Analiza zmogljivosti oblačnih in strežniških storitev

Uredil prof. dr. Miha Mraz

Maj 2019

Kazalo

Predgovor

Pričujoče delo je razdeljeno v deset poglavij, ki predstavljajo analize zmogljivosti nekaterih tipičnih strežniških in oblačnih izvedenk računalniških sistemov in njihovih storitev. Avtorji posameznih poglavij so slušatelji predmeta *Zanesljivost in zmogljivost računalniških sistemov*, ki se je v štud.letu 2018/2019 predaval na 1. stopnji univerzitetnega študija računalništva in informatike na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Vsem študentom se zahvaljujem za izkazani trud, ki so ga vložili v svoje prispevke.

prof. dr. Miha Mraz, Ljubljana, v maju 2019

Poglavje 1

Testiranje zmogljivosti Amazon EC2 platforme

Peter Matičič, Jan Pelicon, Blaž Rojc

1.1 Opis problema

Med programerji je veliko takšnih, ki sanjajo o tem, da bi bili naslednji Bill Gates, Mark Zuckerberg ali Steve Jobs. Imajo idejo, za katero verjamejo, da bo zavzela svet in jim prinesla miljone ter večno slavo. Ampak potrebujejo platformo, na kateri bo njihova storitev tekla. En sam prenosnik ne more vendar streči tisočem uporabnikom s celega sveta hkrati. Platforma mora biti cenovno dostopna, hkrati pa tudi poljubno razširljiva, da v primeru, ko se zgodi neizogiben povečan obisk uporabnikov, lahko programer enostavno in hitro aktivira dodatno procesno moč.

Tu nastopi Amazonov Elastic Cloud [1]. Obljublja dostopne cene, fleksibilno alokacijo računskih virov in za nadobudnega podjetnika najpomembnejša možnost uporabe določenih storitev brezplačno. Med temi storitvami je na voljo tudi najem tako imenovanih "mikro instanc" [2]. To so virtualni spletni strežniki z enim procesnim jedrom in 1 GB pomnilnika [3]. Predstavljajo minimalno konfiguracijo, ki lahko gosti poljubno spletno storitev, hkrati pa predstavlja tudi procesno ozko grlo, katerega omejitve moramo upoštevati pri tvorbi storitve.

S tem v mislih želimo stestirati platformo Amazon EC2. Ustvarili bomo enostavno storitev, ki bo uporabniku omogočala iskanje vzorcev v večjem naboru slik, shranjenih v oblaku. Predstavljala bo generično spletno aplikacijo, ki potrebuje ravno dovolj računske moči, da se bodo pojavile slabosti mikro instanc v obliki upočasnjenega ali onemogočenega delovanja na strani uporabnika. Osnova storitve je iskanje vzorca v naboru slik. Uporabnik od storitve zahteva podatek o tem, v katerih slikah se ta vzorec nahaja, pričakuje pa hiter

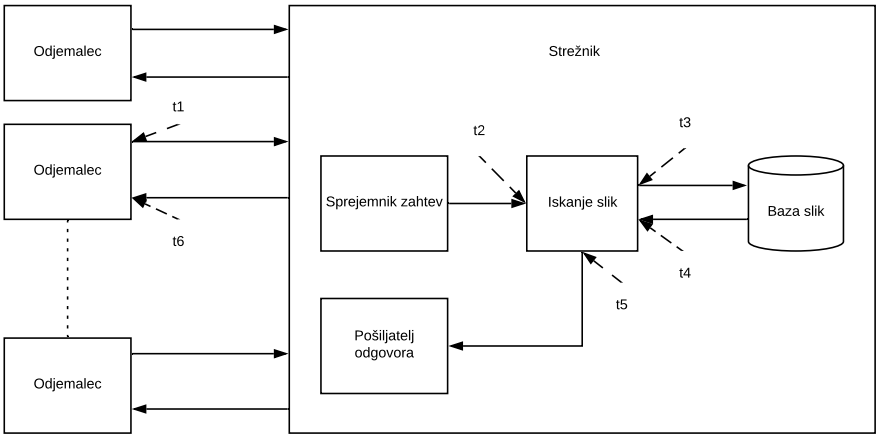
odgovor, z ne več kot nekaj sekundami zamika. Taka storitev nam bo omogočala relativno enostavno merjenje odzivnosti platforme, modularnost nalaganja kode in morebitno razširljivost v primeru večjega števila hkratnih uporabnikov.

