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

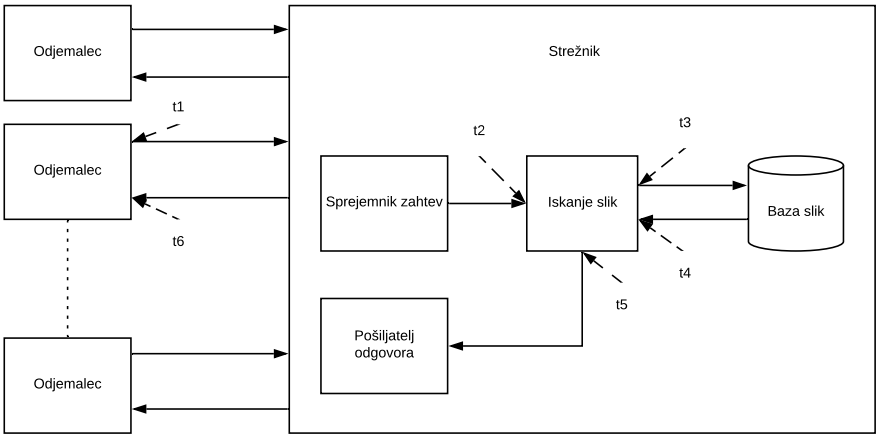
odgovor, z ne več kot nekaj sekundami zamika. Taka storitev nam bo omogočala relativno enostavno merjenje odzivnosti platforme, modularnost nalaganja kode in morebitno razširljivost v primeru večjega števila hkratnih uporabnikov.

1.2 Realizacija

Storitev je sestavljena iz dveh delov, in sicer 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. Zaporedoma preiskuje slike v oblačni podatkovni bazi. Takoj ko najde prvo pojavitev iskanega piksla, iskanje ustavi in vrne podatke o tej pojavitvi - zaporedno številko slike in koordinati x ter y najdenega piksla. Če ne najde nobene pojavitve, preišče vse slike v podatkovni bazi in vrne sporočilo, da piksel ni bil najden.

Iskanje piksla, podobnega specifični barvi

Poleg iskane barve storitev prejme od odjemalca še največje dovoljeno odstopanje. Zaporedoma preiskuje slike v podatkovni bazi, dokler ne najde prvega piksla, katerega barva se od iskane po komponentah razlikuje za največ toliko, kot določa odstopanje. Razlika se izračuna po enačbi

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

Če je iskana barva naključno izbrana, potem lahko za dano odstopanje izračunamo približno verjetnost, da bomo dobili ujemanje z naključnim pikslom. Za vsako barvo in odstopanje n obstanja največ $\frac{1}{3}(n^3 + 6n^2 + 14n + 3)$ barv, ki se od iskane barve po komponentah razlikujejo za n. Vsak barvni kanal lahko zavzema vrednosti od 0 do 255, torej je za izbiro barve na voljo $2^{24} = 16777216$ vrednosti. Verjetnost aproksimiramo kot $\frac{n^3+6n^2+14n+3}{3 \cdot 16777216}$. Nekaj verjetnosti zadetka in ustreznih odstopanj:

verjetnost	0.05%	0.1%	0.2%	0.5%	1%	2%	5%	10%
odstopanje	18	23	29	39	50	63	85	107

Ampak piksli v slikah niso naključni. Boljšo oceno dobimo, če privzamemo, da je barva cele slike *približno* enaka - če ne najdemo ustreznega piksla na začetku slike, potem je velika verjetnost, da ga tudi v preostanku slike ne bomo. Tako lahko izračunamo, koliko slik bo v povprečju pregledanih s pomočjo matematičnega upanja po formuli

$$E(p) = \sum_{i=1}^{\text{število slik}} i \cdot (1-p)^{i-1} \cdot p = \frac{1}{p}. \tag{1.2}$$

To pomeni, da bomo pri verjetnosti 1% v povprečju pregledali približno 100 slik pri vsaki zahtevi. Izkaže se, da je ta ocena v primeru te storitve kar dobra.

Iskanje slikovnega izseka

Storitev prejme slikovni izsek. Zaporedoma preiskuje slike v podatkovni bazi, dokler ne najde prvega ujemanja podanega izseka z izsekom na neki sliki. Izseka se ujemata, če je barva vsakega piksla danega izseka za največ 7 oddaljena od barve ustreznega piksla na sliki. Razlika barv se računa tako kot pri podobni barvi piksla (enačba 1.1). Odstopanje je tu potrebno le zato, ker pri shranjevanju izsekov pride do artefaktov stiskanja [4].

