Analiza zmogljivosti oblačnih in strežniških storitev

Uredil prof. dr. Miha Mraz

Maj 2018

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 2017/2018 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 2018

Poglavje 1

Analiza zmogljivosti pomnjenja različnih oblačnih storitev

Miha Pečnik, Mihael Podplatnik, Nik Zupančič

1.1 Opis problema

V zadnjih letih se predvsem pri prenosnih računalnikih pojavlja težnja po zmanjšanju velikosti trdega diska, zato se velikokrat zanašamo na oblačne storitve za zagotavljanje dodatnega prostora za shranjevanje. Namen tega poglavja je med seboj primerjati zmogljivosti več različnih oblačnih storitev z vidika pomnjenja in ugotoviti, ali kateri izmed ponudnikov dosega izrazito boljše rezultate od drugih. Preverili bomo kako se oblačne storitve med seboj razlikujejo, zasedenost posamezne storitve v različnih delih dneva, oddaljenost lokacije in čas prenosa različnih vrst datotek (slike, video, glasba...). Iz zbranih podatkov bomo skušali izbrati najprimernejšo oblačno storitev za uporabnika. Predlagani ponudniki storitev za testiranje in primerjavo pomnjenja so Dropbox [1], Mega [2] in Google Drive [3]. Prva razlika, ki jo opazimo med temi ponudniki so različne ponudbe za brezplačne uporabniške račune. V tem pogledu nam najmanj nudi Dropbox, ki nam z novim brezplačnim uporabniškim računom dodeli samo 2 GB prostora, medtem ko Google Drive in Mega oba nudita kar 15 GB prostora.

Eno izmed del, ki se ukvarja s podobnim področjem, je magistrsko delo [4] v okviru katerega je tudi nastalo programsko orodje, ki smo ga uporabljali pri naših meritvah. Celotna magistrska naloga sicer ni osredotočena le na testiranje, ampak vsebuje tudi splošno analizo delovanja oblačnih sistemov ter se v večji meri osredotoča predvsem na oblikovanje ustrezne programske opreme

za testiranje pomnjenja oblačnih ponudnikov, v zadnjem delu naloge pa lahko najdemo nekaj meritev, ki so bile opravljene na Dropboxu. Glavni zaključki v okviru teh meritev so, da večnitno pošiljanje datotek v oblak sicer poveča povprečni čas potreben za shranjevanje posamezne datoteke, vendar pa se kljub temu skupni čas potreben za shranjevanje vnaprej določenega števila datotek zmanjša v primerjavi s pošiljanjem le ene datoteke naenkrat. Še eden od zanimivih zaključkov v nalogi je bil, da v kolikor uporabnik na svojem računalniku uporablja Dropboxovo programsko storitev, pred pošiljanjem datoteke v oblak pride do kompresije datotek, kar privede do razlike med velikostmi omrežnih paketov ter dejanskimi velikostmi datotek, ki jih želimo shranjevati.

Članek Personal Cloud Storage Benchmarks and Comparison [5] se še bolj podrobno osredotoči na to tematiko in med seboj primerja 11 različnih ponudnikov oblačnih storitev, to pa vključuje tudi vse naše predlagane ponudnike, zato bomo pri nadaljnem delu naše ugotovitve lahko primerjali z rezultati raziskovalcev v članku.

1.2 Orodja

Za pomoč pri izvajanju ter analizi meritev smo si izbrali nekaj orodij, ki so brezplačno dostopna na spletu.

1.2.1 BenchCloud

BenchCloud [6] je orodje izdelano v programskem jeziku Python in omogoča interakcijo z oblačnimi storitvami, ki nudijo možnosti nalaganja in prenašanja datotek preko storitve API. Pri pošiljanju datotek v oblak omogoča specifikacijo različnih načinov generiranja datotek, ki jih shranjujemo v oblak. Lahko generiramo datoteke, ki imajo popolnoma naključno vsebino, datoteke katerih vsebina je identična, datoteke, ki se razlikujejo le v določenem delu vsebine ali datoteke, ki vsebujejo ponavljajoči se niz - velikost teh datotek je možno učinkovito zmanjšati s kompresiranjem. Pri vseh načinih generiranja lahko izberemo poljubno velikost in končnico ter število datotek, ki jih želimo shraniti v oblak. Po koncu zadane naloge nam BenchCloud generira poročilo, v katerem zabeleži koliko časa so trajale posamezne operacije generiranja datotek in koliko časa je vzelo nalaganje v oblak ter koliko časa smo potrebovali za prenos ene ali vseh datotek iz oblaka.

