Računanje u oblaku

Tadić, Mia

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Mia Tadić

RAČUNANJE U OBLAKU

Diplomski rad

Voditelj rada:
prof. dr. sc. Robert Manger

Zagreb, rujan, 2020.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Ovaj diplomski rad posvećujem svojoj obitelji za beskrajnu podršku i razumijevanje tijekom studiranja. Posebna i najveća hvala mojem Domagoju na strpljenju i beskompromisnoj vjeri u mene. Hvala mojim prijateljima bez kojih bi sve bilo puno teže. Zahvaljujem mentoru prof. dr. sc. Robertu Mangeru na ukazanoj pomoći u stvaranju ovog diplomskog rada. Zahvaljujem tvrtci Syntio na usmjeravanju tijekom pisanja i na resursima koje su mi omogućili.

Sadržaj

Uvod

Računanje u oblaku je poseban distribuirani sustav usluga računanja koje se koriste *na daljinu*. Radi se o tehnologiji koja je još uvijek u svojim mladim danima te poslovni korisnici tek počinju shvaćati koje su sve mogućnosti oblaka.

U ovom diplomskom radu objasnit ćemo pojam računanja u oblaku: tehnologije, arhitekturu, sigurnost, resurse i ostale bitne aspekte.

Zatim ćemo uvesti konkretniji primjer proizvoda koji se može razviti u oblaku i koji jako dobro prikazuje dobre primjene i prednosti oblaka. Prikazuje više različitih resursa/tehnologija tipičnih i često korištenih u oblaku. Taj primjer je tzv. *podatkovna platforma u oblaku* koja opisuje kako se podaci mogu dovesti iz različitih izvora na neku platformu u oblaku, kako se na platformi u oblaku ti podaci zatim sređuju i spremaju za razne vrste korisnika, te konačno samu primjena tih podataka. Podatkovna platforma će biti opisana generički te se može implementirati u različitim platformama u oblaku.

Kao studijski primjer i praktični dio ovog diplomskog rada, opisat ćemo i implementirati računanje s visokim performansama (engl. *High Performance Computing*) na poznatoj Amazonovoj platformi u oblaku AWS. Ukratko, radi se o pokretanju zadatka distribuiranog na više računala paralelno. Ovo je idealan primjer efikasnosti distribuiranog računanja koji veliki zalet u razvoju dobiva s tehnologijom oblaka zbog brzog pružanja neograničenog broja namještenih resursa. Također prikazuje i glavnu prednost oblaka – skalabilnost. Postupak strojnog učenja za klasifikaciju slika distribuirat ćemo na klaster računala te uspoređivati performanse u ovisnosti o različitom broju računala na kojem se zadatak izvršava.

Poglavlje 1

Računanje u oblaku

> *"Računanje u oblaku je specijalizirani oblik distribuiranog računanja koji uvodi modele korištenja skalabilnih i mjerljivih resursa na daljinu."* ([6])

U ovom poglavlju precizno ćemo objasniti pojam računanja u oblaku (engl. *cloud computing*), zašto i kako je došlo do njegovog nastanka, te sve aspekte njegove realizacije. Najprije ćemo opisati motivaciju i začetke računanja u oblaku te navesti njegove karakteristike, prednosti i nedostatke. Opisat ćemo koje su se uloge formirale u oblaku te kako se računanje u oblaku isporučuje korisnicima. Zatim ćemo objasniti kakve opasnosti vrebaju na sigurnost korisnika i njihovih podataka. Nadalje objašnjavamo koje tehnologije, mehanizmi i arhitekturne komponente sudjeluju u realizaciji oblaka. Konačno, opisat ćemo kako se mjere troškovi korištenja oblaka te SLA kao jedini službeni pravni dokument.

1.1 Razvoj oblaka

Koncept oblaka počeo se razvijati u zadnjem desetljeću 20. stoljeća. Razni tadašnji postojeći alati i određene tehnologije počinju se kombinirati i dovode do razvoja oblaka kakav je danas. Opisat ćemo kako se koncept oblaka kretao od samih početaka, koja je uopće motivacija dovela do potrebe za razvojem oblaka te koje se to tehnologije smatraju začetcima i temeljem oblaka.

1.1.1 Kratka povijest nastanka

Temeljne koncepte koji čine osnovu suvremenog računanja u oblaku popularizirali su razni internetski alati sredinom 1990-ih, poput pretraživača (Yahoo!, Google), e-pošte (Hotmail, Gmail), otvorenih izdavačkih platformi (MySpace, Facebook, YouTube) i drugih društvenih medija (Twitter, LinkedIn).

Krajem 1990-ih, Salesforce.com je uveo ideju o uvođenju daljinskih usluga u poduzetnički svijet.

U 1990-tima je bio aktivan pojam *oblak* u kontekstu apstrakcijskog sloja u metodama isporuke poruka baziranima na paketima preko heterogenih (polu)javnih mreža.

2002. godine Amazon je pokrenuo platformu Amazon Web Services (AWS) – paket usluga usmjerenih na organizacije, koje pružaju pohranu i računalne resurse na daljinu. Upravo je AWS prva platforma za računanje u oblaku.

Tek 2006. godine se pojam *računanje u oblaku*, odnosno *cloud computing*, pojavljuje u komercijalnoj areni. Tada je Amazon pustio na korištenje svoje usluge Elastic Compute Cloud (EC2) koje su organizacijama omogućile "zakup" računalnog kapaciteta i procesorske moći za pokretanje njihovih poslovnih aplikacija.

1.1.2 Motivacija za računanjem u oblaku

Prije oblaka nastali su *on-premise* sustavi. Dok IT resurs u oblaku predstavlja daljinski dostupan resurs, IT resurs koji je smješten *u prostorijama organizacije* (i koji posebno ne predstavlja oblak) smatra se on-premise resursom. On-premise je pojam suprotan oblaku. Informatički resurs koji je on-premise, ne može biti u oblaku, i obrnuto. Primarni poslovni pokretači koji su ukazali na manjkavost on-premise sustava, odnosno na potrebu za računanjem u oblaku i doveli do njegovog formiranja su *planiranje kapaciteta*, *smanjenje troškova* i *organizacijska agilnost*.

Planiranje kapaciteta proces je utvrđivanja budućih zahtjeva u organizaciji za informatičkim resursima, proizvodima i uslugama. U tom kontekstu kapacitet predstavlja maksimalnu količinu posla koju informatički resurs može isporučiti u određenom vremenskom periodu. Nesklad između kapaciteta informatičkog resursa i zahtjevima prema njemu može dovesti do toga da sustav postaje ili neefikasan (prekomjerna opskrba resursom) ili nije u mogućnosti ispuniti potrebe korisnika (nedovoljna opskrba resursom). Planiranje kapaciteta usmjereno je na minimiziranje te razlike.
Postoje različite strategije planiranja kapaciteta:

- *strategija ulaganja unaprijed*: dodavanje kapaciteta resursu zbog očekivanja potražnje
- *strategija zaostajanja*: dodavanje kapaciteta kada resurs dosegne svoj puni kapacitet
- *strategija usklađivanja*: dodavanje kapaciteta resursu malo po malo, kako se povećava potražnja

Planiranje kapaciteta može biti izazovno jer zahtijeva procjenu varijacija opterećenja. Postoji stalna potreba za uravnoteženjem dvaju faktora: osiguravanje tečnog korištenja

resursa (čak i u ekstremnom slučaju prekomjernog korištenja) i izbjegavanje nerazumnih troškova za infrastrukturu.

Smanjenje troškova je zahtjevan zadatak jer sklad između informatičkih troškova i poslovnih performansi može biti teško održavati. Rastu informatičkih okruženja često odgovara procjena njihovih maksimalnih zahtjeva za uporabom resursa. Posljedica toga je da poslovanja imaju sve veća ulaganja. Veliki dio ovog potrebnog ulaganja usmjeren je na širenje infrastrukture jer će poslovanje uvijek biti ograničeno procesnom snagom osnovne infrastrukture.

Potrebno je uzeti u obzir trošak nabave nove infrastrukture, trošak stalnog vlasništva nad tom infrastrukturom te operativne troškove. Operativni troškovi predstavljaju značajan udio informatičkog proračuna, često prelazeći već visoke investicijske troškove. Uobičajeni oblici operativnih troškova povezanih s infrastrukturom uključuju sljedeće:

- tehničko osoblje potrebno za održavanje okoliša u radu
- nadogradnje i popravke koje zahtijevaju i dodatne cikluse testiranja i implementacije
- komunalne račune za napajanje energijom i za hlađenje tehnologije
- mjere sigurnosti i kontrole pristupa koje je potrebno održavati i provoditi radi zaštite resursa infrastrukture
- administrativno osoblje koje će morati voditi evidenciju o licenci

Organizacijska agilnost mjerilo je sposobnosti prilagodbe i evolucije organizacije kako bi se uspješno suočila s promjenama uzrokovanim unutarnjim i vanjskim čimbenicima.

IT poslovanje često mora reagirati na poslovne promjene povećavanjem planiranog kapaciteta svojih informatičkih resursa. Na primjer, infrastruktura može biti podložna ograničenjima koja će spriječiti organizaciju da reagira na varijacije u korištenju. Što je još gore, organizacija se možda uopće ne odluči na rješavanje problema nakon pregleda svog infrastrukturnog proračuna jer si to jednostavno ne može priuštiti. Ovaj oblik nemogućnosti reagiranja može spriječiti organizaciju da odgovori na zahtjeve tržišta, da se oduprije konkurentskim pritiscima i da se drži svojih strateških poslovnih ciljeva.

1.1.3 Začetne tehnologije oblaka

Primarne tehnološke inovacije koje su utjecale i inspirirale ključne karakteristike i aspekte računanja u oblaku uključuju *klasteriranje*, *rešetku za računanje* i tradicionalne oblike *virtualizacije*. Sva tri pojma smatraju se temeljima računanja u oblaku.

Klaster je skupina inače neovisnih IT resursa koji su međusobno povezani i djeluju kao jedinstveni sustav. Stopa kvara sustava smanjuje se s rastom raspoloživosti i pouzdanosti njegovih komponenti, iz razloga što klasteri osiguravaju redundantne sustave (radi eliminacije individualnih točaka kvara) i prijelaz na rezervni sustav u slučaju kvara glavnog sustava (engl. *failover*). Opći preduvjet klasteriranja hardvera je da njegove komponente imaju razmjerno identičan hardverski i operacijski sustav koji omogućuju slične razine performansi kada jednu neuspješnu komponentu treba zamijeniti drugom. Dijelovi uređaja koji tvore klaster drže se u sinkronizaciji putem namjenskih, brzih komunikacijskih veza. Osnovni koncept ugrađene redundancije i *failover*a je temelj platformi za računanje u oblaku.

Rešetka za računanje (engl. *computing grid*) pruža platformu u kojoj su resursi organizirani u jedan ili više logičkih skupova. Ovi se skupovi zajednički koordiniraju kako bi se osigurala distribuirana mreža visokih performansi, koja se ponekad naziva i "super virtualno računalo". Rešetka za računanje razlikuje se od klastera u tome što su sustavi utemeljeni na rešetci za računanje mnogo labavije povezani i distribuirani. Kao rezultat toga, sustavi utemeljeni na rešetci za računanje mogu uključivati resurse koji su raznoliki i geografski različito raspoređeni, što općenito nije moguće sa sustavima utemeljenima na klasterima. Rešetka za računanje je aktualno područje istraživanja u području računarstva i računanja od početka 1990-ih. Tehnološki napredak postignut rešetkama za računanje utjecao je na različite aspekte i mehanizme računanja u oblaku, posebno na značajke kao što su mrežni pristup, udruživanje resursa te skalabilnost i otpornost. Rešetka za računanje temelji se na sloju međusoftvera koji je raspoređen po resursima. Ti IT resursi sudjeluju u mrežnom skupu koji implementira niz raspodjela radnog opterećenja i koordinacijskih funkcija. Ovaj srednji sloj može sadržavati logiku balansiranja opterećenja (engl. *load balancing*), kontrole prelaska na rezervni sustav (engl. *failover*) i upravljanje autonomnom konfiguracijom, a svaka ta komponenta je nadahnula i nekoliko sofisticiranijih tehnologija računanja u oblaku. Iz tog razloga neki klasificiraju računanje u oblaku kao potomak ranijih inicijativa za rešetkama za računanje.

Virtualizacija je stvaranje virtualnih instanci informatičkih resursa. Sloj softvera za virtualizaciju omogućava fizičkom informatičkom resursu da izvede više svojih virtualnih slika, tako da ga može koristiti veći broj korisnika, a ne samo jedan. Prije pojave tehnologija za omogućavanje virtualizacije, softver je bio ovisan o statičkim hardverskim okruženjima. Proces virtualizacije nadilazi tu ovisnost softvera o hardveru jer hardverski zahtjevi mogu biti simulirani emulacijskim softverom koji radi u virtualiziranom okruženju. Sad kad smo uveli pojam virtualizacije, definiramo *informatički (ili IT) resurs* u oblaku kao fizički ili virtualni informatički artefakt koji može biti temeljen na virtualizacijskom softveru, poput virtualnog servera, ili temeljen na hardveru, poput fizičkog servera. Tehnologije virtualizacije su začetnici nekoliko karakteristika oblaka i mehanizama računanja u oblaku. Kako se razvijalo računanje u oblaku, pojavila se i generacija suvremenih tehnologija virtualizacije koja je prevladala ograničenja performansi, pouzdanosti i skalabilnosti

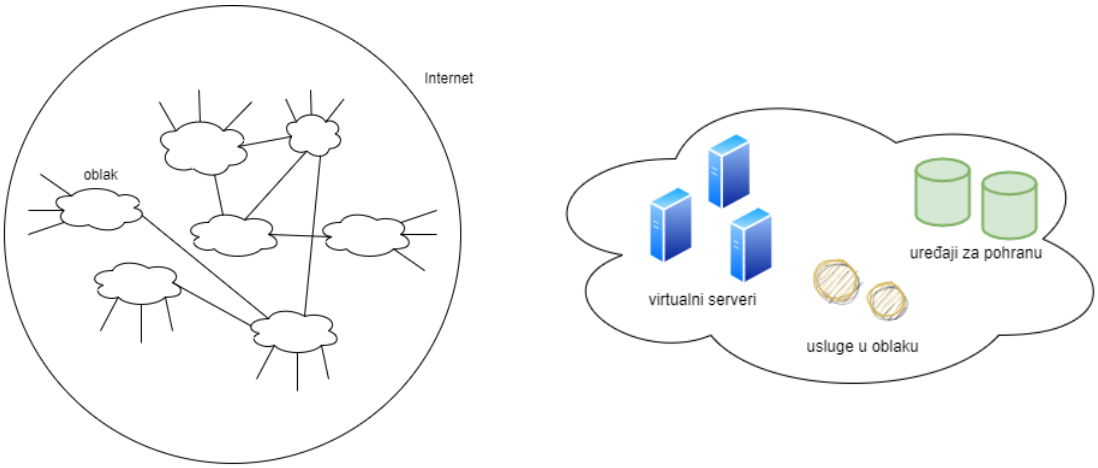
tradicionalnih platformi za virtualizaciju. Suvremena virtualizacija je temelj računanja u oblaku i bit će detaljnije opisana u odjeljku 1.5.3.

1.2 Osobitosti oblaka

Često se pojam oblaka poistovjećuje s Internetom, no to su dva odvojena pojma iako imaju sličan idejni koncept – daljinski pristup. Objasnit ćemo što čini oblak posebnim, njegove karakteristike, prednosti i nedostatke.

1.2.1 Razlika između oblaka i Interneta

Oblak je nastao kao metafora za Internet koji je, u osnovi, mrȅža mrêžā koja pruža daljinski pristup skupu decentraliziranih IT resursa. Prije nego što je računanje u oblaku postalo zasebni formalizirani segment IT industrije, simbol oblaka obično se koristio za predstavljanje Interneta. Važno je razlikovati oblak od Interneta. Kao specifično okruženje koje se koristi za daljinsko pružanje IT resursa, oblak je ograničeno područje. Mnogo je pojedinih oblaka koji su dostupni putem Interneta. Dok Internet pruža otvoreni pristup mnogim informatičkim izvorima utemeljenima na webu, oblak je obično u privatnom vlasništvu i nudi pristup informatičkim izvorima čije se korištenje mjeri.



Slika 1.1: Apstrakcija strukture Interneta u kontekstu oblaka (lijevo) i primjer jednog oblaka s resursima koje sadrži (desno).

1.2.2 Ciljevi i prednosti

Ciljevi koji se oblakom žele postići su postali glavne prednosti oblaka. Glavnom prednošću oblaka smatra se *skalabilnost*, a tu su još i *racionalizacija troškova, dostupnost* i *pouzdanost*.

Racionalizacija troškova. Ekonomski razlog za ulaganje u resurse u oblaku je smanjenje ili potpuno uklanjanje ulaganja "unaprijed". Obično su IT resursi skupi i organizacije moraju unaprijed uračunati njihove troškove u svoje financijske planove, na primjer kupovinu hardvera, održavanja softvera i troškove licence. S druge strane, model plaćanja koji se koristi u oblacima je *pay-for-use* ili *pay-as-you-go* model i on definira plaćanje samo onoga što se zaista koristi. Ovo omogućuje poslovanjima da krenu polako s resursima i s vremenom povećaju zahtjeve za IT resursima prema potrebi. Nadalje, smanjenje kapitalnih troškova i ulaganja "unaprijed" omogućava preusmjeravanje kapitala na temeljna poslovna ulaganja.
Uobičajene koristi za korisnike oblaka uključuju:

- Mogućnost potražnje kratkoročnih resursa u trenutku, uz naplatu po korištenju (na primjer, procesori po satu) i mogućnost odbacivanja tih informatičkih resursa kada više nisu potrebni.
- Percepcija neograničenih informatičkih resursa koji su dostupni na zahtjev, čime se smanjuje potreba za pripremom za rezerviranje.
- Mogućnost dodavanja ili uklanjanja IT resursa na "finoj" razini, kao što je izmjena raspoloživog prostora na disku gigabajt po gigabajt.
- Apstrakcija infrastrukture – aplikacije se ne ograničavaju na uređaje ili lokacije, i mogu se lako premjestiti ako je potrebno.

Skalabilnost. Skaliranje, iz perspektive IT resursa, predstavlja sposobnost IT resursa da podnese povećane ili smanjene zahtjeve za korištenjem. Sljedeće su vrste skaliranja:

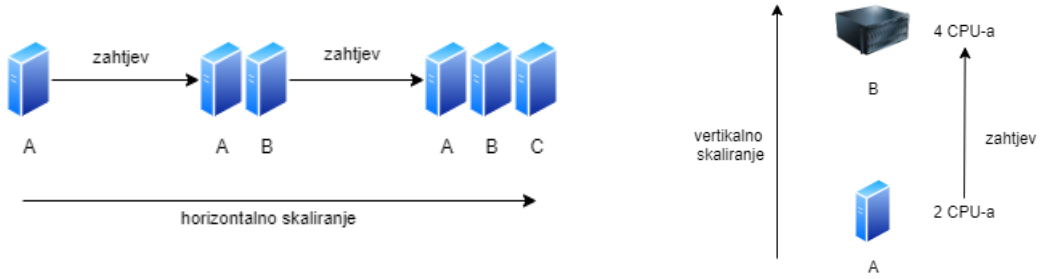
- *horizontalno skaliranje*: skaliranje prema van i skaliranje prema unutra (engl. *scaling out*, *scaling in*)
- *vertikalno skaliranje*: skaliranje prema gore i skaliranje prema dolje (engl. *scaling up*, *scaling down*)

Horizontalno skaliranje je dodjeljivanje ili oslobađanje IT resursa *iste vrste*. Dodjeljivanje istovrsnih resursa naziva se *skaliranjem prema van*, a oslobađanje istovrsnih resursa naziva se *skaliranjem prema unutra*. Horizontalno skaliranje uobičajen je oblik skaliranja u oblaku.

Vertikalno skaliranje je kad se postojeći informatički resurs zamijeni drugim informatičkim resursom s većim ili manjim kapacitetom, dakle resursom *različite vrste*. Konkretno, zamjena IT resursa nekim drugim koji ima veći kapacitet naziva se *skaliranjem prema gore*, a zamjena IT resursa nekim drugim koji ima manji kapacitet smatra se *skaliranjem prema dolje*. Vertikalno skaliranje manje je uobičajeno u oblaku zbog prekida rada tijekom zamjene.

Horizontalno skaliranje	***Vertikalno skaliranje***
jeftinije (uobičajene hardverske komponente)	skuplje (specijalizirani resursi)
IT resursi odmah dostupni	IT resursi uglavnom odmah dostupni
replikacija resursa i automatizirano skaliranje	potrebna dodatna podešavanja
potrebni dodatni IT resursi	nisu potrebni dodatni IT resursi
nije ograničeno mogućnostima hardvera	ograničeno mogućnostima hardvera

Tablica 1.1: Razlike između horizontalnog i vertikalnog skaliranja.



Slika 1.2: Horizontalno skaliranje (lijevo) i vertikalno skaliranje (desno).

Sposobnost oblaka da poveća obujam IT resursa omogućuje organizacijama da odgovore na nepredvidive varijacije u korištenju, a da nisu ograničene unaprijed definiranim pragovima. Suprotno tome, sposobnost oblaka da smanji obujam IT resursa je značajka koja izravno dopridonosi smanjenju troškova poslovanja.

Dostupnost i pouzdanost. Informatički resurs s visokom razinom *dostupnosti* dostupan je kroz dulje vremensko razdoblje (na primjer, 22 od 24 sata dnevno). Pružatelji usluga u oblaku općenito nude otporne IT resurse za koje su u stanju jamčiti visoku razinu dostupnosti. Informatički resurs s visokom razinom *pouzdanosti* može efikasno izbjeći i oporaviti se od nepoželjnih situacija. Modularna arhitektura oblaka osigurava iscrpnu zaštitu te tako povećava pouzdanost.

1.2.3 Rizici i izazovi

Sigurnosna ranjivost. Prijenos poslovnih podataka u oblak znači da se odgovornost za sigurnost podataka dijeli s pružateljem usluga u oblaku. Daljinsko korištenje IT resursa zahtijeva proširenje granica povjerenja od strane korisnika oblaka. Obično je teško uspostaviti sigurnosnu arhitekturu koja osigurava granice povjerenja koje korisniku odgovaraju, a da je bez posljedica ranjivosti – osim ako korisnici oblaka i pružatelji usluga u oblaku podržavaju iste ili kompatibilne sigurnosne okvire, što je malo vjerojatno s javnim oblacima. Nadalje, mogu se preklapati granice povjerenja od različitih korisnika oblaka međusobno zbog činjenice da se informatički resursi u oblaku uobičajeno dijele. Preklapanje granica povjerenja i povećana izloženost podataka mogu zlonamjernim korisnicima oblaka (ljudskim i automatiziranim) pružiti veće mogućnosti napada na informatičke resurse te ukrasti ili oštetiti poslovne podatke.

Smanjena kontrola operativnog upravljanja. Korisnicima oblaka obično je dodijeljena razina upravljanja koja je niža od one nad informatičkim resursima koji se nalaze unutar organizacije (on-premise). Ovo može uvesti rizike povezane s načinom na koji pružatelj usluga u oblaku upravlja svojim oblakom, kao i vanjskim vezama koje su potrebne za komunikaciju između oblaka i korisnika oblaka. Na primjer, nepouzdan pružatelj usluga u oblaku možda neće održati obećane dogovore i to može ugroziti imovinu korisnika u oblaku. Još jedan primjer gdje smanjena operativna kontrola postaje problem, jest kod veće geografske udaljenosti između korisnika oblaka i pružatelja usluga u oblaku, koje mogu zahtijevati dodatne mrežne mehanizme koji mogu dovesti do spore mreže sa smanjenom propusnošću.

Ograničena prenosivost između različitih pružatelja usluga u oblaku. Zbog nedostatka utvrđenih industrijskih standarda u industriji računanja u oblaku, javni su oblaci različitih standarda. Za korisnike oblaka koji imaju prilagođena rješenja ovisna o okruženju može biti teško prijeći s jednog pružatelja usluga u oblaku na drugog.

Multiregionalna neusklađenost i pravna pitanja. Pružatelji usluga u oblaku obično uspostavljaju svoje podatkovne centre na pristupačnim i povoljnim geografskim lokacijama. Korisnici oblaka obično nisu svjesni fizičkih lokacija svojih IT resursa i podataka koji se nalaze u oblaku. Za neke organizacije to može predstavljati ozbiljne pravne probleme koji se odnose na industrijske ili vladine propise koji određuju politiku privatnosti podataka i pravila pohrane. Na primjer, neki britanski zakoni zahtijevaju da se osobni podaci građana Velike Britanije čuvaju isključivo u Ujedinjenom Kraljevstvu. Drugi potencijalni pravni problem odnosi se na dostupnost i otkrivanje podataka. Zemlje imaju zakone koji zahtijevaju da se neke vrste podataka moraju otkriti određenim državnim agencijama na njihov zahtjev. Na primjer, podacima europskog korisnika oblaka koji se nalaze u SAD-u američke vladine agencije mogu lako pristupiti (zbog američkog Zakona o patriotima), za razliku od podataka smještenih u većini zemalja Europske unije. Međutim, većina regulatornih okvira prepoznaje da su organizacije korisnika oblaka u konačnici odgovorne za si-

gurnost, integritet i pohranu vlastitih podataka, čak i kad se nalaze kod vanjskog pružatelja usluga u oblaku.

