritam: $-2.22325304*10^{-4}$; *space-shared* algoritam: $-2.21114971*10^{-4}$; *time-shared autoscaling* algoritam: $4.43440275*10^{-4}$

U ovom modelu linearne regresije vidljivo je da algoritmi dodjele resursa imaju sličan utjecaj. *Time-shared* raspored s automatskim skaliranjem ima pozitivan težinski faktor jer, u slučaju unaprijed definirane opterećenosti sustava, instancira novu virtualnu jedinicu te, zbog toga, trošak raste. Kod ostala dva algoritma (*time-shared* i *space-shared*) ne dolazi do zakupa novih virtualnih

strojeva, neovisno o opterećenju sustava. Povećanjem broja korisničkih zahtjeva, usprkos opadanju performansi usluge, svi radni zadaci se smještaju na postojeću virtualnu jedinicu. Rezultat toga jest da ukupan trošak ostaje isti, a prosječan trošak *cloudlet*-a opada. Najveći utjecaj na izlaznu vrijednost troška ima cijena virtualnog stroja. Očekivano, povećani zahtjevi za resursima potrebnim za izvršavanje radnog zadatka također utječu na povećanje prosječne cijene izvođenja korisničkog zahtjeva.



Slika 10. Polinomijalna regresija, stupanj polinoma = 3, reg. faktor = 5

Empirijska pogreška = 0.031242536257

Generalizacijska pogreška = 0.0104979797133

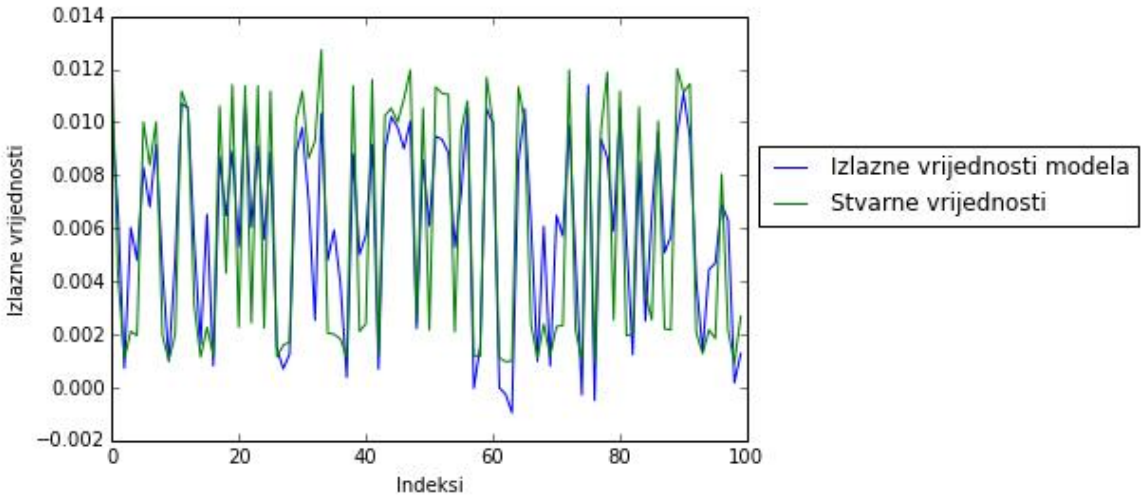
Srednja kvadratna pogreška na ispitnom skupu = $1.11978450275*10^{-7}$

Relativna pogreška: 0.0809574534708

Izlazne vrijednosti ispitnog skupa primjera kreću se između 0.00016 i 0.00716.

Polinomijalna regresija očekivano daje bolje rezultate (model se bolje prilagođava podacima).

2. *Pay-as-you-go naplatni model, kombinirana sva tri rasporeda dodjele resursa*



Slika 11. Linearna regresija, reg. faktor = 0

Empirijska pogreška = 0.29080034587

Generalizacijska pogreška = 0.0973643824858

Srednja kvadratna pogreška na ispitnom skupu = $4.94662310043*10^{-6}$

Relativna pogreška: 0.265461304901

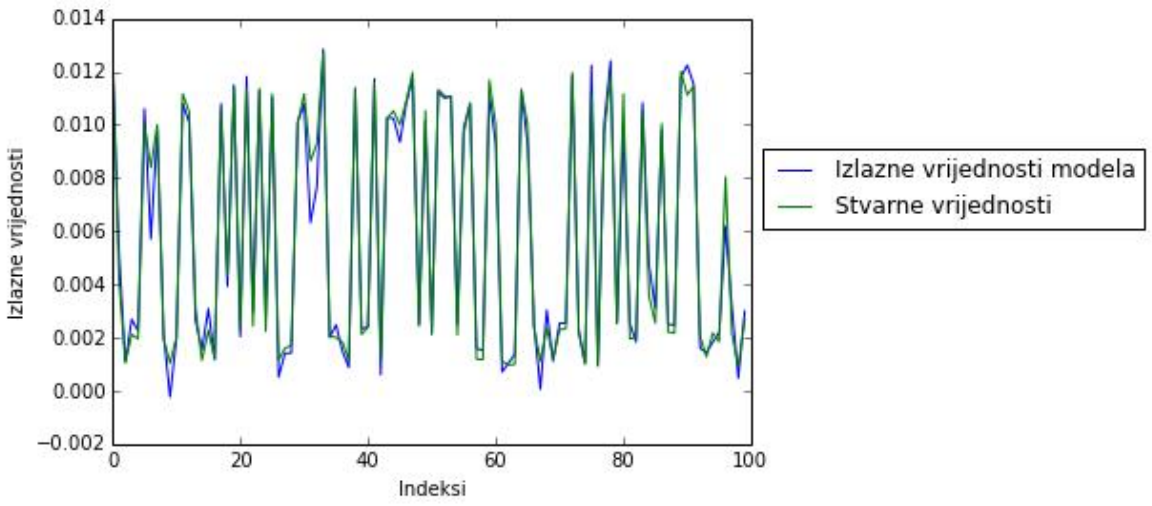
Atribut: težinski koeficijent

broj korisničkih zahtjeva: $-1.94081303*10^{-4}$; međudolazno vrijeme zahtjeva: $-8.63740552*10^{-5}$; RAM memorija *cloudlet*-a: $3.67453760*10^{-6}$; utilizacija procesora kod *clodulet*-a: $4.69086841*10^{-5}$; broj instrukcija *cloudlet*-a: $2.00110138*10^{-8}$; RAM memorija VM: $-5.24658473*10^{-9}$; brzina procesora VM: $-2.02092918*10^{-7}$; cijena RAM memorije: $9.86165722*10^{-2}$; cijena CPU jedinice: $9.43887854*10^{-2}$; cijena memorije skladištenja: $1.09647152*10^{-1}$; cijena mrežne propusnosti: $9.57526520*10^{-2}$; *time-shared* algoritam: $-4.57479095*10^{-4}$; *space-shared* algoritam: $-5.73641382*10^{-4}$; *time-shared autoscaling* algoritam: $1.03112048*10^{-3}$

Kao i u prethodnom, *on-demand-instance*, slučaju, svi algoritmi dodjele resursa imaju slične težinske koeficijente gledajući njihovu apsolutnu vrijednost. Isto tako, cijene resursa, koje definira pružatelj usluge, imaju najveći utjecaj na izlazni trošak.

Uz to vrijedi spomenuti i utjecaj koeficijenta koji stoji uz broj korisničkih zahtjeva. Taj parametar određuje potrošnju resursa (više korisnika zahtijeva veću količinu rezerviranih resursa). Također, ako se radi o algoritmu s *autoscaling*-om, povećanjem broja korisničkih zahtjeva, koje rezultira unaprijed zadanim opterećenjem sustava, dolazi do instanciranja novih virtualnih jedinica. Treba naglasiti da je težinski faktor, koji stoji uz broj korisničkih zahtjeva, negativan jer porastom broja zahtjeva, u principu, opada prosječni trošak po korisniku. Iako dolazak novih zahtjeva, može uzrokovati veću potrebu za resursima, a samim time i veću potrošnju, uglavnom se događa da je određena količina resursa već zakupljena te dolazak novih korisničkih zahtjeva kao posljedicu ima to da ukupan trošak ostaje isti (nije potrebno rezervirati nove *cloud* resurse). Time prosječan trošak po *cloudlet*-u opada. Ključno je da se zakup resursa obično radi prema unaprijed definiranoj mjeri, primjerice, minimalno 1GB RAM memorije. Stoga, ukoliko se radi o aplikacijskom zahtjevu koji treba 66 MB RAM-a za svoje izvođenje, dolazak novih *cloudlet*-a neće zahtijevati novi zakup od 1 GB RAM memorije. Ukupan trošak za ovaj resurs ostaje isti u slučaju jednog ili deset zahtjeva, ali prosječan trošak se, naravno, mijenja. Slično vrijedi i za ostale korištene resurse.