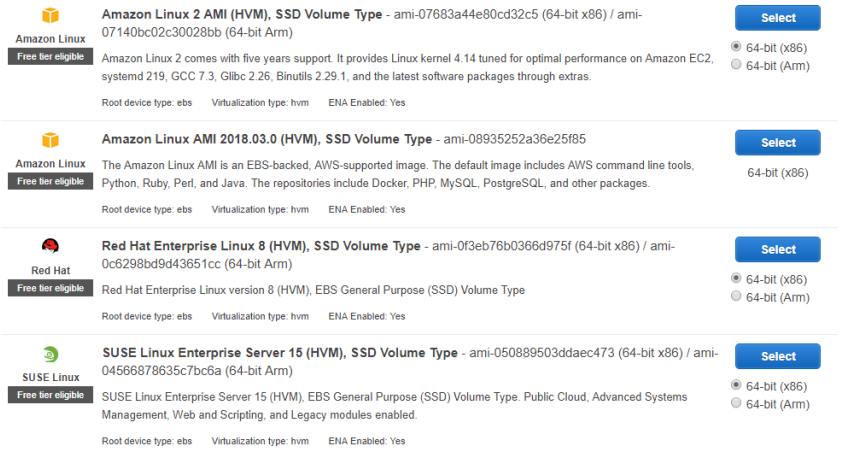
1.2.3 Slikovni material

Za potrebe zgoraj naštetih metod in njihovega testiranja smo na strežnik naložili 540 slik v JPEG formatu, ki smo jim zmanjšali velikost na 300 pikslov po širini in 400 pikslov po dolžini ali obratno za hitrejše procesiranje. Hkrati nam to zagotavlja, da je časovna zahtevnost primerjave približno enaka za ves slikovni material. Prav tako je bil cilj zbrati slike s čim večjo vsebinsko raznolikostjo, saj je to pomembno pri iskanju barve piksla in iskanju piksla podobnega specifični barvi. Nekaj primerov slik se nahaja na sliki 1.2.

Slika 1.2: Nekaj primerov slik iz nabora.

1.2.4 Amazon Web Services (AWS)

Za delo z Amazon Web Services [5] 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 [6], mi pa to naredimo tako, da se postavimo v AWS Management Console [7], 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 [8]. Izbor možnih slik diska je prikazan na sliki 1.3. Za naš projekt izberemo sliko Amazon Linux 2 AMI c.



Slika 1.3: 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 brezplačno hranjenje podatkov na mrežno povezani shrambi [9]. 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.

1.2.5 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 odgovora.

```
{
   "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 odjemalca do strežnika in čas potovanja paketa od strežnika do odjemalca, 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 omrežju.

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 Meritve

Zmogljivost storitve smo merili večkrat. Testirali smo hitrost iskanja podobne barve, opazovali smo razlike v hitrosti odziva iz različnih lokacij in ob različnih časih.

1.4.1 Poskusno testiranje

V soboto, 27. aprila 2019 ob 20.37 smo poskusno testirali iskanje piksla v okolici barve. Strežniku je bilo poslanih 500 zahtev, vsaka je vsebovala naključno izbrano barvo, strežnik pa je iskal piksel z odstopanjem, manjšim ali enakim 50. Zahteve so bile poslane zaporedno, kar pomeni, da je po oddaji zahteve odjemalec počakal na odgovor strežnika, preden je poslal naslednjo. Odstopanje je bilo izbrano tako, da so bili zadetki čim bolj enakomerno razporejeni po slikah v podatkovni bazi.

Zahteve je generiral osebni računalnik, nahajajoč se na Viču v Ljubljani. Prek brezžičnega omrežja je bil priklopljen na kabelski internet ponudnika A1, ki za zakupljen paket *A1 Kombo L* oglašuje hitrost povezave 40 megabitov na sekundo k uporabniku in 10 megabitov na sekundo od uporabnika. Test na strani *Speedtest by Ookla* te vrednosti potrdi (rezultat testa: `https://www.speedtest.net/result/8218379266`). Čas obhoda paketa od odjemalca do storitve in nazaj (t_{mreza}) je povprečju znašal 62 milisekund. Rezultat orodja `tracert` se nahaja v izpisu 1.3.

Listing 1.3: Rezultat orodja `tracert`.

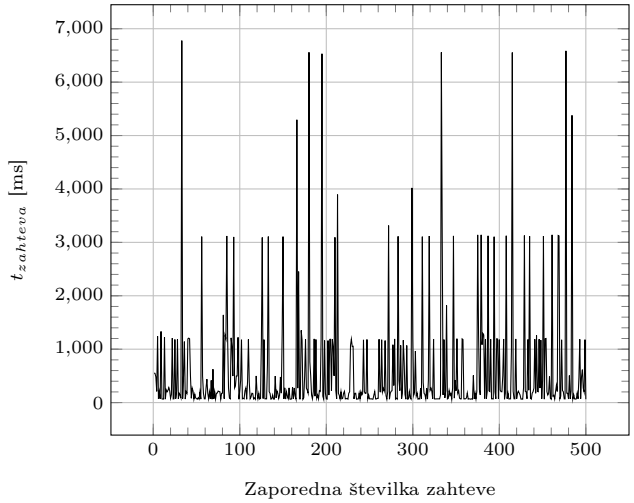
```
Tracing route to ec2-52-212-203-50.eu-west-1.compute.amazonaws.com
    [52.212.203.50]
over a maximum of 30 hops:

