Uvod

Pojam, motivacija, i osnove CC kao koncepta

Sadržaj

Servisne informacije

Opšti podaci o predmetu.

Malo o predavaču

- Veljko Petrović
- Sedim u NTP 330 kancelariji
- Termin konsultacija je za sada Sreda u 16:15, odmah posle ovih predavanja.
- Uvek je dobro najaviti se na konsultacije — to može i na ovom času.
- Konsultacije mogu i po posebnom dogovoru kada je potrebno
- Možemo da se čujemo i online, naravno.

Kontaktiranje - elektronska pošta

- Najbolje me je tražiti na pveljko@uns.ac.rs
- Vodite računa da neretko dobijam čitav *tsunami* elektronske pošte, možda vašu ne vidim ili ne vidim na vreme.
- Ako ste zabrinuti da vam nisam video poruku, slobodno je šaljite opet.
- Ako želite da dobijete odmah odgovor (makar on bio samo 'video') molim stavite reč HITNO u subject poruke.

Šta je Računarstvo u Oblaku

- U pitanju je kalk engleskog 'Cloud Computing.'
- Očekujte da koristimo 'Cloud Computing' u velikoj meri ovde i još češće skraćenicu CC
- Kakav oblak?
  - Na mrežnim dijagramima se često spoljašnja mreža, najčešće internet, predstavlja kao stilizovani crtež oblaka.
- Odatle je oblak postao neka vrsta metonimije za računarske sisteme koji su, fizički, 'tamo negde' i povezani spoljašnjom mrežom.

Šta je cilj predmeta

- Cilj predmeta nije toliko da naučite da koristite nečiji servis CC-a
- Na tržištu ima neverovatan broj 'oblaka' i svaki ima svoje specifičnosti i svoju dokumentaciju
- Nas to ovde toliko ne zanima
- Umesto toga trudimo se da naučimo, koliko možemo, kako funkcionišu oblaci i opštije kako pisati sisteme koji su ekstremno distribuirani.

Literatura

- Dokumentacija, ovi slajdovi, i beleške sa predavanja bi trebali da budu sasvim dovoljni.

O slajdovima

- Slajdovi su novi za ovu generaciju i predstavljaju proširenu verziju slajdova ranijih godina
- Promenila se i tehnologija slajdova koji se sada generišu kao ovi interaktivni sajtovi. Mogu se, na zahtev dobiti i u drugoj formi: PDF, PPTX, ili Markdown izvorni kod
- Možete pratiti prezentaciju dok je slušate preko online verzije. Ovo bi trebalo da se nalazi na https://pveljko-ftn.github.io/predavanja-CC/01/

Kako se polaže?

- Predispitne obaveze nose 70 bodova
- Kako te bodove stičete ćućete na vežbama
- Ispit nosi 30 bodova
- Te bodove stičete na klasičnom ispitu koji će gotovo sigurno biti usmen i pokrivaće gradivo koje ste radili na predmetu.

Infrastruktura

- Budući da je predmet zahtevan, nemamo baš najbolju moguću podršku u laboratorijama fakulteta
- Trebaće vam `Linux,`` idealno najsvežija` Fedora `ili `Ubuntu` bilo direktno instaliran, bilo u virtuelnoj mašini.
- Dosta posla će se raditi iz komandne linije.
- Naš primarni jezik će uglavnom biti `Python``
- Najvažniji alati su Docker, Vagrant, i QEMU
- Sav naš alat će biti Open Source

Pogled na CC sa visine od 10km

O čemu u stvari govorimo?

Od čega počinjemo

- Najbolje se može razumeti šta je ova tehnologija osmišljena da reši ako se posmatra problem
- Zamislite da pravimo nekakvu aplikaciju koja radi online
- Da bi ta aplikacija radila potrebani su nam nekakvi računari na kojima se izvršava njen kod
- Prirodno rešenje je da kupimo X računara, i podesimo ih kako nam odgovara.
- Šta je problem ovde?