Težinski koeficijent ispred parametra međudolaznog vremena *cloudlet* zahtjeva također je negativnog predznaka. Razlog tome jest da s većim vremenskim intervalom između nadolazećih zahtjeva opada opterećenje sustava te se smanjuje potreba za zakupom novih resursa.



Slika 12. Polinomijalna regresija, stupanj polinoma = 3, reg. faktor = 5

Empirijska pogreška = 0.0175346408246
Generalizacijska pogreška = 0.00588477493583
Srednja kvadratna pogreška na ispitnom skupu = $2.98977540814*10^{-7}$
Relativna pogreška: 0.0160446766429

Izlazne vrijednosti ispitnog skupa primjera kreću se između 0.000912 i 0.0128.

On-demand-instance **naplatni model, rasporedi dodjele resursa razdvojeni na zasebne modele**

3. *On-demand-instance naplatni model, time-shared raspored*

U ovom slučaju imamo deterministički određeno rješenje te nije potrebna primjena regresijskih modela. Naime, *time-shared* algoritam radi na principu da sve pristigle aplikacijske zadatke smješta na istu virtualnu jedinicu. Ne dolazi do instanciranja novih virtualnih strojeva, nego *cloudlet*-i dijele resurse postojećeg. Generirani skup podataka ne razmatra utilizaciju virtualne mašine dulju od jednog sata koji je promatran kao vremenska jedinica naplate. Iz toga je vidljivo da izlazna varijabla, odnosno prosječan trošak *cloudlet*-a, odgovara omjeru:

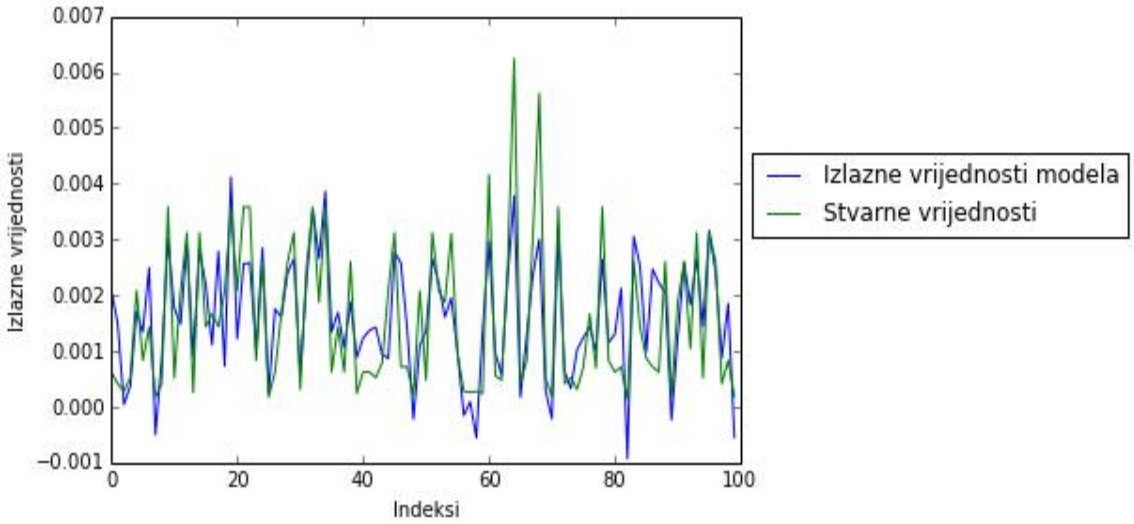
$$\frac{\text{cijena korištenja virtualne jedinice za jedan sat}}{\text{broj pristiglih radnih zadataka}} . \qquad (9)$$

4. *On-demand-instance naplatni model, space-shared raspored*

Space-shared raspored radi na sličnom principu kao *times-shared*. Svi pristigli aplikacijski zadaci koriste resurse samo jedne virtualne jedinice, odnosno ne dolazi do zakupljenja novih strojeva. Iz tog razloga vrijedi ista formula za izračun prosječnog troška *cloudlet*-a, odnosno rješenje je deterministički određeno.

Iz formule (9) vidljivo je da, u ovim slučajevima, jedini utjecaj na trošak imaju cijena virtualne jedinice te broj pristiglih *cloudlet*-a, odnosno upućenih korisničkih zahtjeva.

5. *On-demand-instance naplatni model, time-shared raspored* s automatskim skaliranjem



Slika 13. Linearna regresija, reg. faktor = 0

Empirijska pogreška = 0.00842694908633

Generalizacijska pogreška = 0.00278966122255

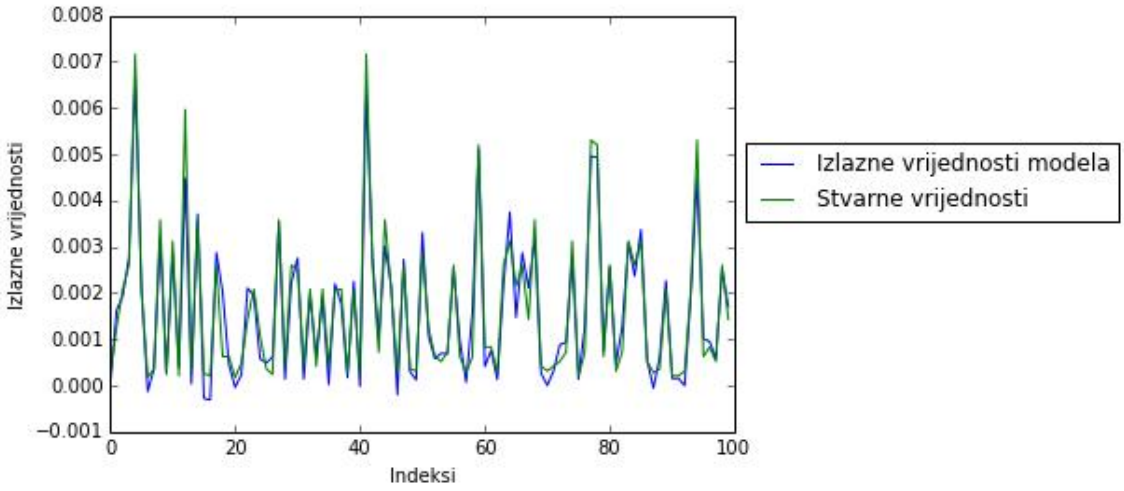
Srednja kvadratna pogreška na ispitnom skupu = $8.07193640784*10^{-7}$

Relativna pogreška: 0.439133906816

Atribut: težinski koeficijent

broj korisničkih zahtjeva: $-2.86142781*10^{-5}$; međudolazno vrijeme zahtjeva: $-9.15518952*10^{-5}$; RAM memorija *cloudlet*-a: $3.81622930*10^{-6}$; utilizacija procesora kod *clodulet*-a: $2.76380257*10^{-5}$; broj instrukcija *cloudlet*-a: $1.69875964*10^{-8}$; RAM memorija VM: $-2.31282450*10^{-7}$; brzina procesora VM: $-1.14594863*10^{-7}$; cijena virtualne jedinice: $1.04445820*10^{-1}$

Za ovaj slučaj linearne regresije najveći težinski faktor pripada atributu cijene. Uz njega se mogu spomenuti još i broj korisnika, vrijeme između dva zahtjeva upućena za izvođenjem *cloudlet*-a te utilizacija procesora. Ta tri parametra uvelike utječu na automatsko skaliranje. Povećanje faktora utilizacije procesora uzrokuje veću potrebu za novim virtualnim strojevima zbog opterećenosti postojećih.