  1    2 ms   <1 ms   <1 ms 192.168.1.254
  2    *       *       *    Request timed out.
  3    *       *       *    Request timed out.
  4    *       *       *    Request timed out.
  5   25 ms   26 ms   25 ms 195.3.102.57
```

```
 6    *        *        *     Request timed out.
 7   26 ms    26 ms    28 ms  lg4-9072.as8447.a1.net [195.3.64.142]
 8   26 ms    27 ms    25 ms  52.95.219.250
 9   31 ms    33 ms    32 ms  52.93.38.102
10   25 ms    26 ms    27 ms  52.93.38.109
11    *       58 ms    57 ms  54.239.44.47
12   63 ms    58 ms    58 ms  52.93.128.141
13   57 ms    60 ms    58 ms  52.93.128.2
14    *       58 ms    63 ms  54.239.44.158
15    *        *        *     Request timed out.
16   72 ms    93 ms    80 ms  52.93.7.190
17   59 ms    62 ms    59 ms  52.93.101.127
18   72 ms     *       79 ms  52.93.101.126
19   61 ms    61 ms    61 ms  52.93.36.143
20    *        *        *     Request timed out.
21    *        *        *     Request timed out.
22    *        *        *     Request timed out.
23    *        *        *     Request timed out.
24    *        *        *     Request timed out.
25    *        *        *     Request timed out.
26    *        *        *     Request timed out.
27   60 ms    62 ms    60 ms
    ec2-52-212-203-50.eu-west-1.compute.amazonaws.com [52.212.203.50]