1.2 Realizacija

Storitev je sestavljena iz dveh delov, in siccer strežnika na strani storitve EC2 in odjemalca na lokalnem računalniku. Opazujemo odzivnost strežnika, tako z meritvami na strežniku samem, kot pri odjemalcu.

1.2.1 Opazovano okolje

Aplikacija je realizirana kot spletna storitev, t.j. strežnik, dostopen na spletu, ki se odziva na zahteve uporabnikov. Zasnovana je po shemi na sliki 1.1. Uporabnik strežniku pošlje zahtevo v obliki JSON niza, ki vsebuje zaporedno številko zahteve, podatek o tipu zahteve in potrebne parametre. Strežnik zahtevo primerno obdela in vrne rezultat v obliki JSON niza, katerega oblika je odvisna od tipa zahteve.



Slika 1.1: Shema opazovane storitve.

1.2.2 Tipi zahtev

V osnovi vsi tipi zahtev vključujejo iskanje vzorca v naboru slik. Razlikujejo se v tipu vzorca, ki ga uporabnik želi najti v sliki. Storitev nudi tri tipe iskanj:

- iskanje specifične barve piksla,
- iskanje piksla, podobnega specifični barvi,
- iskanje slikovnega izseka.

Iskanje specifične barve piksla

Storitev prejme podatek o iskani barvi piksla, preišče vse slike v oblačni podatkovni bazi, dokler ne najde prvega ujemanja pojavitev iskanega piksla.

Iskanje piksla, podobnega specifični barvi

Poleg iskane barve storitev prejme od odjemalca še največje dovoljeno odstopanje. Preišče vse slike dokler ne najde prvega piksla, katerega barva se od iskane po komponentah razlikuje za največ toliko, kot določa odstopanje. Razlika se izračuna tako:

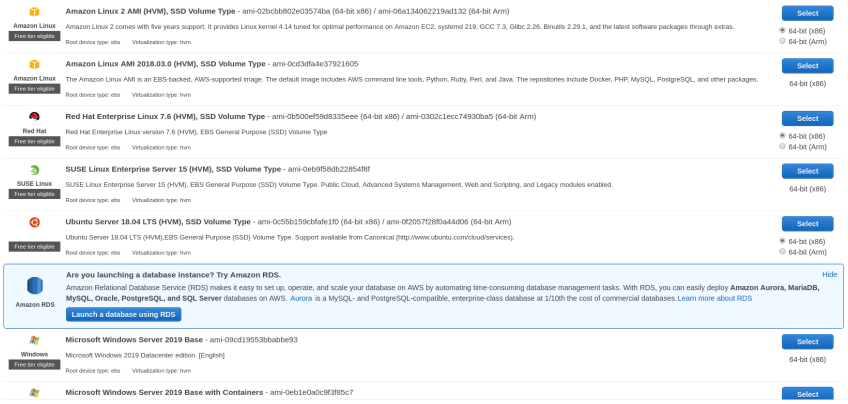
$$\begin{aligned} Razlika(RGB_{piksel}, RGB_{iskan}) = \\ = |R_{piksel} - R_{iskan}| + |G_{piksel} - G_{iskan}| + |B_{piksel} - B_{iskan}| \end{aligned} \quad (1.1)$$

Iskanje slikovnega izseka

Storitev prejme slikovni izsek. Preišče vse slike dokler ne najde prvega ujemanja podanega izseka z izsekom na neki sliki.

1.2.3 Amazon Web Services (AWS)

Za delo z Amazon Web Services [4] si moramo ustvariti račun v njihovem sistemu. Po uspešni registraciji si lahko ustvarimo svojo instanco EC2 storitve, ki nam jo Amazon ponuja zastonj za eno leto, pri čemer lahko na mesec porabimo največ 750 ur delovanja ponujenih instanc. Obsežnejša navodila so na voljo tudi na Amazonovi spletni strani [5], mi pa to naredimo tako, da se postavimo v AWS Management Console [6], kjer lahko izberemo opcijo *Launch a virtual machine with EC2*. Takoj nam konzola ponudi izbiro *slike navideznega diska* - vnaprej priravljenega nabora datotek, ki ga lahko neposredno zaženemo na instanci [7]. Izbor možnih slik diska je prikazan na sliki 1.2. Za naš projekt izberemo sliko Amazon Linux 2 AMI c.