Slika 14. Polinomijalna regresija, stupanj polinoma = 3, reg. faktor = 0

Empirijska pogreška = 0.00144581651222

Generalizacijska pogreška = 0.000480718892616

Srednja kvadratna pogreška na ispitnom skupu = 1.39096901799e-07

Relativna pogreška: 0.0756722585842

Izlazne vrijednosti ispitnog skupa primjera kreću se između 0.000139 i 0.00716.

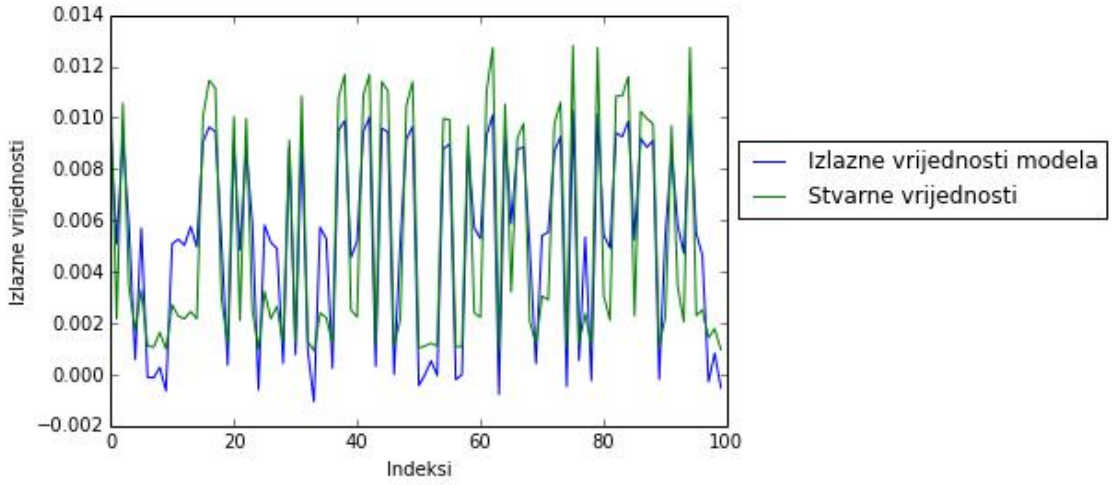
Razdvajanje ulaznih podataka na tri skupa rezultiralo je time da *time-shared* i *space-shared* algoritmi sada imaju deterministično određivanje izlazne varijable troška, što je, svakako, bolja situacija u odnosu na korištenje modela strojnog učenja po pitanju predikcije na ispitnom skupu.

Usporedba između regresijskih modela *time-shared* rasporeda s automatskim skaliranjem i početnih modela, kod kojih su svi algoritmi dodjele resursa bili unutar iste simulacije, napravljena je pomoću relativne pogreške.

Modele nije moguće usporediti isključivo preko srednje kvadratne pogreške jer nemaju isti ispitni skup. Relativna pogreška kod navedenih dviju simulacija govori da su prema točnosti predikcije slični. Početni model s kombiniranim primjerima rasporeda dodjele resursa ima manju relativnu pogrešku u linearnom slučaju, dok za slučaj polinomijalne regresije manju pogrešku ima *time-shared* algoritam s automatskim skaliranjem.

Pay-as-you-go naplatni model, rasporedi dodjele resursa razdvojeni na zasebne modele

6. *Pay-as-you-go naplatni model, time-shared raspored*



Slika 15. Linearna regresija, reg. faktor = 0

Empirijska pogreška = 0.026889688363

Generalizacijska pogreška = 0.00923836545002

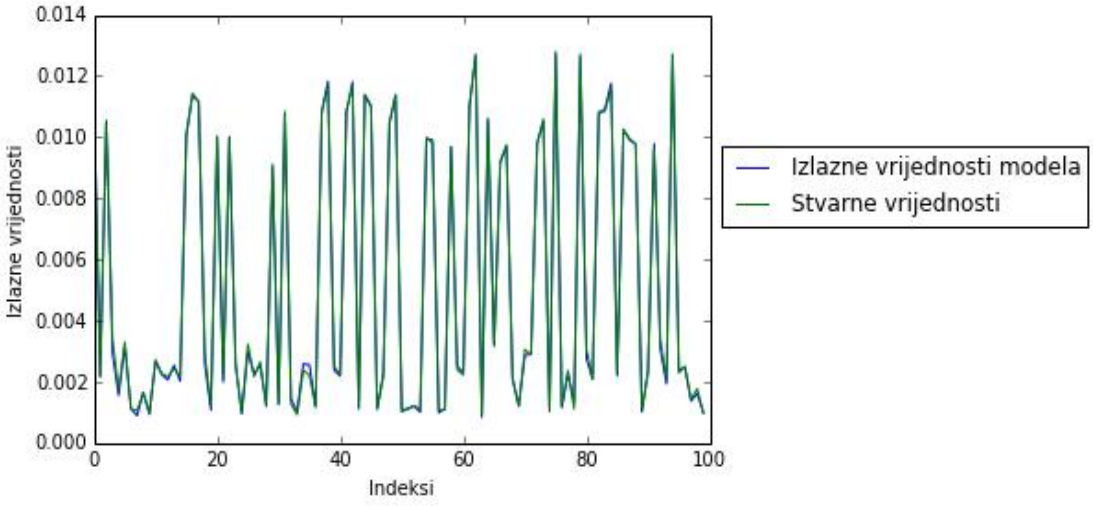
Srednja kvadratna pogreška na ispitnom skupu = $4.22421831277*10^{-6}$

Relativna pogreška: 0.219149225425

Atribut: težinski koeficijent

broj korisničkih zahtjeva: $-2.09553739*10^{-4}$; međudolazno vrijeme zahtjeva: $-1.17842244*10^{-5}$; RAM memorija *cloudlet*-a: $1.55654373*10^{-6}$; broj instrukcija *cloudlet*-a: $3.15620773*10^{-9}$; RAM memorija VM: $1.84789771*10^{-7}$; brzina procesora VM: $-5.53064265*10^{-8}$; cijena RAM memorije: $9.34504821*10^{-2}$; cijena CPU jedinice: $8.79559170*10^{-2}$; cijena memorije skladištenja: $1.30905189*10^{-1}$; cijena mrežne propusnosti: $9.21933491*10^{-2}$

Parametri s najvećim težinskim koeficijentima su ponovno cijene resursa. Slično kao i kod prethodno opisanog linearnog modela (*on-demand-instance* s algoritmom automatskog skaliranja), vrijedno je spomenuti atribute broja korisnika koji pristupaju aplikaciji te vrijeme između njihovih zahtjeva. Oni utječu na razinu utilizacije resursa na virtualnim jedinicama te time uzrokuju veću ili manju potrošnju resursa.



Slika 16. Polinomijalna regresija, stupanj polinoma = 3, reg. faktor = 0

Empirijska pogreška = $8.70991537457*10^{-5}$

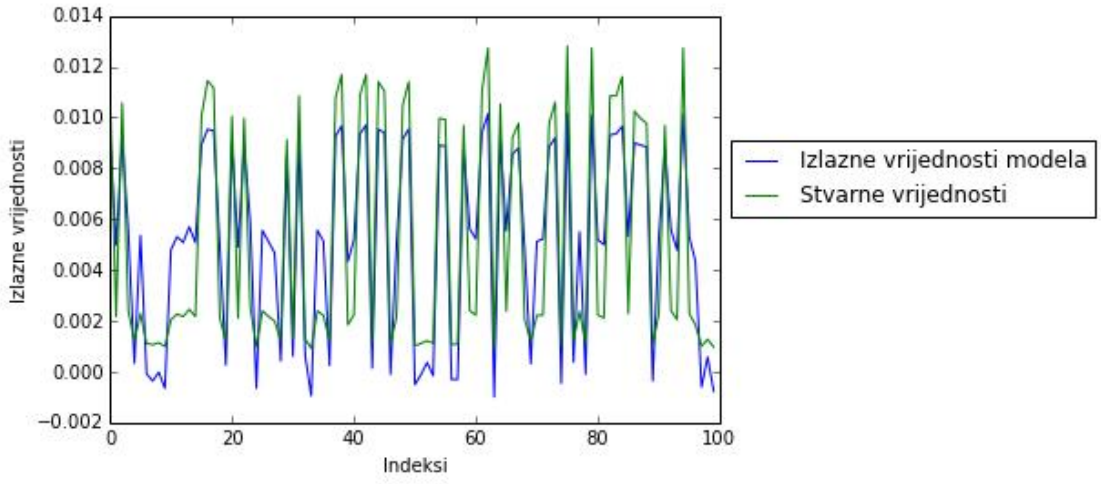
Generalizacijska pogreška = $2.95922345541*10^{-5}$

Srednja kvadratna pogreška na ispitnom skupu = $1.35309714468*10^{-8}$

Relativna pogreška: 0.000701976482336

Izlazne vrijednosti ispitnog skupa primjera kreću se između 0.000912 i 0.0128.

