Računanje bez servera i obrada događaja

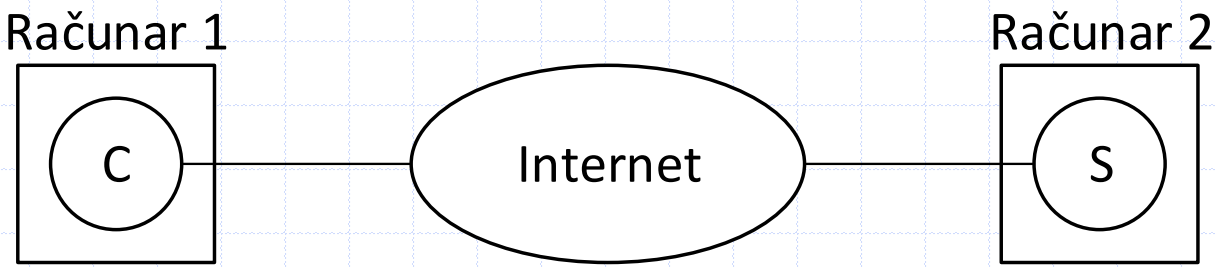
- Računanje bez servera
- Serveri bez stanja i kontejneri

Tradic. arhitektura klijent-server

- Klijent-server praksa deli apps u 2 kategorije:
  - Server: app koja se prva startuje i čeka kontakt
  - Klijent: app koja kontaktira server
  - Podela se odnosi samo na uspostavu veze
  - Nakon uspostave veze, dvosmerna komunikacija
- Uobičajena primena:
  - Klijent i server se smeštaju na 2 računara povezana na internet
  - Klijent sa rač. 1 kontaktira server na rač. 2, i time započinje njihova komunikacija – vidi sledeći slajd

Arhitektura klijent-server

- Klijent i server na 2 rač. povezana na internet:



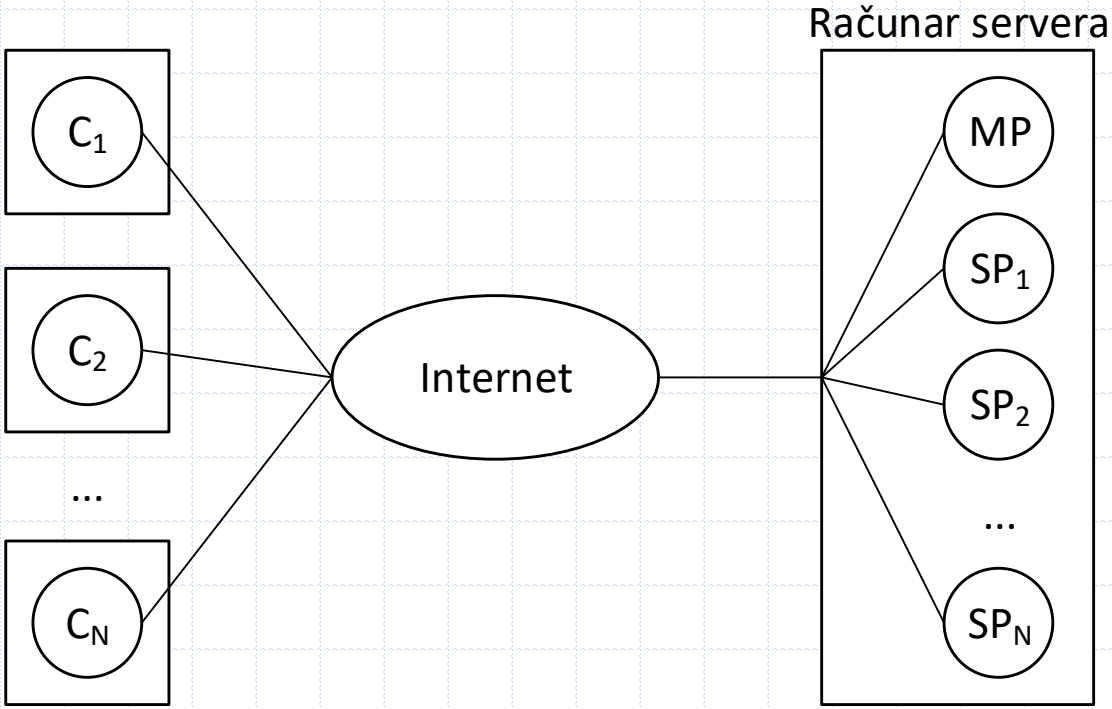
Legenda: C – klijent, S - server

Skaliranje tradic. servera da posluži više klijenata (1/2)