Iako većini ljudi kao prvi nedostatak oblaka na pamet padne sigurnosna ranjivost, sigurnost je zapravo značajno veća nego na on-premise sustavima. Razlog tomu je što se sve akcije u oblaku bilježe te tako pružatelji usluga u oblaku puno lakše mogu pratiti sve radnje u oblaku te ih ili na vrijeme spriječiti, ili iscrpno naučiti iz njih. Sigurnost u oblaku je konstantno poboljšavana i danas se ne može čuti za situaciju u kojoj je ta sigurnost probijena, u kojoj se dogodio uspješan hakerski napad. Nijedan od navedenih rizika zapravo nije toliko realan na ozbiljnim platformama u oblaku, no jedan rizik postoji, a to su pravna pitanja različitih geografskih lokacija. S obzirom na sve aktivnija pitanja o zaštiti privatnih podataka građana, te s obzirom na međusobne sukobe nekih zemalja, lokacije pružatelja usluga u oblaku mogu postati izrazit problem. Na primjer, Kina ne dopušta čuvanje podataka svojih stanovnika u drugim državama, pogotovo u SAD-u gdje se i nalaze danas najveći pružatelji usluga u oblaku, te stoga organizacije iz Kine neće sklopiti ugovor ni sa jednim od njih.

1.2.4 Karakteristike oblaka

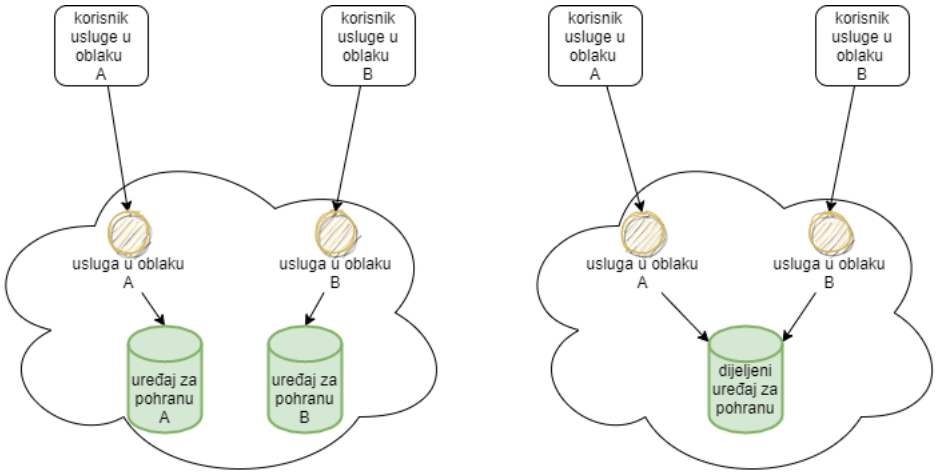
IT okruženje zahtijeva specifičan skup od šest karakteristika kako bi se na učinkovit način omogućilo daljinsko pružanje skalabilnih i mjerljivih IT resursa. Upravo tih šest karakteristika specificiraju oblak:

- *Korištenje na zahtjev.*
  Korisnik oblaka može samoinicijativno pristupiti IT resursima u oblaku, što mu pruža slobodu samoosiguravanja IT resursa. Nakon konfiguriranja "samoosiguranih" IT resursa, njihova upotreba se može automatizirati, ne zahtijevajući daljnje ljudsko sudjelovanje niti korisnika oblaka, niti pružatelja usluga u oblaku.

- *Sveprisutnost.*
  Sveprisutnost predstavlja mogućnost da usluga u oblaku bude široko dostupna. Uspostavljanje sveprisutnosti inače može zahtijevati podršku za čitav niz uređaja, transportnih protokola, sučelja i sigurnosnih tehnologija, što u oblaku nije briga korisnika oblaka, već pružatelja usluga u oblaku.

- *Multitenancy (i udruživanje resursa).*
  Karakteristika softverskog programa koja omogućuje da se jedna instanca programa vrti na različitim instancama servera, pri čemu je svaki izoliran od drugog, naziva se *multitenancy*. Pružatelj usluga u oblaku svoje IT resurse objedinjuje kako bi opskrbio više korisnika usluga u oblaku koristeći multitenancy arhitekturne modele koji

se često oslanjaju na korištenje tehnologija virtualizacije. Korištenjem multitenancy tehnologije, IT resursi mogu se dinamički dodjeljivati prema zahtjevu korisnika usluge u oblaku. Udruživanje resursa omogućuje pružatelju usluga da udružene resurse ponudi na korištenje većem broju korisnika.



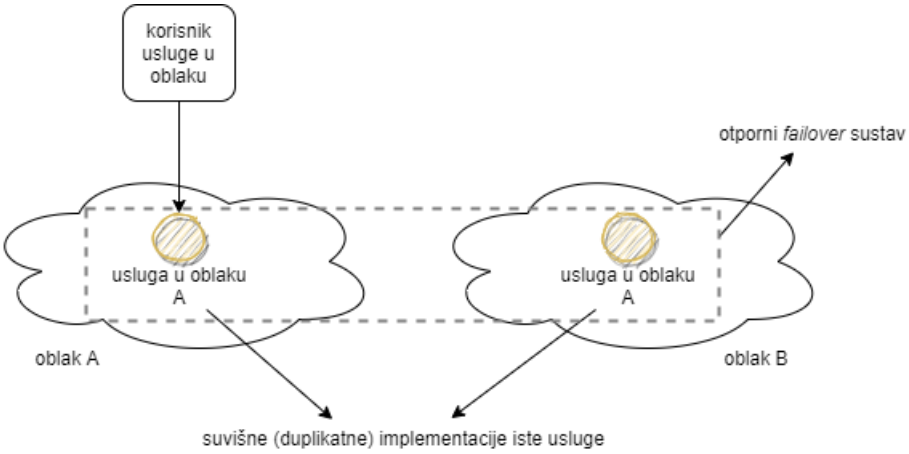
Slika 1.3: Usporedba oblaka bez i sa korištenjem *multitenant* tehnologije.

- *Elastičnost.*
  Elastičnost je automatizirana sposobnost oblaka da transparentno skalira IT resurse, kao odgovor na uvjete izvršavanja ili na ono što je unaprijed utvrđeno od strane korisnika ili pružatelja usluga u oblaku. Elastičnost se često smatra osnovnim razlogom uvođenja računanja u oblaku, prvenstveno zbog činjenice da je usko povezano s racionalizacijom troškova.

- *Mjerljivo korištenje.*
  Na temelju mogućnosti mjerenja korisnikove upotrebe IT resursa, pružatelj usluga u oblaku može naplatiti korisniku oblaka samo one IT resurse koje stvarno koristi i to za točan vremenski period tijekom kojeg mu je odobren pristup IT resursima. Osim za naplaćivanje, praćenje korištenja je efikasno i za raznovrsne analize i izvještaje.

- *Otpornost.*
  Otporno računanje oblik je *failover*a koji distribuira kopije IT resursa na više fizičkih lokacija. IT resursi mogu se unaprijed konfigurirati tako da, ako jedan postane nedostupan, obrada se automatski predaje drugom (kopiji). Karakteristika otpornosti može se odnositi na IT resurse unutar istog oblaka (ali na različitim fizičkim lokacijama) ili na IT resurse raspoređene na više oblaka. Korisnici oblaka mogu povećati

pouzdanost i dostupnost svojih aplikacija povećanjem otpornosti IT resursa u oblaku.



Slika 1.4: Otporni sustav sa redundantnim implementacijama iste usluge.

1.3 Uloge u oblaku

Iako je oblak udaljeno dostupno okruženje, ne mogu se svi IT resursi koji se nalaze u oblaku učiniti dostupnima za daljinski pristup. Na primjer, baza podataka ili fizički server smješten u oblaku može biti dostupan samo drugim IT resursima koji su unutar istog oblaka. Taj se problem može riješiti softverskim programom s objavljenim API[1]-jem kako bi se udaljenim klijentima omogućio pristup.

Usluga u oblaku je svaki informatički resurs (ili skup informatičkih resursa) u oblaku koji je dostupan na daljinu. Obično im se pristupa preko korisničkog sučelja u web pregledniku ili aplikacijskim programskim sučeljem (API-jem) iz terminala.

Uvjeti korištenja usluge u oblaku obično se definiraju u *sporazumu o razini usluge* (SLA) (engl. *service-level agreement*) koji je ljudski čitljiv dio ugovora o usluzi između pružatelja usluga u oblaku i korisnika oblaka te koji opisuje kvalitetu, značajke, ponašanje i ograničenja usluge. SLA pruža detalje o različitim mjerljivim karakteristikama usluge, kao što su sigurnosne karakteristike i druge specifične QoS (engl. *quality-of-service*) značajke, uključujući dostupnost, pouzdanost i performanse. Budući da se sama implementacija usluge skriva od korisnika oblaka, SLA postaje izuzetno bitna specifikacija.

[1] API (Application Programming Interface) je programsko sučelje za programiranje aplikacije ili pristup podacima. API jedne aplikacije pruža niz funkcija za pristup ili rad s tom aplikacijom. To je vrsta softvera koja omogućuje dvjema aplikacijama da komuniciraju.

Korisnik usluge u oblaku privremena je uloga koju uređaj ili softver preuzima kada pristupa usluzi u oblaku. Korisnik usluge u oblaku može biti mobilni telefon, laptop, softverski program, ...

Organizacije i ljudi mogu preuzeti različite vrste unaprijed definiranih uloga, ovisno o načinu na koji komuniciraju s oblakom i njegovim informatičkim resursima. Uobičajene uloge uključuju *pružatelja usluga u oblaku, korisnika oblaka, vlasnika usluge u oblaku* i *administratora resursa u oblaku*.

Pružatelj usluga u oblaku. Organizacija koja pruža informatičke resurse u oblaku je pružatelj usluga u oblaku (engl. *cloud provider*). Kada preuzima ulogu pružatelja usluga u oblaku, organizacija je odgovorna za pružanje usluga u oblaku korisnicima oblaka, u skladu s dogovorenim SLA jamstvima. Pružatelj usluga u oblaku je, osim za konfiguracijske, također zadužen za sve potrebne upravljačke i administrativne dužnosti kako bi se osigurao kontinuirani rad cjelokupne infrastrukture oblaka. Pružatelji usluga u oblaku obično posjeduju IT resurse koje korisnici oblaka uzimaju u zakup. Međutim, neki pružatelji usluga u oblaku također "preprodaju" IT resurse zakupljene od drugih pružatelja usluga u oblaku. Kao tri ponajveća primjera pružatelja usluga u oblaku navodimo Google, Microsoft i Amazon.

Korisnik oblaka. Korisnik oblaka (engl. *cloud consumer*) je organizacija ili čovjek koji ima formalni ugovor ili dogovor s pružateljem usluga u oblaku o korištenju IT resursa koje mu je pružatelj ustupio. Konkretno, korisnik oblaka koristi korisnika usluge u oblaku (npr. mobilni uređaj, laptop, softverski program) da bi pristupio usluzi u oblaku. Podrazumijeva se da se organizacije ili ljudi koji na daljinu koriste informatičke resurse u oblaku smatraju korisnicima oblaka. Samo radi jasnijeg prikaza interakcije između korisnika i IT resursa zapravo razlikujemo pojam "korisnik usluge u oblaku" od pojma "korisnik oblaka". Prvi se obično koristi za označavanje softverskih programa ili aplikacija koje se programskim sučeljem povezuju s API-jem usluge u oblaku. Potonji izraz je širi jer se može koristiti za označavanje organizacije/pojedinca koji pristupa korisničkom sučelju ili softverskom programu koji preuzima ulogu korisnika oblaka tijekom interakcije s oblakom, informatičkim resursom u oblaku ili pružateljem usluga u oblaku.

Vlasnik usluge u oblaku. Osoba ili organizacija koja zakonski posjeduje uslugu u oblaku naziva se vlasnik usluge u oblaku. Vlasnik usluge u oblaku može biti korisnik oblaka (korisnik oblaka može biti vlasnik usluge u oblaku kada smjesti vlastitu uslugu u oblak) ili pružatelj usluga u oblaku (pružatelj usluga u oblaku postaje vlasnik usluge u oblaku kada smjesti vlastitu uslugu u vlastiti oblak, a obično za upotrebu drugim korisnicima oblaka). Razlog zašto vlasnik usluge u oblaku nije nazvan vlasnikom resursa u oblaku je taj što se uloga vlasnika usluge u oblaku primjenjuje samo na usluge u oblaku (koje su vanjski dostupni IT resursi koji se nalaze u oblaku).

Administrator resursa u oblaku. To je osoba ili organizacija odgovorna za administra-

ciju informatičkih resursa u oblaku (uključujući usluge u oblaku). Administrator resursa u oblaku može biti korisnik oblaka ili pružatelj usluge u oblaku u kojem se nalazi resurs (ili usluga). Alternativno, on može pripadati trećoj organizaciji i sklopiti ugovor s vlasnikom usluge u oblaku za administraciju usluge u oblaku. Razlog zbog kojeg se administrator resursa u oblaku ne naziva "administrator usluge u oblaku" je taj što ova uloga može biti odgovorna za administraciju IT resursa u oblaku koji ne postoji kao usluga u oblaku.

Postoje i još neke, manje uobičajene, uloge u oblaku, kao što su *revizor oblaka, posrednik u dogovaranju* i *nositelj oblaka*.

Revizor oblaka (engl. *cloud auditor*) je treća strana (često akreditirana) koja provodi neovisne procjene okruženja oblaka. Tipične odgovornosti su procjena sigurnosnih kontrola, utjecaja na privatnost i performansi. Glavna svrha uloge revizora oblaka je pružiti nepristranu procjenu okruženja oblaka kako bi se ojačao odnos povjerenja između korisnika oblaka i pružatelja usluga u oblaku.

Posrednik u dogovaranju (engl. *cloud broker*) je stranka koja preuzima odgovornost za upravljanje i pregovaranje o upotrebi usluga u oblaku između korisnika oblaka i pružatelja usluga u oblaku.

Nositelj oblaka (engl. *cloud carrier*) je stranka odgovorna za pružanje povezivosti na razini mreže između korisnika oblaka i pružatelja usluga u oblaku. Tu ulogu često preuzimaju mrežni i telekomunikacijski operateri.

1.4 Modeli oblaka

Oblake možemo razvrstati s obzirom na isporuku usluga u njemu i s obzirom na vrstu vlasništva nad oblakom, tj. njegovog okruženja.

1.4.1 Modeli isporuke u oblaku

Pružatelj usluga u oblaku mora definirati način na koji će upakirati svoj proizvod, odnosno svoje usluge i resurse u oblaku, te na koji način će ga isporučiti korisnicima. To upravo opisuje model isporuke u oblaku. Model isporuke u oblaku predstavlja specifičnu, unaprijed upakiranu kombinaciju IT resursa koju nudi pružatelj usluga u oblaku. Opisat ćemo tri uobičajena modela isporuke u oblaku: *infrastruktura kao usluga, platforma kao usluga* i *softver kao usluga*.

Najprije objasnimo pojam "kao usluga". Kada se nešto nudi "kao usluga", onda je to korisnicima na raspolaganju kao usluga, a ne kao nešto što sami instaliraju i upravljaju kroz vlastiti hardver. Ovim se uslugama pristupa putem Interneta. Za primjer uzmimo Microsoft Word, poznati alat za stvaranje i uređivanje dokumenata. Postoje dvije opcije kako ga koristiti. Može ga se kupiti kao proizvod (jednokratna kupovina softvera, odnosno trajne

licence) i instalirati na računalo. Taj se softver pokreće s računala i pristupa mu se samo s računala na kojem je instaliran. Druga opcija je koristiti Word kao uslugu. Umjesto da se instalira na jedno računalo, kupci mogu koristiti SaaS verziju (model isporuke koji je opisan kasnije) koja se nalazi na Microsoftovim internim serverima, tako da Wordu pristupe preko Microsoftove web stranice. Umjesto da izravno plaćaju softver, kupci plaćaju pretplatu za pristup – Word tada postaje manje proizvod koji ljudi kupuju, a više usluga koju plaćaju da bi ga koristili. Prednosti "kao usluge" su niži troškovi, fleksibilnost (pristup s više uređaja), stalna ažurnost, bolje mogućnosti (vlasnici proizvoda potiču ih "kao usluge" te stoga pružaju neke prednosti koje bi privukle kupce), manja zahtjevnost za infrastrukturu (korisnik nije ograničen kvalitetom vlastitog računala).

Objasnimo i pojam infrastrukture. Infrastruktura je digitalni kostur, konstrukcija koja podržava računalni sustav, sastoji se od važnih pozadinskih resursa kao što su serveri i pohrana, mreže, sigurnost i podatkovni centri.

Opišimo sada modele isporuke usluga i resursa koji se koriste u oblaku.

Infrastruktura kao usluga (IaaS) (engl. *Infrastructure-as-a-Service*). IaaS model isporuke predstavlja samostalno IT okruženje koje se sastoji od IT resursa koji grade infrastrukturu te kojima se može pristupiti i upravljati putem sučelja i alata u oblaku. Ovo okruženje uključuje hardver, mrežu, mogućnost povezivanja, operacijske sustave i druge "sirove" IT resurse. Opća svrha IaaS okruženja je omogućiti korisnicima oblaka visoku razinu kontrole i odgovornosti nad konfiguracijom i upotrebom resursa. IT resursi koje pruža IaaS uglavnom nisu unaprijed konfigurirani, što administrativnu odgovornost prebacuje izravno na korisnika oblaka te stoga ovaj model biraju korisnici oblaka koji zahtijevaju visoku razinu kontrole nad oblakom. IT resursi dostupni putem IaaS okruženja uglavnom se nude kao svježe inicijalizirane virtualne instance. Središnji i primarni IT resurs unutar tipičnog IaaS okruženja je virtualni server. Virtualni serveri se iznajmljuju tako da se odrede zahtjevi na hardver servera, poput kapaciteta procesora, radne memorije i prostora za pohranu.

Platforma kao usluga (PaaS) (engl. *Platform-as-a-Service*). PaaS model isporuke predstavlja unaprijed definirano okruženje „spremno za upotrebu“, koje se obično sastoji od već uspostavljenih, konfiguriranih i pripremljenih IT resursa. Radeći na gotovoj platformi, korisnik oblaka izbjegava administrativni teret postavljanja i održavanja infrastrukturnih IT resursa, a što se zahtijeva u IaaS modelu. Sukladno tome, korisniku oblaka je dodijeljena niža razina nadzora nad IT resursima.

Softver kao usluga (SaaS) (engl. *Software-as-a-Service*). Softverski program objavljen svima na korištenje kao usluga u oblaku i dostupan kao "proizvod" predstavlja tipičnu značajku SaaS modela. Korisniku oblaka uglavnom je dodijeljena vrlo ograničena administrativna kontrola.

Dakle, kada govorimo o modelima isporuke govorimo o isporuci usluga i resursa. Često se pogrešno shvaća modele isporuke kao modele isporuke cijelog oblaka. Pružatelji usluga

u oblaku nikad neće korisnicima isporučiti svoj cijeli oblak, već samo resurse i usluge u njemu. Kako bismo dodatno precizirali modele isporuke, uzmimo za primjer Microsoft Azure[2] platformu u oblaku. Azure koristi sva tri modela IaaS, PaaS i SaaS u isporuci svojih usluga. Na primjer, nudi *Virtual Machines* uslugu kao IaaS (virtualne mašine), *Azure App Service* uslugu kao PaaS (potpuno upravljana usluga za izgradnju i ugošćivanje (engl. *hosting*) web aplikacija) te *Outlook* uslugu kao Saas (servis za pristup e-pošti). ([41])

Slika 1.5: Razlike modela isporuke u oblaku.

S vremenom i potrebama korisnika su se pojavile i mnoge specijalizirane varijacije ova tri osnovna modela isporuke u oblaku, a svaka se sastoji od drugačije kombinacije IT resursa. Neki primjeri su: pohrana kao usluga (engl. *Storage-as-a-Service*), baza podataka kao usluga (engl. *Database-as-a-Service*), sigurnost kao usluga (engl. *Security-as-a-Service*), komunikacija kao usluga (engl. *Communication-as-a-Service*), integracija kao usluga (engl. *Integration-as-a-Service*), testiranje kao usluga (engl. *Testing-as-a-Service*), obrada kao usluga (engl. *Process-as-a-Service*).

Također, moguće su i različite kombinacije osnovnih modela isporuke u oblaku, ovisno o tome kako korisnici oblaka i pružatelji usluga u oblaku odluče iskoristiti prirodnu hijerarhiju uspostavljenu ovim osnovnim modelima isporuke u oblaku.

IaaS + PaaS. Recimo da je pružatelj usluga u oblaku koji nudi PaaS okruženje odlučio iznajmiti IaaS okruženje od drugog pružatelja usluga u oblaku. Na motivaciju za takav aranžman može utjecati više razloga, na primjer ekonomija ili premašen kapaciteta prvog pružatelja ili možda određeni korisnik oblaka nameće zakonski zahtjev da se podaci

[2]Microsoft Azure, koji se obično naziva Azure, usluga je računanja u oblaku koju je stvorio Microsoft za izgradnju, testiranje, postavljanje i upravljanje aplikacijama i uslugama putem podatkovnih centara kojima upravlja Microsoft.

fizički pohranjuju u određenoj regiji (za razliku od mjesta u kojem je smješten oblak prvog pružatelja usluga u oblaku).

IaaS + PaaS + SaaS. Na primjer, dodavanjem dodatnog sloja prethodno opisanoj slojevitoj arhitekturi, takvo spremno okruženje korisnik oblaka može koristiti za razvoj i implementaciju vlastitih SaaS usluga u oblaku koje onda može učiniti dostupnima kao komercijalni proizvod.

1.4.2 Modeli okruženja oblaka

Model okruženja oblaka se prvenstveno razlikuje po vlasništvu, veličini i pristupu. Postoje četiri uobičajena modela okruženja oblaka.

Javni oblak je u vlasništvu treće strane i on obično nudi javno dostupne (ali naplative) komercijalizirane usluge i IT resurse u oblaku korisnicima oblaka. Pružatelj usluga u oblaku odgovoran je za stvaranje i kontinuirano održavanje javnog oblaka i njegovih IT resursa.

Privatni oblak je u vlasništvu pojedine organizacije i nalazi se u prostorijama organizacije. Iako je privatni oblak fizički smješten u prostorijama organizacije, IT resursi koje privatni oblak sadrži i dalje se smatraju ”u oblaku”, a ne ”on-premise”, i to sve dok korisnici tim resursima pristupaju kao da se radi o ”daljinskim” resursima.

Oblak zajednice obično je ograničen na određenu zajednicu (grupu) korisnika oblaka koja obično i međusobno dijeli vlasništvo.

Hibridni oblak je kombinacija dva ili više drugih modela okruženja oblaka. Na primjer, korisnik oblaka može rasporediti usluge u oblaku koje obrađuju osjetljive podatke u privatni oblak, a druge, manje osjetljive usluge u javni oblak.

Relevantna su i sljedeća dva modela okruženja oblaka.

Virtualni privatni oblak (VPC). Ovaj model rezultira samostalnim oblačnim okruženjem kojim upravlja javni pružatelj usluga u oblaku te je u njegovom vlasništvu, i koji je dostupan korisniku oblaka.

Međupovezani oblaci. Ovaj se model temelji na arhitekturi koja se sastoji od dva ili više međusobno povezanih oblaka.

1.5 Tehnologije za omogućavanje oblaka

U ovom poglavlju opisujemo skup primarnih tehnologija koje zajedno omogućuju ključne karakteristike povezane sa suvremenim računanjem u oblaku.

1.5.1 Internet

Korisnici oblaka i pružatelji usluga u oblaku obično koriste Internet za komunikaciju koja se temelji na decentraliziranom modelu pružanja usluga i upravljanja, te ju ne kontrolira nijedan centralizirani subjekt. Glavne komponente arhitekture Interneta su (žično i bežično) slanje paketa i međusobno povezivanje na temelju rutera. Prijenos podataka u jedinici vremena (engl. *network bandwidth*) i kašnjenje (engl. *latency*) karakteristike su koje utječu na QoS oblaka, a na koje snažno utječe zagušenje mreže.

1.5.2 Podatkovni centri

Podatkovni centar (engl. *data center*) je specijalizirana informatička infrastruktura koja sadrži centralizirane informatičke resurse, poput servera (poslužitelja), baza podataka i softverskih sustava. Hardver podatkovnih centara obično se sastoji od standardiziranih servera povećane računalne snage i kapaciteta za pohranu, dok tehnologije sustava za pohranu uključuju diskove i virtualizaciju prostora.

1.5.3 Virtualizacija

Virtualizacija je proces pretvaranja fizičkog IT resursa u virtualni IT resurs. Većina IT resursa može se virtualizirati, kao na primjer:

- *serveri*: fizički server može se apstrahirati u virtualni server
- *pohrana*: fizički uređaj za pohranu može se apstrahirati u virtualni uređaj za pohranu ili virtualni disk
- *mreža*: fizički ruteri i sklopke mogu se apstrahirati u logičke mreže, poput VLAN-ova
- *struja*: fizički UPS i jedinice za distribuciju električne energije mogu se apstrahirati u ono što se obično naziva virtualnim UPS-om

Usmjerit ćemo se na stvaranje i implementaciju virtualnih servera tehnologijom virtualizacije servera.

Prvi korak u stvaranju novog virtualnog servera putem softvera za virtualizaciju jest dodjela fizičkih IT resursa, nakon čega slijedi instalacija operacijskog sustava. Virtualni serveri koriste vlastite gostujuće operacijske sustave koji su neovisni o operacijskom sustavu u kojem su stvoreni. Ni gostujući operacijski sustav ni aplikacijski softver koji se izvode na virtualnom serveru nisu svjesni procesa virtualizacije, što znači da su ti virtualizirani IT resursi instalirani i izvršavaju se kao da se izvode na zasebnim fizičkim serverima. Ova

uniformnost izvršavanja koja omogućava rad programa na virtualnim sustavima ekvivalentan radu na fizičkim sustavima vitalno je obilježje virtualizacije. Gostujući operacijski sustavi obično zahtijevaju neometano korištenje softverskih proizvoda i aplikacija koje nije potrebno prilagođavati ili konfigurirati da bi se izvodile u virtualiziranom okruženju. Softver za virtualizaciju radi na fizičkom serveru zvanom *host*, čiji je osnovni hardver postao dostupan virtualnom serveru pomoću softvera za virtualizaciju. Funkcija softvera za virtualizaciju obuhvaća sistemske usluge, posebno one povezane s upravljanjem virtualnim strojevima i koje se obično ne nalaze u standardnim operacijskim sustavima. Zbog toga se ovaj softver ponekad naziva upraviteljem ili monitorom virtualnog stroja, ali najčešće *hypervisor*om.