Trace complete.
```

Celotni časi zahtev ($t_{zahteva}$) so prikazani na sliki 1.4.

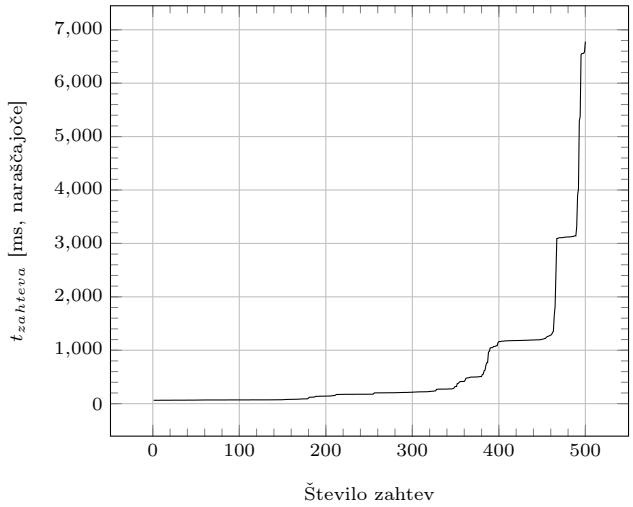


Slika 1.4: Diagram celotnih časov zahtev.

Najhitrejši odziv $t_{zahteva}$ je znašal 63 milisekunde, najdaljši pa 6778 milise-

kund. V povprečju je čas odziva znašal 587 milisekund.

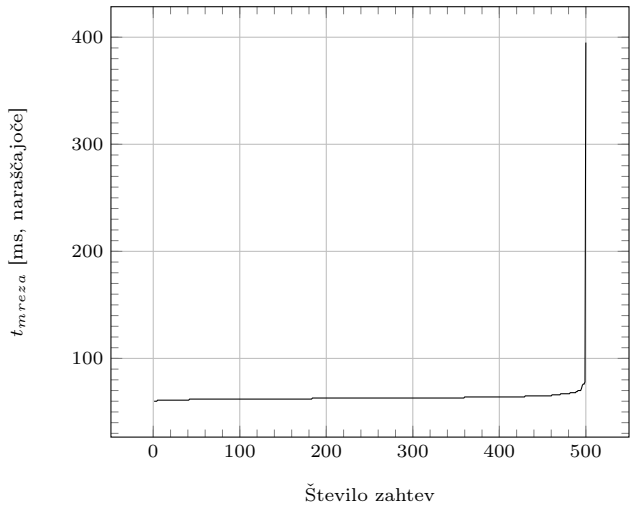
Iz slike 1.4 ni enostavno razvidno, kakšna je razporeditev časov zahtev. Na sliki 1.5 so celotni časi zahtev ($t_{zahteva}$) prikazani po velikosti urejeni.



Slika 1.5: Diagram celotnih časov zahtev, urejenih naraščajoče po velikosti.

Iz slike 1.5 je lažje razvidno, da je večina časov zahtev manj kot 500 milisekund, le peščica pa jih preseže 3500 milisekund.

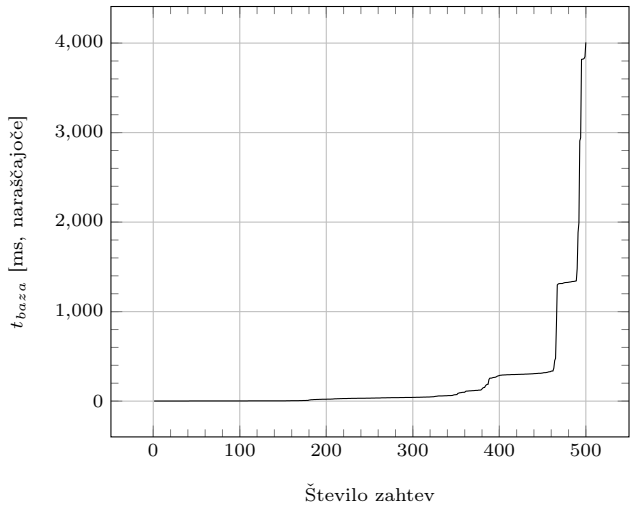
Časi zahteve na mreži (t_{mreza}) so po velikosti urejeni prikazani na sliki 1.6. Najhitrejši čas obhoda je znašal 60 milisekund, najdaljši pa 395 milisekund. V povprečju je čas paketa na mreži znašal 64 milisekund, v 499 izmed 500 primerov ni presegel 78 milisekund.



Slika 1.6: Diagram časov obhoda paketa, urejenih naraščajoče po velikosti.

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

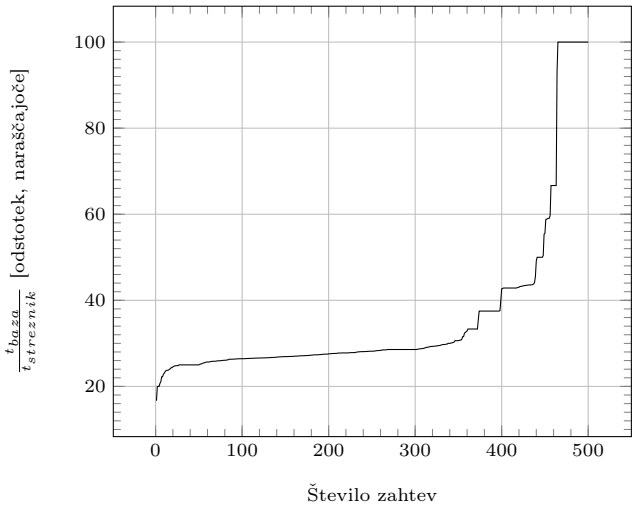
Časi, ki jih storitev porabi za nalaganje slik v pomnilnik, so po velikosti urejeni prikazani na sliki 1.7. Zavzemajo vrednosti med 1 in 4008 milisekund, v povprečju pa 201 milisekundo.



Slika 1.7: Diagram časov, porabljenih za nalaganje slik z diska v pomnilnik, urejenih naraščajoče po velikosti.

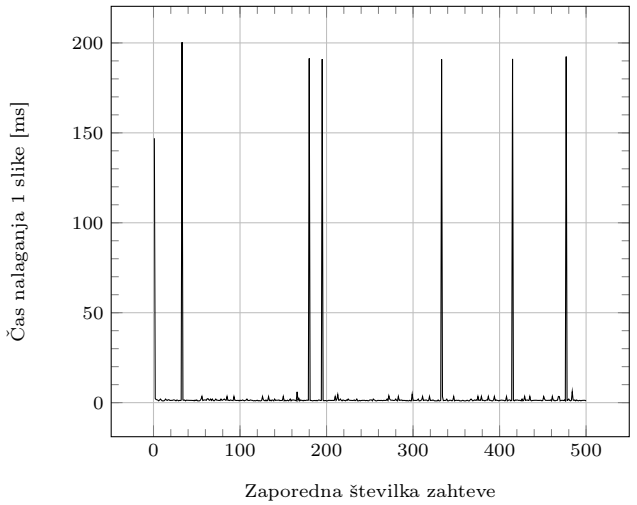
Odstotki časa, ki ga storitev porabi za nalaganje slik v pomnilnik, so po

velikosti urejeni prikazani na sliki 1.8. Zavzemajo vrednosti med 16 in 100 odstotkov, v povprečju pa 36 odstotkov.



Slika 1.8: Diagram odstotkov časa, porabljenega za nalaganje slik z diska v pomnilnik, urejenih naraščajoče po velikosti.

Povprečne vrednosti časa nalaganja 1 slike so prikazani na sliki 1.9. Vrednosti so izračunane kot količnik časa nalaganja slik s številom preiskanih slik pri posamezni zahtevi. Zavzemajo vrednosti med 1 in 200 milisekund, v povprečju pa 4 milisekunde. V 493 izmed 500 primerov ni presegel 7 milisekund.



Slika 1.9: Diagram povprečnih časov nalaganja ene slike z diska v pomnilnik, v vrstnem redu zahtev.

1.4.2 Primerjava odzivov na zahteve iz različnih lokacij