Problemi

- Koliko računara?
- Koliko vremena ti računari provode neupotrebljeni?
- Šta sa navalama?
- Gde su računari? Gde su korisnici?
- Šta ako deo računara otkaže? Šta ako nestane struja?

Problem Google-a

- Zamislite da morate da osmislite sistem koji služi da drži u životu glavnu stranicu internet pretraživača Google
- Pustite pretragu i sve te komplikacije: treba da samo prikažete logo, polje za unos, i par dugmića.
- Da li ima ijednog računara na svetu koji može da izdrži 95 000 zahteva za pretragom svake sekunde?

O čemu maštamo

- Da naši računari nikada ne zastarevaju i nikad se ne kvare.
- Da kad nema korisnika imamo malo računara, a kad je navala imamo mnogo.
- Da nam se vreme odziva glatko skalira bez obzira koliko zahteva imamo.
- Da računari nisu nigde posebno negu su nekako *svuda*.
- Da su računari imuni na nestanke struje ili elementarne nepogode.

Idejno rešenje

- Da li je moguće da se nekako okupe računari širom sveta, a da mi zakupimo ili na neki drugi način dobijemo pristup taman onom delu tih računara koji nam treba da bi izvršavali naš posao?
- *Baš to* je idejno rešenje računarstva u oblaku.

Tehničke prepreke

- **Spajanje.** Treba nekako da nam se mnogo računara ponaša kao jedan integrisan sistem.
- **Razdvajanje.** Treba nam nekako da jedan računar tretiramo kao mnogo računara.
- **Fleksibilnost.** Treba nam neki način da sistemu možemo dodati ili oduzeti računare dok on radi.
- **Konzistentnost.** Treba dan nam se svi podaci u rarzuđenom sistemu 'slažu'

Skica tehničkog rešenja

- Uzmemo brdo računara instaliranih u centre sa podacima širom sveta
- Koristeću tehnike *virtualizacije* i *kontejnerizacije* podelimo te individualne računare u mnogo, mnogo, *mnogo* virtuelnih malih računara.
- Kad name treba više računarske moći 'zavrtimo' još virtuenih malih računara i time skaliramo angažovanost sistema
- Ti računari se fizički nalaze na mnogim individualnim računarima širom sveta, te u slučaju prekida veze ili nestanka struje ili bilo kakve kataklizme, samo deo sistema je pogođen
- Koristeći kompleksne forme komunikacije i tehnologiju poznatu kao *load balancing.* postaramo se da zahtevi ka sistemu idu različitim čvorovima.

Skica tehničkog rešenja

- Koristeći kompleksan sistem čuvanja podataka koji podržava sinhronizaciju održavamo deljeno stanje unutar sistema
- Koristeći posebne tehnike projektovanja softvera kao što su, na primer, mikroservisi napravimo arhitekturu aplikacije kojoj se trivijalno dodaju novi čvorovi ili uklanjaju oni koji nisu potrebni.

"There is no cloud. There are only other people’s computers."

—*Važna istina, nepoznati autor.*

Alternative?

- Ovo što je do sada opisano je klasičano računarstvo u oblaku.
- Ovo nije jedini pristup problemu
- Ovo je optimizovano za jednu, fundamentalno centralizovanu, aplikaciju koja radi jednu stvar pod kontrolom jednog entiteta.
- Drugim rečima, iako je tehnički distribuiran, ovaj sistem je fundamentalno *centralizovan*.
- Kako izgleda decentralizovan cloud?