Virtualizacija pruža *neovisnost hardvera*, *konsolidaciju servera* i *repliciranje resursa* te dodatno podržava (već opisane) karakteristike *objedinjavanje resursa* i *elastičnu skalabilnost*.

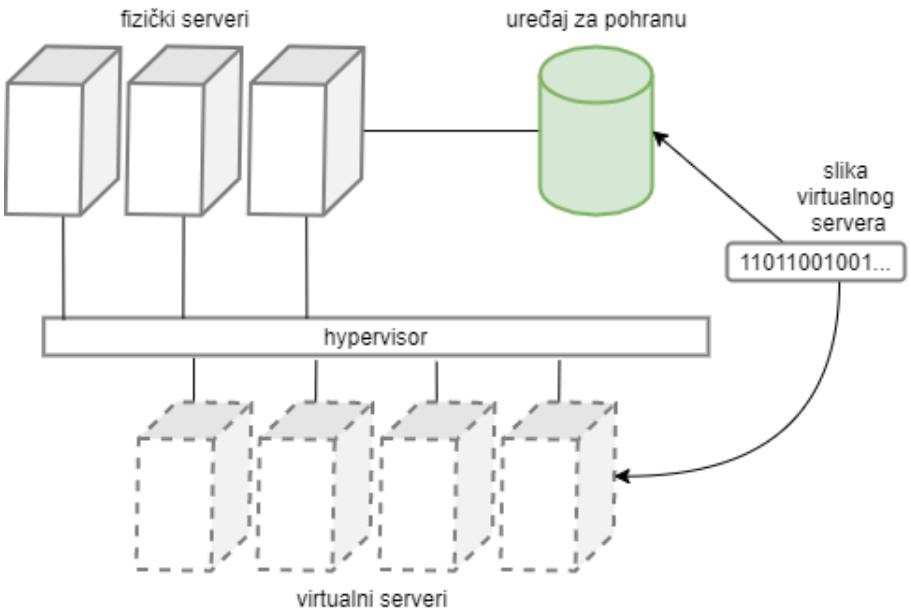
Hardverska neovisnost. Virtualizacija je proces pretvorbe koji prevodi jedinstveni IT hardver u standardizirane softverske kopije. Kroz neovisnost hardvera virtualni se serveri mogu lako premjestiti na drugi *host* virtualizacije, automatski rješavajući velike probleme s nekompatibilnošću hardvera i softvera. Kao rezultat toga, kloniranje i manipuliranje virtualnim IT resursima mnogo je lakše od dupliciranja fizičkog hardvera.

Konsolidacija servera. Tehnologija virtualizacije omogućuje različitim virtualnim serverima dijeljenje jednog fizičkog servera. Taj se postupak naziva konsolidacija servera i obično se koristi za veće iskorištavanje hardvera, uravnoteženje opterećenja i optimizaciju dostupnih IT resursa. Rezultirajuća fleksibilnost je takva da različiti virtualni serveri mogu pokrenuti različite gostujuće operacijske sustave na istom računalu. Ova temeljna sposobnost izravno podržava uobičajene značajke oblaka, poput korištenja na zahtjev, udruživanja resursa, elastičnosti, skalabilnosti i otpornosti.

Replikacija resursa. Virtualni serveri stvoreni su kao slike virtualnog diska koje sadrže kopije binarnih datoteka sadržaja tvrdog diska. Te su slike virtualnog diska dostupne operacijskom sustavu *host*a, što znači da se jednostavnim operacijama nad datotekama, poput kopiranja, premještanja i lijepljenja, može replicirati, migrirati i sigurnosno kopirati virtualni server. Ova jednostavnost manipulacije i repliciranja jedna je od najvažnijih značajki tehnologije virtualizacije jer omogućuje stvaranje standardiziranih slika virtualnog stroja za trenutnu implementaciju, brzinu migracije i stvaranja novih instanci virtualnog stroja te vraćanje unatrag na zdravo stanje sustava ukoliko dođe do kvara.

Virtualni serveri ostvaruju se putem virtualizacije temeljene ili *na operacijskom sustavu* ili *na hardveru*.

Virtualizacija na bazi operacijskog sustava je instalacija softvera za virtualizaciju u prethodno postojeći operacijski sustav koji se naziva *host* operacijski sustav. Budući da



Slika 1.6: *Hypervisor* replicira nekoliko primjeraka virtualnog servera, koristeći pohranjenu sliku virtualnog servera.

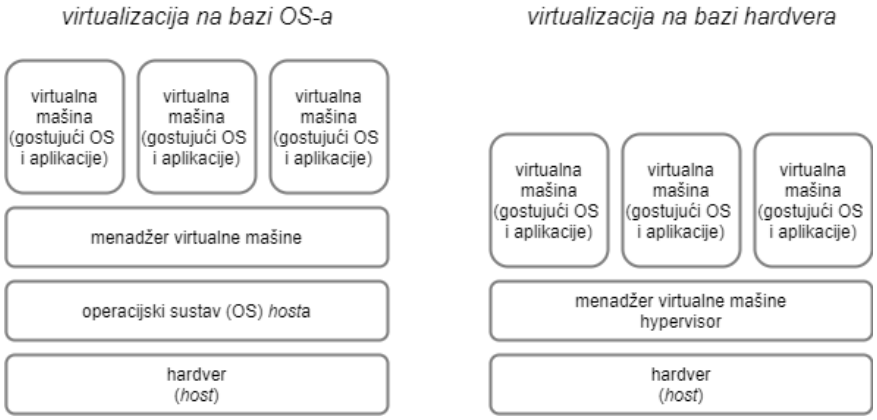
host operacijski sustav može hardverskim uređajima pružiti potrebnu podršku, virtualizacija na bazi operacijskog sustava može riješiti probleme nekompatibilnosti hardvera. Neovisnost hardvera koja je omogućena virtualizacijom omogućuje fleksibilnije korištenje hardverskih IT resursa.

Mane ovog pristupa su dodatni troškovi: *host* operacijski sustav troši procesore, memoriju i druge hardverske informatičke resurse, operacije vezane za hardver gostujućih operacijskih sustava moraju prijeći nekoliko slojeva do hardvera, što smanjuje ukupne performanse, te licence koje nisu potrebne samo za *host* operacijski sustav, već i za svaki njegov gostujući operacijski sustav.

Virtualizacija na bazi hardvera predstavlja instalaciju softvera za virtualizaciju izravno na fizički hardver glavnog računala kako bi se zaobišao glavni operacijski sustav. Dopuštanje virtualnim serverima da komuniciraju s hardverom bez traženja posredničkih radnji od glavnog operacijskog sustava obično čini ovaj tip virtualizacije efikasnijim. Softver za virtualizaciju obično se naziva *hypervisor* za ovu vrstu obrade. *Hypervisor* ima jednostavno korisničko sučelje koje zahtijeva zanemarivu količinu prostora za pohranu. Postoji kao tanki sloj softvera koji upravlja funkcijama koje upravljaju hardverom radi uspostavljanja sloja upravljanja virtualizacijom.

Mana ovog pristupa je obavezna kompatibilnost s hardverom. Sloj virtualizacije dizaj-

niran je za izravno komuniciranje s hardverom *hosta*, što znači da svi pridruženi upravljački programi uređaja moraju biti kompatibilni s *hypervisor*om.



Slika 1.7: Razlike u logičkim slojevima između virtualizacije na bazi operacijskog sustava i virtualizacije na bazi hardvera.

1.5.4 Web tehnologija

Zbog ovisnosti računanja u oblaku o Internetu, univerzalnosti web preglednika i jednostavnosti razvoja web usluga, web tehnologija se uglavnom koristi kao medij za implementaciju upravljačkog sučelja za usluge u oblaku.

1.5.5 *Multitenant* tehnologija

Dizajn *multitenant* aplikacije (aplikacije koju koristi više korisnika) kreiran je kako bi više korisnika moglo istovremeno pristupiti istom logičkom sloju aplikacije. Svaki korisnik ima vlastiti prikaz aplikacije koju koristi, administrira i prilagođava, a da pritom nije svjestan ostalih korisnika koji koriste istu aplikaciju. *Multitenant* aplikacije osiguravaju da korisnici nemaju pristup podacima i konfiguracijskim informacijama koje nisu njihove.

Multitenant tehnologija se ponekad pogrešno poistovjećuje s virtualizacijom. Razlika je u tome *čega* je veći broj unutar fizičkog servera koji djeluje kao *host* (domaćin):

- *virtualizacija*: jedan fizički server može ugostiti *više virtualnih kopija*, svaka se kopija može pružiti različitim korisnicima, može se samostalno konfigurirati i može sadržavati vlastite operacijske sustave i aplikacije

- *multitenancy*: fizički ili virtualni server koji ugošćava multitenant aplikaciju dizajniran je tako da dopušta upotrebu *više različitih korisnika*, svaki korisnik osjeća se kao da isključivo on koristi aplikaciju

1.5.6 Kontejneri

Kontejnerizacija je tehnologija virtualizacije na razini operacijskog sustava koja se koristi za pokretanje aplikacija i usluga u oblaku bez potrebe za kreiranjem virtualnog servera za svaku aplikaciju i uslugu. Umjesto toga, aplikacije su raspoređene u kontejnere. Korištenje kontejnera omogućuje pokretanje više izoliranih usluga u oblaku na jednom fizičkom ili virtualnom serveru koje pristupaju istoj jezgri operacijskog sustava. Kada se usluga u oblaku izvršava unutar kontejnera, ona misli da se izvodi na stvarnom računalu. Usluga u oblaku koja radi na fizičkom ili virtualnom operacijskom sustavu servera može vidjeti sve pružene resurse, poput povezanih uređaja, priključaka, datoteka, mapa, mrežnih udjela, CPU-ova, kao i fizičku memoriju koja se može adresirati. Međutim, usluga u oblaku koja radi unutar kontejnera može vidjeti samo sadržaj kontejnera i pripadne uređaje.

Treba uočiti razliku između virtualizacije i korištenja kontejnera. *Hypervisor* omogućuje više virtualnih servera da se pokreću na jednom fizičkom *host*u. Virtualni serveri vide emulirani hardver kojeg im je *hypervisor* predstavio kao pravi hardver, a svaki virtualni server ima svoj operacijski sustav koji se mora instalirati te se njime treba upravljati i održavati ga kao da se radi o fizičkom serveru. Suprotno tome, kontejner je apstrakcija na aplikacijskom ili servisnom sloju koji upakira kod i pripadne zavisne pakete. Na istom stroju može se rasporediti više kontejnera, te oni međusobno dijele istu jezgru operacijskog sustava. Svaki kontejner pokreće se kao izolirani proces. Kontejneri ne zahtijevaju gostujući operacijski sustav koji je potreban za virtualne servere i može se pokrenuti izravno na operacijskom sustavu fizičkog servera. Kontejneri također troše manje prostora za pohranu od virtualnih servera.

Prednosti kontejnera su:

- *prenosivost*: administratori resursa u oblaku mogu premjestiti kontejnere u bilo koje okruženje koje dijeli isti matični operacijski sustav i motor kontejnera (opisan malo kasnije), i to bez potrebe za promjenom aplikacije ili softvera (što bi obično zahtijevalo promjene izvornog koda)
- *učinkovito korištenje resursa*: postiže se značajnim smanjenjem korištenja CPU-a, memorije i pohrane u usporedbi s virtualnim serverima (što povlači jeftinije troškove)
- *praćenje verzija softverskog koda i njegovih ovisnosti*

Slika 1.8: Razlike u logičkim slojevima između virtualizacije i kontejnera.

Obično se jedna usluga u oblaku implementira u jedan kontejner, ali moguće je i više usluga u oblaku svrstati u jedan kontejner. Također, više kontejnera se može raporediti u jednu logičku kontrukciju zvanu *pod*. *Pod* se obično koristi kad postoje različite usluge u oblaku koje su dio iste aplikacije ili prostora s imenima i koje se trebaju izvoditi pod jednom IP adresom. *Pod* uspostavlja okolinu osiguravajući izolaciju usluga u oblaku tako da si usluge međusobno ne utječu na rad.

Da bismo shvatili kako kontejneri rade, potrebno je identificirati najosnovnije arhitektonske elemente koji čine okruženje kontejnera i koji mu omogućuju da obavlja svoje osnovne funkcije.

Motor kontejnera. Ključna komponenta, specijalizirani softver koji je implementiran u operacijski sustav kako bi apstrahirao potrebne resurse i omogućio definiciju i upotrebu kontejnera. Svaki motor kontejnera nudi skup alata za upravljanje i naredbi/API-ja za stvaranje, izmjenu, zakazivanje, pokretanje, zaustavljanje ili brisanje kontejnera.

Datoteka za izgradnju kontejnera. To je deskriptor (kreiran od strane korisnika ili usluge) koji predstavlja zahtjeve aplikacije ili usluge koja se izvodi unutar kontejnera, kao i konfiguracijske parametre za stvaranje kontejnera.

Slika kontejnera. Na temelju definiranog opisa kontejner prilagođava sliku operacijskog sustava i potrebne naredbe za uslugu ili aplikaciju. Ova prilagođena slika obično je nepromjenjiva slika koja je samo za čitanje, što omogućava da aplikacija ili usluga u kontejneru funkcionira i izvršava zadatke, ali sprječava da se naprave bilo kakve promjene.

Kontejner. Kontejner je izvršna instanca unaprijed definirane ili prilagođene slike kontejnera. Svaki kontejner može imati jednu ili više aplikacija/usluga. Aplikacije/usluge u kontejneru obično su tijesno ovisne o kontejneru, odnosno pokreću se i zaustavljaju s kontejnerom.

Adresa za umrežavanje. Svaki kontejner ima svoju mrežnu adresu (poput IP adrese) koja se koristi za komunikaciju s drugim kontejnerima i vanjskim komponentama.

Uređaj za pohranu. Slično mrežnoj adresi, kontejner se može povezati s jednim ili više uređaja za pohranu koji su kontejnerima dostupni putem mreže.

1.6 Sigurnost u oblaku

Povjerljivost, integritet, autentičnost i *dostupnost* karakteristike su koje se koriste za mjerenje sigurnosti.

Povjerljivost je svojstvo onoga koje je učinjeno dostupnim samo ovlaštenim stranama. U okruženju oblaka povjerljivost se prvenstveno odnosi na ograničavanje pristupa podacima u tranzitu i pohrani.

Integritet je svojstvo onoga kojeg neovlaštena strana nije izmijenila. Važno pitanje koje se tiče integriteta podataka u oblaku jest može li se korisniku oblaka zajamčiti da se podaci koje prenosi u oblak podudaraju s podacima primljenima u oblak.

Autentičnost je svojstvo onoga koje je nespornog podrijetla ili autorstva, tj. koje je pruženo od strane *ovlaštenog* izvora.

Dostupnost je svojstvo upotrebljivosti tijekom određenog vremenskog razdoblja. U tipičnim oblačnim okruženjima dostupnost usluga u oblaku može biti odgovornost koju dijele pružatelj usluga u oblaku i nositelj oblaka.

Nekoliko je vrsta prijetnji za sigurnost u oblaku, i nazivamo ih *prijetećim agentima*. To su entiteti koji predstavljaju prijetnju jer su sposobni za napad. Mogu napadati iznutra ili izvana, mogu biti ljudi ili softverski programi.

Anonimni napadač je agent koji nije od povjerenja i koji obično pokušava napasti izvan oblaka.

Zlonamjerni agent presreće mrežnu komunikaciju u pokušaju zlonamjerne upotrebe ili povećanja volumena podataka.

Pouzdani napadač postoji kao ovlašteni korisnik usluga u oblaku s legitimnim vjerodajnicama koje koristi za iskorištavanje pristupa informatičkim resursima u oblaku.

Zlonamjerni član pokušava zloupotrijebiti privilegije pristupa oblacima.

Sada slijede najčešći oblici prijetnji u oblaku.

Prisluškivanje u prometu događa se kada se podaci koji se prenose u (ili unutar) oblak (obično od korisnika oblaka do pružatelja usluga u oblaku) presreću od strane zlonamjernog agenta u svrhu nelegitimnog prikupljanja podataka. Cilj ovog napada je izravno ugrožavanje povjerljivosti podataka i, eventualno, povjerljivosti odnosa između korisnika oblaka i pružatelja usluga u oblaku.

Prijetnja *zlonamjernog posrednika* nastaje kada poruke presreće i promijeni zlonamjerni agent i na taj način potencijalno ugrozi povjerljivost i/ili integritet poruke. Također može umetnuti štetne podatke u poruku prije nego što ih proslijedi odredištu.

Cilj napada *uskraćivanja usluge (DoS)* jest preopterećenje IT resursa do trenutka u kojem više ne mogu pravilno funkcionirati. Ovaj oblik napada obično se izvodi tako da se opseg rada u oblačnim uslugama umjetno povećava oponašanjem poruka, ponovljenim zahtjevima za komunikacijom ili zahtjevima koji troše prekomjernu memoriju i resurse za obradu. Mreža se preoptereti prometom kako bi se smanjila reaktivnost usluge u oblaku i osakatile njezine performanse.

Do napada *nedovoljne autorizacije* dolazi kada se napadaču omogući pogrešno ili preširoko djelovanje, što rezultira time da napadač dobije pristup IT resursima koji bi inače trebali biti zaštićeni. Do varijacije ovog napada, poznate kao *slaba provjera autentičnosti*, može doći ako se koriste slabe lozinke za zaštitu IT resursa.

Moguć je i *napad na virtualizirane resurse*. Virtualizacija pruža višestrukim korisnicima oblaka pristup IT resursima koji dijele temeljni hardver, ali su međusobno logički izolirani. Budući da pružatelji usluga u oblaku pružaju korisnicima oblaka administrativni pristup virtualiziranim IT resursima (kao što su virtualni serveri), postoji rizik da bi korisnici oblaka mogli zloupotrijebiti taj pristup kako bi napali temeljne fizičke IT resurse.

Ako fizičke IT resurse unutar oblaka dijele različiti korisnici usluga u oblaku, ti korisnici imaju *preklapajuće granice povjerenja*. Zlonamjerni korisnici usluga u oblaku mogu ciljati dijeljene IT resurse s namjerom da ugroze korisnike oblaka ili druge IT resurse koji su unutar iste granice povjerenja.

Poznati slučaj je i *napad na kontejnere*. Korištenje kontejnera unosi nedostatak izolacije od glavnog operacijskog sustava. Budući da kontejneri smješteni na istom stroju dijele isti matični operacijski sustav, sigurnosne prijetnje su veće jer se može dobiti pristup cijelom sustavu. Ako je osnovni domaćin (*host*) ugrožen, može se utjecati na sve kontejnere koji su na njemu.

Kod sigurnosti u oblaku treba obratiti pažnju i na još neke situacije. Korisnici oblaka moraju biti svjesni da mogu doprinijeti sigurnosnim rizicima ako u oblak uvedu aplikacije s manama u implementaciji. Nadalje, pružatelj usluga u oblaku mora paziti da ne definira nespojive sigurnosne politike[3] oblaka. Korisnik oblaka, s druge strane, treba proučiti pružateljeve sigurnosne politike pri odabiru, i provjeriti da nisu kontradiktorne s njegovim pravilima. Također, bitno je jasno definirati odgovornost, naknadu štete i krivicu za moguće kršenje sigurnosti kroz pravne sporazume koje potpisuju pružatelj i korisnik oblaka.

[3]Sigurnosna politika uspostavlja skup sigurnosnih pravila i propisa. Često će sigurnosne politike dalje definirati kako se ta pravila i propisi primjenjuju i provode.

1.7 Mehanizmi u oblaku

Razni mehanizmi sudjeluju u različitim zadaćama izgradnje oblaka. Opisat ćemo nekoliko vrsta takvih mehanizama.

1.7.1 Mehanizmi infrastrukture oblaka

Mehanizmi oblačne infrastrukture osnovni su građevni blokovi oblačnog okruženja i čine temelj osnovne arhitekture oblaka.

Perimetar logičke mreže. Definiran kao izoliranje mrežnog okruženja od ostatka komunikacijske mreže, perimetar logičke mreže uspostavlja granicu virtualne mreže koja može obuhvatiti i izolirati skupinu povezanih informatičkih resursa u oblaku. Ovaj se mehanizam može primijeniti za izolaciju IT resursa u oblaku od neautoriziranih korisnika, za izolaciju IT resursa u oblaku od nekorisnika i za kontrolu propusnosti izdvojenim IT resursima. Perimetri logičke mreže obično se uspostavljaju putem mrežnih uređaja koji opskrbljuju i kontroliraju povezanost podatkovnog centra i obično se primjenjuju kao virtualizirano IT okruženje koje uključuje *virtualni vatrozid* (engl. *virtual firewall*) (IT resurs koji aktivno filtrira mrežni promet prema i iz izolirane mreže, kontrolirajući njezine interakcije s Internetom) i *virtualnu mrežu* (engl. *virtual network*) (obično realizirana putem VLAN-a, ovaj resurs izolira mrežnu okolinu unutar infrastrukture podatkovnog centra).

Virtualni server. Virtualni server, već spominjan i opisan, predstavlja najtemeljniji građevni blok oblaka.

Uređaj za pohranu u oblaku. Mehanizam uređaja za pohranu u oblaku predstavlja uređaje za pohranu koji su posebno dizajnirani za korištenje u oblaku. Ovi uređaji se mogu virtualizirati, slično kao i fizički serveri koji mogu izraditi slike virtualnog servera. Uređaji za pohranu u oblaku mogu biti izloženi daljinskom pristupu putem usluga pohrane u oblaku. Glavne brige vezane za pohranu u oblaku su sigurnost, integritet i povjerljivost podataka, koje postaju podložnije kompromisu kada je pohrana povjerena vanjskim pružateljima usluga u oblaku. Može doći do pravnih i regulatornih posljedica koje proizlaze iz premještanja podataka preko geografskih ili nacionalnih granica. Drugi problem odnosi se posebno na performanse velikih baza podataka. LAN pruža lokalno pohranjene podatke s razinom pouzdanosti i latencije mreže koji su bolji od WAN-a.

Monitor korištenja oblaka. Mehanizam nadzora uporabe oblaka lagan je i autonomni softverski program odgovoran za prikupljanje i obradu podataka o korištenju IT resursa.

Replikacija resursa. Definirana kao stvaranje više instanci istog IT resursa, replikacija se obično izvodi kad je potrebno poboljšati dostupnost i performanse IT resursa. Tehnologija virtualizacije koristi se za implementaciju mehanizma replikacije resursa za kopiranje informatičkih resursa u oblaku.

Spremno okruženje. Spremno okruženje definirajuća je komponenta modela isporuke u oblaku PaaS koji predstavlja unaprijed definiranu platformu u oblaku koja se sastoji od

skupa već instaliranih IT resursa (npr. baza podataka), spremnih za upotrebu i prilagođenih korisniku oblaka. Ova okruženja koriste korisnici oblaka za daljinsko razvijanje vlastitih usluga i aplikacija unutar oblaka.

Kontejneri. Kontejneri, također već opisani, pružaju učinkovito sredstvo isporuke usluga u oblaku.

1.7.2 Specijalizirani mehanizmi oblaka

Tipična arhitektura oblaka sadrži brojne razvijajuće dijelove za rješavanje različitih zahtjeva vezanih uz korištenje IT resursa. Svaki mehanizam opisan u ovom poglavlju ispunjava određenu funkciju izvođenja kao podršku jednoj ili više karakteristika oblaka.

Automatizirani osluškivač skaliranja. Uslužni agent koji nadgleda i prati komunikaciju između korisnika usluga u oblaku i usluga u oblaku za potrebe dinamičkog skaliranja. Automatizirani osluškivači skaliranja raspoređeni su unutar oblaka, obično u blizini vatrozida, odakle automatski prate informacije o statusu radnog opterećenja.

Održavanje ravnoteže opterećenja. Uobičajena potreba za horizontalnim skaliranjem javlja se kad treba uravnotežiti opterećenje rada kroz dva ili više IT resursa radi povećanja performansi i kapaciteta većeg od onog što jedan IT resurs može pružiti.

SLA Monitor. Koristi se za posebno promatranje performansi usluga u oblaku za vrijeme izvršavanja kako bi se osiguralo da ispunjavaju ugovorne QoS zahtjeve koji su objavljeni u SLA dokumentu. Sustav može proaktivno djelovati na usluge u oblaku kad se pojave nepogodne situacije, primjerice kada SLA monitor prijavi da usluga u oblaku ne radi.

Monitor plaćanja po upotrebi. Mjeri i evidentira korištenje informatičkih resursa u oblaku u skladu s unaprijed definiranom cijenom za potrebe izračuna i naplate naknada. Neke tipične varijable praćenja su količina zahtjeva/poruka i prenesena količina podataka.

Monitor revizije. Koristi se za prikupljanje podataka radi potpore (ili diktiranja) regulatornih i ugovornih obveza. Monitor može presretati zahtjeve za prijavu i pohranjivati sigurnosne vjerodajnice podnositelja zahtjeva, kao i neuspješne i uspješne pokušaje prijave, i to sve u bazu podataka za buduće potrebe izvještavanja revizije.

Sustav za oporavak prebacivanjem (engl. failover). Koristi se za povećanje pouzdanosti i dostupnosti IT resursa pomoću klasteriranja, pružanjem redundantnih implementacija. Sustav za prebacivanje konfiguriran je za automatsko prebacivanje na duplikatnu ili pripravnu instancu IT resursa kad god trenutno aktivni IT resurs postane nedostupan.

Hypervisor. Temeljni je dio infrastrukture za virtualizaciju. *Hypervisor* je uglavnom ograničen na jedan fizički server i stoga može stvoriti samo virtualne slike tog servera.

Klaster resursa. Informatički resursi u oblaku koji su geografski razmješteni na različitim lokacijama mogu se logički kombinirati u skupine radi poboljšanja njihove upotrebe. Mehanizam grupiranja resursa koristi se za grupiranje više instanci IT resursa tako da se njima

može operirati kao jednim IT resursom. Uobičajene vrste klastera resursa su klasteri servera i klasteri baza podataka.