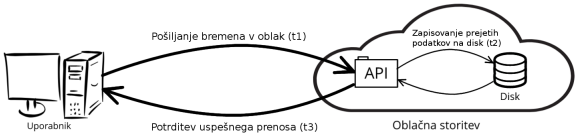
1.2.2 Wireshark

Orodje BenchCloud sicer nudi funkcijo spremljanja paketov ob pošiljanju ter prejemanju datotek, vendar bomo za bolj natančno analizo prometa po omrežju potrebovali orodje Wireshark [7]. S pomočjo Wiresharka bomo lažje ugotovili kam v resnici pošiljamo svoje podatke, saj lokacija strežnika lahko igra pomembno vlogo pri pošiljanju in prejemanju podatkov.

1.3 Metrike

1.3.1 Čas prenosa

Kot je razvidno iz slike 1.1, ki prikazuje potek pošiljanja bremena v oblak, je čas shranjevanja posamezne datoteke sestavljen iz treh komponent. Čas t_1 predstavlja čas potreben za pošiljanje bremena z uporabnikovega sistema v oblak, t_2 je čas, ki ga oblačna storitev porabi za shranjevanje prejete datoteke ter t_3 čas, ki traja od končanega shranjevanja datoteke v oblaku do prejema potrditve na uporabnikovem sistemu. Potrditev, ki jo prejmemo ob koncu prenosa na Dropbox, je sestavljena podatkovna struktura, ki vsebuje osnovne podatke o shranjeni datoteki v oblaku. Primer odgovora na uspešno pošiljanje lahko vidimo v izpisu 1.1.



Slika 1.1: Primer pošiljanja enega bremena v oblačno storitev.

V potrditvi na izpisu 1.1 lahko najdemo osnovne podatke o datoteki, ki smo jo ravnokar shranili v oblak. V prvih dveh vrsticah vidimo informacije o imenu datoteke ter unikatni identifikator datoteke. Sledita datum in točna ura, ko je bila datoteka poslana oz. modificirana. Vidimo lahko, da prvi datum prikazuje čas, ki ga Dropboxu posreduje klient, vendar ta sodeč po dokumentaciji ni najbolj zanesljiv, zato naj bi se uporabljal naslednji, to je datum, ki ga zabeleži sam strežnik in je bolj natančen. Med svojimi pošiljanji razlike med njima nismo opazili. Polje *rev* je enolični identifikator, vezan na trenutno različico datoteke. S tem se pozneje lahko ugotovi, ali je bila datoteka po prvem shranjevanju spremenjena ali ne. *Size* prikazuje velikost datoteke v bajtih, *path_lower* pa je pot do datoteke v našem Dropbox direktoriju z zgolj malimi črkami. V primeru izpisa je datoteka shranjena v korensko mapo, *path_display* pa je namenjen lepšemu izpisu poti datoteke, ki bi ga potrebovali v programih oz. aplikacijah, saj prikazuje dejansko ime datoteke brez spreminjanja v male črke. Sledi še nekaj podatkov, ki niso nujni in so v primeru izpisa brez vrednosti. Dropbox omogoča uporabo skupnih map, ki si jih lahko med seboj deli več uporabnikov. V tem primeru identifikacijski niz skupne mape, v katero smo shranili datoteko, dobimo v polju *parent_shared_folder_id*, *media_info* vsebuje dodatne podatke o tem ali je datoteka video oz. fotografija. Tudi *sharing_info* in *has_explicit_shared_members* nam pove ali je datoteka deljena tudi z drugimi uporabniki. V *property_groups* pa datoteki lahko določimo dodatne vrednosti v obliki slovarja (ključ - vrednost). S pomočjo tega lahko uporabnik datoteki doda tudi dodatne lastnosti. V zadnjem polju imamo še zgoščeno vrednost vsebine datoteke, ki jo strežnik uporablja za primerjavo lokalnih datotek z datotekami

shranjenimi v oblaku.