Zaradi odločitve, da bomo testirali tudi z različnih lokacij, smo se najprej odločili, da preverimo kakšne zakasnitve (ping) ima posamezen član na različnih lokacijah do storitve. Rezultati po 30 ponovitvah so prikazani v tabeli 1.1. Te razlike v časih pripisujemo predvsem temu dejstvu, da ima vsaka od teh lokacij drugega ponudnika omrežnih storitev in zato naši paketi prepotujejo različne poti. V povprečju opravijo 7 skokov, dokler ne pridejo do skupne točke, ki je v lasti podjetja Amazon Technologies Inc. [10]. Te razlike so pomembne samo pri merjenju časa, ko se paket nahaja na mreži, in niso odvisne od zmogljivosti same storitve.

Tabela 1.1: Rezultat orodja `ping`

Lokacija	Min [ms]	Max [ms]	Povprečje [ms]
Ljubljana Vič	58	62	58
Postojna	41	41	41
Vrtojba	42	43	42

Testirali smo iskanje piksla, podobnega specifični barvi. Poslali smo 500 zahtev. Vsaka je vsebovala naključno barvo, strežnik pa je iskal piksel z odstopanjem, manjšim ali enakim 50. Zahteve so bile poslane zaporedno, po oddaji zahteve je odjemalec počakal na odgovor strežnika, preden je poslal naslednjo. Minimalni, maksimalni in povprečni časi zahtev iz vsake lokacije so prikazani v tabeli 1.2.

Tabela 1.2: Časi zahtev $t_{zahteva}$, poslanih iz različnih lokacij.

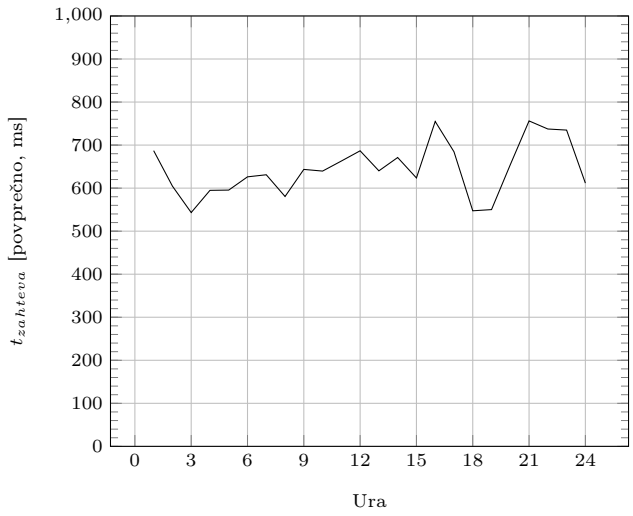
Lokacija	Min [ms]	Max [ms]	Povprečje [ms]
Ljubljana Vič	63	6778	587
Postojna	42	6850	624
Vrtojba	44	6570	549

Minimalni časi zahtev se z zanemarljivim odstopanjem ujemajo z izmerjenimi zakasnitvami do storitve. Variacijo pri maksimalnih in povprečnih časih zato pripisujemo nakjučni izbiri barve v zahtevah. Upoštevajoč to sklepamo, da ima lokacija odjemalca zanemarljiv pomen pri zmogljivosti storitve.

1.4.3 Primerjava odzivov na zahteve ob različnih časih

Kvaliteta storitve je odvisna tudi od zmožnosti odzivanja ob različnih časih. To smo testirali tako, da smo v urnih intervalih pošiljali 500 zahtev tipa iskanja piksla, podobnega specifični barvi. Vsaka zahteva je vsebovala naključno izbrano barvo, strežnik pa je iskal piksel z odstopanjem, manjšim ali enakim 50. Zahteve so bile poslane zaporedno, po oddaji zahteve pa je odjemalec počakal na odgovor

strežnika, preden je poslal naslednjo. Povprečni časi $t_{zahteva}$ glede na uro so prikazani na sliki 1.10.

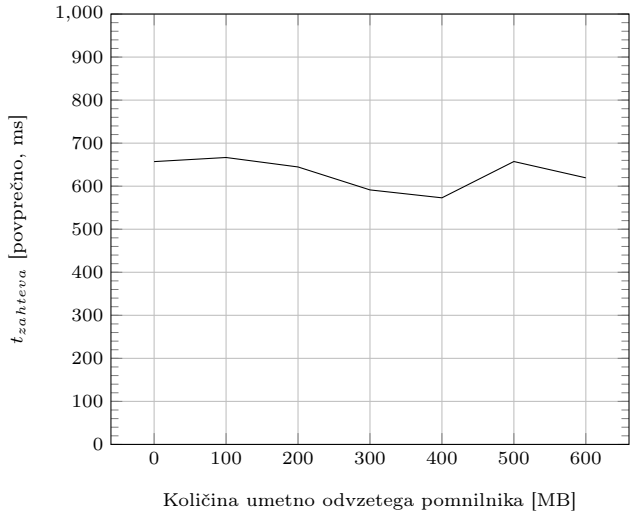