Posrednik za različite uređaje (engl. Multi-Device Broker). Pojedinačnoj usluzi u oblaku obično treba pristupiti niz *različitih* korisnika usluga u oblaku (mobilni uređaj, laptop, i sl.) koji se razlikuju po svojim hardverima i zahtjevima. Kako bi se prevladala nespojivost između usluge u oblaku i različitih korisnika usluge u oblaku potrebno je stvoriti posrednički logički sloj mapiranja informacija koje se razmjenjuju. Posrednik za različite uređaje uglavnom postoji kao *gateway*[4]. Obično prevodi transportni protokol, protokol razmjene poruka, protokol pohranjivanja ili koristi podatkovne sheme.

1.7.3 Mehanizmi za upravljanje u oblaku

Informatičke resurse u oblaku potrebno je postaviti, konfigurirati, održavati i nadzirati. Navest ćemo sustave koji obuhvaćaju i omogućavaju ove vrste upravljačkih zadataka. Oni čine ključne dijelove arhitekture oblaka olakšavajući kontrolu i razvoj IT resursa.

Mehanizam daljinskog upravljanja pruža alate i korisnička sučelja za vanjske administratore resursa u oblaku za konfiguraciju i upravljanje IT resursima u oblaku. Zadaci ovog mehanizma su konfiguriranje i postavljanje usluga u oblaku, pružanje i oslobađanje IT resursa za usluge u oblaku na zahtjev, nadgledanje stanja, upotrebe i performansi usluga u oblaku, praćenje ispunjenja QoS i SLA, upravljanje troškovima korištenja, upravljanje korisničkim računima, autorizacijom i kontrolom pristupa, praćenje unutarnjeg i vanjskog pristupa iznajmljenim uslugama, planiranje i procjena pružanja IT resursa.

Sustav upravljanja resursima podrazumijeva upravljanje slikama virtualnih IT resursa, raspoređivanje virtualnih IT resursa u raspoloživu fizičku infrastrukturu kao odgovor na pokretanje, pauziranje i prestanak instanci virtualnih IT resursa, provođenje politika upotrebe i sigurnosti tijekom životnog ciklusa usluga u oblaku.

Mehanizam upravljanja SLA pravilima predstavlja niz komercijalno dostupnih proizvoda za upravljanje oblakom koji pružaju značajke koje se odnose na administraciju, prikupljanje, pohranu i izvještavanje o SLA podacima.

Mehanizam upravljanja naplatama posvećen je prikupljanju i obradi podataka o upotrebi. Sustav upravljanja naplatama omogućuje definiranje različitih cjenovnih politika, kao i prilagođenih modela cijena s obzirom na korisnike oblaka ili IT resurse. Modeli određivanja cijena mogu se razlikovati od najčešćeg modela plaćanja po upotrebi, do fiksne cijene ili plaćanja po alokaciji resursa. Kad se uspostave ograničenja, obično su u obliku kvota. Kada se kvote prekorače, sustav upravljanja naplatama može blokirati daljnje korištenje od strane korisnika oblaka.

[4] *Gateway* je uređaj koji se nalazi u čvoru mreže računala i služi za komuniciranje sa nekom drugom mrežom koja koristi drugačiji mrežni protokol. Naziva se još i prevodiocem protokola. Zadatke *gateway*a može izvršavati i računalo. ([38])

1.7.4 Mehanizmi za sigurnost u oblaku

Opisat ćemo nekoliko opće prihvaćenih načina zaštite u oblaku.

Šifriranje. Originalni podaci u čitljivom obliku nazivaju se otvoreni tekst. Kad se prenosi putem mreže, otvoreni tekst je ranjiv na neovlašteni i zlonamjerni pristup. Mehanizam šifriranja je digitalni sustav kodiranja posvećen očuvanju povjerljivosti i integriteta podataka. Koristi se za kodiranje podataka iz otvorenog teksta u zaštićeni i nečitljivi format. Tehnologija šifriranja obično se oslanja na standardizirani algoritam koji se naziva šifra (engl. *cipher*) radi pretvaranja podataka otvorenog teksta u šifrirane podatke koji se nazivaju šifrirani tekst. Kada se enkripcija primijeni na podatke otvorenog teksta, podaci su upareni s nizom znakova koji se naziva ključ za dešifriranje. Ključ za dešifriranje koristi se za dešifriranje šifriranog teksta u otvoreni tekst.

Hashiranje. Mehanizam hashiranja koristi se kada je potreban jednosmjerni, nereverzibilni oblik zaštite podataka. Jednom kada se na poruku primijeni *hashing* (funkcija koja uzima niz podataka bilo koje veličine i uvijek daje izlaz unaprijed određene dužine), ona se zaključava i ne postoji ključ za otključavanje poruke. Uobičajena primjena ovog mehanizma je pohranjivanje lozinki. Ako organizacija pohranjuje hashirane lozinke umjesto samih lozinki, može provjeriti podudaraju li se dvije hashirane lozinke kad se korisnik prijavi. Korisnici unose svoje lozinke koje se zatim hashiraju. Zatim se ova hashirana lozinka uspoređuje s hashiranom lozinkom koja je pohranjena u bazi podataka. Ako se te dvije hashirane lozinke podudaraju, tada je unesena ispravna lozinka i korisniku se daje pristup. Ovo omogućuje da se lozinke nikada ne moraju pohranjivati. Ako napadač uđe u bazu podataka, sve što će pronaći su šifre lozinki, a ne lozinke. ([19])

Digitalni potpis. Mehanizam digitalnog potpisa sredstvo je za pružanje autentičnosti i integriteta podataka provjerom. Prije prijenosa poruci se dodjeljuje digitalni potpis, koji se smatra nevažećim ako poruka doživi bilokakve neovlaštene izmjene. Digitalni potpis pruža dokaz da je primljena poruka jednaka onoj koju je stvorio njezin zakoniti pošiljatelj.

Infrastruktura javnog ključa (PKI). Uobičajeni pristup za izdavanje određenih vrsta ključeva za dešifriranje temelji se na mehanizmu infrastrukture javnih ključeva (PKI), koji postoji kao sustav protokola, formata podataka, pravila i praksi koji omogućuje velikim sustavima sigurno korištenje šifriranja s javnim ključevima. PKI se oslanja na upotrebu digitalnih certifikata, koji su digitalno potpisane strukture podataka koje vežu javne ključeve za identitet vlasnika certifikata.

Upravljanje identitetom i pristupom (IAM). Mehanizam upravljanja identitetom i pristupom (IAM) obuhvaća komponente i politike potrebne za kontrolu i praćenje identiteta korisnika i pristupnih privilegija za IT resurse. Naime, IAM mehanizmi postoje kao sustavi koji se sastoje od četiri glavne komponente:

- *provjera autentičnosti*: kombinacije korisničkog imena i lozinke ostaju najčešći oblici vjerodajnica za provjeru autentičnosti korisnika kojima upravlja IAM sustav koji

također može podržavati digitalni potpis, digitalne certifikate, biometrijski hardver (čitači otisaka prstiju), specijalizirani softver (poput programa za analizu glasa) i zaključavanje korisničkih računa na registriranu IP ili MAC adresu

- *autorizacija*: komponenta autorizacije definira ispravnu kontrolu pristupa i nadzire odnose između identiteta, prava kontrole pristupa i dostupnosti IT resursa
- *upravljanje korisnicima*: vezano za administrativne mogućnosti sustava, program za upravljanje korisnicima odgovoran je za stvaranje novih identiteta korisnika i pristupnih grupa, postavljanje lozinki, itd.
- *upravljanje vjerodajnicama*: sustav upravljanja vjerodajnicama utvrđuje identitete i pravila kontrole pristupa definiranim korisničkim računima, što umanjuje prijetnju nedovoljne autorizacije

Iako su njegovi ciljevi slični onima mehanizma PKI, opseg provedbe mehanizma IAM je različit jer njegova struktura obuhvaća kontrolu pristupa i politike uz dodjeljivanje specifičnih razina korisničkih privilegija.

Jedinstvena prijava (SSO). Korisnik oblaka obično tijekom jedne sesije pristupa više usluga u oblaku. Kako se korisnik nebi trebao autentificirati prilikom pristupanja svakoj od njih, ovaj mehanizam treba omogućiti korektnu autorizaciju dok se dijele informacije između resursa unutar pojedinog sigurnosnog konteksta. SSO mehanizam posebno je koristan kada korisnik usluge u oblaku treba pristupiti uslugama koje žive u različitim oblacima.

Sigurnosne grupe u oblaku. Segmentacija resursa u oblaku proces je kojim se izrađuju zasebna fizička i virtualna IT okruženja za različite korisnike i skupine. Na primjer, jedna mreža može biti zaštićena vatrozidom za vanjski pristup Internetu, dok druga ne koristi vatrozid jer su njezini korisnici interni i ne mogu pristupiti Internetu. Proces segmentacije resursa u oblaku stvara mehanizme sigurnosnih grupa u oblaku koje se utvrđuju sigurnosnim politikama. Mreže su segmentirane u sigurnosne skupine u oblaku koje tvore perimetre logičke mreže. Svaki informatički resurs u oblaku dodijeljen je barem jednoj sigurnosnoj grupi u oblaku. Svakoj se sigurnosnoj grupi u oblaku dodjeljuju posebna pravila koja upravljaju komunikacijom između sigurnosnih skupina. Na primjer, jednoj grupi se može zadati pravilo da svi resursi unutar grupe imaju otvoren port 6644.

Ojačane slike virtualnog servera. Kao što je prethodno opisano, virtualni server stvara se iz konfiguracijskog predloška koji se naziva slika virtualnog servera. Ojačavanje je postupak uklanjanja nepotrebnog softvera iz sustava radi ograničavanja potencijalnih ranjivosti koje napadači mogu iskoristiti. Uklanjanje suvišnih programa, zatvaranje nepotrebnih portova servera, onemogućavanje neiskorištenih usluga i ograničavanje pristupa gostiju sve su primjeri ojačavanja.

1.8 Arhitektura oblaka

Arhitektura oblaka formalizira funkcionalne domene u oblaku uspostavljanjem dobro definiranih rješenja koja se sastoje od interakcija, ponašanja i različitih kombinacija mehanizama računanja u oblaku i drugih specijaliziranih tehnologija oblaka.

Arhitektura raspodjele radnog opterećenja. IT resursi mogu se horizontalno skalirati dodavanjem jednog ili više identičnih IT resursa i uravnoteživača opterećenja koji može ravnomjerno rasporediti radni teret među raspoloživim IT resursima. Rezultirajuća arhitektura raspodjele radnog opterećenja barata i prekomjernom upotrebom IT resursa i nedovoljnom upotrebom IT resursa.

Arhitektura udruživanja resursa. Temelji se na korištenju jednog ili više skupova resursa, u kojima se identični IT resursi grupiraju i održavaju u sustavu koji automatski osigurava njihovu sinkronizaciju. To može biti skup fizičkih servera, skup virtualnih servera, skup uređaja za pohranu, skup CPU-ova, itd.

Arhitektura dinamičke skalabilnosti. Ovaj arhitektonski model zasnovan je na sustavu unaprijed definiranih uvjeta skaliranja koji pokreće dinamičku raspodjelu IT resursa iz skupova udruženih resursa. Dinamička raspodjela omogućava promjenjivu uporabu diktiranu varijacijom potražnje, na primjer nepotrebni IT resursi učinkovito se uklone bez potrebe za ručnom interakcijom.

Arhitektura kapaciteta elastičnih resursa. Primarno je povezana s dinamičkim pružanjem virtualnih servera. Koristi se sustav koji dodaje i uklanja CPU-ove i RAM kao neposredan odgovor na promjenjive potrebe korisnika.

Arhitektura uravnoteženja opterećenja usluga. Smatra se specijaliziranom vrstom arhitekture raspodjele radnog opterećenja koja je usmjerena posebno na skaliranje implementacija usluga u oblaku. Stvore se redundantne implementacije usluga u oblaku, te se onda dodaju sustavi za uravnoteženje opterećenja kako bi dinamički raspodijelili radno opterećenje među uslugama. Duplirane implementacije usluge u oblaku organizirane su u resursni bazen, dok je uravnoteženje opterećenja pozicionirano kao vanjska ili ugrađena komponenta.

Arhitektura bujanja u oblak. Kada *on-premise* resursi dosegnu svoje kapacitete, arhitektura bujanja u oblak skalira te resurse na način da alocira te IT resurse u oblaku. Kada se zahtjevi smanje dovoljno da ih kapaciteti *on-premise* resursa podnose, resursi iz oblaka se otpuštaju. Odgovarajući redundantni IT resursi u oblaku su unaprijed raspodijeljeni, ali su neaktivni dok se ne pojavi potreba za njima. Temelj ovog arhitektonskog modela zasnovan je na mehanizmima za automatsko skaliranje i replikaciju resursa.

Arhitektura pružanja elastičnog diska. Korisnicima oblaka se obično naplaćuje fiksni memorijski prostor na tvrdom disku, što znači da su troškovi unaprijed određeni kapacitetom diska i nisu usklađeni sa stvarnom potrošnjom prostora za pohranu podataka. Na primjer, korisnik oblaka koristi virtualni server s operacijskim sustavom Windows Server i

tri tvrda diska od 150 GB. Korisniku oblaka naplaćuje se korištenje 450 GB prostora za pohranu nakon instaliranja operacijskog sustava, iako još nije instalirao nijedan softver. Kako bi se doskočilo takvim situacijama, koristi se arhitektura pružanja elastičnih diskova koja uspostavlja dinamički sustav za skladištenje koji osigurava da se korisniku oblaka precizno naplati točna količina prostora za pohranu koju stvarno koristi.

Arhitektura redundantne pohrane. Uređaji za pohranu u oblaku povremeno su podložni kvarovima i poremećajima koji su uzrokovani problemima s mrežnim povezivanjem, općim oštećenjem hardvera ili sigurnosnim kršenjima. Kvar uređaja za pohranu u oblaku može imati domino efekt i prouzrokovati neuspješan rad na svim servisima, aplikacijama i komponentama infrastrukture u oblaku koje se oslanjaju na njegovu dostupnost. Arhitektura redundantne pohrane uvodi sekundarni duplikatni (redundantni) uređaj za pohranu u oblaku kao dio sustava za oporavak prebacivanjem koji sinkronizira svoje podatke s podacima u primarnom uređaju za pohranu u oblaku. Preusmjeravaju se zahtjevi korisnika oblaka na sekundarni uređaj kad god primarni uređaj ne radi kako treba. Pružatelji usluga u oblaku mogu smjestiti sekundarne uređaje za pohranu u oblaku u različite geografske regije, obično iz ekonomskih razloga. Međutim, to može dovesti do pravnih pitanja za neke vrste podataka.

1.9 Metrike plaćanja

Ovaj se odjeljak bavi opisom uobičajenih vrsta metrika koje se koriste za procjenu troškova zakupa informatičkih resursa u oblaku te usporedbom ulaganja u oblačne i on-premise IT resurse.

Treba spomenuti *početno ulaganje* i *tekuće ulaganje* u resurse. Početno ulaganje kod on-premise resursa je uglavnom visoko. Uključuju se troškovi za hardver, softver i radnu snagu potrebnu za uspostavljanje. Početno ulaganje kod oblaka je nisko, eventualno je potrebno postaviti okolinu u oblaku. Tekuće ulaganje u resurse je kod on-premise resursa varirajuće. Uključene su naknade za licenciranje, električnu energiju, osiguranje i rad. Tekuće ulaganje u rad informatičkih resursa u oblaku može također varirati, ali često je više od tekućeg ulaganja u on-premise informatičke resurse (posebno tijekom dužeg vremenskog razdoblja). Uključeni su troškovi najma hardvera, naknade za promet na mreži, naknade za licenciranje i radnu snagu. Već smo objasnili u odjeljcima 1.2.2 i 1.7.3 da je uobičajeni model plaćanja u oblaku *pay-as-you-go*, odnosno da se resursi plaćaju po upotrebi.

Treba uzeti u obzir da je lakše dati veći iznos novca kroz duži vremenski period (kao u oblaku), nego odjednom na početku (kao za on-premise sustave). S druge strane, treba uzeti u obzir i integracijske troškove, odnosno imati na umu da je potrebno svoje rješenje prilagoditi tuđem okruženju (što je slučaj oblaka).

Najčešće metrike za izračun troškova su:

- *Upotreba mreže*: ulazni i izlazni mrežni promet, kao i mrežni promet unutar oblaka. Npr. za izlazni promet: 1 GB mjesečno besplatno, 0.01$/GB između 1 i 10 TB mjesečno.
- *Upotreba servera*: dodjela i korištenje virtualnog servera.
  Npr. 0.10$/sat manja instanca, 0.20$/sat srednja instanca, 0.90$/sat velika instanca.
- *Upotreba pohrane*: dodjela kapaciteta za pohranu.
  Npr. 0.10$/TB.
- *Upotreba usluge u oblaku*: trajanje pretplate, broj korisnika, broj transakcija.
  Npr. 69.90$ mjesečno.

Cijene se mogu razlikovati ovisno i o usvojenom modelu isporuke u oblaku:

- *IaaS*: cijene se obično temelje na dodjeli i korištenju IT resursa, što uključuje količinu prenesenih mrežnih podataka, broj virtualnih servera i dodijeljeni kapacitet za pohranu.
- *PaaS*: slično kao IaaS, ovaj model obično definira cijene za mrežni prijenos podataka, virtualne servere i pohranu. Cijene su promjenjive ovisno o čimbenicima poput konfiguracije softvera, razvojnih alata i naknada za licenciranje.
- *SaaS*: budući da se ovaj model odnosi samo na korištenje aplikacije, cijene se određuju brojem aplikacijskih modula u pretplati, brojem korisnika usluga u oblaku i brojem transakcija.

1.10 SLA

Sporazum o razini usluge (SLA) je središnja točka pregovora, zakonskih obveza i dogovorenih naplaćivanja. SLA je sastavni dio načina na koji organizacije grade automatizaciju poslovanja oko korištenja IT resursa u oblaku.

SLA, izdan od pružatelja usluga u oblaku, čitljivi je dokument koji opisuje značajke, usluge, jamstva i ograničenja jednog ili više IT resursa ili usluge u oblaku.

SLA koristi metrike kvalitete usluge za mjerenje QoS karakteristika:

- *dostupnost*: trajanje rada, prekidi rada, trajanje usluge
- *pouzdanost*: minimalno vrijeme kvara, zajamčena stopa uspješno prihvaćenih zahtjeva

- *performanse*: kapacitet, vrijeme odziva i garancija vremena isporuke
- *skalabilnost*: upravljanje kapacitetom
- *otpornost*: prosječno vrijeme oporavka od kvara

Primjer opisa resursa servera u SLA dokumentu:

- *Opis*: mjerljive karakteristike kapaciteta servera
- *Mjere*: broj CPU-ova, frekvencija CPU-a u GHz, veličina RAM-a u GB, veličina memorije u GB
- *Dostupnost*: stalna
- *Model isporuke u oblaku*: IaaS, PaaS
- *Primjer*: 1 jezgra jačine 1,7 GHz, 16 GB RAM-a, 80 GB prostora za pohranu

Poglavlje 2

Podatkovna platforma u oblaku

Među glavnim prednostima oblaka su brzo pokretanje raznolikih resursa i olakšano povezivanje među njima, što u samostalnom pristupu van oblaka može biti, i u pravilu jest, izrazito zahtjevno. Kao primjer proizvoda kojeg se može implementirati u oblaku te koji koristi spomenute prednosti, opisat ćemo *podatkovnu platformu*. Najprije ćemo precizirati pojam podatkovne platforme, a zatim ćemo opisati podatkovnu platformu *u oblaku*.

2.1 Podatkovna platforma

Podatkovna platforma je softverska platforma za prikupljanje, analizu i upravljanje podacima. Omogućuje poslovanju identifikaciju homogenih skupina primjena podataka, na primjer ciljanje određenih korisnika u internetskim reklamnim kampanjama. Podatkovna platforma je dobro upravljani, centralizirani ili distribuirani sustav u kojem mnogi organizacijski subjekti mogu generirati podatke i koristiti podatke iz kojih se stječe uvid, a onda i povećava organizacijska učinkovitost. Podatkovna platforma ključni je faktor za *demokratizaciju podataka* te od podataka stvara *podatkovna dobra* (engl. *data asset*), to jest glavno sredstvo za napredak u poslovanju. Demokratizacija podataka i podatkovna dobra su dva izrazito aktualna pojma danas te će biti opisana u sljedećim potpoglavljima. Stručnjaci koji sudjeluju u izgradnji podatkovne platforme, i koji će se spominjati u ovom radu, su *podatkovni arhitekti* (engl. *data architects*), *podatkovni inženjeri* (engl. *data engineers*) i *podatkovni znanstvenici* (engl. *data scientists*).

2.1.1 Demokratizacija podataka

Demokratizacija podataka je sposobnost da informacije u digitalnom formatu budu dostupne i korisne prosječnom krajnjem korisniku. Cilj demokratizacije podataka je omogućiti nespecijaliziranim ljudima prikupljanje i analiziranje podataka bez traženja dodatne pomoći.

2.1.2 Podatkovna dobra

Podatkovna dobra (engl. *data asset*) su rezultat uzimanja sirovih podataka iz aktivnog poslovanja i generiranje podataka visoke kvalitete kao krajnji proizvod, radi integracije i nadziranja poslovanja. Time se bavi tim stručnjaka posvećenih pružanju visokokvalitetnih podataka za poslovnu upotrebu.

Podaci unutar poslovanja mogu se razvrstati u tri skupine:

- *Podaci aktivnog poslovanja*. Proizvode ih aplikacije unutar poslovanja, poput one koju tvrtka koristi za upravljanje financijskim transakcijama. To su sirovi podaci za skladište podataka (engl. *data warehouse*, bit će opisano u odjeljku 2.2.2.2).
- *Podaci za integriranje poslovanja*. Podaci za poboljšanje kvalitete i za sinkronizaciju dviju ili više aplikacija unutar poslovanja, poput glavnog popisa kupaca. Podaci služe za integraciju aplikacija koje nisu bile osmišljene za međusobnu suradnju.
- *Podaci za praćenje poslovanja*. Služe za predstavljanje krajnjim korisnicima koji ih onda koriste za izvještavanje i kao podršku pri donošenju odluka, puput financijskih izvještaja. Podaci se čiste kako bi korisnici mogli bolje razumjeti razvoj i napredak te procijeniti uzročno-posljedične veze u podacima.

Na podatke treba gledati kao na strateška dobra. Kako bi pronašle način kako kapitalizirati ili monetarizirati svoja podatkovna dobra, neke organizacije počnu analizirati vrijednost svojih podataka o klijentima. To nije nova ideja, to se događa u maloprodajnim trgovinama koje koriste kartice vjernosti kod svojih kupaca već desetljećima. Ali podaci o kupcima mogu proizvesti materijalnu vrijednost (ako se podaci prodaju, trguju) ili ”slučajno” kada se iz tih podataka stvori nova usluga, materija, iako se podaci se zapravo ne prodaju – Google i Facebook su dobri primjeri ovog slučaja. Organizacije mogu preuzeti javno dostupne podatke za izradu korisnih skupova podataka za kupnju ili za korištenje kao uslugu. Primjeri za to su tvrtke za istraživanje tržišta poput tvrtke Nielsen.

Pitanje je kako, odnosno koju metriku koristiti za određivanje vrijednosti podatkovnih dobara, a to se efektivno može mjeriti učestalošću korištenja podataka. Materijalnim dobrima (strojevi, oprema, zgrade, itd.) vrijednost obično pada što se više koriste. Ali podacima kao nematerijalnim dobrima se vrijednost može povećavati što se oni više koriste. Ako na podatke gledate kao na imovinu, sredstvo, dobro, onda oni dobivaju veću vrijednost (obnovljivi su, trajni i imaju stratešku svrhu).

Još uvijek ne postoji standardizirana definicija kako bi podatkovna dobra trebala biti registrirana u knjigama organizacije. Postoji nekoliko različitih metoda pomoću kojih se pokušava staviti vrijednost na nematerijalnu imovinu. Jedan od načina jest traženje slične ili identične imovine i dobara u nedavnim tržišnim transakcijama kojima je na taj način

ocijenjena vrijednost. Također se mogu procijeniti izračunom koliko prihoda stvaraju ili koliki trošak bi proizvelo njihovo generiranje ili zamjena.