- Server čeka klijente, i pokreće procese za njih:
  - Ukupno N+1 procesa
  - Klijenti kontaktiraju glavni (master) proces (MP)
  - N konkurentnih procesa poslužuje N klijenata
  - Pravi se N+1 kopija jedne serverske app (fork poziv)
- U jednu app su integrisana 2 aspekta servera:
  - Ispunjavanje usluge koja se nudi
  - Repliciranje i skaliranje servera
  - I aspekt se odnosi na komunikaciju sa klijentom, a II aspekt su tehnike za repliciranje radi skaliranja

Skaliranje tradic. servera da posluži više klijenata (2/2)

- Rešenje skaliranja tradic. servera:



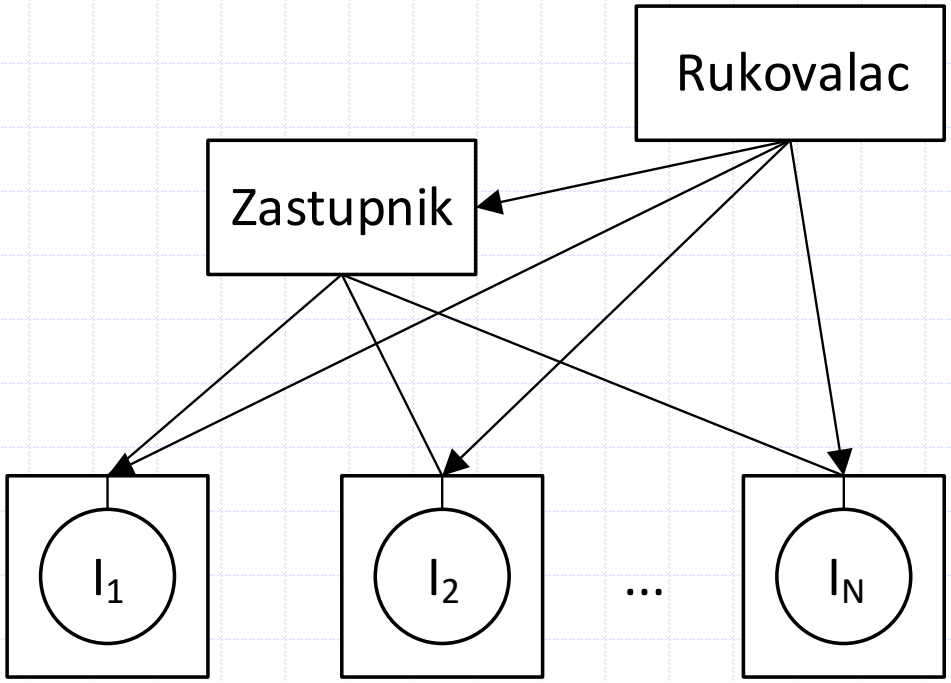
Legenda: C_i – klijent br. i, MP – glavni proces servera, SP_i – proces za klijenta br. i.

Skaliranje servera u oblaku (1/2)

- Ograničenja tradic. skaliranja:
  - Sve kopije servera moraju biti na istom fizičkom rač.
  - Neprimenljivo u oblaku gde su instance na raz. rač.
- Rukovalac u DC u oblaku:
  - Po potrebi raspoređuje instance (tj. kopije) na različite fizičke mašine
  - Aranžira da saobraćaj od klijenata bude usmeren kroz zastupnika do korektne instance

Skaliranje servera u oblaku (2/2)

- Rešenje skaliranja u oblaku:



Legenda: I_i – instanca br. i

Ekonomija servera u oblaku (1/2)

- Postoji puno tehnologija za rukovanje uslugama:
  - Sistemi za orkestraciju, zastupnici, balanseri opterećenja, rukovanje mrežom servisa, itd.
- U pogledu troškova:
  - Mnoge SW tehnologije su otvorene, tj. besplatne
  - Na prvi pogled se čini da bi korisnici oblaka mogli da postave i skaliraju servere uz mali dodatni trošak
  - Tj. da dovoljno imati osnovni SW, iznajmiti skup VMs, i iskoristiti otvoreni SW za raspoređivanje i skaliranje

Ekonomija servera u oblaku (2/2)