7. *Pay-as-you-go naplatni model, space-shared raspored*



Slika 17. Linearna regresija, reg. faktor = 0

Empirijska pogreška = 0.0285858218337

Generalizacijska pogreška = 0.0098241594563

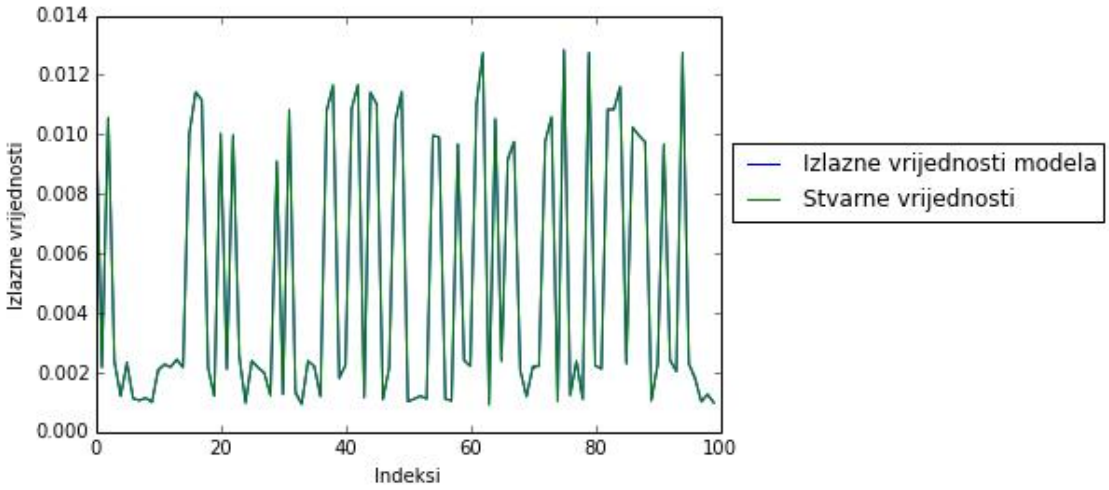
Srednja kvadratna pogreška na ispitnom skupu = $4.49207108198*10^{-6}$

Relativna pogreška: 0.226639771151

Atribut: težinski koeficijent

broj korisničkih zahtjeva: $-2.11722832*10^{-4}$; međudolazno vrijeme zahtjeva: $-3.64353365*10^{-6}$; RAM memorija *cloudlet*-a: $1.87803796*10^{-8}$; broj instrukcija *cloudlet*-a: $-1.03753525*10^{-9}$; RAM memorija VM: $-1.61961869*10^{-8}$; brzina procesora VM: $-1.24603666*10^{-8}$; cijena RAM memorije: $8.84144393*10^{-2}$; cijena CPU jedinice: $8.83092071*10^{-2}$; cijena memorije skladištenja: $1.37247109*10^{-1}$; cijena mrežne propusnosti: $9.19185370*10^{-2}$

Najveći težinski faktori pripadaju cijenama resursa te, u drugom redu, broju korisnika koji pristupaju aplikaciji.



Slika 18. Polinomijalna regresija, stupanj polinoma = 3, reg. faktor = 0

Empirijska pogreška = $6.1001345208*10^{-6}$

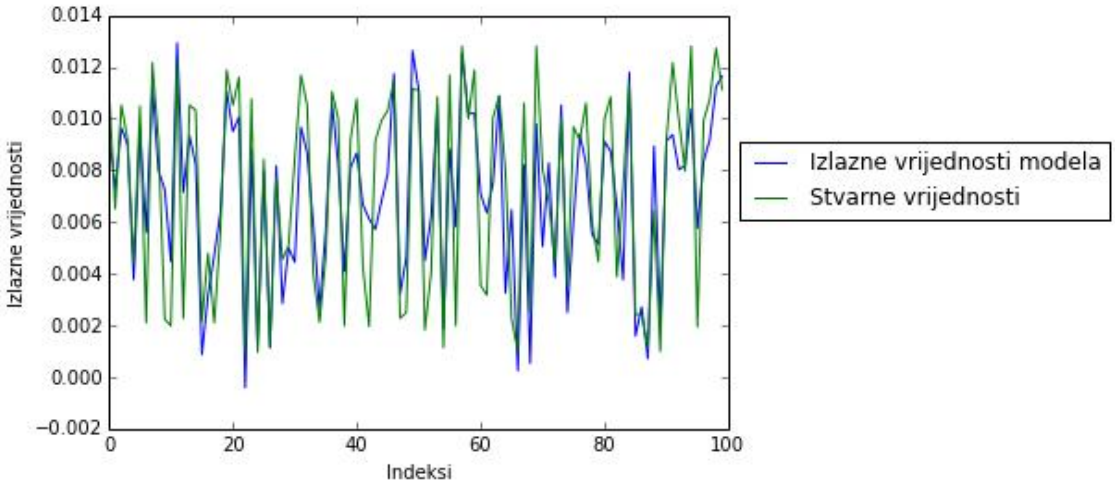
Generalizacijska pogreška = $2.02273417234*10^{-6}$

Srednja kvadratna pogreška na ispitnom skupu = $9.24889882184*10^{-10}$

Relativna pogreška: $4.66637387104*10^{-5}$

Izlazne vrijednosti ispitnog skupa primjera kreću se između 0.000912 i 0.0128.

8. *Pay-as-you-go naplatni model, time-shared raspored* s automatskim skaliranjem



Slika 19. Linearna regresija, reg. faktor = 0

Empirijska pogreška = 0.0902850610547

Generalizacijska pogreška = 0.029381404284

Srednja kvadratna pogreška na ispitnom skupu = $4.47818995335*10^{-6}$

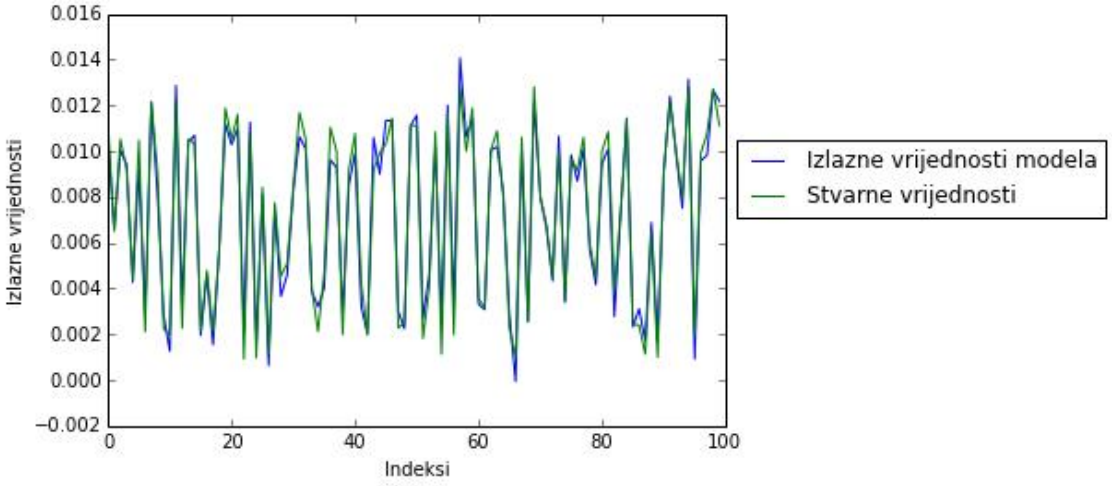
Relativna pogreška: 0.291532621949

Atribut: težinski koeficijent

broj korisničkih zahtjeva: $-1.60915150*10^{-4}$; međudolazno vrijeme zahtjeva: $-2.49678370*10^{-4}$; RAM memorija *cloudlet*-a: $9.64598766*10^{-6}$; utilizacija procesora kod *clodulet*-a: $1.41728790*10^{-4}$; broj instrukcija *cloudlet*-a: $5.57332460*10^{-8}$; RAM memorija VM: $-1.96806562*10^{-7}$; brzina procesora VM: $-5.68461158*10^{-7}$; cijena RAM memorije: $1.18581609*10^{-1}$; cijena CPU jedinice: $1.15976547*10^{-1}$; cijena memorije skladištenja: $1.26459765*10^{-1}$; cijena mrežne propusnosti: $1.13926570*10^{-1}$

Kao što je slučaj i s prethodna dva modela koja pripadaju *pay-as-you-go* naplatnoj metodi, izlaz u najvećoj mjeri formiraju cijene resursa te atributi koji utječu na utilizaciju, korisničko ponašanje i iskoristivost procesora od strane radnog zadatka. U korištenim primjerima aplikacija puno manji utjecaj na utilizaciju čini RAM memorija *cloudlet*-a, odnosno instanciranje novih virtualnih jedinica većinom je uzrokovano zauzećem CPU resursa. Predznak težinskog faktora koji

stoji uz parametar RAM memorije radnog zadatka je, također, pozitivan jer njegovim povećanjem raste opterećenost sustava i potreba za novim resursima.