Slika 1.2: Slike navideznih diskov za izdelavo virtualke

V naslednjem koraku izberemo tip instance slike, to je Amazonov način izbire paketov, ki vključujejo različne funkcionalnosti. Ker v našem primeru izbiramo brezplačno različico, nam ponujajo tip t2.micro, ki vsebuje 1 jedro, 1GB pomnilnika, nizko prioriteto hitrosti povezave do same instance strežnika (umetno omejena hitrost s spremenljivo največjo hitrostjo) in hrambo podatkov na platformi Elastic Block Storage, ki nam ponuja zastonjsko hranjenje podatkov na mrežno povezani shrambi. [8] Za tem lahko nadaljujemo z nastavljanjem različnih konfiguracij naše virtualke ali pa preprosto kliknemo Review and Launch, ki nam ponudi še en pregled izbranih nastavitev in zažene virtualko. Po zagonu virtualke nam sistem ponudi opcijo generiranja para ključev za varno SSH povezavo do nje. Ko zaključimo z ustvarjanjem, se premaknemo v EC2 management console kjer kliknemo na *instances*. Od tam lahko opazujemo status naše storitve in pridobimo tudi naslov, na katerem se nahaja. Za povezavo uporabimo javni naslov storitve, uporabnika $ec2 - user$, za varnost pa uporabimo *.pem* datoteko (Privacy Enhanced Mail Security Certificate), ki smo jo v prejšnjem koraku prenesli. Ko se uspešno povežemo na storitev, lahko pričnemo z razvojem naše aplikacije.

1.2.4 Aplikacija

Aplikacija je napisana v programskem jeziku Java, ki je interpretirana preko sprotnega prevajalnika, kar se lahko potencialno izkaže kot ozko grlo. Sestavljata jo odjemalska in strežniška komponenta, ki uporabljata skupne enumeratorje za določanje tipa zahteve. Zahteve sestavlja odjemalec in jih pošlja strežniku, ki te zahteve obdela - začne iskanje v slikah in pripravi prvi najden rezultat. Povezava med strežnikom in odjemalcem je trajna, dokler je eden od njiju ne prekine, kar nam omogoči, da pri meritvah ne upoštevamo časa vzpostavitve povezave. Podatki se prenašajo v obliki JSON, ki vsebuje sekvenčno številko zahteve, tip zahteve in morebitne dodatne podatke, ki so potrebni za obdelavo. V izpisu 1.1 je predstavljen primer zahteve v obliki dokumenta JSON, ki jo odjemalec pošlje strežniku.

Listing 1.1: Primer JSON zahteve

```
{
   "reqId": 1,
   "reqType": "PIXEL_NEAR",
   "pixelValue": "0xFFFA3881",
   "maxDistance": 86,
   "image": "",
   "req_start": 1555162220550,
   "err": ""
}
```

Strežnik ob prejemu podatkov začne z delom na ustrezni zahtevi ter nato pošlje odgovor klientu s številko zahteve in rezultatom. Odgovor vsebuje tudi čas proceiranja in branja slik. Strežniški del je napisan tako, da lahko paralelno obdeluje več zahtev. V izpisu 1.2 je predstavljen primer odgovora na zahtevo, ki ga strežnik vrne odjemalcu.

Listing 1.2: Primer JSON zahteve

```
{
   "reqId": 1,
   "imageId": 4,
   "location": {
      "x": 124,
      "y": 87
   },
   "proc_time": 4528,
   "req_start": 1555162220550,
   "image_fetch_time": 78,
   "err": ""
}
```

1.3 Uporabljene metrike

Kot glavno metriko bomo opazovali skupni čas zahteve in odgovora.