Slika 1.10: Diagram povprečnih časov odziva $t_{zahteva}$ glede na čas v dnevu.

V grobem se storitev odziva približno enako skozi dan, dopoldan in zgodaj zvečer nekoliko hitreje kot pozno popoldan in pozno zvečer.

1.4.4 Stresni test - primerjava odzivov na zahteve pri zmanjšani količini delovnega pomnilnika

Za izvedbo stresnega testa smo umetno zmanjševali količino delovnega pomnilnika, ki ga je storitev imela na voljo. To smo storili tako, da smo pred testiranjem zagnali program, ki je alociral poljubno količino delovnega pomnilnika in ga tako odvzel storitvi. Začeli smo z vsem pomnilnikom (1 GB), nato pa ga postopoma zmeraj več alocirali v korakih po 100 MB. Ustavili smo se pri 600 MB alociranega pomnilnika, pri 700 MB je strežnik po nekaj zahtevah zmrznil in zahteval ponovni zagon programske opreme (vnos `Ctrl+C` v terminalu). Povprečni časi glede na količino umetno odvzetega pomnilnika so prikazani na sliki 1.11.



Slika 1.11: Diagram povprečnih časov odziva $t_{zahteva}$ glede na količino umetno odvzetega pomnilnika.

Opazili smo, da se storitev odziva približno enako, ne glede na količino razpoložljivega pomnilnika, v kolikor je na voljo vsaj približno 400 MB delovnega pomnilnika.

1.4.5 Bremenski test - primerjava odzivov na zahteve pri povečanemu število sočasnih uporabnikov

Za izvedbo bremenskega testa smo primerjali čase odzivov storitve pri enem uporabniku in čase odzivov pri treh sočasnih uporabnikih. Testirali smo iskanje piksla, podobnega specifični barvi. Za simulacijo treh sočasnih uporabnikov smo iz treh lokacij (Vič, Postojna in Vrtojba) hkrati zagnali iskanje. Časi so bili merjeni na Viču. Minimalni, maksimalni in povprečni časi zahtev enega in treh sočasnih uporabnikov so prikazani v tabeli 1.3.

Tabela 1.3: Časi zahtev $t_{zahteva}$ glede na število sočasnih uporabnikov.

Število uporabnikov	Min [ms]	Max [ms]	Povprečje [ms]
1	58	6551	662
3	58	23072	1960

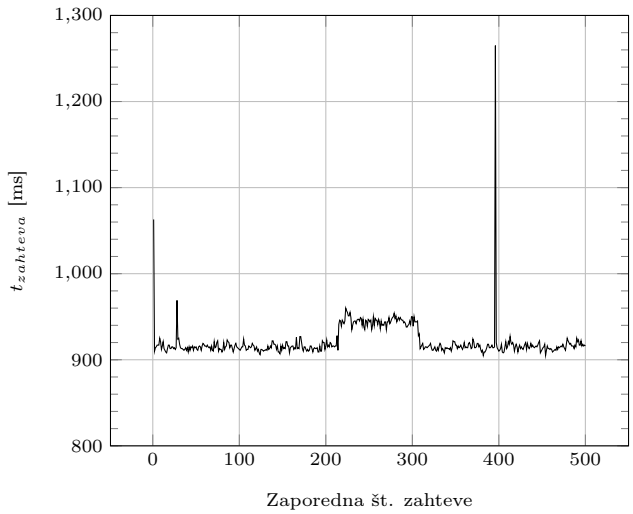
Minimalni čas je v obeh primerih enak. Je le čas, ki ga paket porabi od uporabnika do storitve in nazaj. Povprečni čas je v primeru treh uporabnikov približno trikrat daljši kot v primeru enega. To si lahko razlagamo na sledeči način: Storitev popolnoma zaposli že en uporabnik. Če imamo tri hkratne