Slika 20. Polinomijalna regresija, stupanj polinoma = 4, reg. faktor = 100

Empirijska pogreška = 0.0100199295441

Generalizacijska pogreška = 0.00339926896259

Srednja kvadratna pogreška na ispitnom skupu = $5.18102265294*10^{-7}$

Relativna pogreška: 0.0337287416149

Izlazne vrijednosti ispitnog skupa primjera kreću se između 0.000912 i 0.0128.

Odvajanjem rasporeda resursa na tri modela *time-shared* i *space-shared* algoritam daju manju relativnu pogrešku u odnosu na početni *pay-as-you-go* model koji je sadržavao sve kombinacije rasporeda.

Algoritam *time-shared* rasporeda koji implementira automatsko skaliranje ima malo manju predikcijsku točnost u odnosu na početni model. Ipak, ta razlika je manja u odnosu na onu između *time-shared* ili *space-shared* na jednoj strani i početnog modela na drugoj strani.

Iz svega navedenog može se zaključiti da je bolje odvojiti rasporede dodjele resursa na posebne modele. Osim što dobiveni rezultati ukazuju na to, pojedine značajke nemaju isti utjecaj kod svakog algoritma.

Primjerice, pojedini atributi koji određuju izlaznu varijablu kod *autoscaling*-a nemaju nikakvog utjecaja na izlaz kod preostala dva rasporeda.

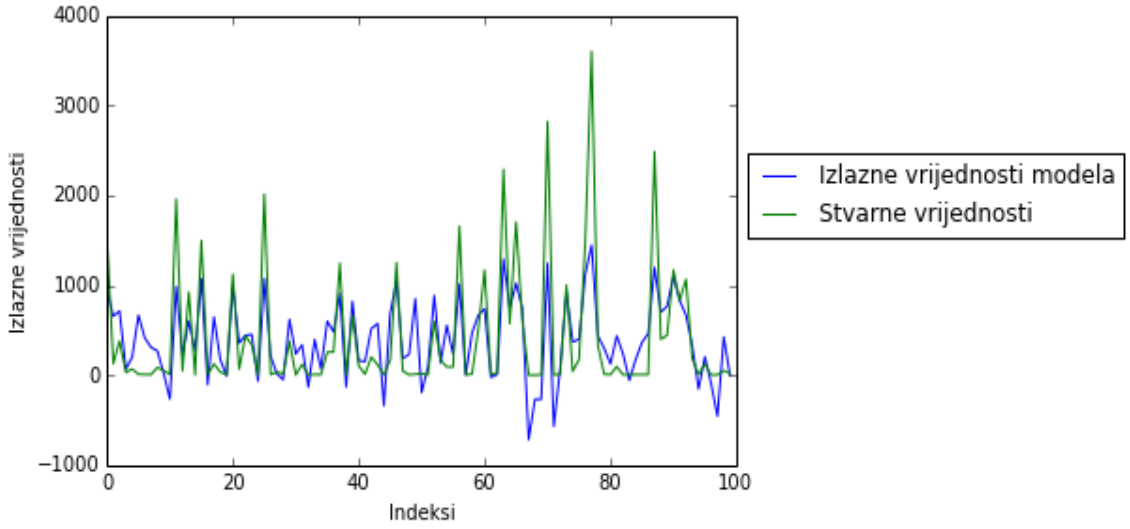
Također je potrebno naglasiti da razdvajanjem algoritama manji broj značajki ulazi u regresijski model koji se onda lakše prilagođava podacima. Kod

simulacije koja sadrži sve algoritme dodjele u svojim primjerima, postoje dodatne tri značajke koje govore koji se algoritam dodjele resursa koristio.

Vrijeme izvođenja, rasporedi dodjele resursa razdvojeni na zasebne modele

Za simulacije kojima je vrijeme odziva izlazna varijabla, istraživanje je provedeno isključivo na modelima s odvojenim pravilima dodjele resursa. Razlog tome objašnjen je nekoliko redova iznad.

9. Vrijeme izvođenja*, time-shared raspored*



Slika 21. Linearna regresija, reg. faktor = 100

Empirijska pogreška = 37326051.7568

Generalizacijska pogreška = 16080635.4012

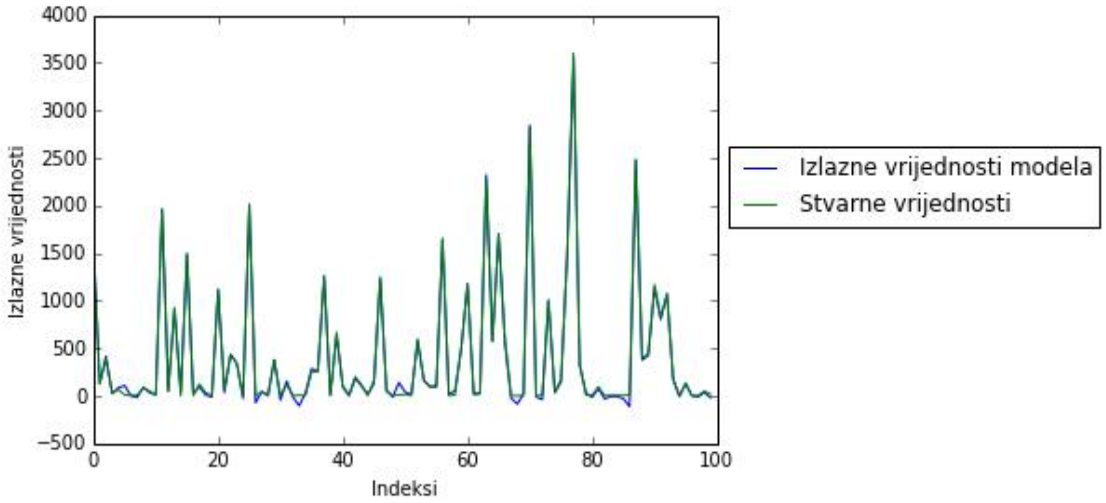
Srednja kvadratna pogreška na ispitnom skupu = 204848.858614

Relativna pogreška: 0.421208687821

Atribut: težinski koeficijent

broj korisničkih zahtjeva: 7.34152831; međudolazno vrijeme zahtjeva: -27.46223413; broj instrukcija *cloudlet*-a: 0.03779254; brzina procesora VM: -0.18376498

U slučaju kada je izlazna varijabla vrijeme odziva, očekivano, najveći utjecaj imaju broj korisničkih zahtjeva i međudolazna vremena. Ako je vremenski interval između dva zahtjeva dovoljno velik da se prethodni zadaci stignu obaviti na virtualnoj jedinici, vrijeme odziva će biti manje, stoga je težinski koeficijent modela uz taj parametar negativnog predznaka. Kod *time-shared* algoritma zadaci dijele resurse virtualnog stroja tako da svaki od njih dobije dio vremena u kojem se izvodi (opisano u poglavlju 3.1.1). Ukoliko je stroj opterećen većim brojem zadataka, vrijeme izvođenja *cloudlet*-a raste. To je razlog pozitivnog predznaka faktora vezanog uz broj nadolazećih zahtjeva.