Listing 1.1: Primer potrditve, ki jo pošlje Dropbox po končanem prenosu.

```
FileMetadata(name=u'benchmarking-tKfG8O',
    id=u'id:IKsdqYoRnMAAAAAAAAAnwA',
    client_modified=datetime.datetime(2018, 4, 12, 8, 40, 16),
    server_modified=datetime.datetime(2018, 4, 12, 8, 40, 16),
    rev=u'4f6ca14d7960', size=5242880,
    path_lower=u'/benchmarking-tkfg8o',
    path_display=u'/benchmarking-tKfG8O', parent_shared_folder_id=None,
    media_info=None, sharing_info=None, property_groups=None,
    has_explicit_shared_members=None,
    content_hash=u'a7d9abfad87bdbcf808cb9461222f7
     5b0fd914e7ed0b7ed6c9464ef4667bbd88')
```

1.3.2 Ponudniki

V začetku poglavja smo si izbrali 3 glavne ponudnike, ki uporabnikom ponujajo brezplačne storitve. S pomočjo orodja Wireshark smo skušali pridobiti IP naslov oblačne storitve kamor pošiljamo datoteke, s storitvijo Ipdata [8] pa smo določili približno lokacijo, kje naj bi se strežniki nahajali. Za vsakega izmed njih smo nato še z orodji ping in tracert preizkusili koliko časa potrebuje posamezen paket, da pride do strežnika, ter po kateri poti potuje. Pri ugotavljanju lokacije smo naleteli na nekaj težav, saj za Dropbox različne storitve dajejo različne rezultate za isti IP naslov. Tako smo za pridobljen IP naslov dobili podatek da se nahaja v Nemčiji, medtem ko nekaj drugih storitev ta naslov postavlja v San Francisco.

Meritve iz Ribnice

Meritve so bile izvedene 19.4.2018 iz Ribnice. Na vsak naslov smo poslali 500 paketov v velikosti 32 B in v tabelo 1.1 vključili glavne podatke, rezultate izvedbe klica tracert pa lahko vidimo v izpisih 1.2, 1.3, 1.4.

Ponudniki	*Lokacija strežnikov*	*Povprečni čas za 1 paket*	*Št. izgubljenih paketov*
Dropbox	Frankfurt (Nemčija)	38 ms	2
Mega	Luksemburg (Luksemburg)	50 ms	0
Google Drive	Kalifornija (ZDA)	37 ms	1

Tabela 1.1: Podatki o lokaciji ponudnikov.

Kot je razvidno v tabeli 1.1, so časi za Mego občutno daljši v primerjavi z Dropboxom in Googlom, zato lahko sklepamo, da je eden izmed skokov paketa na poti iz Ribnice počasnejši, zaradi česar je daljši tudi končni čas pinga.

Listing 1.2: Primer klica tracert za strežnik Dropboxa iz Ribnice.

```
 1    1 ms    <1 ms    <1 ms whrhpgn [192.168.0.1]
 2   81 ms    70 ms    17 ms
    upc.si.94.140.84.1.dc.cable.static.telemach.net [94.140.84.1]
 3   93 ms    90 ms   150 ms 217-72-74-9.ipv4.telemach.net [217.72.74.9]
 4   36 ms    42 ms    47 ms 185.66.148.237.ipv4.telemach.net
    [185.66.148.237]
 5   46 ms    39 ms    55 ms 185.66.148.237.ipv4.telemach.net
    [185.66.148.237]
 6   94 ms    80 ms    54 ms 185.66.148.219.ipv4.telemach.net
    [185.66.148.219]
 7   83 ms   128 ms   111 ms 185.66.148.219.ipv4.telemach.net
    [185.66.148.219]
 8   67 ms    60 ms    68 ms 10ge1-5.core1.vie1.he.net [216.66.82.73]
 9   72 ms    95 ms    87 ms 100ge14-2.core1.prg1.he.net [72.52.92.185]
10   72 ms    69 ms   103 ms 100ge16-1.core1.fra1.he.net
   [184.105.213.233]
11    *        *        *    Request timed out.
12  106 ms    79 ms   105 ms 185.45.10.167
13   68 ms    87 ms   105 ms 162.125.66.8
```