- Na žalost, 2 troška mogu biti značajna:
  - Nekorišćen kapacitet
  - Ekspertiza i treninzi
- Nekorišćen kapacitet:
  - Korisnik planira br. VM prema vršnom saobraćaju
  - Mora da plaća besposlene VM van vršnog saobra.
- Ekspertiza i treninzi:
  - Korišćenje otvorenog SW zahteva značajnu ekspertizu
  - Korisnik mora da zaposli eksperte
  - Plus mora da plaća treninge za zaposlene

Pristup računanju bez servera (1/2)

- Kao odgovor na potrebu za rukovanje serverima
  - Dobavljači su to ponudili kao uslugu pod nazivom:
  - Računanje bez servera (serverless computing)
- Smisao: korisnici ne vode računa o serverima
  - Ne moraju da konfigurišu mrežna imena i adrese,
  - Ne moraju da iznajmljuju VMs, itd.
- Sinonim koji je nastao da ukloni konfuziju:
  - Funkcija kao servis (Function as a Service, FaaS)
  - Jer korisnik piše kod za glavnu funkciju servera
  - A dobavljaču prepušta rukovanje serverima

Pristup računanju bez servera (2/2)

- Ovako korisnik smanjuje ukupan trošak:
  - Prvo, ne plaća za nekorišćeni kapacitet VMs (dobavljač pravi VMs samo kad su potrebne)
  - Drugo, dobavljač amortizuje troškove ekspertize i treninga, posebno za korisnike više oblaka
- Šta ako niko ne koristi server? Da li se plaća?
  - Neki dobavljači nude „skaliranje do 0“ (scale to zero)
  - Ako se ne koristi ni jedan server, ništa se ne plaća
- Pojam „skaliranje do beskonačnosti“:
  - Usluga računanja bez servera koja ne ograničava skaliranje, tj. može da ide preko očekivanja korisnika

Serveri bez stanja i kontejneri

- Kako se može postići skaliranje od 0 do ∞?
  - Tj. kako ostvariti i efikasnost i nisku cenu?
- Odgovor: kombinovanjem tehnologija
  - Za brzo raspoređivanje servera: kontejenri
  - Za rukovanje serverom: orkestracija
  - Za skaliranje: pristup na bazi kontrolera
- 2 ključne (nove) osobine računanja bez servera:
  - Korišćenje servera bez stanja
  - Oslanjanje na paradigmu pobude događajima

Korišćenje servera bez stanja (1/2)

- Stanje = klijentski podaci
  - Server sa stanjem (stateful) čuva te podatke
  - Server bez stanja (stateless) ne čuva te podatke
- Server sa stanjem čuva podatke iz 2 razloga:
  - Obezbeđivanje kontinuiteta u više kontakata
  - Moguće deljenje info između više klijenata
- Prvi pristup dobro radi za tradic. server
  - Jer on radi na jednom računaru
  - Više niti (threads) može koristiti deljenu memoriju
  - Info. o stanju se održava tokom mnogih kontakata

Korišćenje servera bez stanja (2/2)

- S druge strane, kontejneri su kratkotrajni:
  - Startuju, urade jednu f-iju, i izađu
  - Info o stanju ne traje duže od jednog kontakta
  - Kontejner izvršava jednu f-iju bez stanja (stateless)
  - Usluga FaaS (Function as a Service)
- Razlika između servera sa stanjem i DB servera:
  - Prvi čuva informaciju u operativnoj memoriji, koja nije perzistentna
  - Drugi čuva info. u spoljnem skladištu (DB, dat u NFS ili u objektnom skladištu)
  - Po potrebi, i server bez stanja može skladištiti info.