2.1.3 Stručnjaci na podatkovnoj platformi

Data architect (podatkovni arhitekt) je osoba koja je odgovorna za dizajn, implementaciju, upotrebu i upravljanje arhitekturom podataka organizacije. Definira kako će se podaci pohranjivati, koristiti, integrirati i upravljati od strane različitih korisnika ili sustava. Osigurava da organizacija slijedi formalni standard podataka, održava registar metapodataka[1], optimizira baze podataka i još mnogo toga. ([31])

Data engineer (podatkovni inženjer) radi sa skupovima podataka kako bi unaprijedio ciljeve znanosti o podacima (engl. *data science*). Za razliku od ostalih uloga, poput podatkovnih znanstvenika, podatkovni inženjer uglavnom nije uključen u sveukupnu stratešku analizu, već je dublje uključen u životni ciklus skupova podataka kako bi podaci bili korisni za potrebnu svrhu. Prvenstveno su usmjereni na objedinjavanje neobrađenih podataka i njihovo pretvaranje u korisne, uredne i strukturirane formate podataka. Na primjer, projekt može uključivati stvaranje podatkovne platforme, odnosno prikupljanje podataka s raznih izvora, njihovo prebacivanje u središnje spremište, pročišćavanje tih podataka te njihovo prikazivanje u vrlo preciznom formatu za upotrebu u umjetnoj inteligenciji i strojnom učenju. Ovdje bi podatkovni inženjer prvenstveno sudjelovao u promatranju načina na koji se ti podaci prikupljaju, kako se ti podaci čiste i kako se kreću kroz tzv. *cjevovod podataka* (engl. *data pipeline*), kao i u osiguravanju da poslovanja završavaju s pravim podatkovnim dobrima (engl. *data assets*). Sigurnost podataka i upravljanje podacima (engl. *data governance*) također su dio rada podatkovnog inženjera. Općenito, podatkovni inženjeri široko su traženi jer podaci postaju jedno od najbitnijh dobara svih poslovanja. ([32])

Data scientist (podatkovni znanstvenik) vrši statističke analize, procese vađenja i pretraživanja velike količine podataka kako bi se otkrili trendovi, brojke i drugi relevantni podaci. Vrši analizu podataka pohranjenih u skladištima podataka ili podatkovnim centrima za rješavanje različitih poslovnih problema, optimiziranje performansi i u svrhu poslovne inteligencije. Gotovo da i nisu od koristi što se tiče strateške ili novčane koristi, već su opremljeni statističkim modelima i analiziraju prošle i trenutne podatke iz takvih spremišta podataka kako bi dobili preporuke i prijedloge za optimalno donošenje poslovnih odluka. Uglavnom su dio procesa marketinga i planiranja radi prepoznavanja korisnih uvida i dobivanja statističkih podataka za planiranje, provođenje i praćenje marketinških strategija usmjerenih na rezultate. ([30])

Ukratko, podatkovni arhitekt brine se o osmišljavanju cijele podatkovne platforme promatrajući sve aspekte tako da je u potpunosti prilagođena određenom poslovanju. Podat-

[1] Metapodaci su podaci koji opisuju određeni skup podataka (format, semantiku, ...) ili pružaju informaciju s kojom je te podatke lako pronaći, evaluirati i razumjeti.

kovni inženjer brine se o samim podacima, na koji način ih dovući, pročistiti i pospremiti onako kako je to podatkovni arhitekt osmislio. Podatkovni znanstvenik uglavnom samo koristi sređene podatke (zadnja faza podatkovne platforme), ali svojim zahtjevima za određenim izgledom podataka može utjecati na samu izgradnju podatkovne platforme.

2.2 Općenita podatkovna platforma u oblaku

Za opisanu podatkovnu platformu ćemo pokazati i objasniti shemu generalizirane podatkovne platforme implementirane *u oblaku*, neovisne o konkretnoj platformi za pristup oblaku. Shema je prikazana na slici 2.1. Platformu ćemo opisati po njezinim fazama: doprema podataka u oblak, tranformacije podataka unutar oblaka i korištenje podataka iz podatkovne platforme u oblaku, dodatno uz opis infrastrukture podataka.

2.2.1 Doprema podataka

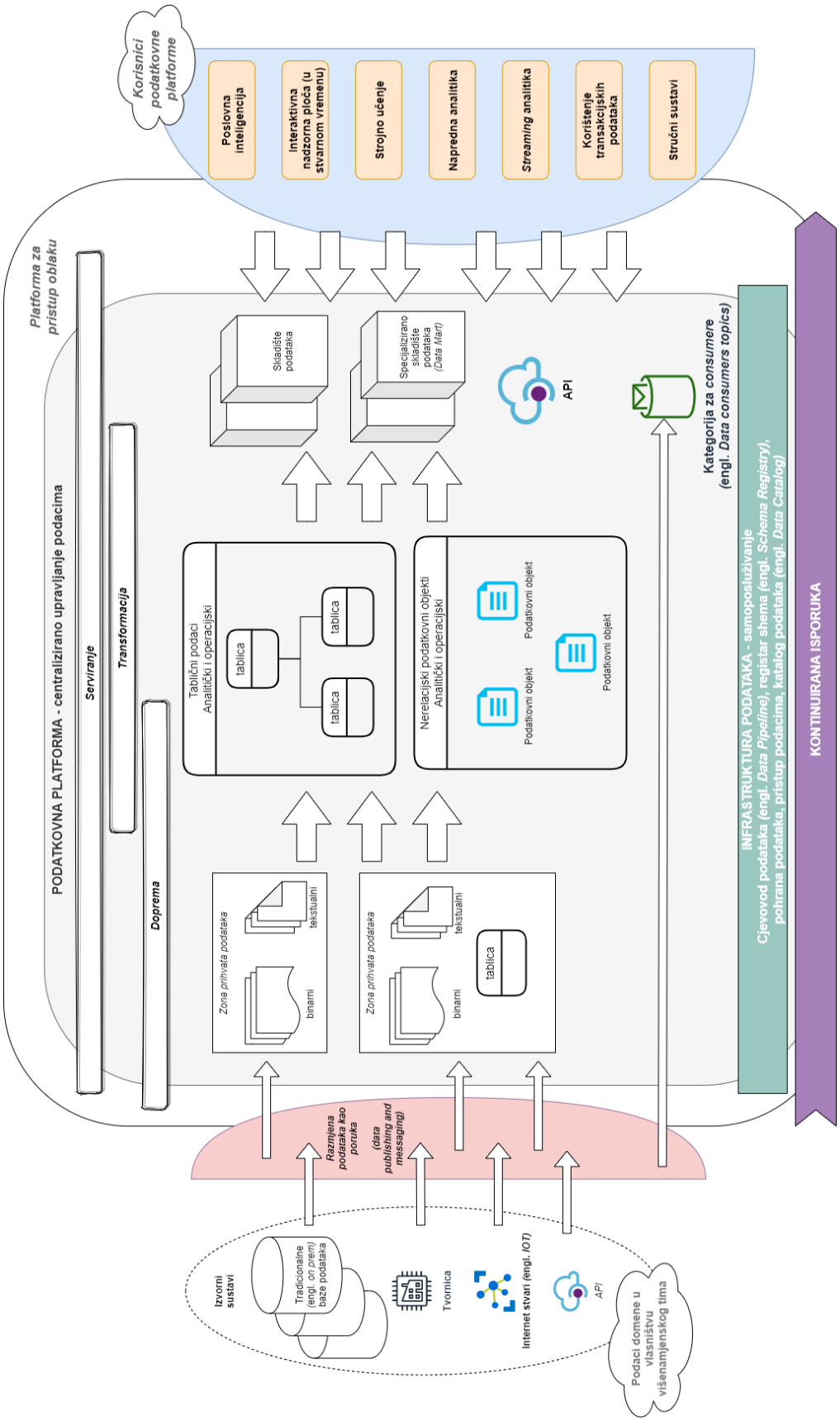
Prva faza implementacije podatkovne platforme u oblaku je doprema podataka s nekog izvora podataka, koji nije ili jest u oblaku. Izvorni sustavi su mnogobrojni i raznoliki, primjeri se kreću od tradicionalnih (engl. *on-premise*) poslovnih sustava za obradu podataka u podatkovnim centrima, preko tvornica i Interneta stvari (engl. *Internet of Things, IOT*), do raznih sučelja za programiranje aplikacija (engl. *APIs*).

2.2.1.1 Izvorni sustavi

Opisat ćemo neke od čestih izvora podataka.

Tradicionalni (engl. on-premise*) poslovni sustavi za obradu podataka u podatkovnim centrima* su sustavi čiji se podaci spremaju na lokalnim serverima unutar poslovanja. Kao primjer poslovnih sustava mogu se navesti ERP i CMR. ERP (engl. *Enterprise Resource Planing*) je integrirani poslovni informacijski sustav koji podržava odvijanje mnogih operativnih procesa poduzeća u području procesa nabave, skladištenja, proizvodnje, prodaje, financija itd. CRM (engl. *Customer Relationship Management*) je strategija i skup poslovnih procesa za upravljanje odnosima s klijentima, kupcima, dobavljačima, partnerima i zaposlenicima. ([26]) Sve je popularnija implementacija ovih sustava u oblaku, ali često se i samo podaci iz sustava šalju u oblak za daljnje procesiranje, na primjer za podatkovnu platformu.

Internet stvari (engl. Internet of Things, IOT*)* opisuje tehnologiju povezivanja fizičkih uređaja, vozila i drugih stvari, koje prikupljaju, dijele i razmjenjuju podatke putem interneta. Spajanje uređaja može biti žično ili bežično te omogućuje potpuno nove mogućnosti za međusobnu interakciju (komunikaciju) između ljudi i različitih sustava. Na taj način komunicirati mogu stvari, uređaji i stvari/uređaji s ljudima, uz zajednički cilj olakšavanja



Slika 2.1: Shema općenite podatkovne platforme u oblaku.

i pojednostavljivanja života ljudi. Neki od uređaja koji rade na tehnologiji interneta stvari su bankomati, pametne klupe, hladnjaci koji putem aplikacije kažu da je potrebno kupiti neki proizvod koji je potrošen, perilica koja započne pranje tako da bude gotovo upravo kad se osoba vrati s posla, pametna narukvica koja prati tjelesnu aktivnost, ... Svi ti uređaji umrežavanjem razmjenjuju podatke s ciljem olakšavanja života ljudi. ([13]) Ti podaci su idealni za daljnje procesiranje, na primjer za analizu ponašanja tijela ljudi na temelju dobivenih podataka iz pametne narukvice.

API (Application Programming Interface) je programsko sučelje za programiranje aplikacije ili pristup podacima. To je vrsta softvera koja omogućuje dvjema aplikacijama da funkcioniraju zajedno i komuniciraju. Na primjer, svaki put kada se provjeri vremenska prognoza na mobitelu, koristi se API. ([2]) API navodi veliki broj funkcija koje razvojni programeri mogu koristiti, zajedno s opisom onoga što rade. Razvojni programer ne mora nužno znati kako se, na primjer, gradi operacijski sustav i predstavlja dijaloški okvir ”Spremi kao”. Oni samo trebaju znati da je dostupno za upotrebu u svojoj aplikaciji. ([9]) Pomoću API-ja se mogu dohvatiti podaci koje vlasnik API-ja dopušta. Na primjer, Twitter pruža API kojim se mogu dohvatiti sve objave objavljene na određeni datum.

2.2.1.2 Načini dopreme podataka u oblak

Podatke koji se nalaze ili su generirani u spomenutim sustavima treba dopremiti s pripadajućih servera/spremišta u oblak. Taj proces naziva se *unos podataka (engl.* data ingestion*)*. Unos podataka je proces dobivanja i uvoza podataka za neposrednu upotrebu ili pohranu u bazu podataka. ([23])

Iako je ovaj proces intuitivno jednostavan, on zahtijeva posebno ulaganje vremena i promišljanja te ova faza podatkovne platforme često predstavlja i jednu od najzahtjevnijih. Razlog zahtjevnosti procesa unosa podatka je velik broj izvora podataka različitih formata (izvori se često mogu nabrojati na stotine, a formati na desetine). Stoga za tvrtke može biti teško unijeti podatke razumnom brzinom i učinkovito ih obraditi kako bi zadržali konkurentsku prednost te se ovime (kao i ostatkom podatkovne platforme) bavi specijalizirani tim stručnjaka – podatkovni inženjeri (engl. *data engineers*).

Razlikujemo unos podataka *ovisno o redoslijedu unosa podataka* i *ovisno o učestalosti unosa podataka*.

Unos podataka ovisno o redoslijedu unosa podataka može biti:

- sinkroni
- asinkroni

Pri *asinkronom unosu podataka* podaci se ne sinkroniziraju kad se šalju. To se obično odnosi na podatke koji se prenose u povremenim intervalima, a ne u neprekidnom strujanju, što znači da prvi dijelovi cijele datoteke nisu uvijek prvi koji su poslani i stigli na odredište. Različiti dijelovi skupa podataka šalju se u različitim intervalima, ponekad istovremeno, ali slijede različite putove do odredišta. Unos asinkronih podataka ne zahtijeva koordinaciju bitova između dvije krajnje točke. Unos asinkronih podataka ne potiče satni signal prilikom slanja podataka, za razliku od *sinkronog unosa podataka*, gdje se slanje podataka mjeri vremenskom referencom.
Prednost asinkronog unosa podataka je fleksibilnost, sustavi mogu razmjenjivati podatke svojim tempom. Nedostaci su osjetljivost na kvar na mreži te veći volumen podataka za prijenos s obzirom da se moraju prenijeti i dodatni (opisni) podaci koji osiguravaju ispravan prijenos svih podataka. ([33])

Unos podataka ovisno o učestalosti unosa podataka dijeli se na:

- unos podataka u stvarnom vremenu (engl. *real time streaming ingestion*)
- unos podataka u serijama (engl. *batch ingestion*)

Unos podatka u stvarnom vremenu se realizira tako da se svaki podatak unosi u trenutku kada je i generiran. Zato se ovaj proces naziva i strujanje podataka.
Unos podataka u serijama se realizira tako da se podaci unose grupa po grupa u periodičnim intervalima.

2.2.1.3 Doprema i prihvat podataka u oblak sa sustava koji ih kreira

Nakon opisanih izvora i načina unosa podataka, opisujemo konkretnu tehnologiju dopreme i pojmove usko vezane uz samu dopremu, odnosno prihvat podataka.

U procesu dopreme podataka sa izvornog sustava koji ih generira sudjeluju komponente koje možemo definirati kao poseban sloj prilagođen razmjeni podataka *u obliku poruka* i distribuciji prema platformi u oblaku. Taj sloj poznat je pod terminom *publish/subscribe obrazac* i to je zapravo poseban mrežno orijentirani arhitekturni obrazac koji opisuje kako komuniciraju dva različita dijela sustava za razmjenu poruka (engl. *message pattern*) ([36]). Njegova posebnost je upravo u tim dvama različitim dijelovima sustavima: pošiljatelj poruka (u nastavku rada ćemo koristiti engleski prijevod *publisher*) i primatelj poruka (u nastavku rada ćemo koristiti engleski prijevod *consumer*). Naime, *publisher* ne šalje poruke izravno ciljanom *consumer*u, već razvrstava poruke i šalje ih u kategorije (engl. *topic*) bez ikakvog znanja o tome postoje li *consumer*i i tko su oni. Slično, *consumer*i izraze zanimanje za jednu ili više kategorija te primaju samo poruke iz tih kategorija bez ikakvog znanja o tome postoje li *publisher*i i tko su oni. ([40])

Publisher dio se koristi za unos podataka u oblak, dok se *consumer* koristi u kasnijoj fazi podatkovne platforme. *Publisher* šalje podatke iz nekog od izvornih sustava u kategoriju koja se nalazi u oblaku. *Publisher* je realiziran programskim kodom te koristi API one platforme za pristup oblaku na kojoj se nalazi kategorija za prihvat poruka. Na primjer, ako se podaci učitavaju u kategoriju na GCP-ju[2], koristi se GCP-jev API. Dakle, *publisher* je spona između izvornog sustava i oblaka, dok se kategorija, a uglavnom i *consumer*, nalaze u oblaku.

Prije samog prihvata podataka u oblaku poslanih od strane *publisher*a, posebnu pažnju pri unosu podataka treba posvetiti tipu podataka koji se unose. Datoteke se širokom podjelom mogu razvrstati na *binarne* i *tekstualne*. ([18]) Čest tip podataka su i tablični podaci, uglavnom kao posljedica široko raširenih relacijskih baza podataka, no oni ovom širom podjelom pripadaju skupini tekstualnih podataka. Ove dvije kategorije imaju različite karakteristike i potrebni su različiti alati za rad s takvim datotekama. Sve datoteke, bilo binarne ili tekstualne, sastavljene su od bajtova. Razlika između binarnih i tekstualnih datoteka je u tome kako se ovi bajtovi tumače. Svaka tekstualna datoteka jest binarna datoteka, ali obrnuto nije istina. Poznavanje razlika između binarnih i tekstualnih datoteka značajno štedi vrijeme i pogreške prilikom čitanja podataka. Primjeri binarnih datoteka su slike (jpg, gif), pjesme (mp3, wav), kompresirane datoteke (zip, 7z). Primjeri tekstualnih datoteka su dokumenti (txt, markdown), web standardi (html, json), programski kodovi (c, java). Osim što podaci izvorno mogu biti binarnog oblika kao spomenuti primjeri, razlog zašto su binarni podaci česti jest što se nerijetko i tekstualni podaci pretvaraju u binarne prije pohranjivanja. Na primjer kod poslovanja gdje su podaci toliko veliki da se moraju kompresirati ili kod poslovanja poput banaka gdje su podaci od velike tajnosti te se moraju enkriptirati.

Upravo zbog različitih tipova podataka nastaje motivacija za dijelom unutar podatkovne platforme kojeg nazivamo *zona prihvata podataka*. U njoj se podaci pohranjuju u obliku najbližem izvornom/sirovom obliku u kojem su generirani i uneseni u oblak pomoću *publisher*a. Osim spomenute motivacije o različitim tipovima sirovih podataka, svrha uvođenja zone prihvata jest i mogućnost naknadne obrade i općenita nemogućnost ili skupa cijena procesa ponovnog unosa podataka s izvornog sustava koji je te podatke generirao. Zonu prihvata mogu utjeloviti bilokakva spremišta podataka koja ne zahtijevaju određenu strukturu podataka, na primjer *jezero podataka* (engl. *data lake*). Jezero podataka je spremište podataka pohranjenih u sirovom obliku, sposobno za čuvanje velikih količina strukturiranih, polustrukturiranih i nestrukturiranih podataka.

[2]Google Cloud Platform, koji nudi Google, skup je usluga računanja u oblaku koje rade na istoj infrastrukturi koju Google interno koristi za proizvode krajnjih korisnika, kao što su Google Search, Gmail i YouTube. ([39])

2.2.2 Transformacija podataka u oblaku

Proces transformacije podataka podrazumijeva mijenjanje originalnog oblika podatka ili skupa podataka u oblik koji je prilagođen korištenju od strane sustava ili korisnika. Transformacije podataka mogu biti:

- promjena oblika (deserijalizacija binarnog oblika)
- organizacija skupa podataka (modeliranje podataka)
- operacije nad sadržajem (grupiranje, filtriranje, čišćenje)
- prilagodba transformiranog podatka konzumentu (sustavu ili korisniku)

Deserijalizacija pretvara binarni oblik u razumljiv, čitljiv objekt. Obično je potrebna kako bi se provele daljnje transformacije, ili barem provjerila potreba za njima.

Modeliranje podataka je proces kreiranja podatkovnog modela za spremanje podataka u bazu podataka. Model predstavlja konceptualnu reprezentaciju podatkovnih objekata i veza između njih te objašnjava kako bi se taj koncept trebao implementirati. Modeliranje podataka pomaže u vizualnoj reprezentaciji podataka te osigurava konzistentnost u konvenciji imena, semantici podataka, i sličnome. ([8])

Kada su podaci razumljivog i jasnog oblika, mogu se urediti i na temelju *semantike*. Podaci su često inkonzistentni (postoje i relevantni i nerelevantni podaci), neprecizni (postoje netočne informacije ili nedostajuće vrijednosti) i ponavljajući (postoje duplikati) te je ovo faza u kojoj se takvi problemi trebaju riješiti. ([34])

Ovisno o korisnicima podataka, podaci se mogu dalje transformirati i *specijalizirati* (prilagoditi) tako da pogoduju svrsi za koje ih korisnik upotrebljava.

Općenito se tranformacije na podatkovnoj platformi provode u dvije faze: najprije se provode neke osnovnije transformacije kako bi se podaci mogli spremiti u baze podataka, a zatim se podaci dalje transformiraju kako bi se specijalizirali za krajnje korisnike.

2.2.2.1 Prvotno uređivanje podataka: spremanje u baze podataka

Opisane tranformacije služe kao uvod u prvotnu organizaciju podataka na podatkovnoj platformi. U organizaciji podataka susrećemo se sa bitnim pojmovima kao što su *oblikovni obrasci* kao konceptualno rješenje organizacije podataka te *metapodaci* i *skrb o podacima* (engl. *data curation*). Tako uređeni i organizirani podaci spremaju se u *baze podataka*.

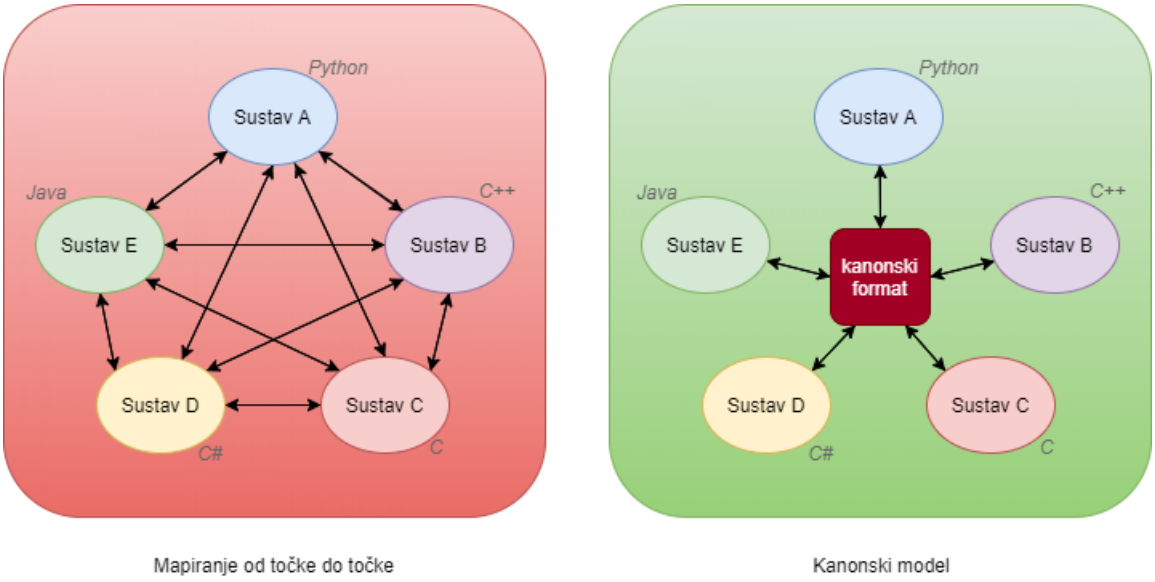
Oblikovni obrazac (engl. *design pattern*) u softverskom inženjerstvu je općenito rješenje za probleme koji se često pojavljuju. ([24]) Oblikovni obrasci se koriste i u obradi podataka, od strane *data architect*a, *data engineer*a i *data scientist*a. To nije gotov dizajn koji se može izravno transformirati u programski kod ili alat, već opis ili predložak za rješavanje problema koji se može opetovano koristiti u mnogim situacijama – rješenje za višekratnu upotrebu. Predstavljaju najbolju praksu te služe za postizanje standardizacije jednog procesa.
Na primjer, ako u modelu skladišta podataka postoji jedna činjenična i 15 dimenzijskih tablica, efektivno je imati jedan oblikovni obrazac za punjenje činjenične tablice i jedan za punjenje dimenzijskih tablica, gdje parametri mogu pomoći prilagodbi različitim situacijama. Cilj je ne imati 15 različitih procedura, već jednu.

Prednosti (i ciljevi) korištenja oblikovnih obrazaca u obradi podataka su:

- smanjenje trajanja ukupnog razvojnog ciklusa
- olakšavanje određenog postupka – standardizacija (postizanje neovisnosti o programskom jeziku ili alatu)
- stvaranje sustava koji je jednostavan za rad s njim
- osiguravanje kvalitete podataka od samog početka

Popularan oblikovni obrazac koji se koristi u optimizaciji komunikacije među sustavima koji komuniciraju različito (npr. različiti programski jezici – potreban prevoditelj) jest *kanonski model*. *Kanonski model* (engl. *canonical model*) je oblikovni obrazac koji se koristi za komunikaciju između *različitih* formata podataka. Ima za cilj predstaviti podatke i međusobne odnose u najjednostavnijem mogućem obliku kako bi se integrirali procesi u *različitim* sustavima i bazama podataka. Motivacija za kanonskim modelom je u tome što se podaci koji se razmjenjuju preko različitih sustava oslanjaju na različite jezike, sintaksu i protokole te ih je onda teško integrirati. Ovaj model mora biti u mogućnosti sadržavati i prevoditi različite vrste podataka. Na primjer, kada jedan sustav treba poslati podatke u drugi sustav, on svoje podatke prvo prevodi u standardnu sintaksu (kanonski format ili uobičajeni format) koja nije ista kao u drugom sustavu. Kad drugi sustav primi podatke iz prvog sustava, taj kanonski format prevodi u svoj vlastiti format podataka. Upotrebom kanonskog modela koristi se kanonski pristup u kojem svaka aplikacija prevodi svoje podatke u jedinstven, zajednički model koji razumiju i sve ostale aplikacije. ([10])

Drugi spomenuti aspekt organizacije podataka je *skrb o podacima* (engl. *data curation*). *Skrb o podacima* podrazumijeva organizaciju, očuvanje i upravljanje podacima, ali više se odnosi na upravljanje *metapodacima*, nego što se odnosi na same podatke. Stoga je veliki dio skbi o podacima upravo unošenje metapodataka, poput shema, čestih načina



Slika 2.2: Usporedba mapiranja od točke do točke i kanonskog modela.

korištenja podataka, bitnih i čestih operacija *join*ova, filtriranja, upita (engl. *query*), a sve u svrhu nalaženja najbolje prakse za cjelokupan rad s pripadnim podacima, odnosno pridavanja točne vrijednosti podacima poslovanja kako bi odgovarali konkretnim svrhama i zahtjevima koje poslovanje zahtijeva. ([12])

Ovisno o poslovanju, podacima i svrsi za koju su namijenjeni, podaci se nakon prvotnih tranformacija i organizacije mogu premjestiti iz zone prihvata u relacijske ili u nerelacijske *baze podataka*. Svaka ozbiljna platforma u oblaku nudi obje vrste baza podataka s obzirom da prepoznaju važnost obiju u *big data* svijetu. To upravo prikazuje prednosti oblaka – *na jednom mjestu* svako poslovanje može pronaći baš ono što njemu odgovara, pa čak i relativno brzo isprobati obje vrste te tako ispitati što mu bolje odgovara bez posebnih instalacija i konfiguriranja baza i potrebnih alata.