Listing 1.3: Primer klica tracert za strežnik Mega iz Ribnice.

```
 1    3 ms     2 ms     1 ms whrhpgn [192.168.0.1]
 2   16 ms    11 ms    15 ms
    upc.si.94.140.84.1.dc.cable.static.telemach.net [94.140.84.1]
 3   12 ms    15 ms    11 ms 217-72-74-9.ipv4.telemach.net [217.72.74.9]
 4   11 ms    13 ms    12 ms 185.66.148.237.ipv4.telemach.net
    [185.66.148.237]
 5   10 ms    12 ms    11 ms 185.66.148.237.ipv4.telemach.net
    [185.66.148.237]
 6   18 ms    18 ms    18 ms 185.66.148.219.ipv4.telemach.net
    [185.66.148.219]
 7   72 ms   144 ms   241 ms 185.66.148.219.ipv4.telemach.net
    [185.66.148.219]
 8   18 ms    19 ms    17 ms 10ge1-5.core1.vie1.he.net [216.66.82.73]
 9  266 ms   240 ms    24 ms 100ge14-2.core1.prg1.he.net [72.52.92.185]
10   31 ms    28 ms    35 ms 100ge16-1.core1.fra1.he.net
   [184.105.213.233]
11   34 ms    44 ms    35 ms 10ge11-1.core1.lux1.he.net [184.105.64.14]
12  197 ms   288 ms   274 ms AS24611-1.members.lu-cix.lu [188.93.171.3]
13   47 ms    42 ms    43 ms 80.92.71.58
14   46 ms    45 ms    45 ms 31.216.145.85
```

Listing 1.4: Primer klica tracert za strežnik Google Drive iz Ribnice.

```
 1    2 ms     2 ms     2 ms whrhpgn [192.168.0.1]
 2   12 ms    13 ms    22 ms
```

```
    upc.si.94.140.84.1.dc.cable.static.telemach.net [94.140.84.1]
 3   13 ms   13 ms   10 ms 217-72-74-9.ipv4.telemach.net [217.72.74.9]
 4   15 ms   11 ms   11 ms 185.66.148.237.ipv4.telemach.net
    [185.66.148.237]
 5  263 ms   11 ms   13 ms 185.66.148.237.ipv4.telemach.net
    [185.66.148.237]
 6   20 ms   16 ms   19 ms 185.66.148.220.ipv4.telemach.net
    [185.66.148.220]
 7   21 ms   16 ms   16 ms 185.66.148.220.ipv4.telemach.net
    [185.66.148.220]
 8   16 ms   16 ms   17 ms peer-AS31042.sbb.rs [82.117.193.217]
 9   32 ms   60 ms  488 ms bg-tp-m-0-be1.sbb.rs [89.216.5.77]
10   45 ms   31 ms   30 ms 72.14.219.230
11  168 ms  367 ms  228 ms 209.85.243.121
12   34 ms   34 ms   34 ms 66.249.94.183
13  110 ms  217 ms  264 ms bud02s24-in-f234.1e100.net [216.58.214.234]
```

Meritve iz Vodic

Meritve so bile izvedene 20.4.2018 iz Vodic. Na vsak naslov smo poslali 500 paketov v velikosti 32 B in v tabelo 1.2 vključili glavne podatke, rezultate izvedbe klica tracert pa lahko vidimo v izpisih 1.5 , 1.6 , 1.7.

Ponudniki	*Lokacija strežnikov*	*Povprečni čas za 1 paket*	*Št. izgubljenih paketov*
Dropbox	San Francisco (Kalifornija)	122 ms	0
Google Drive	Mountain View (Kalifornija)	39 ms	0
Mega	Luksemburg (Luksemburg)	34 ms	0

Tabela 1.2: Podatki o lokaciji ponudnikov.

Rezultate izvedbe klica ping lahko vidimo v tabeli 1.2.

Listing 1.5: Primer klica tracert za strežnik Dropbox iz Vodic.

```
 1    1 ms    1 ms    1 ms DD-WRT [192.168.1.1]
 2   13 ms   21 ms   18 ms 93-103-0-1.gw.t-2.net [93.103.0.1]
 3   12 ms   13 ms   13 ms 84-255-209-153.core.t-2.net [84.255.209.153]
 4   13 ms   12 ms   13 ms 212.162.28.9
 5    *       *     111 ms ae-2-3601.ear2.Washington1.Level3.net
    [4.69.206.73]
 6    *       *       *    Request timed out.
 7  118 ms  119 ms  118 ms ae2-iad6-bb02.net.dropbox.com [45.58.66.66]
 8  128 ms  124 ms  123 ms ae16-iad4-dr01.net.dropbox.com [45.58.66.79]
 9    *       *       *    Request timed out.
10    *       *       *    Request timed out.
11    *       *       *    Request timed out.
12  117 ms  117 ms  116 ms 162.125.18.133
```