Slika 22. Polinomijalna regresija, stupanj polinoma = 5, reg. faktor = 100

Empirijska pogreška = 101563.251067

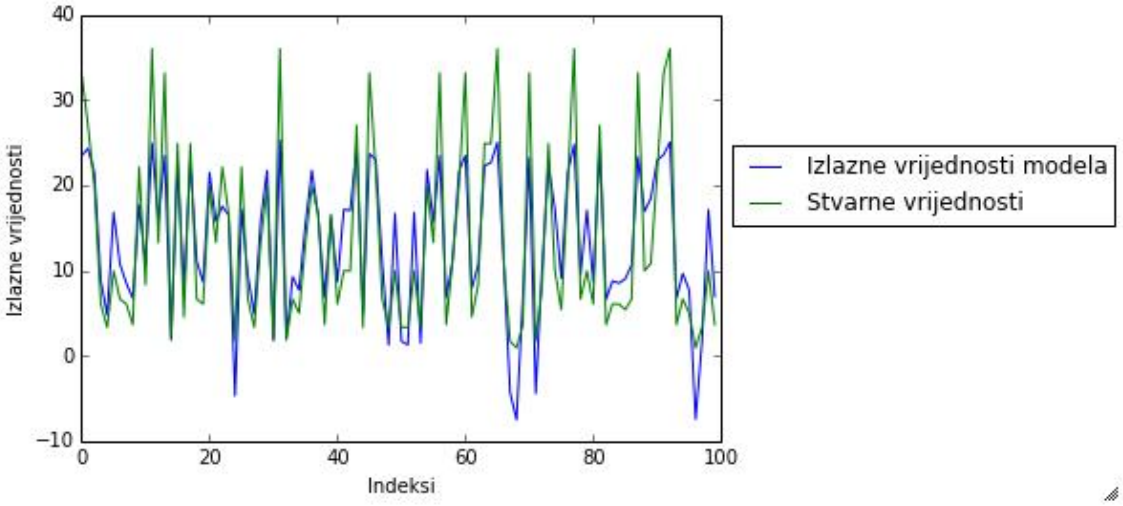
Generalizacijska pogreška = 76358.6741857

Srednja kvadratna pogreška na ispitnom skupu = 972.721964148

Relativna pogreška: 0.00200010361252

Izlazne vrijednosti ispitnog skupa primjera kreću se između 0.90775 i 3599.008333.

10. Vrijeme izvođenja*, space-shared raspored*



Slika 23. Linearna regresija, reg. faktor = 100

Empirijska pogreška = 5021.08537161

Generalizacijska pogreška = 1947.12841348

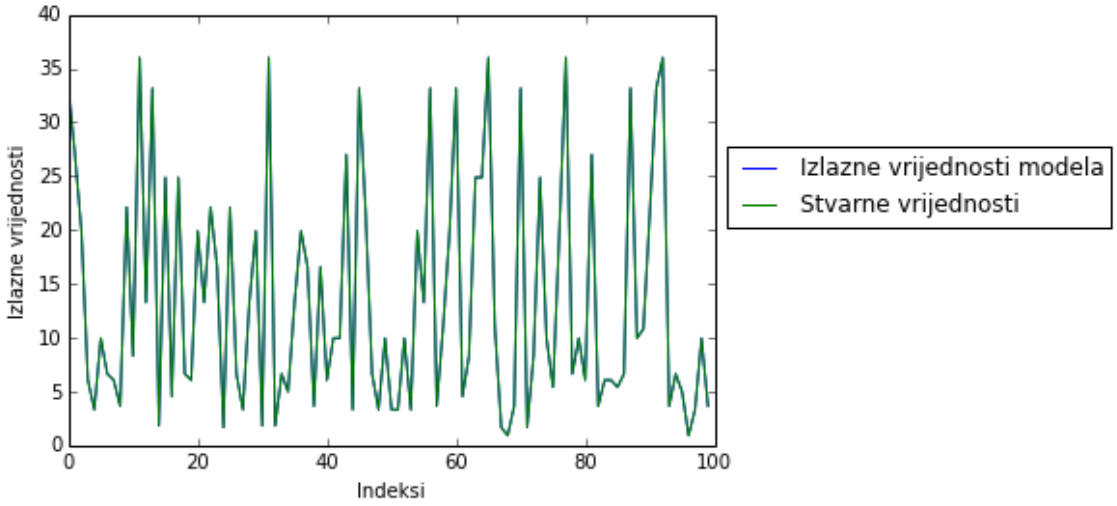
Srednja kvadratna pogreška na ispitnom skupu = 24.8041836113

Relativna pogreška: 0.222536688779

Atribut: težinski koeficijent

broj korisničkih zahtjeva: $-4.96918480*10^{-3}$; međudolazno vrijeme zahtjeva: $3.24195724*10^{-3}$; broj instrukcija *cloudlet*-a: $9.13739181*10^{-4}$; brzina procesora VM: $-4.67891140*10^{-3}$

Kod *space-shared* algoritma simulacija korisničkog ponašanja (broj zahtjeva i interval među njima) ima manji utjecaj na izlaz u odnosu na prethodno opisanu *time-shared* politiku. Kao što je opisano u poglavlju 3.1.1.1, ovaj algoritam radi na principu da svaki zadatak dobije potrebnu količinu resursa dok ostali, za koje trenutno nema prostora, čekaju na svoj red izvođenja. Takvim načinom rada svi navedeni parametri imaju sličan utjecaj na konačno vrijeme odziva.



Slika 24. Polinomijalna regresija, stupanj polinoma = 5, reg. faktor = 5

Empirijska pogreška = 0.000151464770896

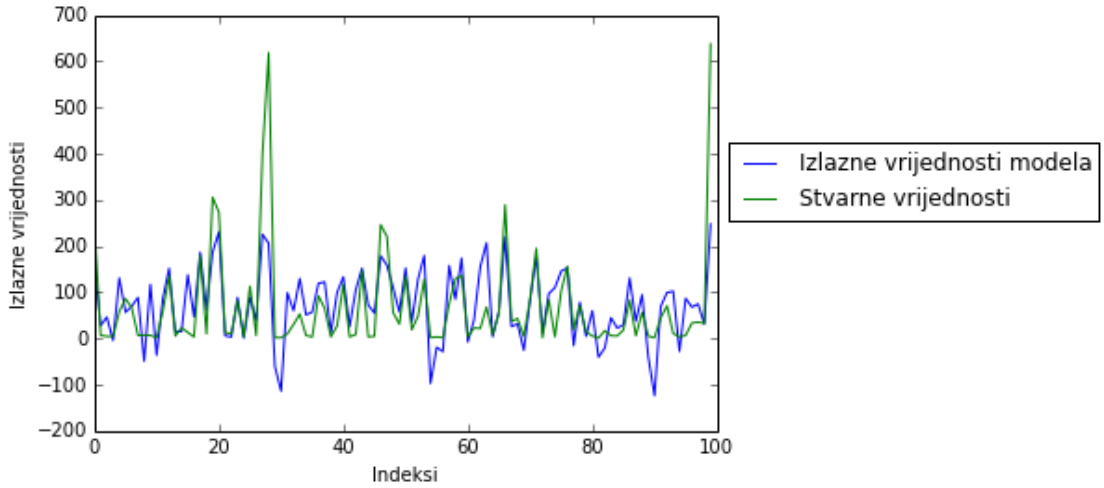
Generalizacijska pogreška = $5.17246780237*10^{-5}$

Srednja kvadratna pogreška na ispitnom skupu = $6.58913095844*10^{-7}$

Relativna pogreška: $5.91159704509*10^{-9}$

Izlazne vrijednosti ispitnog skupa primjera kreću se između 0.90775 i 35.991667.