Relacijska baza podataka je baza podataka koja se zasniva na relacijskom logičkom modelu[3]. Relacijski model zasnovan je na matematičkom pojmu *relacije* i zahtijeva da se baza podataka sastoji od skupa pravokutnih tablica – takozvanih *relacija* koje predstavljaju *entitet* ili *vezu*. Jedan stupac u tablici predstavlja *atribut* (entiteta ili veze). Jedan redak

[3](Logički) model podataka je skup pravila koja određuju kako sve može izgledati logička struktura baze podataka. Model čini osnovu za oblikovanje i implementiranje baze. Točnije rečeno, podaci u bazi moraju biti logički organizirani u skladu s onim modelom kojeg podržava odabrani DBMS. Dosadašnji najčešći modeli su relacijski model, mrežni model, hijerarhijski model, objektni model.

relacije obično predstavlja jedan primjerak entiteta, ili bilježi vezu između dva ili više primjeraka. Redak nazivamo *n-torka*. ([14])

Nerelacijska baza podataka je baza podataka koja ne koristi tabličnu shemu redaka i stupaca koja se nalazi u većini tradicionalnih baza podataka. Umjesto toga, nerelacijske baze podataka koriste model pohrane koji je optimiziran za specifične zahtjeve vrste podataka koji se pohranjuju. ([17]) Četiri su bitna tipa nerelacijskih baza ([16]):

- *Ključ-vrijednost baze podataka* (engl. *key-value database*).
  Najjednostavniji tip nerelacijske baze, to je kolekcija ključ-vrijednost parova. Izrazito su skalabilne i sposobne spremiti jako velike količine podataka.

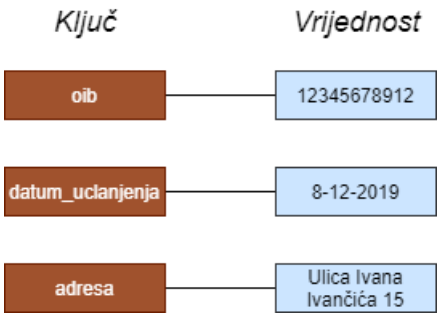
- *Dokumentske baze podataka* (engl. *document database*).
  Pretpostavlja se određena dokumentska shema. Najčešće su korištene među nerelacijskim bazama jer su dizajnirane da spremaju svakodnevne dokumente i jer dopuštaju kompleksne upite i operacije na svojem već agregiranom obliku podataka.

- *Stupčano orijentirane baze podataka* (engl. *column-oriented database*).
  Tradicionalne relacijske baze su orijentirane prema retcima, ali u nekim situacijama su češći i bitniji upiti koji zahtijevaju pretraživanje po stupcima. Na primjer, baza orijentirana prema stupcima je efikasna u analizama i generiranju izvještaja, dok je baza orijentirana prema retcima bolja za same operacije nad bazom (na primjer, prodaja).

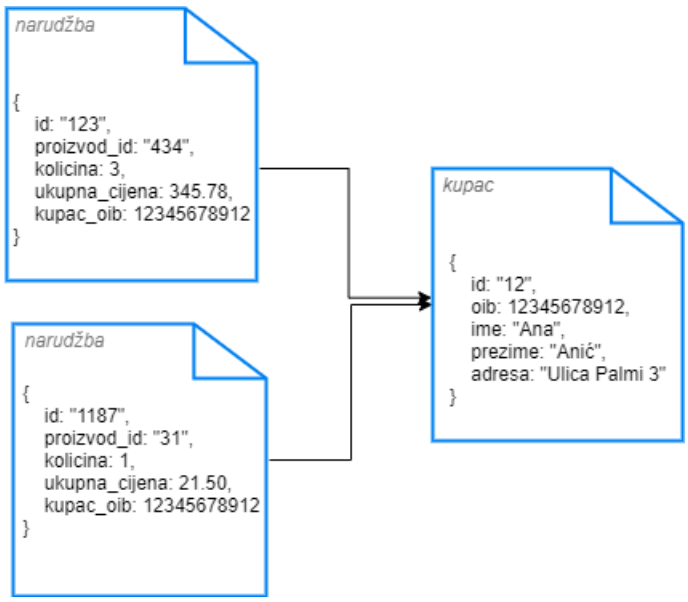
- *Grafovske baze podataka* (engl. *graph database*).
  Najkompleksniji tip među nerelacijskim bazama. Model podataka je graf. Efektivan je kod podataka u kojima postoji izrazito puno međupovezanosti, na primjer, u socijalnoj mreži. Graf, odnosno mreža podataka, sastoji se od čvorova (engl. *node*) i bridova (engl. *edge*). Ovisno o implementacijama, vrhovi i bridovi mogu imati oznake (engl. *label*) i svojstva (engl. *property*), a bridovi mogu biti i usmjereni.

Prva tri modela pripadaju agregatnom modelu podataka jer sadrže skupine podataka (dokumente, parove) – agregate. Zajedničko je svim četirima tipova baza to što ne koriste relacijski model. Nadalje, optimizirane su za određene vrste podataka i imaju specijalizirane načine izvršavanja upita (engl. *query*) nad takvim podacima. Na primjer, baze za spremanje podataka usmjerenih na praćenje vremena optimizirane su za vremenski utemeljene upite, dok su grafovske baze optimizirane za istraživanje težinskih veza između entiteta. Nijedan format ne bi dobro odradio zadatak upravljanja transakcijskim podacima. Bitan izraz vezan uz nerelacijske baze podataka je *NoSQL* i često se koristi kao sinonim za nerelacijske baze. Odnosi se na spremišta podataka koja ne koriste SQL za upite, već umjesto toga koriste druge programske jezike i konstrukcije za pretraživanje podataka. U praksi, ”NoSQL” znači ”nerelacijska baza podataka”, iako mnoge od ovih baza podataka

podržavaju SQL-kompatibilne upite. Međutim, temeljna strategija izvršavanja upita obično se jako razlikuje od načina na koji bi tradicionalni RDBMS izvršio isti SQL upit.



Slika 2.3: Shema ključ-vrijednost NoSQL baze podataka.

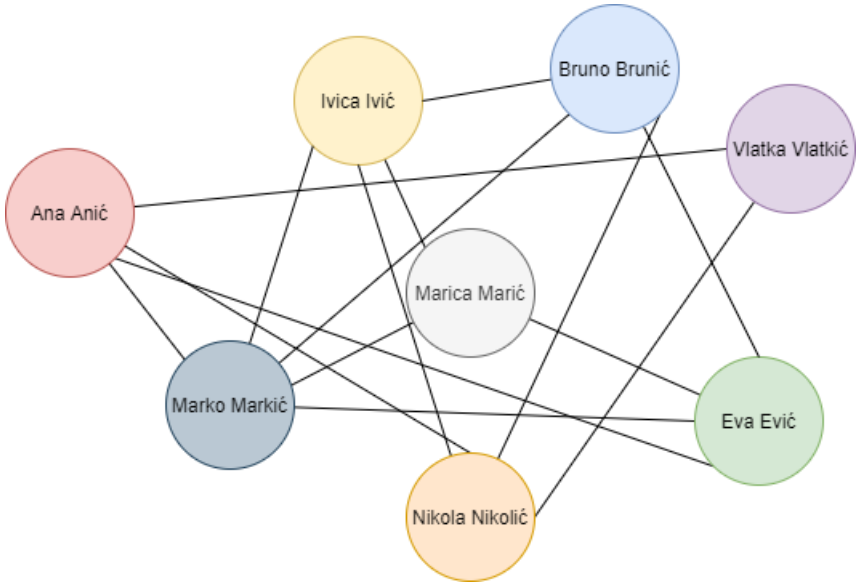


Slika 2.4: Shema dokumentske NoSQL baze podataka.

2.2.2.2 Priprema podataka za specijalizirane domene

Sljedeća faza u transformaciji podataka na podatkovnoj platformi je prilagoditi podatke konzumentima (korisnicima ili sustavima). Takvi prilagođeni podaci spremaju se u skladišta

Slika 2.5: Shema stupčano orijentirane NoSQL baze podataka.



Slika 2.6: Shema grafovske NoSQL baze podataka.

podatka ili u još prilagođenije *data mart*ove. Za podatke se može kreirati API pomoću kojeg će ih konzument dohvatiti. Također postoji mogućnost da se podaci direktno dohvate sa kategorije na koju ih je *publisher* poslao, bez tranformacija, a to rade *consumer*i koji su pokazali interes za tu kategoriju.

Skladište podataka (engl. *data warehouse*) je subjektno orijentiran, integriran, postojan i vremenski uvjetovan skup podataka koji služi kao potpora odlučivanju.

- *Subjektno orijentiran*. Podaci su organizirani oko subjekata od interesa, npr. student, predmet.
- *Integriran*. Podaci su integrirani iz više (heterogenih) izvora podataka, npr. relacijskih baza, i ujednačeni su (mjerne jedinice, oznake, tipovi podataka, ...).

- *Postojan*. Podaci su postojani, odnosno ne mijenjaju se od zadnjeg stvaranja skladišta podataka (jednom dnevno, tjedno, ...).
- *Vremenski uvjetovan*. Skladište je vremenski uvjetovano, svi zapisi imaju vremensku komponentu – svaki zapis vrijedi od nekog trenutka. Podaci se skoro uvijek analiziraju u vremenskom kontekstu (broj položenih ispita u godini, broj objavljenih radova u zadnjih 5 godina, prodaja u kvartalu, ...).

Skladište podataka je strateški sustav organizacije i predstavlja jedno mjesto s agregiranim informacijama (rijetko se bavi pojedinačnim zapisima).

Skladišta podataka obično imaju velike skupove podataka, ali analiza podataka zahtijeva lako pronalaženje i lako dostupne podatke. Treba li poslovna osoba obavljati složene upite samo radi pristupa podacima potrebnim za svoja izvješća? Ne – i to je razlog zašto pametne tvrtke koriste *data mart*ove, tzv. specijalizirana skladišta podataka.

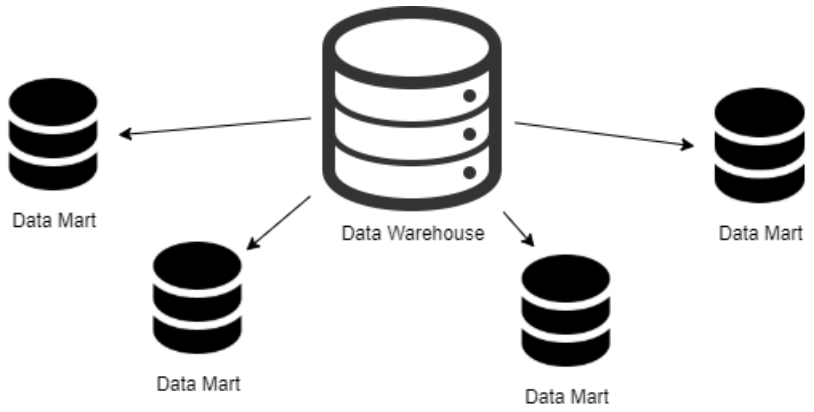
Data mart je subjektno orijentirana baza podataka koja je često samo odrezani segment skladišta podataka. Podskup podataka koji se nalazi u *data mart*u obično se usklađuje s određenom poslovnom jedinicom, poput prodaje, financija ili marketinga. *Data mart*ovi ubrzavaju poslovne procese omogućujući izrazito brži pristup važnim informacijama u skladištu podataka (s obzirom da je manji broj podataka za pretraživati, i manja potreba za filtriranjem). Budući da *data mart* sadrži samo podatke koji se primjenjuju na određeno poslovno područje, to je isplativ način za brzo stjecanje korisnih uvida. ([20])

*Data mart*ovi i skladišta podataka su strukturirana spremišta u kojima se podaci pohranjuju dok ne postanu potrebni za svoju svrhu. Međutim, oni se razlikuju u količini pohranjenih podataka: skladišta podataka izgrađena su da služe kao središnja pohrana podataka za cijelo poslovanje, dok *data mart* ispunjava zahtjev određene poslovne funkcije. Stoga je primarna svrha *data mart*a izoliranje manjeg skupa podataka iz cjeline kako bi se krajnjim potrošačima olakšao pristup podacima.

Čak i uz poboljšanu fleksibilnost i učinkovitost koje nude, *data mart*ovi su još uvijek preveliki za mnoga lokalna rješenja s tradicionalnim (*on-premise*) podatkovnim centrima i serverima fizički smještenima unutar poslovanja. Kako se skladišta podataka i jezera podataka premještaju u oblak, isto tako se premještaju i *data mart*ovi. Prednosti *data mart*ova u oblaku su:

- fleksibilna arhitektura
- jedinstveno mjesto koje sadrži sve *data mart*ove
- resursi se koriste na zahtjev (engl. *on-demand*)
- neposredni pristup informacijama u stvarnom vremenu
- povećana učinkovitost

- integracija resursa koja smanjuje troškove
- interaktivna analitika u stvarnom vremenu



Slika 2.7: Prikaz odnosa skladišta podataka i *data mart*ova.

Spomenuti *consumer* u poglavlju 2.2.1.3 dohvaća poruke (podatke) iz kategorija prema kojima je pokazao zanimanje. Ako u tim kategorijama ima bilokakvih poruka, tj. ako ih je *publisher* tamo poslao, *consumer* će sve te poruke dohvatiti. *Consumer* će dakle uzeti sirove poruke, tj. netransformirane podatke.

2.2.3 Infrastruktura podataka

Kvalitetna infrastruktura podataka, koja se gradi kroz cijelu podatkovnu platformu, za cilj ima omogućiti podatkovnoj platformi svojstvo *samoposluge*. Samoposluga podacima je svrha podatkovne platforme – da se konzumenti (korisnici ili sustavi) bez ikakve dodatne pomoći podatkovnih stručnjaka mogu poslužiti njima potrebnim podacima. Kao faktore koji sudjeluju u izgradnji infrastrukture podataka i koji potpomažu omogućavanju samoposlužnosti podatkovne platforme, opisat ćemo cjevovod podataka, registar shema, pohranu podataka, pristup podacima i katalog podataka.

Cjevovod podataka (engl. *data pipeline*) je softver koji eliminira mnoge ručne korake iz procesa i omogućuje nesmetan, automatizirani protok podataka s jedne stanice na drugu. Sve započinje definiranjem što, gdje i kako se podaci prikupljaju. Automatizira procese uključene u unos, tranformaciju, organizaciju i prijenos podataka za daljnju analizu i vizualizaciju. Omogućuje brzinu od kraja do kraja uklanjanjem grešaka i brigom o uskim

grlima (engl. *bottlenecks*) i kašnjenjima. Može obraditi više tokova podataka odjednom. Cjevovod podataka dijeli svaki tok podataka na manje komade koje paralelno obrađuje, čime osigurava dodatnu računalnu snagu. ([1])

Cjevovod podataka širi je pojam od ETL-a (*extract, transform and load*), obuhvaća ETL kao podskup. Odnosi se na sustav za premještanje podataka iz jednog sustava u drugi. Za razliku od ETL-a, podaci se mogu, ali i ne moraju transformirati, i mogu se obraditi u realnom vremenu umjesto serija. Osim toga, podaci se ne moraju učitati u bazu podataka ili skladište podataka kao u ETL-u, već na bilokakvo spremište podataka, poput AWS[4] *bucket*a ili jezera podataka.

Iako cjevovod za podatke nije nužan za svako poduzeće, ova je tehnologija posebno korisna onima koji:

- barataju velikim količinama podataka
- koriste više izvora podataka
- zahtijevaju analizu podataka u stvarnom vremenu
- spremaju podatke u oblaku

Navest ćemo listu nekoliko popularnih tipova cjevovoda podataka, s tim da se oni međusobno ne moraju isključivati, već se mogu kombinirati (npr. cjevovod u oblaku baziran na serijama podataka):

- *Cjevovod baziran na serijama podataka.* (engl. *batch*). Obrada u serijama je najkorisnija za premještanje velike količine podataka u pravilnom intervalu i kada ih nije potrebno premještati u stvarnom vremenu. Na primjer, ovo bi moglo biti korisno za integriranje marketinških podataka u veći sustav za analizu.
- *Cjevovod baziran na podacima u stvarnom vremenu* (engl. *real time*). Optimizirano za obradu podataka u stvarnom vremenu. U stvarnom vremenu korisno je obrađivati podatke koji *struje*, poput podataka s financijskih tržišta ili signala iz prometa.
- *Cjevovod u oblaku*. Optimizirano za rad s podacima koji se nalaze u oblaku. Alati iz cjevovoda se nalaze u oblaku i omogućuju uštedu novca na infrastrukturi i stručnim resursima jer se može pouzdati u infrastrukturu i stručnost pružatelja platforme, odnosno resursa, u oblaku.
- *Javno dostupan cjevovod* (engl. *open source*). Ovaj cjevovod je najkorisniji kada je potrebna jeftina alternativa komercijalnom prodavaču i ako korisnik posjeduje

[4] Amazon Web Services (AWS) produkt je Amazona koji pruža platformu i API za pristup resursima na Amazonovom oblaku.

stručnost za razvoj ili proširenje alata u svoje svrhe. Alati *open source* koda često su jeftiniji od svojih komercijalnih protivnika.

Registar shema (engl. *Schema Registry*) nudi zajedničko spremište shema koje aplikacijama omogućavaju fleksibilnu interakciju, odnosno pruža sloj za posluživanje metapodacima. Aplikacije obično razmjenjuju tri dimenzije metapodataka: format podataka, shemu, semantiku. ([3]) Oblikovni obrazac registra shema pruža način za rješavanje izazova upravljanja i dijeljenja shema između komponenti tako da su sheme dizajnirane da podrže evoluciju: *producer* i *consumer* moraju moći razumjeti različite verzije tih shema, dakle podaci se smiju razvijati s vremenom i moraju funkcionirati i sa starijim i sa novijim *producer*ima i *consumer*ima.

Na primjer, modeli koji se temelje na serijalizaciji podataka, pogotovo oni koji imaju za cilj prenosivost na različite platforme i jezike, oslanjanju se na shemu koja opisuje kako se podaci serijaliziraju u binarni format. Da bi serijalizirali podatke i zatim ih opet interpretirali, obje strane koje šalju i primaju moraju imati pristup shemi koja opisuje binarni format. Shema je uglavnom reprezentirana kao objekt koja mora sadržavati logičko ime, verziju i format sheme. ([27])

Dakle, vrijednosti koje pruža registar shema i aplikacije koje se s njim integriraju su sljedeće:

- *centralizirani registar*: shema treba biti dovoljno standardizirana tako da bude upotrebljiva na većem broju podataka, to jest da nije potrebna shema za svaki podatak
- *upravljanje verzijama*: uz definiran odnos između verzija sheme, *producer* i *consumer* se mogu razvijati različitim, neovisnim brzinama
- *validacija sheme*: omogućuje općenitu, neovisnu pretvorbu formata i kvalitetu podataka

Pohrana podataka (engl. *data storage*) opći je pojam za arhiviranje podataka u oblicima za upotrebu od strane računala. Novija opcija za daljinsku pohranu podataka je računanje u oblaku. ([29])

Pohrana podataka u oblaku (engl. *cloud storage*) je model pohrane podataka na računalu u kojem se digitalni podaci pohranjuju u logičke bazene. ([37]) Fizička pohrana obuhvaća više poslužitelja (ponekad na više lokacija), a fizičko okruženje obično posjeduje i njime upravlja pružatelj platforme za pristup oblaku. Ovi pružatelji usluga pohrane u oblaku odgovorni su za omogućavanje dostupnosti podataka cijelo vrijeme te za njihovu zaštitu. Ljudi i organizacije kupuju ili iznajmljuju kapacitet za pohranu podataka od pružatelja usluga za pohranu podataka. Uslugama pohrane u oblaku može se pristupiti pomoću web

sučelja ili programskog sučelja aplikacije za web uslugu (API). Pohrana u oblaku temelji se na visoko virtualiziranoj infrastrukturi i, poput cijelog računanja u oblaku, posjeduje pristupačno sučelje te gotovo trenutnu elastičnost i skalabilnost. Pohrana u oblaku sastavljena je od mnogo raspodijeljenih resursa, ali se predstavlja i vidljiva je kao jedan resurs. Visoko je otporna na kvar, a to omogućuju redundancija uređaja za pohranu i distribucija podataka. Prednosti pohrane u oblaku su:

- Korisnici trebaju platiti samo za pohranu koju stvarno koriste. To ne znači da je pohrana u oblaku jeftinija, već da ima operativne troškove, a ne kapitalne troškove.
- Tvrtke koje koriste pohranu u oblaku mogu smanjiti potrošnju energije do 70%, čineći ih zelenijim poduzećem.
- Trenutna dostupnost pohrane i zaštita podataka od izuzetne su važnosti za arhitekturu pohrane objekata u oblaku, pa se, ovisno o aplikaciji, može ukloniti dodatna tehnologija, napor i troškovi za osiguravanje dostupnosti i zaštite.
- Zadaci održavanja skladišta, poput kupnje dodatnog kapaciteta za skladištenje, prebačeni su na odgovornost pružatelja resursa u oblaku. Pohrana u oblaku pruža korisnicima neposredan pristup širokom rasponu resursa putem web sučelja usluga.
- Pohrana u oblaku može se koristiti za kopiranje slika virtualnog stroja (engl. *virtual machine image*) iz oblaka na lokalna mjesta ili za uvoz slike virtualnog stroja s lokalnog mjesta u oblak.
- Pohrana u oblaku može se koristiti kao sigurnosna kopija prilikom prirodnih nepogoda, jer obično postoji više različitih računala koja se nalaze na različitim mjestima širom svijeta.
- Pohrana u oblaku može predstavljati centralnog poslužitelja podataka za organizacije s više lokacija ureda.

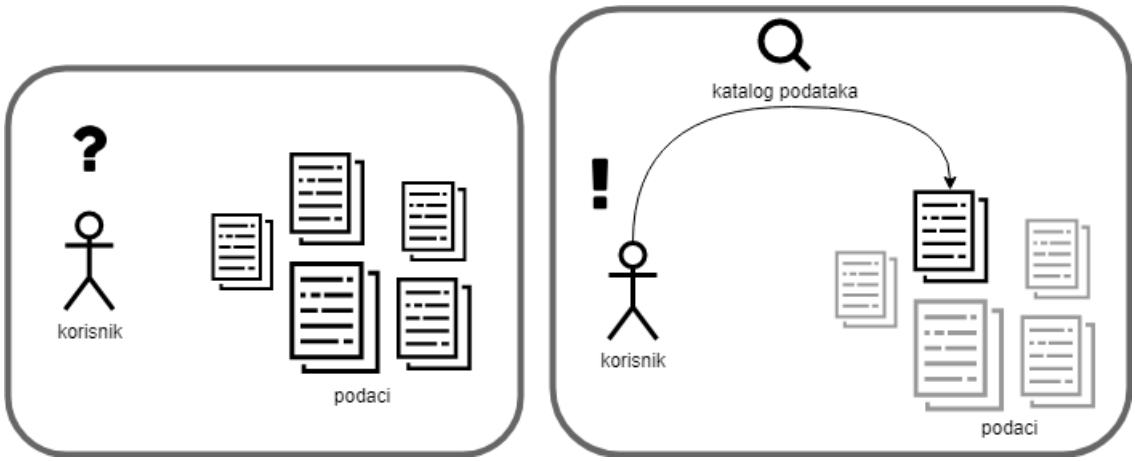
Pristup podacima (engl. *data access*) je ovlaštena sposobnost dohvaćanja, izmjene, kopiranja ili premještanja podataka iz sustava, u ovom slučaju iz podatkovne platforme. Pristup podacima je način na koji korisnici mogu doći do tih podataka, na vjerodostojan način odobren od organizacije koja ih posjeduje. Pravilno reguliran pristup podacima jedan je od glavnih rezultata dobro odrađenog upravljanja podacima (engl. *data governance*). Organizacije bi u idealnom slučaju trebale imati dobro osmišljene, strukturirane načine omogućavanja pristupa podacima različitim korisnicima. To je pojačano raznim dopuštenjima i razinama sigurnosti potrebnima za pristup podacima. Često se ta dopuštenja

temelje na organizacijskim ulogama ili odgovornostima te su strukturirana u skladu s politikama upravljanja podacima. U oblaku se obično koriste upravljanje identitetom i pristupom (IAM) i sigurnosne grupe za davanje pristupa korisnicima. ([15])

Katalog podataka (engl. *data catalog*) je detaljan popis svih podatkovnih dobara u organizaciji, dizajniran tako da pomogne podatkovnim stručnjacima da brzo pronađu najprikladnije podatke za bilo koju analitičku ili poslovnu svrhu. ([5]) Katalog podataka koristi *metapodatke* da bi se stvorio takav informativan i pretraživ popis podatkovnih dobara. Metapodaci mogu biti shema podataka, opis podataka, vrijeme nastanka, ... Kada govorimo o podatkovnim dobrima, to mogu biti strukturirani (tablični) podaci, nestrukturirani podaci (dokumenti, web stranice, sadržaj na društvenim mrežama, slike, video zapisi, ...), izvješća i rezultati upita, vizualizacije podataka, modeli strojnog učenja, ... Katalog podataka služi kao jedinstven, centralizirani izvor istine za podatkovne inženjere i podatkovne znanstvenike kako bi dobili *samoposlužni* pristup podacima u koje mogu vjerovati. On bitno olakšava upravljanje podacima (engl. *data governance*) te značajno dopridonosi kvaliteti podataka i aktivnom upravljanju politikama koje pomažu organizaciji da zaštiti i upravlja osjetljivim podacima. Omogućuje korisnicima podataka da ne moraju tražiti nikoga za pomoć kako bi pronašli potrebne podatke. Prednosti uvođenja kataloga podataka u podatkovnu platformu su:

- *Bolje razumijevanje podataka kroz poboljšani kontekst*. Analitičari mogu pronaći detaljne opise podatkovnih dobara i bolje razumjeti koliko su podaci relevantni za poslovanje.
- *Povećana operativna učinkovitost*. Katalog podataka stvara optimalnu podjelu rada između korisnika podataka i IT-a: korisnici mogu sami pristupiti i analizirati podatke, a IT osoblje se može usredotočiti na druge prioritetne zadatke.
- *Smanjeni rizik*. Analitičari imaju veće povjerenje kad rade s podacima za koje su ovlašteni upotrijebiti u određenu svrhu, pogotovo jer su podaci u skladu s industrijskim propisima i propisima o privatnosti podataka. Oni također mogu brzo pregledati napomene i metapodatke kako bi uočili nedostajuće ili netočne vrijednosti koje inače znatno utječu na njihovu analizu.
- *Veći uspjeh u upravljanju podacima*. Što je analitičarima teže pronaći, pristupiti, pripremiti i vjerovati podacima, to je manje vjerojatno da će njihove analize biti uspješne.
- *Bolji podaci i bolja analiza – brža konkurentska prednost*. Stručnjaci mogu brzo reagirati na probleme i izazove analizom i odgovorima na temelju prikladnih i brzo razumljivih podatkovnih dobara u organizaciji.