Listing 1.6: Primer klica tracert za strežnik Google Drive iz Vodic.

```
 1    1 ms    1 ms    2 ms DD-WRT [192.168.1.1]
 2   70 ms   70 ms   72 ms 93-103-0-1.gw.t-2.net [93.103.0.1]
 3   56 ms   51 ms   53 ms 84-255-250-237.core.t-2.net [84.255.250.237]
 4   91 ms   90 ms   37 ms google.telecity-2-equinix-am7.nl-ix.net
    [193.239.117.142]
 5   37 ms   37 ms   38 ms 108.170.241.225
 6   36 ms   36 ms   36 ms 72.14.239.45
 7   37 ms   36 ms   36 ms ams16s31-in-f10.1e100.net [172.217.19.202]
```

Listing 1.7: Primer klica tracert za strežnik Mega iz Vodic.

```
 1    1 ms    1 ms    1 ms DD-WRT [192.168.1.1]
 2   14 ms   13 ms   13 ms 93-103-0-1.gw.t-2.net [93.103.0.1]
 3   13 ms   13 ms   13 ms 84-255-209-153.core.t-2.net [84.255.209.153]
 4   13 ms   13 ms   13 ms 212.162.28.13
 5    *      24 ms   24 ms ae-1-3103.ear3.Frankfurt1.Level3.net
    [4.69.163.86]
 6   24 ms   24 ms   25 ms Cogent-level3-200G.Frankfurt1.Level3.net
    [4.68.111.178]
 7   30 ms   30 ms   29 ms be2846.ccr42.fra03.atlas.cogentco.com
    [154.54.37.29]
 8   35 ms   34 ms   34 ms be2377.rcr21.lux01.atlas.cogentco.com
    [154.54.38.146]
 9   35 ms   35 ms   35 ms
    be3456.nr51.b038844-0.lux01.atlas.cogentco.com [154.25.12.70]
10   40 ms   35 ms   34 ms 149.6.66.198
11   34 ms   34 ms   35 ms lu2.api.mega.nz [31.216.147.133]
```

Meritve iz Ljubljane

Meritve so bile izvedene 13.5.2018 iz Ljubljane. Na vsak naslov smo poslali 500 paketov v velikosti 32 B in v tabelo 1.3 vključili glavne podatke, rezultate izvedbe klica tracert pa lahko vidimo v izpisih 1.8 , 1.9.

Ponudniki	*Lokacija strežnikov*	*Povprečni čas za 1 paket*	*Št. izgubljenih paketov*
Dropbox	San Francisco (Kalifornija)	22 ms	1
Mega	Luksemburg (Luksemburg)	28 ms	1

Tabela 1.3: Podatki o lokaciji ponudnikov.

Listing 1.8: Primer klica tracert za strežnik Dropbox iz Ljubljane.

```
 1    2 ms    1 ms    1 ms 192.168.1.1
 2    2 ms    4 ms    1 ms 89-212-0-1.gw.t-2.net [89.212.0.1]
```

```
 3    4 ms    1 ms    1 ms  89-212-88-65.dynamic.dsl.t-2.net
     [89.212.88.65]
 4    3 ms    1 ms    2 ms  84.255.208.230
 5    *       *       *     Request timed out.
 6   22 ms   23 ms   22 ms  185.45.10.167
 7   21 ms   21 ms   20 ms  162.125.66.8
```

Listing 1.9: Primer klica tracert za strežnik Mega iz Ljubljane.

```
 1    2 ms    2 ms    1 ms  192.168.1.1
 2    2 ms    1 ms    6 ms  89-212-0-1.gw.t-2.net [89.212.0.1]
 3    1 ms    1 ms    1 ms  84-255-250-201.core.t-2.net [84.255.250.201]
 4   12 ms   10 ms   10 ms  win-b4-link.telia.net [213.248.104.157]
 5   27 ms   27 ms   27 ms  be1299.ccr51.vie01.atlas.cogentco.com
     [130.117.14.89]
 6   24 ms   34 ms   23 ms  be2974.ccr21.muc03.atlas.cogentco.com
     [154.54.58.5]
 7   22 ms   23 ms   22 ms  be2959.ccr41.fra03.atlas.cogentco.com
     [154.54.36.53]
 8   26 ms   26 ms   28 ms  be2376.rcr21.lux01.atlas.cogentco.com
     [130.117.50.70]
 9   27 ms   33 ms   27 ms
     be3456.nr51.b038844-0.lux01.atlas.cogentco.com [154.25.12.70]
10   27 ms   27 ms   29 ms  149.6.66.198
11   27 ms   27 ms   27 ms  31.216.144.31
```

Meritve s fakultete

Ker smo nekatere meritve izvedli tudi na Fakulteti za računalništvo in informatiko v Ljubljani, smo vključili še podatke za ping in tracert iz omrežja fakultete in jih nademo v tabeli 1.4 ter izpisi 1.10, 1.11, 1.12.