Podrobneje ga bomo razdelili na čas dostopa do datotečnega sistema (zbirke slik v mapi na datotečnem sistemu, recimo ji baza slik) ($t_{baza} = t_4 - t_3$) in celoten čas obdelave na strežniku ($t_{streznik} = t_5 - t_2$). Hkrati merimo celoten čas trajanja zahteve ($t_{zahteva} = t_6 - t_1$) (časi označeni na sliki 1.1). Z izmerjenimi časi lahko izračunamo tudi druge kot sta čas procesiranja na strežniku ($t_{procesiranje} = t_{streznik} - t_{baza}$) in čas paketa na mreži ($t_{mreza} = t_{zahteva} - t_{streznik}$). S tem smo se znebili problema sinhronizacije ur med odjemalcem in strežnikom. Če bi želeli meriti tudi čas potovanja paketa od od-

jemalca na strežni in čas potovanja paketa od strežnika na odjemalec, pa bi potrebovali tudi sinhronizacijo ur, ker pa naš cilj ni meriti čase potovanja paketov, saj je to namreč lastnost omrežja in ne same platforme, bomo zadovoljni s skupnim časom paketa na mreži.

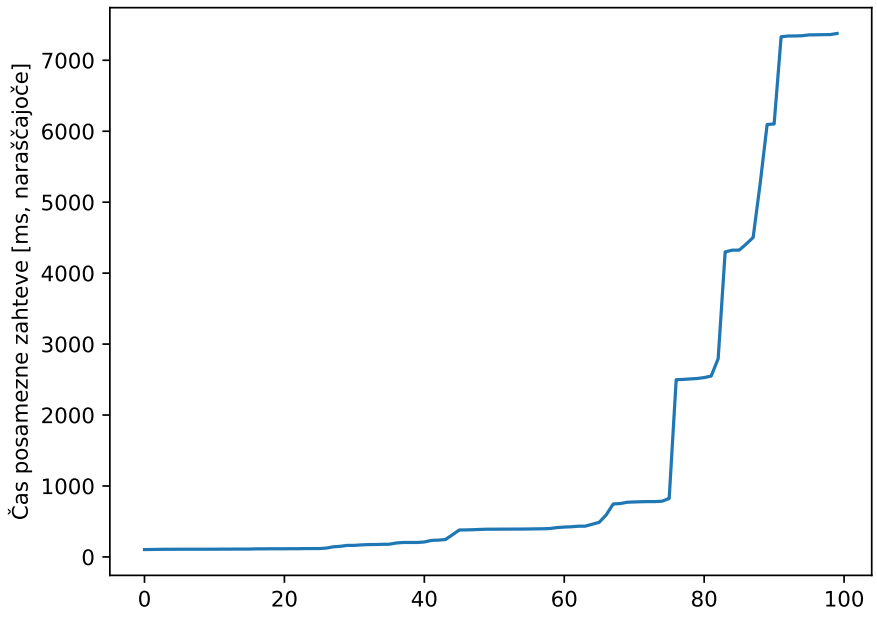
Zanima nas, kako se storitev odziva na zahteve ob različnih urah. Za potrebe meritev bomo odziv sistema merili ob treh različih časih v dnevu. Želimo izvedeti tudi, kako se časi odgovorov podaljšajo glede na število hkratnih uporabnikov.

1.4 Rezultati meritev

V soboto, 13. aprila 2019 ob 18.10 smo poskusno testirali iskanje piksla v okolici barve. Strežniku je bilo zaporedno poslanih 100 zahtev, vsaka je vsebovala naključno izbrano barvo, strežnik pa je iskal piksel z odstopanjem, manjšim ali enakim 86.