uporabnike, mora potemtakem storitev vsakemu nameniti eno tretjino svoje računske moči. Povprečen čas odziva je zato trikrat daljši. Maksimalni čas je težje obrazložiti. V primeru treh uporabnikov je 3,5-krat daljši kot pri enem. Sklepamo, da smo naleteli na neznano ozko grlo, za natančnejšo določitev regresije pa bi morali natančneje opazovati obnašanje storitve pri več uporabnikih in določiti, kateri del obdelave povzroči to upočasnitev.

1.4.6 Frekvenčno testiranje

Zanima nas kako se storitev odziva če zahteve pošiljamo konstantno vsakih Δt časovnih enot, ta čas pa po določenem času t_{step} zmanjšamo in pri tem opazujemo kako se podaljšuje čas odziva na zahtevo $t_{zahteva}$.

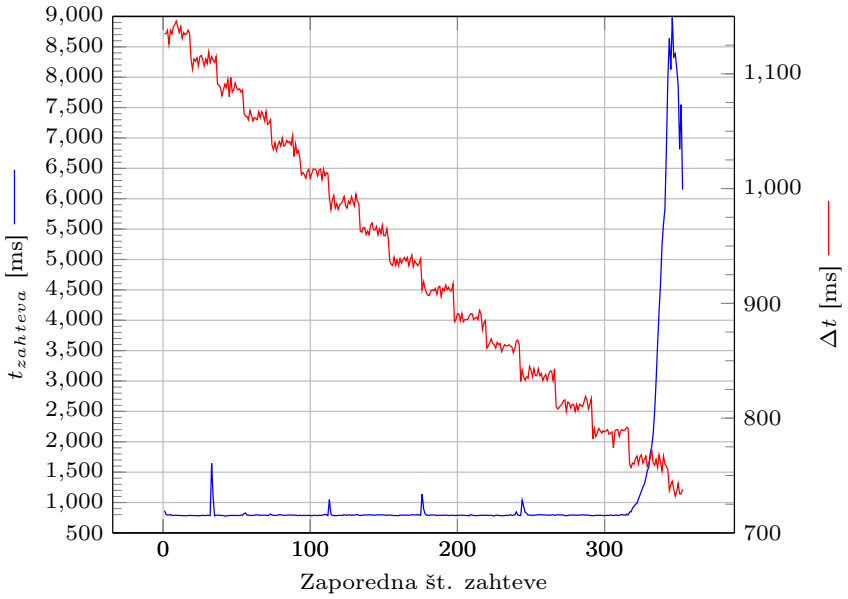
Za pravilno testiranje takega tipa smo se odločili, da bomo ves čas pošiljali isto zahtevo brez naključnosti na strežnik, kar pa lahko povzroči probleme če imajo strežniki predpomnjenje. To smo stestirali tako, da smo poslali 500 enakih zahtev in opazovali čase $t_{zahteva}$, kar je prikazano na grafu 1.12.



Slika 1.12: Diagram časov odziva $t_{zahteva}$ pri istih zahtevah.

Če pogledamo graf je ta skoraj da ravna premica z razliko nekaj špic. Če se posvetimo času $t_{processing}$, ki ni vrisan v graf pa opazimo da so si med zahtevami zelo podobni in lahko te špice pripišemo zastojem na omrežju, ne pa času procesiranja. Posledično lahko razberemo, da pred našim strežnikom ni nobenega predpomnilnika, saj so naši časi procesiranja ostajali isti, časi zahtev pa so bili v določenih primerih višji, ne pa nižji kot bi bili ob uporabi prepomnilnika. Hkrati ta naš zaključek potrjuje tudi dejstvo, da je vsaka od zahtev malo drugačna od druge, saj vsebuje svoj unikaten identifikator in pa uro kdaj je bila zahteva poslana, kar preprostemu predpomnilniku ne omogoča delovanja.

Za frekvenčno testiranje smo kot začetni Δt izbrali vrednost $1100ms$, za trajanje koraka t_{step} smo izbrali vrednost $20000ms$ in po vsakem koraku smo Δt zmanjšali za $25ms$. Testiranje smo zaključili, ko je izmerjen čas odgovora presegel $8000ms$. Rezultati meritev so vidni na grafu 1.13



Slika 1.13: Graf frekvenčnega testiranja.