HPC naučna varijanta

- Neretko, naučnici se nađu suočeni sa računarskim problemom koji zahteva divovske kapacitete računanja ali se, srećom, jako lako paralelizuje.
- U tim situacijama, profitabilnim se pokazalo da se generiše za potrebe tog proračuna ogroman decentralizovan oblak
- Tu obični korisnici mogu, kada god žele, donirati procesorski kapacitet svog računara
- Ovo je oblak bez centralne organizacije (mada sa centralnim zadatkom!) i u kome se čvorovi nepredvidivo aktiviraju i deaktiviraju

Decentralizacija bez poverenja

- U svetu kriptografskih protokola za komunikaciju osobina 'bez poverenja' (odnosno trustless) je jako poželjna
- Ideja je da nema nijednog entiteta u sistemu kome se posebno veruje da će delovati u interesu sistema a ne svom interesu.
- Šta ako želimo 'trustless' oblak? Gde bilo koji proračun ili posao želimo da poverimo bilo kojoj skupini korisnika sistema bez potrebe da im verujemo?
- Ovo postoji u dve forme: forma zasnovana na heš-baziranim strukturama podataka i algoritmima konsenzusa, i forma kriptografskog prikrivanja komputacija

Heš bazirane strukture podataka i konsenzus

- Ovo je mnogo poznatije pod (blago netačnim) terminom *blokčejn*
- Ova tehnologija omogućava da se prozivoljni broj čvorova od kojih je (neki) broj možda nepošten složе oko stanja sistema, pravila za modifikaciju tog stanja, i izemna stanja po tim pravilma.
- Metod kojim se ovo slaganje postiže se zove 'algoritam konsenzusa.'
- Blokčejn je vama poznat isključivo kao *finansijska* tehnologija
- Otkud on ovde?

The World Computer

- Ako su vaši podaci deo stanja blokčejna, a vaša logika je deo tkzv. pametnih ugovora koji takav sistem napajaju, onda u velikoj meri blokčej predstavlja distribuiran, robustan, i bezbedan računarski sistem pristup kojem vi iznajmljujete na neverovatno granularan način.
- Blokčejn zna da naplaćuje uslugu na nivou individualne *instrukcije* i *bajta*.
- Dakle, blokčejn se može tretirati kao još jedna forma 'oblaka' mada neobična i često veoma skupa forma oblaka.

Kriptografsko prikrivanje informacija

- Veliki problem sa bločkejn pristupom (osim cene i problema komunikacije i Oracle problem i…) jeste problem tajnosti
- Zbog toga kako blokčejn radi, nemoguće je prikriti podatke koje on mora da obrađuje
- Ako bilo ko može da se pridruži, onda bilo ko mora da može da podatke i pregleda u celosti
- U slučaju klasičnog pristupa CC-u ne mogu *svi* da vide, ali vi implicitno verujete vašm provajderu usluga oblaka.
- Šta ako sui vaši prohtevi prema bezbednosti veći?

Kriptografsko prikrivanje informacija

- Ono što nam treba jeste nekakav način da odradimo proračun nad nekakvim podacima na takav način da entiteti koji nam rade taj proračun niti mogu da 'privire' u podatke koje obrađuju niti mogu da maliciozno vrate pogrešan rezultat.
- Postoji nekoliko pristupa ovome:
  - Potpuno Homomorfna Enkripcija
  - Enklave/Računarstvo od Poverenja
  - Dokazi Nultog Znanja (delimično)

Potpuno Homomorfna Enkripcija

- Poptuno Homomotfna Enkripcija, na engleskom Fully Homomorphic Encryption je tehnika za šifrovanja takva da, kada izvršite proračune nad šifrovanom formom podataka dobijete rezultat koji, kada se dešifruje, bude rezultat operacije nad podacima.
- Nešto formalnije $E(x)$ je homomorfno akko:

$$(\forall x \in D)(\forall f, f : D \to D)$$
$$E^{-1}(f(E(x))) = f(x)$$

Za neki domen D

Potpuno Homomorfna Enkripcija

- Praktičan sistem za FHE onda omogućava da šifrujemo naše podatke, pošaljemo ih nekome ko ih obradi po specifikacijama, i vrati obrađeno bez mogućnosti da zna šta su ulazi ili šta znači izlaz.
- To omogućava privatnost poruka, a i znači da je nemoguće 'namestiti' rezultat pošto bi to zahtevalo da se poruka adekvatno enkriptuje.
- Naravno, izazov je naći ovakvu šemu za enkripciju koja još podržava željene transformacije, i ne zahteva gotovo beskonačno računarsko vreme.