Oslanjanje na paradigmu pobude događajima

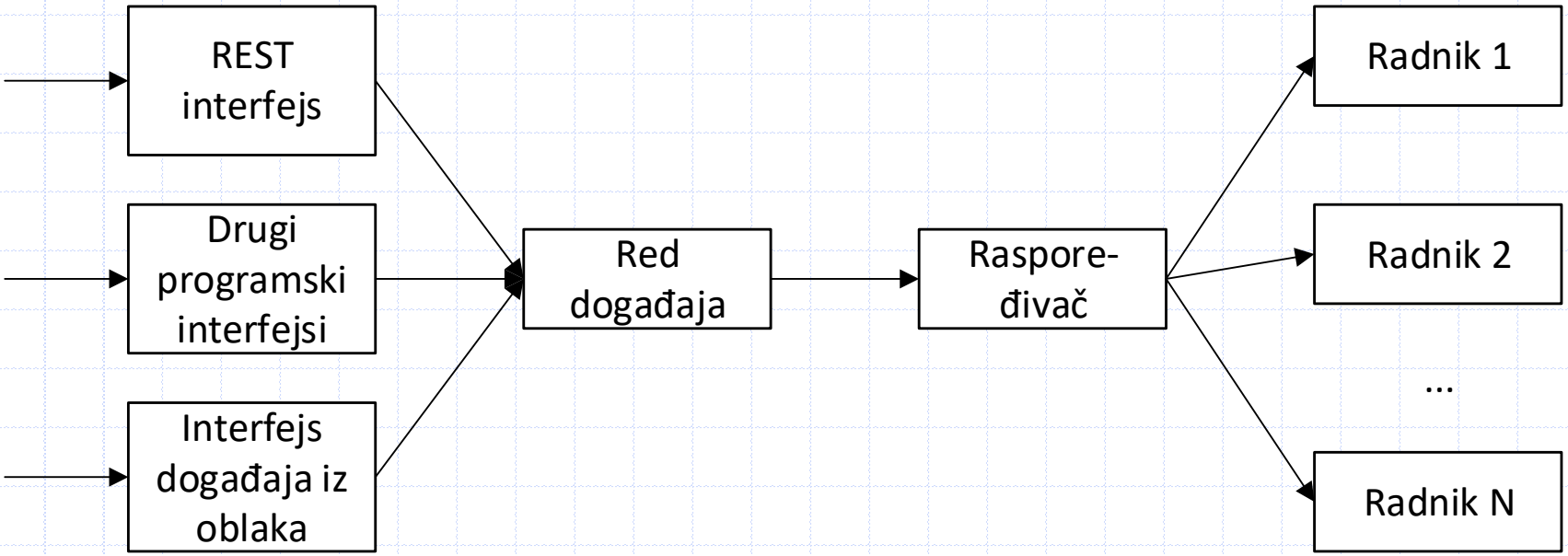
- Već smo videli da:
  - K8s kontroleri koriste paradigmu pobude događajima
- Računanje bez servera generalizuje tu paradig.
  - Kada dolazi do promena, oblak generiše događaje
  - Npr. kada fizički server otkaže
  - Rač. bez servera tretira svaki kontakt kao događaj
- Pored REST (na HTTP) prihvata druge interfejse
  - Putem kojih ljudi interaguju sa sistemom ili
  - Programi korsite interfejs koji je različit od HTTP
  - Kontakti sa svih interfejsa se tretiraju kao događaji

Arhitektura infrastrukture za računanje bez servera (RBS) (1/2)

- RBS usvaja K8s tehnologiju za kontrolere:
  - I koristi istu opštu arhitekturu
- Glavne komponente su:
  - Red događaja
  - Skup interfejsa, koji ubacuju događaje u red
  - Raspoređivač, koji raspoređuje događaje na radnike
  - Radnici, koji izvršavaju serverski kod

Arhitektura infrastrukture za računanje bez servera (RBS) (2/2)

- Arhitektura infrastrukture za RBS:



Primer obrade RBS (1/3)