11. Vrijeme izvođenja, *time-shared raspored* s automatskim skaliranjem



Slika 25. Linearna regresija, reg. faktor = 100

Empirijska pogreška = 243043641.695

Generalizacijska pogreška = 76421658.7164

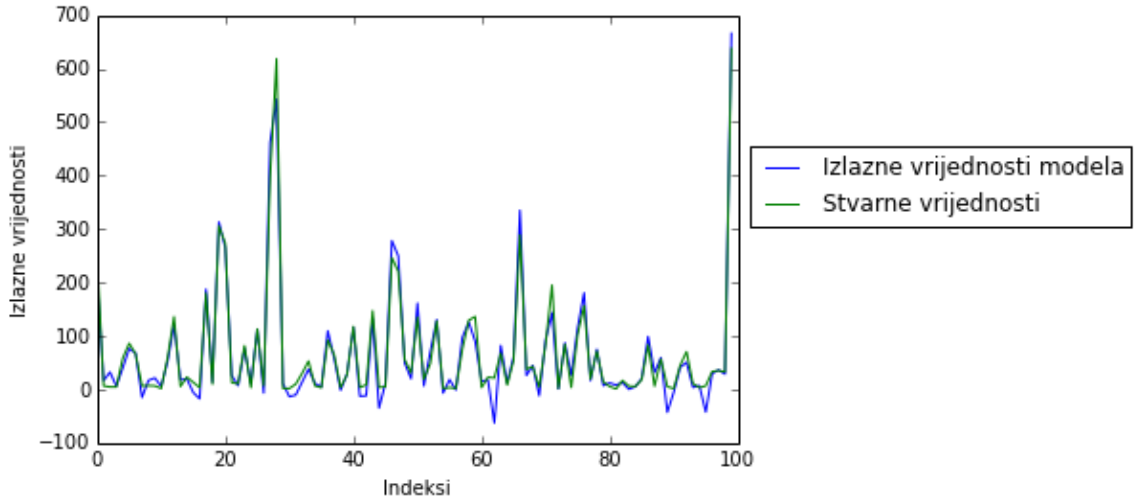
Srednja kvadratna pogreška na ispitnom skupu = 9781.9723157

Relativna pogreška: 0.604217977985

Atribut: težinski koeficijent

broj korisničkih zahtjeva: $4.52534675*10^{-1}$; međudolazno vrijeme zahtjeva: -3.70119367; RAM memorija *cloudlet*-a: $-3.84836461*10^{-1}$; utilizacija procesora kod *clodulet*-a: -7.63943557; broj instrukcija *cloudlet*-a: $7.07010776*10^{-3}$; RAM memorija VM: $1.28842690*10^{-2}$; brzina procesora VM: $-3.45450096*10^{-2}$

Kod *time-shared* rasporeda s automatskim skaliranjem, osim korisničkog ponašanja, velik utjecaj na izlaz čine parametri iskorištenja procesora i RAM memorija *cloudlet*-a. Oni direktno utječu na instanciranje novih virtualnih strojeva. Ukoliko se zadaci podijele na više strojeva, dolazi do rasterećenja sustava i vrijeme odziva se smanjuje, stoga su težinski koeficijenti uz dotične faktore negativnog predznaka.



Slika 26. Polinomijalna regresija, stupanj polinoma = 4, reg. faktor = 80

Empirijska pogreška = 30459534.1691

Generalizacijska pogreška = 9703507.26631

Srednja kvadratna pogreška na ispitnom skupu = 1242.04893009

Relativna pogreška: 0.0767195273997

Izlazne vrijednosti ispitnog skupa primjera kreću se između 0.90775 i 1799.508333.

Prema koeficijentu relativne pogreške *space-shared* daje najbolje rezultate strojnog učenja. Nakon njega slijedi *time-shared* raspored, a najlošiju predikcijsku moć ima, kao i kod slučajeva s troškom, *autoscaling* algoritam. Jedan od glavnih

razloga leži u tome da raspored s automatskim skaliranjem ima najveći broj značajki koje utječu na izlaznu vrijednost te se modeli teže prilagođavaju takvim podacima.

MSE je dosta veći u odnosu na modele vezane uz prosječni trošak, no uzrok toga je puno veći raspon izlaznih vrijednosti. Gledajući vrijednosti relativne pogreške između modela s troškom kao izlaznom vrijednošću i onih s vremenskim odzivom kao izlazom, situacija je nešto drukčija. *Space-shared* ima manju pogrešku za vremenski odziv kao izlaznu varijablu, a preostala dva algoritma imaju bolje rezultate kada je riječ o modelima vezanim uz trošak.

Zaključak

Pružatelji *SaaS* usluge često puta nemaju predodžbu kako će se njihova aplikacija ponašati u okruženju računalnog oblaka po pitanju potrošnje. Okolina *cloud* okruženja može biti prilično složena te velik broj parametara ima utjecaj na potrošnju i izvođenje smještene aplikacije. Zadaća ovog rada bila je primijeniti model strojnog učenja na simulirano ponašanje aplikacijske usluge smještene u okruženje računalnog oblaka. Na taj način može se dobiti jasniji uvid koji parametri aplikacije i *cloud* okruženja imaju najveći utjecaj na trošak izvođenja usluge. Osim financijskog aspekta, promatrano je i vrijeme izvođenja, odnosno kako se promjenom pojedinih značajki mijenja kvaliteta ponuđene usluge. Također se primjenom regresijskih modela želi ostvariti predviđanje navedenih dviju izlaznih varijabli na temelju skupa ulaznih atributa. Mogućnost predikcije svakako je od velikog značaja u konfiguraciji okoline računalnog oblaka u koju se razmatrana usluga smješta.

Modeli linearne i polinomijalne regresije primijenjeni su na podatke dobivene simuliranjem zadane aplikacijske usluge. Pomoću linearne regresije identificirani su parametri aplikacijske usluge i okruženja računalnog oblaka koji najviše utječu na njezin trošak što je temeljna informacija u ovom istraživanju. Pružatelji aplikacijskih usluga mogu iskoristiti taj podatak prilikom optimiranja troška izvođenja aplikacije nastojeći minimizirati utjecaj parametara koji mu najviše doprinose. Dodatno, kako bi se uz predikciju troška mogla razmatrati i kvaliteta usluge, regresija je primijenjena i nad vremenskim odzivom kao izlaznom varijablom modela.

Po pitanju troška, modeli linearne regresije pokazali su da najveći utjecaj imaju cijene resursa. Kod *on-demand-instance* naplatne metode, to se odnosi na cijenu virtualne jedinice, dok kod *pay-as-you-go* najveći utjecaj u formiranju troška izvođenja usluge imaju cijene: RAM memorije, CPU jedinice, memorije skladištenja i mrežne propusnosti.

Osim toga, najveći utjecaj na trošak ima korisničko ponašanje, odnosno broj korisničkih zahtjeva i vremenski interval između njihovih dolazaka. To su faktori na koje pružatelj *SaaS* usluge nema utjecaja, no oni uvelike formiraju cijenu troška.

U trećem redu, korištene simulacije su pokazale da važna uloga pripada atributu vezanom uz *cloudlet*, a radi se o iskoristivosti procesora virtualne jedinice na koju je usluga smještena. Naročito, ako se radi o algoritmu vremenske dodjele resursa koji implementira automatsko skaliranje, velika iskoristivost procesora korištenih radnih zadataka uvjetuje potrebu za instanciranjem, odnosno zakupom, novih virtualnih strojeva.

Što se tiče vremena odziva usluge kao izlazne varijable regresijkih modela, utjecaj parametara varijabilan je u ovisnosti o korištenom algoritmu raspodjele resursa. Korisničko ponašanje, odnosno broj zahtjeva za uslugom te vremenski interval između njihovih dolazaka, ima najveći utjecaj na izvođenje tražene usluge.

Kod algoritma prostorne dodjele resursa, utjecaj tih parametara je nešto manji, no svejedno ima značaj na izlaz. Uz korisničko ponašanje, za dotični algoritam jednak utjecaj imaju brzina procesora virtualnog stroja i broj instrukcija aplikacijske usluge.

Po pitanju algoritma s automatskim skaliranjem, situacija je slična kao u slučajevima kada je trošak postavljen za izlaznu varijablu. Važna uloga pripada parametrima koji određuju potrebu za novim virtualnim jedinicama. Radi se o RAM memoriji *cloudlet*-a i faktoru iskorištenja procesora. Ukoliko dođe do zakupa više virtualnih mašina, korisnički zahtjevi se podijele te je smanjeno opterećenje pojedinog stroja, a samim time se smanjuje vrijeme izvođenja.

Regresijski modeli u dobroj mjeri su odgovorili svojem zadatku i prilagodili se ulaznom skupu podataka. Predikcija budućeg troška i vremena odziva je moguća, uz pogreške koje moraju postojati, ali nisu prevelike. U tu svrhu bolje je koristiti polinomijalnu regresiju. Predviđanje ovih varijabli je od velikog značaja za pružatelja *SaaS* usluge koji, na taj način, može procijeniti ponašanje svoje aplikacije u različitim okolinama računalnog oblaka.