Enklave i Računarstvo od Poverenja

- Alternativa ovome jeste da se sistem implementira ne kroz kriptografski pristup, no kroz eksplicitnu hardversku podršku.
- Zamislite da računar koji iznajmljujete ima u sebi drugi računar, odvojen potpuno od prvog, i takav da ima interne podatke koji su potpuno tajni: u sebe integrisanu, tajnu memoriju.
- Recimo da je jedan od tih podataka par ključeva za asimetričnu kriptografiju takav da se ne može ekstrahovati.
- Recimo, takođe, da taj javni ključ poseduje potpis proizvođača što omogućava da se autentičnog uređaja validira.

Enklave i Računarstvo od Poverenja

- Onda bi mogli da kada želimo da zatražimo ključ takve 'enklave' u računaru, validiramo autentičnost, i njime šifrujemo svoj algoritam, podatke, kao i svoj javni ključ (koji šifrujemo da bi izbegli mogućnost MITM napada).
- Enklava izvrši sve što smo zatražili, radeći operacije interno (u tajnoj memoriji) a koristeći glavnu memoriju računara samo da čuva podatke koji su šifrovani ključem koji se čuva u tajnoj memoriji.
- Onda pošalje šifrovan odgovor nama.
- Ovo je naravno sporo, ali ako se glavni računar koristi za ne-ključne stvari, a enklava za bezbednosno-kritične stvari, dobar nivo performansi i bezbednosti su mogući.

Enklave i Računarstvo od Poverenja

- Prethodno je samo skica rešenja (stvarnost je kao i obično drugačija i kompleksnija, ali ovo prenosi centralnu ideju), i iz toga je već moguće videti dva problema sa ovim pristupom
- Prvo je da bilo koji bezbednosni sistem implementiran u hardveru može imati greške i ako su te greške teške dovoljno, garancije ovog sistema padaju u vodu trenutno.
- Drugo je da poverenje nije distribuirano no je centralizovano: u proizvođača procesora.

Dokazi Nultog Znanja

- Za razliku od dve prethodno navedene stavke, dokazi nultog znanja (eng. Zero Knowledge Proof) ne mogu da se koriste da u potpunosti reše problem.
- Uprkos tome, DNZ-ovi imaju svoju funkciju u konstrukciji ovih sistema.
- Centralna ideja iza DNZ-a jeste da jedna strana (koju možemo zvati 'dokazivač') može da demonstrira nekakvu činjenicu (recimo da zna nekakvu tajnu, ili da nekakav podatak koji zna ima nekakvu osobinu) nekome drugom (tu osobu možemo zvati 'revizor') na način koji odaje *nula znanja*.

Dokazi Nultog Znanja

- Primer primene DNZ metoda, doduše idealizovan, jeste da zamislite da dokazivač zna rešenje nekakve teške matematičke zagonetke.
- Revizor želi da kupi to rešenje, ali neće da plati dok ne vidi dokaz da dokazivač zaista ima validno rešenje.
- Naravno, dokazivač ne želi da podeli rešenje (što bi definitivno dokazalo da ga ima) zato što onda ne može da proda pristup njemu.
- Stoga, koristeći DNZ protkol može da dokaže da poseduje tajnu koja ima matematičku osobinu da rešava tu zagonetku, ali na takav način da ne oda ništa o toj tajni.

Dokazi Nultog Znanja

- DNZ ne može da se koristi sam po sebi na način kao što se FHE može koristiti ili enklave.
- Ono što može uraditi jeste, na primer, postarati se da nekakav eksteran skup validatora može da proveri proračun koji je provajder usluge proračuna obavio validan bez toga da u stvari mora da pristupi podacima proračuna.
- Ovakva kontrola pristupa se može uspešno koristiti da se implementiraju ekstremno distribuirani sistemi sa kompleksnim garancijama bezbednosti i privatnosti.