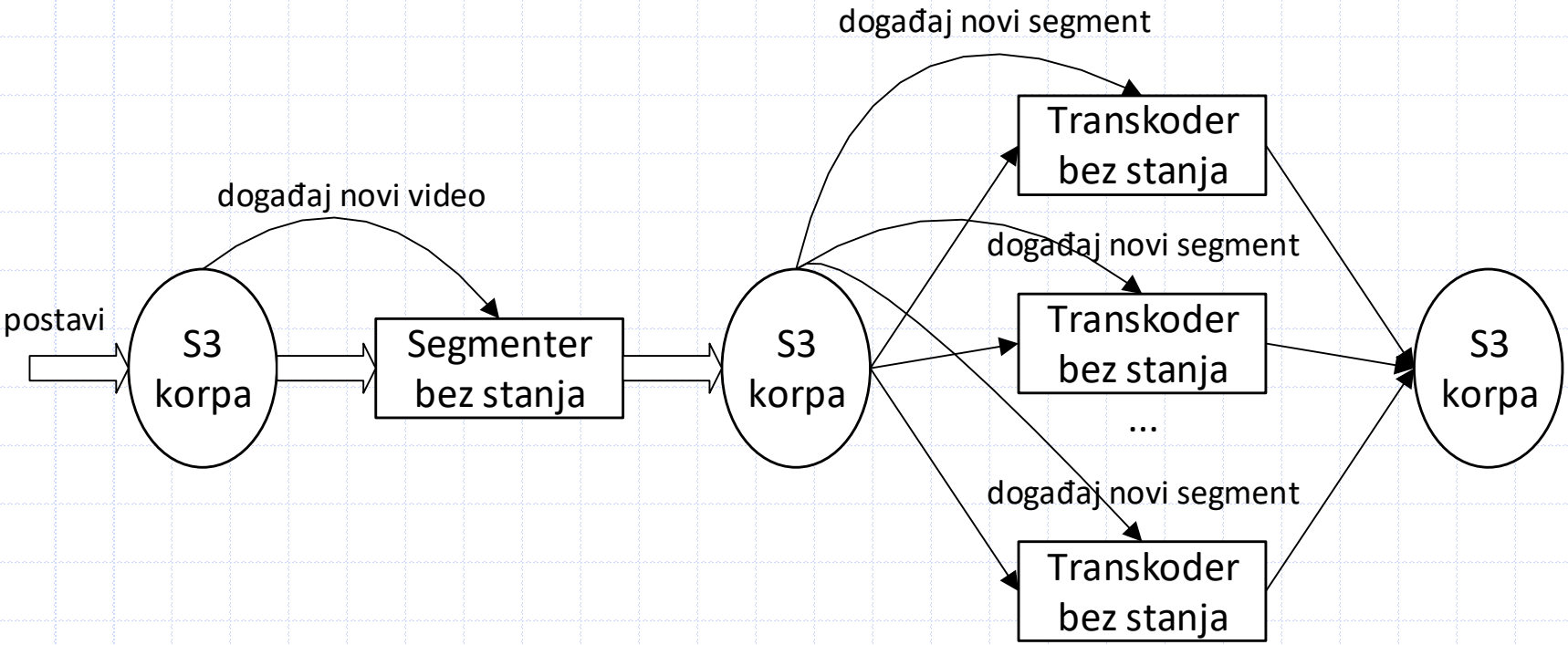
- Netflix koristi AWS Lambda
  - Za svoju video transkoding uslugu, koja priprema svaki novi video za preuzimanje (download)
- 4 koraka transkoding usluge:
  - Dobavljač sadržaja postavlja (upload) novi video
  - F-ija bez stanja deli video na 5 min segmente
  - Svaki segment se daje posebnoj f-iji bez stanja na obradu
  - Segmenti se skupe i video postaje dostupan korisnicima

Primer obrade RBS (2/3)

- Događaji pobuđuju svaki korak obrade:
  - Nov video se postavlja u AWS S3 korpu
  - AWS S3 korpa generiše događaj „nov video“, koji pobuđuje f-iju (bez stanja) za podelu na 5 min seg.
  - Upis segmenta u S3 generiše događaj „nov segment“, koji pobuđuje f-iju za transkodovanje segmenta
  - Transkodovanje segmenata se radi u paraleli

Primer obrade RBS (3/3)

- Netflix-ov sistem za transkodovanje videa:



Potencijalni nedostaci RBS (1/2)

- 3 glavne prednosti RBS:
  - Mogućnost proizvoljnog skaliranja
  - Nema potrebe za rukovanjem servera
  - Niži ukupni troškovi
- I nedostatak: kašnjenje
  - Tradic server startuje pre nego klijent inicira kontakt
  - RBS startuje server na zahtev, kašnjenje može biti nepovoljno za servere koji moraju da brzo reaguju
- II nedostatak: neočekivani troškovi
  - Pošto se naplaćuje svaka instanca, ako ima puno mikroservisa, trošak može biti značajan

Potencijalni nedostaci RBS (2/2)

- I primer neočekivanih troškova:
  - Program ima 4 f-ije: A, B, C, D
  - Za tipičan ulaz, lanac poziva A -> B -> C -> D
  - Slučaj kad su f-ije u posebnim instancama je 4x skuplji, nego slučaj kad se sve u istoj instanci
- II primer neočekivanih troškova:
  - X koristi Y, koji koristi Z
  - Ako Z otkaže, Y šalje upozorenje operateru i javlja X, X šalje upozorenje operateru – „kaskada upozorenja“
  - Može da dođe do „poplave“ događaja/upozorenja
  - Svako upozorenje – nova instanca, i novi trošak