Platforme za pristup oblaku nude katalog podataka kao jedan od svojih resursa. Oblak tu ponovno dolazi u prednost zbog svoje skalabilnosti. Također, s obzirom na lakšu međupovezivost resursa u oblaku, platforme nude automatsko dodavanje u katalog podataka svih podatkovnih dobara koje vide u svojim bazama podataka ili drugim spremištima u oblaku. Na primjer, GCP nudi *Data Catalog* resurs, i opisuje ga kao "potpuno skalabilnu uslugu upravljanja podacima, koja je dio *Google Cloud*ove familije produkata *Data Analytics*". Također navodi da "*Data Catalog* automatski dodaje sve tablice iz *BigQuery*ija[5], te sve *topic*e iz *Pub/Sub*a[6]". ([7])



Slika 2.8: Usporedba kako se korisnik snalazi s podacima bez i sa katalogom podataka.

2.2.4 Korištenje podataka iz oblaka

Zadnja faza podatkovne platforme je upravo korištenje podataka dobivenih brojnim tranformacijama i obradama podataka da bi se kreirala podatkovna dobra. Cilj prethodnih faza je bio tako dobro organizirati podatke, da različiti konzumenti (podatkovni znanstvenici koji trebaju podatke za modeliranje u strojnom učenju, financijski stručnjaci koji trebaju izraditi financijski izvještaj, neki daljni ekspertni sistem, ...) u ovoj fazi mogu brzo i jednostavno pristupiti baš onim podacima koji su njima potrebni. Opisat ćemo česte konzumente/svrhe korištenja podataka.

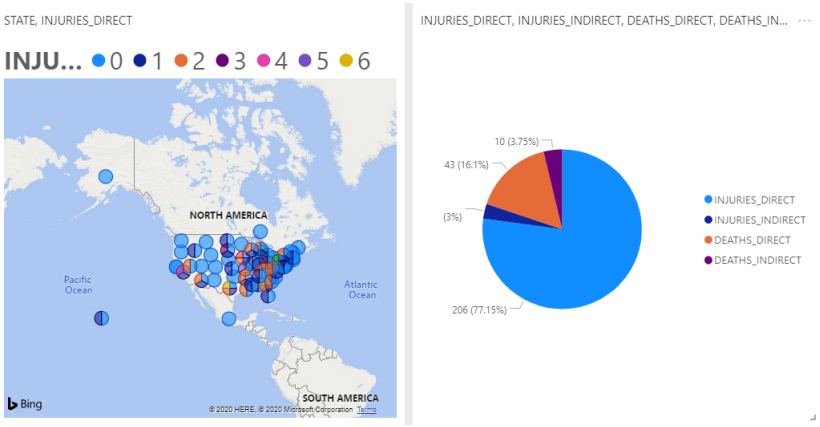
Poslovna inteligencija (engl. *business inteligence*) je skup metodologija, procesa, arhitektura i tehnologija koje transformiraju sirove podatke u smislene i korisne informacije

[5]BigQuery je skladište podataka na GCP-u.

[6]Pub/Sub je *publish-subscriber* tehnologija (opisana u poglavlju 2.2.1.3) na GCP-u. Topic je kategorija na GCP-u.

koje omogućavaju učinkovite strateške, taktičke i operacijske uvide i donošenje odluka. ([35]) Poslovna inteligencija nije proizvod, to je koncept. Glavni cilj je poboljšanje poslovanja. Poslovna inteligencija nisu samo alati, nego ljudi i znanja, ali i odluke. U hrvatskom jeziku pojam poslovne inteligencije može se još odnositi na sposobnost shvaćanja i brzog snalaženja neke tvrtke u novim uvjetima poslovanja. Neke od metoda poslovne inteligencije uključuju rudarenje podataka (engl. *data mining*), skladištenje podataka i OLAP obradu podataka (analitička obrada podataka).

Nadzorna ploča podataka (engl. *data dashboard*) je alat za upravljanje informacijama koji vizualno prati, analizira i prikazuje ključne pokazatelje, metrike i podatke za prilagođeno praćenje performansi poslovnog odjela ili određenog procesa. U pozadini se nadzorna ploča povezuje s podacima i API-ijima, ali na površini se svi ti podaci prikazuju u obliku tablica i grafova. Nadzorna ploča podataka najučinkovitiji je način za praćenje više izvora podataka jer pruža središnje mjesto za tvrtke koje prate i analiziraju svoja poslovanja. Praćenje u stvarnom vremenu smanjuje sate analize i dugu liniju komunikacije što inače zahtijeva iscrpan posao. ([11]) Interaktivne nadzorne ploče omogućuju korisniku da klikom na neki dio grafa, nadzorna ploča prikaže podatke za onaj djelić poslovanja kojeg taj dio grafa predstavlja. Iako su alati za kreiranje nadzornih ploča podataka zaživjele kao on-premise, lokalni alati, ovo je još jedan sjajan primjer adaptacije usluge u oblaku. Razlog prelaska ovakvih alata u oblak je što se na taj način izrađena nadzorna ploča može izrazito lako podijeliti sa šefovima ili strankama te oni na taj način mogu pristupiti nadzornoj ploči u bilokojem trenutku, a nisu ovisni o kreatoru nadzorne ploče i njegovom računalu.



Slika 2.9: Primjer dijela interaktivne nadzorne ploče u Microsoftovom vizualizacijskom alatu *Power BI* u oblaku.

Strojno učenje (engl. *machine learning*) je primjena umjetne inteligencije koja sustavima omogućuje automatsko učenje i poboljšavanje bez eksplicitnog programiranja, već

iz iskustva. Strojno učenje se fokusira na razvoj računalnih programa, tzv. modela, koji mogu pristupiti podacima i koristiti ih za učenje samih sebe. ([28]) Strojno učenje se dijeli na tri skupine: *nadzirano učenje*, *nenadzirano učenje* i *učenje s podrškom*. U sljedećem poglavlju pokazat ćemo prednosti treniranja modela neuronske mreže za klasifikaciju slika – u oblaku. Taj model pripada skupini modela nadziranog učenja.

Napredna analitika (engl. *advanced analytics*) krovni je termin za nekoliko potpolja analitike koje djeluju zajedno koristeći prediktivne mogućnosti. Napredna analitika koristi metode i alate na visokoj razini za predviđanje budućih trendova, događaja i ponašanja. To organizacijama daje mogućnost izvođenja naprednih statističkih modela kao što su predviđenja "što ako", te dokaze o raznim aspektima svog poslovanja u budućnosti. Napredna analitika uključuje i novije tehnologije kao što su strojno učenje i umjetna inteligencija, semantička analiza i vizualizacije. Zajedno, oni pomažu naprednom softveru za analizu podataka da stvori dovoljno precizno platno za pouzdano predviđanje i generiranje uvida poslovne inteligencije na dubljoj razini. ([25])

Streaming analitika (engl. *streaming analytics*), poznata i kao obrada protoka događaja, je analiza ogromnih skupova trenutnih i "u pokretu" podataka pomoću kontinuiranih upita, nazvanih *protok događaja*. Ove struje pokreće određeni događaj koji se događa kao izravni rezultat akcije ili skupa radnji, poput financijske transakcije, kvara opreme, klika na društvenoj objavi ili neke druge aktivnosti. Podaci mogu dolaziti iz Interneta stvari (IoT), transakcija, aplikacija u oblaku, web interakcija, s mobilnih uređaja i senzora stroja. Korištenjem platformi za streaming analitike organizacije mogu izvući poslovnu vrijednost iz podataka u pokretu, baš kao što bi im tradicionalni analitički alati omogućili da rade s podacima u mirovanju. Analitika strujanja u stvarnom vremenu pomaže nizu industrija uočavanjem prilika i rizika. ([4])

Transakcijski podaci (engl. *transactional data*), u kontekstu upravljanja podacima, su podaci zabilježeni iz transakcija. Transakcija je, u ovom kontekstu, niz razmjene informacija i srodnih poslova (poput ažuriranja baze podataka). Transakcijski podaci mogu biti financijski, logistički ili vezani za poslovanje, te uključuju sve, od narudžbe i statusa otpreme, preko radnog vremena zaposlenika, do troškova i potraživanja osiguranja. Transakcijski podaci se grupiraju u transakcijske zapise. Transakcijski podaci bilježe vrijeme i relevantne podatke potrebne za određeni transakcijski zapis. ([22])

Ekspertni sustavi (engl. *expert systems*) podrazumijevaju razne specijalizirane softverske sustave koji pomoću velikih količina podataka obavljaju razne zadatke za koje su specijalizirani, a to, na primjer, mogu biti financijski sustavi, prediktivni sustavi, marketinški sustavi, ...

Upravljanje sustavima pomoću podataka je sve modernija tehnika koja omogućava organizacijama da shvate ponašanje kupaca – u stvarnom vremenu. Struje podataka se automatski guraju prema dolje, odnosno prema krajnjim analitičkim platformama i koriste se u svrhu prepoznavanja bitnih značajki. U industriji se za ovaj proces koristi engleski termin

pushdown integration. Na primjer, klikovi kupaca u web trgovini se automatski prosljeđuju analitičkim sustavima na obradu kako bi odmah prepoznali trenutno najtraženiji proizvod i tako ga pravodobno i adekvatno reklamirali kupcima.

Poglavlje 3

Računanje s visokim performansama na platformi AWS

U ovom poglavlju pokazujemo studijski primjer u kojem testiramo performanse *računanja s visokim performansama*, i to na AWS platformi računanja u oblaku. Računanje s visokim performansama je izrazito napredan pojam koji zahtijeva veliku računalnu snagu. Kupiti, podesiti i međusobno povezati takve računalne resurse je veliki financijski i vremenski pothvat, stoga upravo s tehnologijom *oblaka* računanje s visokim performansama dobiva veliki zalet. Razlog tomu je što oblak nudi neograničene podešene računalne resurse, i to odmah na zahtjev. Pogotovo u situacijama kad korisnik planira upotrijebiti računanje s visokim performansama samo mali broj puta, nema smisla ulagati u on-premise strukturu. Stoga je ovo odličan studijski primjer upotrebe računanja u oblaku.

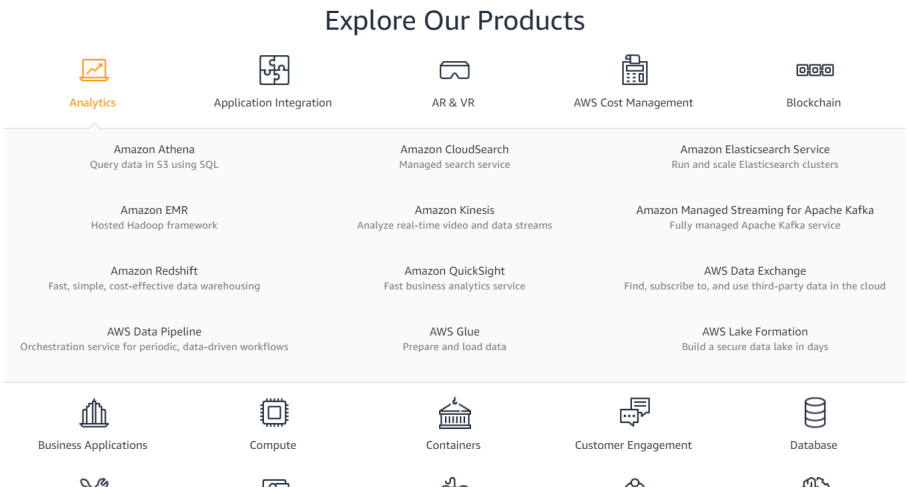
Veliki pružatelji usluga u oblaku (AWS, GCP) također prepoznaju veliki potencijal i budućnost računanja s visokim performansama. Ne samo da pružaju potrebne računalne resurse za uspostavu infrastrukture, već i sami pružaju cjelovitu, već spremnu infrastrukturu za računanje s visokim performansama – kao uslugu. Stoga nije potrebno podešavati resurse i međusobno ih povezivati, već samo konfigurirati uslugu i sve je spremno za korištenje.

U ovom poglavlju upoznajemo se s AWS-om i računanjem s visokim performansama, pokazujemo kako uspostaviti klaster računala koji će implementirati paradigmu računanja s visokim performansama, pokazujemo primjer pokretanja modela strojnog učenja na tom klasteru te konačno pokazujemo rezultate performansi.

3.1 Amazon Web Services (AWS)

Već smo opisali AWS kao Amazonov produkt koji pruža platformu i API za pristup resursima na Amazonovom oblaku. Smatra se prvom platformom za računanje u oblaku te

danas nudi preko 175 cjelovitih usluga iz svojih podatkovnih centara na globalnoj razini.



Slika 3.1: Isječak AWS-ovih usluga. Grupirane su u logičke skupine (analitika, baze podataka, itd.), a u svakoj skupini nalazi se pozamašan broj usluga. Na primjer, za analitiku vidimo usluge Amazon Redshift (skladište podataka) i AWS Data Pipeline (cjevovod podataka) – pojmovi opisani u poglavlju 2.

AWS-u se pristupa preko korisničkog sučelja u web pregledniku, a postoji i opcija korištenja Amazonovog API-ja kojim se usluge koriste iz terminala (na Linuxu), odnosno iz Command Prompta (na Windowsu).

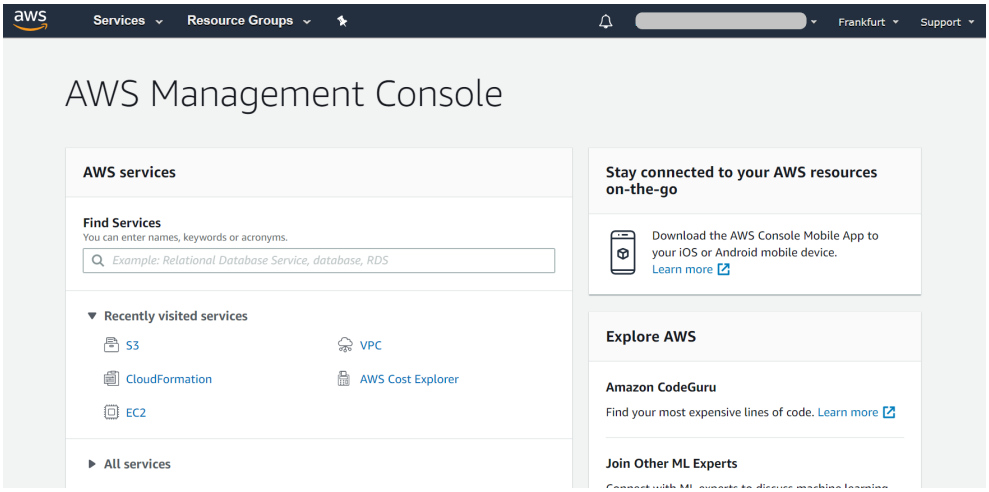
3.2 Računanje s visokim performansama (HPC)

Računanje s visokim performansama (engl. *High Performance Computing*, HPC) je uporaba superračunala i paralelnih tehnika obrade za rješavanje složenih računalnih problema. Superračunala su izuzetno moćna računala koja se koriste za najzahtjevnije obrade podataka.

HPC se obično koristi za rješavanje naprednih problema i obavljanje istraživačkih aktivnosti putem računalnog modeliranja, simulacije i analize koje zahtijevaju puno vremena.

Postoji nekoliko načina na koji se može paralelizirati posao tako da se zadaci izvode istovremeno:

- *pokretanjem programa s više procesa* (model SPMD (engl. *single program, multiple data*) gdje svi procesi obično izvode isti program, npr. s MPI-jem (engl. *Message Passing Interface*) koji predstavlja sučelje za razmjenu poruka između računala)



Slika 3.2: Korisničko sučelje AWS-a nakon prijave.

- *pokretanjem programa s više dretvi* (model zajedničke memorije koji kaže da paralelni procesi dijele globalno adresno područje iz kojeg čitaju i u kojeg pišu asinkrono, npr. koristeći OpenMP ili pthreads API-je za rad s dretvama)
- *pokretanjem nekoliko instanci programa s jednom dretvom* (tzv. sramotna paralelna paradigma koja definira istodobno pokretanje više zadataka)
- *pokretanjem jednog glavnog programa koji kontrolira nekoliko sporednih programa* (paradigma gospodara i roba (engl. *master-slave paradigm*))

Zašto HPC? Kada treba obraditi gomilu podataka i njihova obrada traje jako dugo, korisno je upotrijebiti pristup *divide et impera* (podijeli pa vladaj). Koristeći HPC može se podijeliti posao tako da svako računalo obrađuje drugi dio podataka, time ubrzavajući cjelokupan proces. HPC se realizira klasterom računala na kojima se željeni zadatak raspodjeljuje. Taj zadatak naziva se posao (poznatiji pod engleskim nazivom *job*).

Na primjer, uzmimo slučaj analize velikog skupa slika, točnije klasifikacije slika modelom strojnog učenja. Klasifikacija slika je poznata kao vremenski zahtjevan problem i može potrajati satima (čak i danima) u stvarnom poslovnom svijetu. Korištenjem HPC-a podijeli se skup podataka (slika) u nekoliko grupa i upravlja se s nekoliko računala tako da svako računalo obrađuje jednu grupu slika. Dakle, ideja je da ako je poslu potrebno 20 sati za obradu svih slika, da se upotrijebi nekoliko računala za dovršetak posla u dva sata.

3.2.1 Slurm

Kako bi se pokrenuo određeni posao na HPC klasteru i kako bi se njime upravljalo, potrebno je instalirati *menadžera za upravljanje HPC klasterom* (engl. *HPC cluster manager*) (naziva se još i *raspoređivačem poslova* (engl. *job scheduler*)). Postoji više različitih menadžera od više različitih proizvođača. Neki od primjera menadžera su AWS Batch, SGE, Torque, Slurm. Mi ćemo u ovom studijskom primjeru koristiti Slurm.

Slurm je jedan od vodećih menadžera rada HPC klastera širom svijeta. Slurm nudi *open-source* sustav koji je otporan na kvarove i visoko skalabilan te koji upravlja radnim opterećenjem i rasporedom poslova za male i velike Linux klastere. Slurm ne zahtijeva modifikacije jezgre operacijskog sustava za svoj rad i relativno je samostalan. Kao upravitelj rada klastera, Slurm ima tri ključne funkcije:

1. Korisnicima dodjeljuje pristup resursima (računalima) na neko vrijeme kako bi mogli pokretati posao.

2. Pruža funkcije za pokretanje, izvršavanje i praćenje posla (obično paralelni posao) na skupu dodijeljenih računala.

3. Upravlja skaliranjem resursa upravljajući nizom poslova na čekanju.

Infrastruktura klastera kojeg AWS kreira sa Slurmom kao HPC menadžerom klastera se sastoji od dva tipa računala. Jedno računalo se smatra glavnim računalom i ono se naziva *master* instancom. Ostala računala u klasteru su "radnička" računala te se nazivaju *compute* instancama. Te instance su u AWS-u realizirane kao virtualne mašine (računala, serveri), odnosno kao EC2[1] instance. Master je računalo kojem korisnik pristupa i na kojem pokreće i zaustavlja posao, provjerava stanje posla i stanje resursa u klasteru. Master instanca, kada korisnik pokrene posao, distribuira posao po compute instancama u klasteru. Compute instance su te koje obavljaju sam posao.

3.3 Postavljanje HPC klastera na AWS-u

Kao što smo spomenuli na početku ovog poglavlja, AWS nudi uslugu brzog postavljanja HPC okruženja u svom oblaku. Ta usluga se zove *AWS ParallelCluster*. AWS ParallelCluster je AWS-ov *open-source* alat za upravljanje klasterima koji olakšava postavljanje i upravljanje klasterima računanja s visokim performansama (HPC) na AWS-u. ParallelCluster koristi jednostavnu tekstualnu konfiguracijsku datoteku za automatizirano i sigurno pružanje svih resursa potrebnih za korisnikovu HPC aplikaciju. Također podržava razne

[1] EC2 je usluga na AWS-u koja pruža siguran, skalabilan računalni kapacitet u oblaku.

raspoređivače poslova kao što su AWS Batch, SGE, Torque i Slurm za jednostavno pokretanje poslova.

AWS ParallelCluster je objavljen putem Python Package Index (PyPI) repozitorija, dakle distribuiran je kao Python paket (biblioteka). Izvorni kod implementacije ParallelClustera pohranjen je na GitHubu AWS-a. AWS ParallelCluster dostupan je bez dodatne naknade, a plaćaju se samo AWS resursi potrebni za pokretanje korisnikovih HPC aplikacija.

Općenito, procedura postavljanja HPC klastera na AWS-u pomoću ParallelClustera i pokretanja posla sastoji se od sljedećih pet koraka:

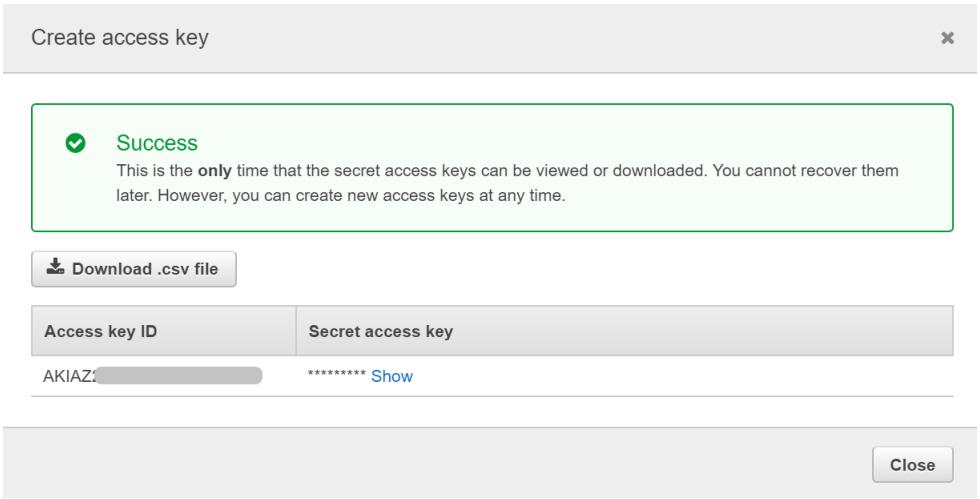
1. instalacija AWS ParallelClustera
2. dizajniranje svog klaster
3. kreiranje svog klastera
4. ulazak u svoj klaster
5. pokretanje posla

3.3.1 Instalacija AWS ParallelClustera

Rekli smo da se ParallelCluster koristi kao Pythonov paket. Stoga ćemo za kreiranje klastera pristupati AWS-u pomoću API-ja. Da bi to bilo moguće, potrebno je povezati lokalno računalo s korisničkim računom u oblaku. Za to služi AWS CLI (sučelje naredbenog retka), jedinstveni alat za programsko upravljanje AWS uslugama. AWS CLI se instalira na lokalno računalo te se konfigurira da odgovara korisniku, što stvori dvije nove datoteke: *credentials* i *config*. *Credentials* datoteka služi za autorizaciju korisnika te se za tu svrhu u nju pohranjuju *access key* i *secret access key*. Kao što korisničko ime i lozinka služe za autorizaciju na korisničkom sučelju, tako access key i secret access key služe za autorizaciju u slučaju pristupanja API-jem. Access key je dugoročna vjerodajnica za IAM ili root korisnika AWS računa. Secret access key služi kao neka vrste lozinke, to je tajni dio vjerodajnice. Access key se generira na AWS-u. Generiranjem access key-ja, automatski se generira i secret access key kojeg se može dohvatiti jedino tada, stoga ga se odmah treba pohraniti na sigurno tajno mjesto, poput lozinke. Ti key-jevi su sada vezani s korisničkim računom te kad ih se unese u *credentials* datoteku, lokalno računalo zna s kojim se korisničkim računom treba povezati.