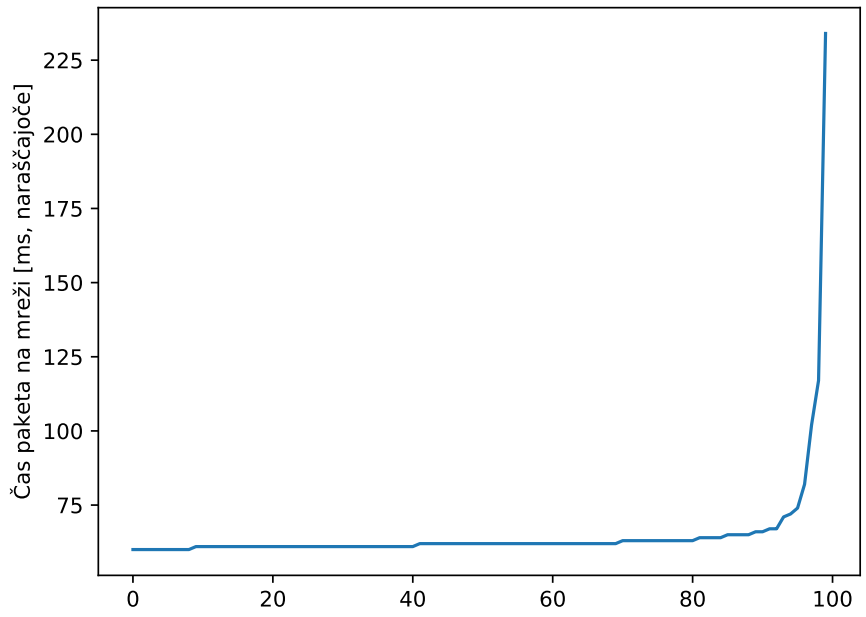
V 91 primerih je bil ustrezen piksel najden. Iskanje se je pri prvi sliki ustavilo v 45 primerih, pri drugi v 21, pri tretji v 10, pri osmi v 7, pri 12. v 5, pri 15. v enem in pri 17. v dveh. V ostalih 9. primerih ustrezen piksel ni bil najden.

Celotni časi zahtev ($t_{zahteva}$) so po velikosti urejeni prikazani v diagramu 1.3. Najhitrejši odziv je znašal 104 milisekunde, najdaljši pa 7378 milisekund. V povprečju je čas odziva znašal 1461 milisekund.



Slika 1.3: Diagram celotnih časov zahtev, urejenih naraščajoče po velikosti.

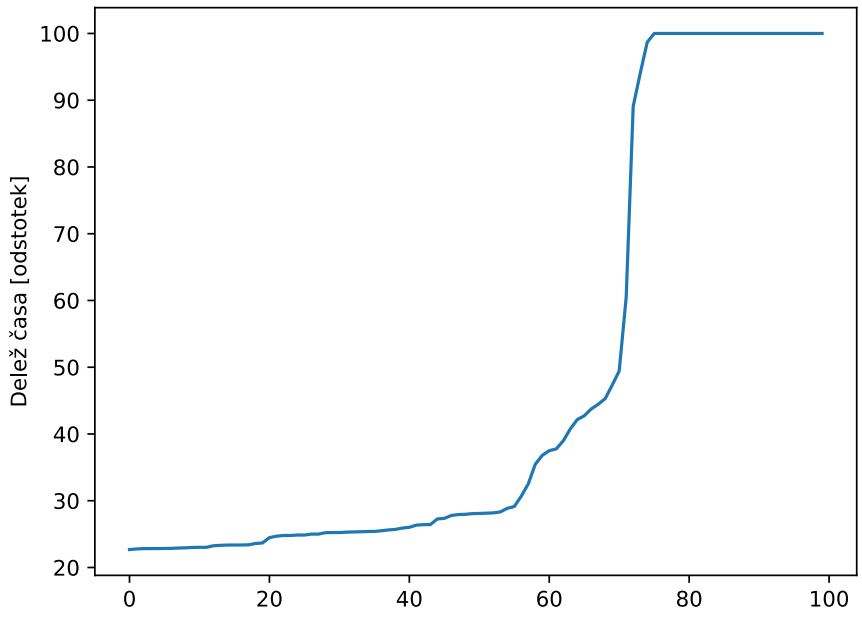
Časi paketa na mreži (t_{mreza}) so po velikosti urejeni prikazani v diagramu 1.4. Najhitrejši čas obhoda je znašal 60 milisekund, najdaljši pa 234 milisekund. V povprečju je čas paketa na mreži znašal 65 milisekund, v 90 izmed 100 primerov ni presegel 66 milisekund.



Slika 1.4: Diagram časov obhoda paketa, urejenih naraščajoče po velikosti.