Klasično Računarstvo u Oblaku

Odlike i organizacija

Samousluga po potrebi

- Jedna ključna osobina klasičnog CC pristupa jeste samouslužnost po potrebi
- To znači, jednostavno rečeno, da klijent pružača te usluge je slobodan da odabere šta god želi od resursa, u zavisnosti od potrebe, i da to alocira za upotrebu
- Ovo se dramatično razlikuje od, recimo, kupovine fizičkog računarskog uređaja: takav uređaj ima određene sposobnosti i sve te sposobnosti su dostupne od samog početka.
- To što nešto nije potrebno ili što je potrebno više ne utiče na situaciju.

Mrežna orijentisanost

- Mreže i CC su vezane neraskidivo.
- Pristup CC servisima je uvek mrežni
- Najčešće, gotovo uvek, u stvari, to je pristup preko Interneta
- To nije neophodno: moglo bi se pristupiti i preko nekakvog direktnog linka.

Unificiranje resursa

- CC sistemi unificiraju resurse računarskih sistema
- Unificiranje ovde je dvostruka operacija koja obuhvata i grupisanje (recimo ako imamo stotinu diskova kapaciteta 2TB onda naš sistem u stvari ima 200 TB na raspolaganju) ali i razdvajanje (sistem može da razlikuje i posebno prati SSD diskove, hard diskove, ili hard diskove koji su normalno u stanju niske energije i ne rotiraju, hard diskove koji nisu ni zakačeni i moraju se robotski povezati, i podatke koji su na magnetnim trakama sekvencijalnog pristupa koje se moraju učitati u slučaju potrebe).
- Raznorodni hardver se objedini, a onda se po kvalitetu usluge i drugim osobinama ceo sistem podeli.

Unificiranje resursa

- Unificiranje resursa je neophodno ne bi li skaliranje i samousluga po potrebi bili mogući.
- Bez mogućnosti da se resursi objedine i rasloje po potrebi, granularnost alokacije bi bila ravna jednoj fizičkoj mašini nekakve, fiksne, konfiguracije.
- Ova unifikacije nije uvek stvar proste aritmetike: unificirana memorija dovodi do nepredvidivih ograničenja programiranja sa NUMA arhitekturama, unificiran disk multiplicira šansu gubitka podataka bez nekakvog mehanizma za redudansu…

Brzo i elastično skaliranje

- CC ne može da reši probleme ako je onaj samouslužni odabir resursa (koji su unificirani i raslojeni) po meri ne može da se menja u oba smera tokom izvršavanja nekakvog projekta.
- To je skaliranje: da komputacioni resursi odgovaraju dimenzijama problema
- Ako na naš servis bude navala, mi povećamo alokaciju onih resursa koji nam nedostaju.
- Elastičnost znači da kada imamo višak komputacionih resursa, ono što nam ne trebe se de-alocira (i ne plaća)

Merljivost i naplativost

- CC zahteva da se upotreba računara precizno meri ne bi li se jednako precizno naplaćivala
- Ta preciznost u naplaćivanju služi ne bi li pružila ekonomsku incentivu za upotrebu elastičnosti servisa
- Ako je cena ista bez obzira da li angažujemo 1 jezgro ili 10, nema incentive da se neupotrebljena jezgra ne koriste i dealociraju.

Merljivost i naplativost

- Sa jedne strane, individualne računarske instance se mogu naplaćivati po satu, minutu, ili sekundi rada
- Drugim rečima, naš sistem od CC provajdera dobija apstraktno-virtualizovane računarske sisteme nominalnih performansi i plaća za to u skladu sa garantovanim performansama.
- Sa druge, moguće je naplaćivati precizno procesorsko vreme, alokaciju memorije na nivou bajta, prostor na disku, i mrežni protok.

Istorija koncepta Računarstva u Oblaku

Kako smo ovde stigli?