Primjer *credentials* datoteke:

```
[default]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```



Slika 3.3: Kreiranje access key-ja i secret access key-ja.

U drugoj, *config* datoteci se spremaju manje tajni parametri:

```
[default]
region=us-west-2
output=json
```

Sada kada je lokalno računalo povezano s korisničkim računom u AWS-u, korisnik treba instalirati ParallelCluster naredbom

```
pip install aws-parallelcluster
```

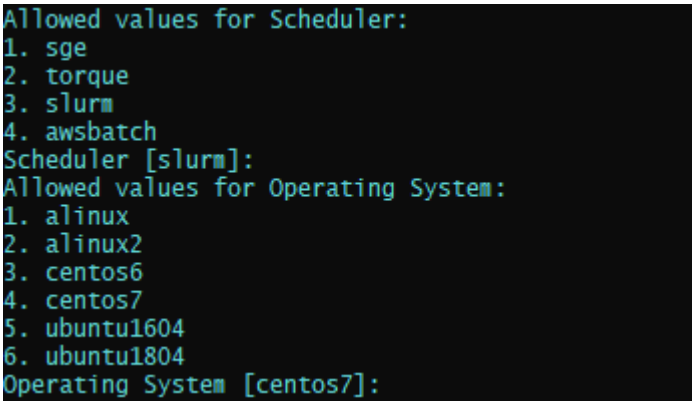
3.3.2 Dizajniranje svog klastera

U ovom koraku se odvija konfiguracija klastera prema vlastitim potrebama pomoću naredbe iz prethodno instaliranog *aws-parallelcluster* paketa:

```
pcluster configure
```

Ova naredba će pokrenuti interaktivnu konfiguraciju tako što će ponuditi niz pitanja s mogućim opcijama za odgovor, sve što je potrebno jest odabrati po jednu opciju za svako pitanje. Nude se čak i različiti HPC menadžeri/rapoređivači poslova koje će zatim AWS instalirati i pripremiti za korisnika bez ikakvog uloženog napora korisnika. Isječak interaktivne konfiguracije je prikazan na slici 3.4.

Tijekom konfiguracije potrebno je specificirati regiju u kojoj će se nalaziti resursi (mi smo odabrali centralnu Europu), EC2 par ključeva za pristup *master* instanci, menadžera



Slika 3.4: Isječak interaktivne konfiguracije klastera gdje korisnik treba odabrati HPC menadžera klastera i operacijski sustav instanci u klasteru.

HPC klastera, operacijski sustav, minimalni i maksimalni broj *compute* instanci u klasteru, tipove mašina *master* i *compute* instanci te VPC[2].

EC2 par ključeva, koji se sastoji od privatnog i javnog ključa, skup je sigurnosnih vjerodajnica koje se koriste za dokazivanje identiteta prilikom povezivanja s EC2 instancom. Amazon EC2 pohranjuje javni ključ, a korisnik privatni. Za siguran pristup instancama upotrebljava se privatni ključ umjesto lozinke. Kada korisnik želi ući u instancu, npr. SSH protokolom, zatražit će ga se EC2 par ključeva za autorizaciju pristupa.

Pri konfiguraciji klastera, korisnik mora odabrati broj compute instanci u klasteru. Ovaj dio pokazuje odličnu prednost oblaka – skalabilnost. Minimalan broj instanci predstavlja broj instanci za koje korisnik želi da budu stalno aktivne, alocirane i spremne. Te instance su aktivne dok god je klaster na životu. Korisno ih je imati jer kada korisnik pokrene određeni posao, ne mora čekati dok se instance alociraju. Ostale instance (do maksimalnog broja instanci) su rezervirane, ali nisu alocirane za korisnika sve dok korisnik ne zatraži posao koji zahtijeva veći broj instanci od minimalnog broja. Na primjer, ako korisnik često pokreće poslove za koje su potrebne samo tri compute instance, a ponekad pokreće i poslove za koje je potrebno deset instanci, korisnik će za minimalan broj postaviti broj tri, a za maksimalan broj 10. Tako korisnik ne mora plaćati aktivno vrijeme svih deset instanci, već samo za tri instance, ali kad god mu je potrebno svih 10, on će ih dobiti na zahtjev.

Još jedna prednost u oblaku je odabir hardvera (kroz odabir tipa mašine EC2 instanci) koji god korisniku odgovara. Amazon nudi veliki izbor tipova EC2 instanci, od običnog

[2]Virtualni privatni oblak (VPC) konfigurabilni je bazen dijeljenih računalnih resursa generiran na zahtjev koji se dodjeljuje unutar javnog oblačnog okruženja, pružajući određenu razinu izolacije između različitih korisnika koji koriste resurse. Izolacija između jednog VPC korisnika i svih ostalih korisnika istog oblaka (ostalih VPC korisnika kao i ostalih korisnika javnih oblaka) obično se postiže dodjelom privatnog IP-a.

tipa računala opće namjene kojeg svi koristimo, do tipova koji su optimizirani za računanje, za memoriju ili za ubrzano računanje (pa sadrže GPU-ove).

Korisnik kroz VPC može odabrati želi li opciju da su i master i compute instance na javnoj mreži (dakle da svaka ima dodijeljen javni IP kojem se može pristupiti), ili sigurniju opciju u kojoj je master instanca na javnoj mreži, a compute instance u privatnoj (pa im se može pristupiti samo "iznutra", tj. kroz master instancu).

Konfiguracija će kreirati *config* datoteku na lokalnom računalu koja će poslužiti kao predložak dizajna za kreiranje klastera.

3.3.3 Kreiranje svog klastera

Nakon podešenih konfiguracija, kreiranje klastera se pokreće jednostavnom naredbom iz instaliranog *aws-parallelcluster* paketa:

```
pcluster create -c <config-datoteka> <ime-klastera>
```

Ova naredba će odraditi cijelo postavljanje klastera na temelju *config* datoteke stvorene u prethodnom odjeljku, a u tu svrhu će pokrenuti niz usluga na AWS-u. Kako bismo dočarali način na koji oblak radi, navest ćemo primjere usluga koje AWS pokreće kako bi ovakva infrastruktura funkcionirala: *CloudFormation Stack* (služi kao menadžer kreiranja klastera i pokreće sve potrebne resurse klastera i potrebne ovisnosti), *EC2 Instance* (master i compute instance), *EC2 Auto Scaling Group* (usluga za brzo skaliranje compute instanci), *EC2 Launch Template* (za specificiranje konfiguracije EC2 instanci, npr. tip mašine, operacijski sustav), *EBS Volume* (uređaji za pohranu koji se prikače na EC2 instance i kojeg one onda koriste kao fizički tvrdi disk), *DynamoDB table* (tablica u NoSQL bazi podataka za spremanje potrebnih metapodataka).

3.3.4 Ulazak u svoj klaster

Korisnik sve instancirane usluge i kreirane resurse iz prethodnog odjeljka može pronaći na AWS-u. Za pristup klasteru, korisnik treba pristupiti master instanci. To se može pomoću SSH klijenta, koristeći javnu IP adresu dodijeljenu master instanci koja se može naći na korisničkom sučelju AWS-a.

3.3.5 Pokretanje posla

Posao koji ćemo pokrenuti na HPC klasteru jest treniranje *Keras*ovog[3] modela strojnog učenja koji klasificira 60,000 slika rukom napisanih znamenki iz poznate baze podataka

[3] Keras je *open-source* biblioteka za rad s neuronskim mrežama napisana u Pythonu.

MNIST. Radi se o jednostavnom algoritmu konvolucijske neuronske mreže[4]. Distribuirat ćemo skup slika na compute instance tako da svaki compute klasificira jedna grupu slika.

Postoji specifičan problem kojeg treba riješiti prilikom uključivanja distribucije u model strojnog učenja. U modelima postoje neke varijable koje se moraju dijeliti među compute instancama, odnosno sve compute instance moraju ažurirati te iste varijable (npr. gradijent ili težine). Inače, ako svaka compute instanca izračunava svoj gradijent ili ažurira svoje težine, na kraju će se efektivno dobiti različit model na svakoj compute instanci. Dakle, compute instance moraju nekako komunicirati. U tu svrhu koristimo API koji omogućuje komunikaciju i upotrebu istih varijabli na compute instancama. U ovom studijskom primjeru koristit ćemo *TensorFlow*ov[5] *tf.distribute.Strategy* API i integrirati ga sa spomenutim Keras modelom.

Kao što je temeljito opisano na službenim stranicama TensorFlowa, *tf.distribute.Strategy* je TensorFlowov API za distribuciju treninga na više GPU-ova, računala ili TPU-ova. U ovom primjeru pokazujemo sinkronu distribuciju treninga s paralelizmom podataka na više strojeva (compute instanci), koristeći *MultiWorkerMirroredStrategy* i slijedeći službeni vodič TensorFlowa. *MultiWorkerMirroredStrategy* strategija stvara kopije svih varijabli modela na svakom stroju. Trening modela strojnog učenja će se izvoditi na CPU-ovima compute instanci.

Nakon što smo opisali postavljanje klastera i izgradnju modela, posljednji je korak konačno pokrenuti posao, tj. započeti treniranje modela. To se izvodi lako – pokretanjem *sbatch* skripte, koja je u osnovi prilagođena *bash* skripta. Sastoji se od dva dijela. Prvi dio skripte specifičan je za Slurm, on određuje parametre potrebne za alokaciju compute instanci i pokretanje posla. Drugi dio sastoji se od *bash* naredbi koje se inače normalno pokreću u terminalu. Ovo je primjer *sbatch* skripte koja nam je odgovarala:

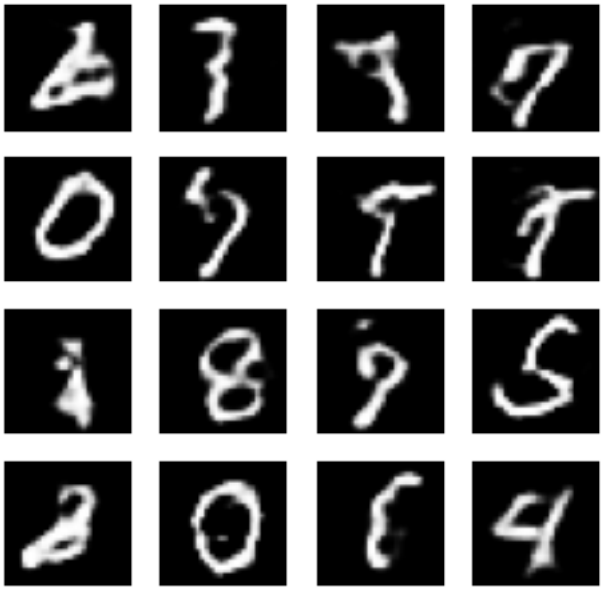
```
#!/bin/sh
#
#SBATCH --partition=compute
#SBATCH --job-name=keras-job
#SBATCH --nodes=4
#SBATCH --output=out-keras.txt

source $HOME/tf-venv/bin/activate
srun --nodes=4 python3 keras.py
```

Parametar `partition` je ime particije u koju je Slurm raspodijelio potrebne resurse, parametar `job-name` je ima posla kojeg pokrećemo, `nodes` je broj compute instanci na kojima

[4]Konvolucijska neuronska mreža je nadogradnja unaprijedne neuronske mreže. Osim od ulaznog i izlaznog sloja te jednog ili više skrivenih potpuno povezanih slojeva, sastavljena je i od konvolucijskih slojeva te slojeva sažimanja. Konvolucijski sloj sastoji se od skupa filtera kvadratnih dimenzija koje su manje od kvadratnih dimenzija ulazne slike. Konvolucijske neuronske mreže su postale vrlo popularne u prepoznavanju objekata na slici i pri klasifikaciji slika. ([21])

[5]TensorFlow je *open-source* biblioteka za strojno učenje.



Slika 3.5: MNIST slike rukom napisanih znamenki 0-9 koje će model strojnog učenja klasificirati.

ćemo pokretati posao, a `output` je datoteka u koju će se zapisivati izlaz programa kojeg bismo inače vidjeli u terminalu. Linije 1-6 predstavljaju prvi dio skripte, a linije 8 i 9 drugi dio. Linijom 8 se aktivira virtualno okruženje u kojem je instaliran TensorFlow, a linija 9 je Slurmova naredba `srun` za paralelno pokretanje Python koda (modela) *keras.py* na četiri compute instance.

Posao se pokreće Slurmovom naredbom:

```
1 sbatch <ime-sbatch-skripte>
```

3.3.5.1 Rezultati

U ovom studijskom primjeru pokazujemo rezultate testiranja na *dva* tipa mašina EC2 instanci. Jedan tip je `m4.large` (mašina za opću namjenu), a drugi je `c4.2xlarge` (mašina optimizirana za napredne računske zadatke, preporučena za HPC). Razlike između ta dva tipa se mogu vidjeti u tablici 3.1.

Rezultati testiranja su prikazani u tablicama 3.2 i 3.3. Primijetiti se može nekoliko činjenica.

Prokomentirajmo prvo tablicu 3.2, odnosno performanse izvršavanja posla na `m4.large` tipu compute instanci. Gledajući vrijeme treniranja modela na jednoj compute instanci i na

Instance	***Broj CPU-ova***	***Memorija (GiB)***	***Mrežne performanse***	***Cijena***
m4.large	2	8	umjerene	$0.12/sat
c4.2xlarge	8	15	visoke	$0.454/sat

Tablica 3.1: Razlike u specifikacijama između EC2 instanci `m4.large` i `c4.2xlarge`.

Broj compute instanci	***Vrijeme izvršavanje posla (u sekundama)***	***Loss (greška)***	***Accuracy (točnost)***
1	381.35	0.0035	0.9993
2	750.05	0.0156	0.9958
3	666.98	0.0282	0.9914
4	563.38	0.0369	0.9890
6	418.38	0.0541	0.9833
8	335.30	0.0599	0.9823
10	272.90	0.0712	0.9789

Tablica 3.2: Rezultati treniranja modela strojnog učenja na `m4.large` tipu compute instance.

Broj compute instanci	***Vrijeme izvršavanje posla (u sekundama)***	***Loss (greška)***	***Accuracy (točnost)***
1	112.34	0.0029	0.9994
2	158.69	0.0139	0.9963
3	140.53	0.0291	0.9911
4	110.21	0.0383	0.9883
6	84.26	0.0545	0.9838
8	65.28	0.0761	0.9774
10	55.65	0.0710	0.9781

Tablica 3.3: Rezultati treniranja modela strojnog učenja na `c4.2xlarge` tipu compute instance.

dvije compute instance, primjećujemo skok u vremenu koji se dogodi u treniranju modela na *dvije* compute instance. Razlog tomu je TensorFlowova implementacija strategije *MultiWorkerMirroredStrategy* u kojoj komunikacija među compute instancama i raspodjela posla ipak uzimaju značajnu količinu vremena. Kao posljedica toga, na manjem broju compute instanci nije moguće uočiti optimizaciju izvršavanja ovog posla na HPC klasteru. No, kako se broj compute instanci povećava, primjećujemo zadovoljavajuće smanjivanje vremena izvršavanja. Vidimo da što je broj compute instanci koje paralelno izvršavaju

program veći, to je vrijeme izvršavanja manje. Da bi se vremenski isplatilo distribuiranje posla na više `m4.large` instanci, potrebno je barem osam takvih compute instanci.

Prokomentirajmo i kako distribucija utječe na performanse samog modela, odnosno na njegove *loss* i *accuracy* metrike. Loss jedne slike se računa funkcijom kategoričke unakrsne entropije $Loss = -\sum_i y_i \cdot \ln \hat{y}_i$, gdje je $\hat{y}$ izlazni vektor modela s 10 koordinata (10 različitih kategorija za klasifikaciju – znamenke od 0 do 9) takav da svaka koordinata predstavlja vjerojatnost da slika pripada toj kategoriji. Zbroj svih koordinata mora biti 1. $\hat{y}_i$ je i-ta koordinata tog vektora. y je točna distribucija slike, odnosno vektor s jednom jedinicom (za kategoriju kojoj slika zaista pripada) i nulama, a y_i i-ta koordinata tog vektora. Što je klasifikacija modela točnija i sigurnija (dakle što je $\hat{y}$ bliži y), to je loss bliži nuli, odnosno model se manje kažnjava. Kako su vrijednosti $\hat{y}_i$ između 0 i 1, tako je prirodni logaritam negativan te se stoga stavlja minus. On je (po apsolutnoj vrijednosti) veći što je modelova klasifikacija $\hat{y}$ netočnija (to jest, što je manja vjerojatnost one kategorije kojoj slika zaista pripada). Na primjer, ako je na slici broj nula, slika pripada prvoj kategoriji, odnosno njezin vektor y ima jedinicu za prvu koordinatu, a sve ostalo su nule. No ako je model krivo klasificirao sliku i procijenio za prvu koordinatu vjerojatnost od samo 0.1 (npr. na drugoj koordinati je 0.9), tada se model kažnjava s velikim iznosom $\ln(0.1) = -2.3$. Accuracy je udio točno klasificiranih slika (u cijelom ulaznom skupu slika za treniranje modela). Loss treba biti što bliži nuli, a accuracy što bliži jedinici.
Možemo primijetiti da se s povećanjem compute instanci loss povećava, a accuracy smanjuje. Razlog tomu je statističke prirode – svaka compute instanca obrađuje manji skup slika te tu dolazi do gubitka preciznosti, odnosno do povećanja greške i smanjenja točnosti. Ovisno o slučaju, treba uzeti u obzir vrijedi li dobiti na vremenu, a izgubiti na točnosti modela. Iako je gubitak preciznosti malen, u nekim slučajevima preciznost je od velike važnosti.

Pogledajmo sada i tablicu 3.3, odnosno performanse izvršavanja posla na `c4.2xlarge` tipu compute instanci. I ovdje možemo primijetiti skok u vremenu izvršavanja posla na dvije compute instance. No, u usporedbi s `m4.large` tipom, ova mašina ima očekivano bolje vremenske performanse zbog boljeg hardvera. Izvršavanje samo na jednoj instanci brže je više od tri puta nego na `m4.large`, a skok s dvije instance je puno manji nego kod tipa `m4.large`. Osim što je svako vrijeme izvršavanja u tablici 3.3 bolje od odgovarajućeg vremena u tablici 3.2, primjećujemo i da se distribucija posla na `c4.2xlarge` instancama isplati već sa četiri compute instance.

Što se tiče loss i accuracy metrika, primjećujemo identično ponašanje kao u tablici 3.2 te zaključujemo da na performanse modela ne utječu različiti tipovi mašina, već utječu samo na vrijeme izvršavanja.

3.3.6 Brisanje klastera

Kako korisnik oblaka ne mora brinuti o svim ”pozadinskim” elementima, tako je i brisanje klastera jednostavno i automatizirano te se svi stvoreni resursi brišu jednom naredbom iz *aws-parallelcluster* paketa:

```
1 pcluster delete <ime-klastera>
```

Bibliografija

[1] Garrett Alley, *What is a Data Pipeline?*, 2018, https://www.alooma.com/blog/what-is-a-data-pipeline, posjećena 2020-08-07.

[2] Booking.com, *Što je API?*, 2020, https://partner.booking.com/hr/pomoc/channel-managerom/sto-je-api, posjećena 2020-08-04.

[3] Cloudera, *Schema Registry Overview*, https://docs.cloudera.com/csp/2.0.1/schema-registry-overview/topics/csp-schema_registry_overview.html, posjećena 2020-08-07.

[4] Databricks, *Streaming Analytics*, https://databricks.com/glossary/streaming-analytics, posjećena 2020-08-10.

[5] IBM Cloud Education, *Data Catalog*, 2020, https://www.ibm.com/cloud/learn/data-catalog, posjećena 2020-08-08.

[6] Thomas Erl, Ricardo Puttini i Zaigham Mahmood, *Cloud Computing: Concepts, Technology & Architecture*, Pearson, London, 2013.

[7] GCP, *Data Catalog overview*, https://cloud.google.com/data-catalog/docs/concepts/overview, posjećena 2020-08-09.

[8] Guru99, *What is Data Modelling? Conceptual, Logical, Physical Data Models*, https://www.guru99.com/data-modelling-conceptual-logical.html, posjećena 2020-08-04.

[9] If-Koubou, *Što je API?*, https://hr.if-koubou.com/articles/how-to/what-is-an-api.html, posjećena 2020-08-04.

[10] Chrissy Kidd, *What is a Canonical Data Model? CDMs Explained*, 2018, https://www.bmc.com/blogs/canonical-data-model/, posjećena 2020-08-06.

[11] Klipfolio, *What is a data dashboard?*, https://www.klipfolio.com/resources/articles/what-is-data-dashboard, posjećena 2020-08-10.

[12] Michelle Knight, *What is Data Curation?*, 2017, https://www.dataversity.net/what-is-data-curation/, posjećena 2020-08-06.

[13] Europska komisija Hrvatska, *Koliko nam IoT pomaže, ali i mijenja svakodnevnicu*, 2020, https://ec.europa.eu/croatia/How_IoT_is_helping_and_changing_our_everyday_life_hr, posjećena 2020-08-04.

[14] Robert Manger, *Baze podataka*, Element, Zagreb, 2012.

[15] Stacey McDaniel, *What is Data Access and (Why is it Important)?*, 2019, https://www.talend.com/resources/what-is-data-access/, posjećena 2020-08-07.

[16] Arno Meysman, *NoSQL Database Types*, 2016, https://dzone.com/articles/nosql-database-types-1, posjećena 2020-08-06.

[17] Microsoft, *Non-relational data and NoSQL*, 2018, https://docs.microsoft.com/en-us/azure/architecture/data-guide/big-data/non-relational-data, posjećena 2020-08-06.

[18] Project Nayuki, *What are binary and text files?*, 2015, https://www.nayuki.io/page/what-are-binary-and-text-files, posjećena 2020-08-04.

[19] Heritage Offshore, *Šifriranje, hashing, soljenje – u čemu je razlika?*, https://heritage-offshore.com/sigurnost-informacija/ifriranje-hashing-soljenje-u-emu-je-razlika/, posjećena 2020-09-03.

[20] Shana Pearlman, *What is a Data Mart?*, 2020, https://www.talend.com/resources/what-is-data-mart/, posjećena 2020-08-07.

[21] Domagoj Ravlić, *Interpretabilnost modela dubokog učenja*, Magistarski rad, Zagreb, 2019.

[22] Margaret Rouse, *What is transactional data?*, 2013, https://whatis.techtarget.com/definition/transactional-data, posjećena 2020-08-10.

[23] Margaret Rouse, *Definition: data ingestion*, 2016, https://whatis.techtarget.com/definition/data-ingestion, posjećena 2020-08-04.

[24] Stephanie Shen, *4 Design Principles for Data Processing*, 2019, https://towardsdatascience.com/4-design-principles-for-data-processing-964d6a45cb7c, posjećena 2020-08-06.

[25] Sisense, *Advanced Analytics*, https://www.sisense.com/glossary/advanced-analytics/, posjećena 2020-08-10.

[26] Softwise, *ERP i CRM*, 2012, https://soft-crm.net/hr_HR/blog/erp-enterprise-resource-planing-i-crm-customer-relationship-management, posjećena 2020-08-03.

[27] Spring, *Spring Cloud Schema Registry*, https://spring.io/projects/spring-cloud-schema-registry, posjećena 2020-08-07.

[28] Expert System Team, *What is Machine Learning? A definition*, 2020, https://expertsystem.com/machine-learning-definition/, posjećena 2020-08-10.

[29] Techopedia, *Data Storage*, 2012, https://www.techopedia.com/definition/23342/data-storage, posjećena 2020-08-07.

[30] Techopedia, *Data Scientist*, 2020, https://www.techopedia.com/definition/28177/data-scientist, posjećena 2020-08-06.

[31] Techopedia, *Data Architect*, https://www.techopedia.com/definition/29452/data-architect, posjećena 2020-08-06.

[32] Techopedia, *Data Engineer*, https://www.techopedia.com/definition/33707/data-engineer, posjećena 2020-08-06.

[33] Techopedia, *Definition: Asynchronous Data*, https://www.techopedia.com/definition/26893/asynchronous-data, posjećena 2020-08-04.

[34] Donald Tobin, *Data Transformation: Explained*, 2020, https://www.xplenty.com/blog/data-transformation-explained/, posjećena 2020-08-04.

[35] Wikipedia, *Poslovna inteligencija*, 2014, https://hr.wikipedia.org/wiki/Poslovna_inteligencija, posjećena 2020-08-10.

[36] Wikipedia, *Messaging pattern*, 2019, https://en.wikipedia.org/wiki/Messaging_pattern, posjećena 2020-08-04.

[37] Wikipedia, *Cloud storage*, 2020, https://en.wikipedia.org/wiki/Cloud_storage, posjećena 2020-08-07.

[38] Wikipedia, *Gateway*, 2020, https://bs.wikipedia.org/wiki/Gateway, posjećena 2020-09-03.

[39] Wikipedia, *Google Cloud Platform*, 2020, https://en.wikipedia.org/wiki/Google_Cloud_Platform, posjećena 2020-08-04.

[40] Wikipedia, *Publish–subscribe pattern*, 2020, https://en.wikipedia.org/wiki/Publish-subscribe_pattern, posjećena 2020-08-04.

[41] Nicola Wright, *IaaS vs SaaS vs PaaS: A guide to Azure cloud service types*, 2019, https://www.nigelfrank.com/blog/iaas-vs-saas-vs-paas-a-guide-to-azure-cloud-service-types/, posjećena 2020-08-29.

Sažetak

Tema ovog diplomskog rada je računanje u oblaku. U prvom poglavlju detaljno objašnjavamo pojam računanja u oblaku – što je to, zašto je nastalo, koje tehnologije i mehanizmi ga implementiraju, kako se isporučuje korisnicima te koje su njegove karakteristike, prednosti i mane. U sljedećem poglavlju opisujemo podatkovnu platformu u oblaku kao primjer kompleksnog proizvoda kojeg je poželjno implementirati u oblaku zbog prednosti opisanih u prvom poglavlju. U posljednjem poglavlju uvodimo studijski primjer računanja s visokim performansama na platformi u oblaku AWS. Postavljamo klaster virtualnih računala na AWS-u te na njemu distribuiramo treniranje modela strojnog učenja za klasifikaciju slika. Testiramo vremenske performanse u odnosu na broj virtualnih računala na kojima je treniranje distribuirano.

Summary

This master thesis deals with cloud computing. In the first chapter, we explain in detail the concept of cloud computing – what it is, why it did evolve, what technologies and mechanisms implement it, how it is delivered to consumers, and what are its characteristics, benefits, and risks. In the next chapter, we describe the cloud data platform as an example of a complex product that is desirable to implement in the cloud environment because of the benefits described in the first chapter. In the last chapter, we introduce a case study of High Performance Computing on the AWS cloud platform. We are deploying a cluster of virtual machines on AWS and distributing training of a machine learning model for image classification on it. We test time performance against the number of virtual machines on which the training is distributed.

Životopis

Rođena sam 21. prosinca 1995. godine u Ženevi, Švicarska. U Zagrebu sam završila osnovnu školu Horvati, nakon koje sam u istom gradu upisala IV. jezičnu gimnaziju na njemačkom dvojezičnom programu. Tijekom srednje škole shvaćam da je moj interes ipak usmjeren na matematiku te 2014. godine upisujem preddiplomski sveučilišni studij Matematika na Matematičkom odsjeku Prirodoslovno-matematičkog fakulteta u Zagrebu. Nakon preddiplomskog studija, odlučujem se za diplomski sveučilišni studij Računarstvo i matematika na istom odsjeku. Nakon prve godine diplomskog studija zapošljavam se u tvrtci Syntio, gdje upoznajem područje računanja u oblaku, te u njoj radim i tijekom pisanja diplomskog rada.