Iz grafa lahko razberemo, da se sistem zelo dobro odziva, dokler ni čas med dvema zahtevama manjši od $825ms$ to pomeni največ $1,2$ $zahteve/s$. Ko presežmo to mejo začne $t_{zahteva}$ silovito naraščati, kar pomeni, da se začne ustvarjati čakalna vrsta za obdelavo, ki pa kot vidimo zelo hitro zraste. Zavajujoče deluje prevoj grafa $t_{zahteva}$, ki se začne spuščati ko dosežemo mejo $\Delta t = 775ms$, kar pa pojasni dejstvo, da smo mi ravno pred tisto točko presegli zgornjo mejo $t_{zahteva} = 8000ms$ in po tem prenehali pošiljati zahteve na strežnik, vendar je strežnik še vedno moral sprazniti čakalno vrsto, kar je vidno v padcu grafa.

1.5 Zaključek

Platforma Amazon Elastic Cloud ponuja zanimivo brezplačno možnost za lansiranje enostavne spletne storitve. Sloni na razsežni oblačni infrastrukturi podjetja Amazon, kar zagotavlja hitre odzivne čase ne glede na lokacijo uporabnika in čas v dnevu, kar je razvidno iz naših testov. Ker so osnovne enote platforme strežniki, na katerih lahko teče odprtokodna programska oprema, osnovana na jedru Linux, je snovanje in izvajanje programske opreme enostavno in dobro podprto.

Največja težava, s katero se mora razvijalec spopasti, je nizka zmogljivost brezplačnih resursov, ki so mu na voljo. Že enostavna storitev hitro zasede 1 jedro in 1 GB delovnega pomnilnika, poleg tega pa je treba pametno izkoristiti 30 GB navideznega diskovnega prostora, ki je na voljo brez plačila. Amazon nam seveda rade volje ponudi boljše, a ne zastonj. Virtualni strežnik z dvemi jedri, 4 GB delovnega pomnilnika in 100 GB navideznega diskovnega prostora stane 23 evrov mesečno za najem [**?**], torej mora razvijalec šteti tudi profitabilnost med svoje cilje.

Amazon Elastic Cloud priporočamo za storitve, ki so računsko razmeroma nezahtevne. Uspeh storitev temelji na več faktorjih, vsekakor pa je zanesljivost infrastrukture zelo pomemben del. Tu Amazon Elastic Cloud zadosti vsaj osnovne potrebe, obljublja pa še veliko več, a ne brez plačila.

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] "Compression artifact." https://en.wikipedia.org/wiki/Compression_artifact/, 2019. [Online; dostopano 27-April-2019].

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

[6] "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].

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

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

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

[10] "Lastnik ip naslove." https://db-ip.com/all/52.95.219, 2019. [Online; dostopano 1-Maj-2019].