Rani dani i centralizovano računarstvo

- U najranijim danima računarstva ništa *slično* računarstvu u oblaku nije bilo moguće
- Računari su bili deljeni resursi, ali je njihova podeljenost bila vremenska
- Programi su se izvršavali ne-interaktivno i obrađivali su se, jedan po jedan, i svaki je, dok se izvršavao, imao svu snagu računara (ne mnogo u tim vremenima)

Prva podela računara

- Prve podele računara su nastale sa povećanom željom za interaktivnim računarskim sistemima
- Prvi interaktivni računari su bile mašine kao što je TX-0 i PDP-1 koje su korišćene kao lični računari (deljeni na nivou sata upotrebe)
- To se pokazalo neefikasnim, a moćniji računari su sugerisali šanse da više ljudi koristi računar u isto vreme
- Prvi operativni sistemi koji podržavaju time sharing kako se to zvalo se pojavljuju

Rani mainframe računar

- Ovo je neizbežno dovelo do generacije ogromnih centralizovanih računara koji su uspešno obavili unificiranje resursa (zato što su se sve finansije slivale u kupovinu jedne maksimalno moćne mašine)
- Ovakav računar izvršava poslove više korisnika paralelno tako što deli procesorsko vreme na kratke vremenske segmente, a memoriju na zaštićene i disjunktne adresne prostore procesa.
- Sam računar se koristi kroz *terminale*, kombinacije tastature i monitora (u ranijim modelima, teleprintere) koji služe da se izdaju komande i čita rezultat.

Zanimljivo pitanje

Danas, znamo da se Cloud Computing ne obavlja na nivou procesa, no na nivou virtuelnih mašina.

Zašto?

Utility Computing

- Ovakav pristup koji manifestuje neke elemente Cloud Computing filozofije (resursi su unificirani pa deljeni, postoje elementi elastičnosti) je dove do generalizacije ideje u koncept koji se zove Utility Computing
- Ideja je bila da će postojati mrežno povezani masivni računari koji će svoje usluge naplaćivati bilo kome ko želi da ih koristi
- U vreme kada je ovo osmišljeno, samo su ogromne institucije mogle da plate da imaju računar - ovo je izgledao kao jedini smislen način da svako ima pristup računarskom vremenu.

Beg na lični računar

- Pojava mikroprocesora je pokvarila ovu viziju
- Prvi lični računari su bili više igračke nego ozbiljne mašine, ali to nije potrajalo
- U ovom novom dobu, velika količina procesorske moći je dostupna individualnom korisniku na individualnom računaru
- Napredak se sastoji u centralizaciji obrade na individualnom računaru i razmeni samo podataka preko mreže

Pojava oblaka

- Kako se povećavala upotreba broadband interneta, tako je bila sve veća potreba za podacima dostupnim odasvud i pojavila se veća potreba za servisima koji se pružaju preko Interneta.
- Ovo, sa druge strane, je dovelo do sve veće potrebe za održavanjem servera koji opslužuju zahteve
- Za korisnike ovo je bio momenat kada je oblak stigao, ali za provajdere tih usluga, ovo je i dalje bilo vreme individualnih serverskih soba za svaki servis

Migracija na oblak i totalna cena vlasništva

- Totalna cena vlasništva (eng. Total Cost of Ownership - TCO) je strahovito bitan faktor u komercijalnoj eksploataciji računarskih sistema
- TCO je cena nekakvog sistema koja uključuje *sve* elemente cene, ne samo koliko je koštao sistem i njegov softver, no i koliko košta struja, održavanje, periodična modernizacija, opravke…
- TCO može da pretvori neke naizgled jeftine stvari u neobično skupe
- Ispostavilo se da je, ekonomski gledano, TCO održavanja serverske sobe veoma nezgodan problem.