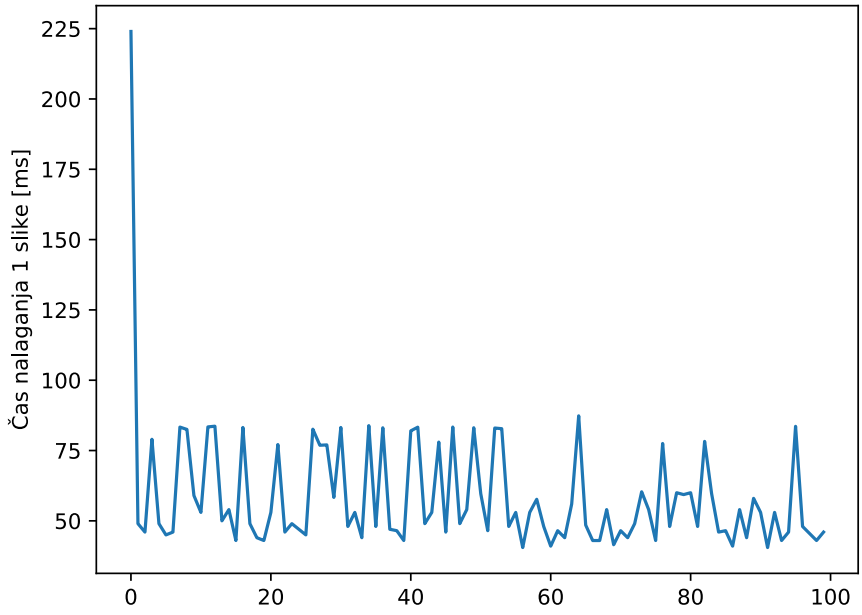
V splošnem nas zanima, kolikšen del procesiranja zavzame nalaganje slik z diska v pomnilnik ($\frac{t_{baza}}{t_{streznik}}$) in koliko časa v povprečju zavzame nalaganje ene slike v pomnilnik ($\frac{t_{baza}}{N_{nalozenih}}$).

Odstotki časa, ki ga storitev porabi za nalaganje slik v pomnilnik, so po velikosti urejeni prikazani v diagramu 1.5. Zavzemajo vrednosti med 23 in 100 odstotkov, v povprečju 49 odstotkov. Ves čas procesiranja se je za nalaganje slik porabil v kar 25 od 100 primerih. To pomeni, da se je naložila le prva slika, primeren piksel pa je bil najden skoraj takoj.



Slika 1.5: Diagram odstotkov časa, porabljenega za nalaganje slik z diska v pomnilnik, urejenih naraščajoče po velikosti.

Povprečne vrednosti časa nalaganja 1 slike so prikazani v diagramu 1.6. Zavzemajo vrednosti med 40 in 224 milisekund, 59 milisekund. V 99 izmed 100 primerov ni presegel 88 milisekund.



Slika 1.6: Diagram povprečnih časov nalaganja ene slike z diska v pomnilnik, v vrstnem redu zahtev.

1.5 Plan dela

- testiranje ostalih dveh iskanj
- določitev bremen - čas meritev, število hkratnih uporabnikov, ... ?
- natančnejša določitev metrik
- določitev orodij - avtomatizacija merjenja, zbiranje rezultatov
- premislek o skalabilnosti

Literatura

[1] "Amazon elastic compute cloud." https://aws.amazon.com/ec2/, 2019. [Online; dostopano 8-April-2019].

[2] "Aws free tier." https://aws.amazon.com/free/, 2019. [Online; dostopano 8-April-2019].

[3] "Amazon ec2 t2 instances." https://aws.amazon.com/ec2/instance-types/t2/, 2019. [Online; dostopano 8-April-2019].

[4] "Amazon web services." https://aws.amazon.com/, 2019. [Online; dostopano 8-April-2019].

[5] "Launch a linux virtual machine." https://aws.amazon.com/getting-started/tutorials/launch-a-virtual-machine/?trk=gs_card, 2019. [Online; dostopano 8-April-2019].

[6] "Aws management console." https://aws.amazon.com/console/, 2019. [Online; dostopano 8-April-2019].

[7] "Aws management console." https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html, 2019. [Online; dostopano 8-April-2019].

[8] "Amazon elastic block store (amazon ebs)." https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AmazonEBS.html, 2019. [Online; dostopano 20-April-2019].