Mogućnosti nastavka istraživanja trebale bi voditi u smjeru optimizacije parametara s najvećim utjecajem na trošak. Jedan primjer pristupa ovom problemu bio bi odabir neke od heurističkih metoda optimizacija. Uspješnim algoritmom i dobrim rezultatima pružatelj *SaaS* usluge imao bi dobar pogled na specificiranje karakteristika aplikacijske usluge i okruženja računalnog oblaka za smještanje iste.

Literatura

[1] Armbruts, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M. Above the Clouds: A Berkley View of Cloud Computing. University of California at Berkley, USA, 2009.
[2] Mell, P., Grance, T. The NIST Definition of Cloud Computing. National Institute of Standards and Technology Special. Publication 800-145, September 2011.
[3] Erl, T., Puttini, R., Mahmood, Z. Cloud Computing: Concepts, Technology & Architecture. 1st ed. NJ, USA: Prentice Hall Press, Upper Saddle River, 2013.
[4] Sinarm Software, Što je Cloud Computing ili usluga u oblaku?, siječanj 2013., http://www.sinarm.net/sto-je-cloud-computing-ili-usluga-u-oblaku/, pregledano 31. svibnja 2015.
[5] Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose C.A.F., Buyya, R. Software - Practice and Experience: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Wiley Online Library. DOI 10.1002/spe.995, 41, str. 23-50
[6] Margaret Rouse, Infrastructure as a Service (IaaS), siječanj 2015., http://searchcloudcomputing.techtarget.com/definition/Infrastructure-as-a-Service-IaaS, pregledano 31. svibnja 2015.
[7] Apprenda, IaaS, PaaS, SaaS (Explained and Compared), 2015., Cloud Computing Overview, http://apprenda.com/library/paas/iaas-paas-saas-explained-compared/, pregledano 31. svibnja 2015.
[8] Margaret Rouse, Platform as a Service (IaaS), siječanj 2015., http://searchcloudcomputing.techtarget.com/definition/Platform-as-a-Service-PaaS, pregledano 1. lipnja 2015.
[9] Tomac R. Tehno-ekonomska analiza usluga zasnovanih na računarstvu u oblaku. Diplomski rad. Fakultet elektotehnike i računarstva, Zagreb, 2013.
[10] Margaret Rouse, Software as a Service (IaaS), kolovoz 2010., http://searchcloudcomputing.techtarget.com/definition/Software-as-a-Service, pregledano 2. lipnja 2015

[11] The Cloud Pricing Codex - 2013. 451 Resarch. 451Research, LLC and/or its Affiliates. 2013

[12] What is Machine Learning?, 2007-2011, Machine learning in The Netherlands, http://www.mlplatform.nl/what-is-machine-learning/, pregledano 10. lipnja 2015.

[13] Margaret Rouse, Definition of Machine Learning, siječanj 2011., http://whatis.techtarget.com/definition/machine-learning, pregledano 10. lipnja 2015.

[14] Bojana Dalbelo Bašić, Jan Šnajder. Uvod u strojno učenje. Fakulet elektrotehnike i računarstva, Sveučilište u Zagrebu. Ak. god. 2014./2015.

[15] Bojana Dalbelo Bašić, Jan Šnajder. Regresija. Fakulet elektrotehnike i računarstva, Sveučilište u Zagrebu. Ak. god. 2012./2013.

[16] Montgomery, D.C., Peck E.A., Vining, G.G. Introduction to Linear Analysis. Hoboken, New Jersey: John Wiley & Sons, Inc., 2012

[17] CloduSim: Framework For Modeling And Simulation Of Cloud Computing Infrastructures And Services,
The Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne, http://www.cloudbus.org/cloudsim/, pregledano 16. lipnja 2015.

[18] Garg, S.K., Buyya, R. NetworkCloudSim: Modelling Parallel Applications in Cloud Simulations. 2011 Fourth IEEE International Conference on Utility and Cloud Computing, The University of Melbourne, Australia, (2011), str. 105-113

[19] Himani, Sidhu, H.S. Comparative Analysis of Scheduling Algorithms of Cloudsim in Cloud Computing. International Journal of Computer Applications (0975 – 8887). Volume 97. No.16, July 2014. str. 29-33

[20] Installing the SciPy Stack, 2015., *SciPy.org*, http://www.scipy.org/install.html, pregledano 15. lipnja 2015.

[21] NumPy, 2013., *SciPy.org*, http://www.numpy.org/, pregledano 15. lipnja 2015.

[22] Google Cloud Platform Compute Engine,
https://cloud.google.com/compute/#pricing, pregledano 22. lipnja 2015.

[23] Amazon EC2 Pricing,
http://aws.amazon.com/ec2/pricing/, pregledano 22. lipnja 2015.

[24] Microsoft Azure Pricing,
http://azure.microsoft.com/en-us/pricing/calculator/?scenario=virtual-machines, pregledano 22. lipnja 2015.

[25] VMware vCloud Air On Demand Pricing Calculator, http://vcloud.vmware.com/uk/service-offering/pricing-calculator/on-demand, pregledano 22. lipnja 2015.

[26] Xiao, Z., Chen, Q., Luo, H. Automatic Scaling of Internet Applications for Cloud Computing Services. Computers, IEEE Transactions on Computer. DOI 10.1109/TC.2012.284, ISSN 0018-9340 (2014), str. 1111-1123

Sažetak

Primjena strojnog učenja u optimizaciji operativnog troška usluga u računalnom oblaku

Računarstvo u oblaku označava model koji omogućava mrežni zahtjev za dijeljenim računalnim resursima. Cilj ovog rada je primjenom strojnog učenja i simulacija temeljenih na zadanom modelu usluge dobiti jasniju sliku o utjecaju pojedinih parametara aplikacijske usluge i okruženja računalnog oblaka, u koji je smještena, na trošak izvođenja i ponašanje aplikacije. Uz to se primjenom regresijskih modela želi ostvariti mogućnost predviđanja ukupnog troška na temelju ulaznih značajki čime bi odluka o konfiguraciji okoline računalnog oblaka bila znatno olakšana. Unutar CloudSim simulatora, korištenog u projektu, implementirana su dva naplatna modela specifična za paradigmu računarstva u oblaku. Korištena su tri algoritma dodjele resursa virtualne jedinice: vremenska dodjela, prostorna dodjela i vremenska dodjela s automatskim skaliranjem. Simulacije aplikacijskih usluga smještenih u računalni oblak prate dvije izlazne varijable: trošak izvođenja i vrijeme odziva usluge. Navedene simulacije čine ulazne podatke za linearnu i polinomijalnu regresiju. Po pitanju troška, modeli linearne regresije pokazali su da najveći utjecaj imaju cijene resursa koje postavlja pružatelj usluge. Što se tiče vremenskog odziva, najvažniju ulogu ima korisničko ponašanje, odnosno broj korisničkih zahtjeva i vremenski intervali među njima.

Ključne riječi: računarstvo u oblaku, strojno učenje, linearna regresija, polinomijalna regresija, trošak aplikacijske usluge, vrijeme odziva, naplatni model

Summary

Using machine learning in optimization of cloud services cost

Cloud computing is a paradigm that enables network access to a shared pool of configurable computing resources. The goal of the project is to identify the impact of parameters, related to the application and the cloud environment, where it will be deployed, on the cost and behavior of cloud-based service using machine learning and simulations based on defined service models. Also, regression models may enable the ability to predict service execution cost based on input features. Such prediction would make the decision about the service cloud deployment environment much easier. Two pricing methods, specific to cloud computing paradigm, were implemented within CloudSim, the simulator used in this project. Three schedulers are used for allocation of virtual machines resources: time-shared, space-shared and time-shared with autoscaling. The research uses simulations of cloud-based application services with two output variables: service cost and response time. These simulations make inputs for linear and polynomial regression. Linear regression models showed that prices of resources, set by the service provider, have the major impact on service cost. On the other hand, user behavior, which consists of number of user requests and time intervals between them, has the most important role in determination value of response time variable.

Keywords: cloud computing, machine learning, linear regression, polynomial regression, application service cost, response time, pricing model