TCO Klasične serverske sobe

- Problem sa održavanjem sopstvenih servera jeste da neobično puno posla odlazi na menadžment konfiguracijom softvera, ažuriranjem verzija, održavanjem konfiguracija, i sličnim poslovima
- Čak i kada se to uzme u obzir, tu su takođe i troškovi kvarova i održvanja koji mogu, za veliku instalaciju, da postanu dramatični
- Konačno, istinski robusna serverska soba mora da bude spremna za razne egzotične scenarije u razna vremena dana i noći, što znači da verovatno mnogo skupe opreme i skupih eksperata sede naokolo u tri smene i ne rade, uglavnom, ništa.

Prednosti povećanja skale

- Sa druge strane, istinski *divovska* instalacija može da deli svu tu skupu opremu i eksperte između jako velikog broja računara što znači da je manje vremena traćeno i da se veliki troškovi isplate.
- Malo koja kompanija zaista mora da ima tako veliku instalaciju, ali one koje jesu imale su otkrile da višak mogu veoma lako da prodaju
- Ovo se naročito rascvetalo usled znatnog napretka tehnologija virtuelizacije o kojima više kasnije.

Razvoj računarstva u oblaku, prvi stadijum

- U ovom stadijumu postoji određeni broj fizičkih servera u serverskoj sobi i administratori (umesto da poslove razdvajaju kao procese na serveru) ih razdvajaju kao više raznih virtuelnih mašina.
- Ovo olakšava rad i efikasnije koristi resurse zato što omogućava da, npr. 'server za baze podataka' u stvari bez problema opslužuje više baza podataka nekompatibilnog tipa više projekata.

Razvoj računarstva u oblaku, drugi stadijum

- U ovom stadijumu sistem evoluira da bude mnogo više automatizovan
- Umesto da se VM-ovi stvaraju rukom, oni se konfigurišu i aktiviraju automatski
- Pristup podacima je potpuno softverski definisan gde se preko apstrakcionog sloja pristupa unificiranim resursima čuvanja podataka
- Mreža između računara je nezavisna od fizičke mreže između njih i softverski je definisana, formirajući virtuelne mreže između računara koji su fizički možda na istom procesoru, a možda na drugom kontinentu

Razdvoj računarstva u oblaku, treći stadijum

- U ovom stadijumu ide se ka apsolutnoj fleksibilnosti prema resursima
- Sve je elastično, sve je skalabilno, svaki zamisliv resurs je unificiran i može se kupiti 'na kilo'
- Da bi se ovo podržalo, aplikacije se konstruišu kao mikroservisi (više za sekundu) tako da aplikacije mogu da se skaliraju anizotropno
- U softverske sisteme se integriše više SaaS rešenja što povlači za svobom komunikaciju između oblaka

Mikroservisi?

- U mnogo čemu mikroservisi su adaptacija u dizajnu softvera na računarstvo u oblaku
- Imaćemo posebno predavanje o ovome, ali evo par detalja
- Mikroservisi su mehanizam kojim se aplikacija ne implementira kao jedan tehnološki stek nego kao međusobno komunicirajuća mreža nezavisnih mikro-aplikacija, svaka od kojih implementira nekakvu svoju funkcionalnost.
- Ova aplikacije mogu biti *potpuno* heterogene, a ne moraju biti

Mikroservisi?

- Mikroservisi su povezani tako da je stepen povezivanja onoliko labav koliko je to moguće i često se oslanjaju na kanale poruka, implementiraju service discovery mehanizme, i slične napredne tehnike organizacije komunikacionih linija aplikacije.
- Ideja je u tome da je svaki mikroservis, osim svog API-ja ka spoljašnjem svetu, svoja priča i da se ne samo može razvijati posebno nego i *skalirati* posebno
- Drugim rečima, ako je u aplikaciji navala na login sistem, onda nema potrebe da se skalira ceo sistem uključujući i logiku za filtriranje podataka ili tako nešo.
- Može se skalirati samo login mikroservis onoliko koliko je potrebno

Kraj

Hvala na pažnji!

Powered by  Slidev