Ponudniki	*Lokacija strežnikov*	*Povprečni čas za 1 paket*	*Št. izgubljenih paketov*
Dropbox	San Francisco (Kalifornija)	129 ms	0
Google Drive	Dublin (Irska)	65 ms	0
Mega	Luxemburg	42 ms	0

Tabela 1.4: Podatki o lokaciji ponudnikov.

Listing 1.10: Primer klica tracert za strežnik Dropbox iz območja fakultete.

```
 1    3 ms    4 ms    1 ms  193.2.176.1
 2    2 ms    5 ms    8 ms  vul-cfkkt.uni-lj.si [193.2.96.114]
 3    4 ms    3 ms    2 ms  arnesul-vul.uni-lj.si [193.2.96.1]
 4    3 ms    3 ms    6 ms  vl819.larnes7.cpe.arnes.si [178.172.80.172]
 5    4 ms    3 ms    2 ms  vl475.larnes6.bb.arnes.si [88.200.2.182]
```

```
 6    3 ms    1 ms    1 ms 1.irb.rarnes1.bb.arnes.si [88.200.7.240]
 7    2 ms    1 ms    3 ms 2.irb.rarnes2.bb.arnes.si [88.200.7.239]
 8    3 ms    2 ms    2 ms
    be4053.nr11.b021176-0.lju01.atlas.cogentco.com [149.6.52.97]
 9    3 ms    2 ms    2 ms te0-0-2-2.rcr11.lju01.atlas.cogentco.com
    [154.25.3.85]
10    9 ms   13 ms    8 ms te0-1-1-10.ccr51.vie01.atlas.cogentco.com
    [130.117.48.81]
11   15 ms   14 ms   15 ms be2974.ccr21.muc03.atlas.cogentco.com
    [154.54.58.5]
12   21 ms   21 ms   20 ms be2959.ccr41.fra03.atlas.cogentco.com
    [154.54.36.53]
13   27 ms   26 ms   26 ms be2813.ccr41.ams03.atlas.cogentco.com
    [130.117.0.121]
14  106 ms  205 ms  207 ms be12194.ccr41.lon13.atlas.cogentco.com
    [154.54.56.93]
15  144 ms  206 ms  205 ms be2982.ccr31.bos01.atlas.cogentco.com
    [154.54.1.117]
16  101 ms  103 ms  101 ms be3471.ccr41.jfk02.atlas.cogentco.com
    [154.54.40.154]
17  109 ms  110 ms  110 ms be2806.ccr41.dca01.atlas.cogentco.com
    [154.54.40.106]
18  110 ms  110 ms  110 ms be3083.ccr41.iad02.atlas.cogentco.com
    [154.54.30.54]
19    *       *       *    Request timed out.
20  213 ms  205 ms  206 ms ae2-iad6-bb02.net.dropbox.com [45.58.66.66]
21  163 ms  205 ms  206 ms ae15-iad4-dr02.net.dropbox.com [45.58.66.81]
22    *       *       *    Request timed out.
23    *       *       *    Request timed out.
24    *       *       *    Request timed out.
25  149 ms  110 ms  204 ms 162.125.18.133
```

Listing 1.11: Primer klica tracert za strežnik Google Drive iz območja fakultete.

```
 1    *      11 ms    *    2001:1470:ffef:fe01::1
 2   61 ms    7 ms   11 ms 2001:1470:ffff:1900::2
 3    *       *       *    Request timed out.
 4    *     217 ms    *    larnes7-v6-v829.arnes.si
    [2001:1470:3004:1::1]
 5    4 ms    *     135 ms 2001:1470:a:16::a
 6  811 ms  128 ms  804 ms 2001:1470:a:60::a
 7  868 ms   44 ms  257 ms arnes-ias-geant-gw.lju.si.geant.net
    [2001:798:1::11]
 8  138 ms   58 ms  115 ms ae2.mx1.vie.at.geant.net
    [2001:798:cc:2d01:1b01::9]
 9  275 ms    *     339 ms 2001:798:cc:1001:1e01::6
10   91 ms    *     824 ms 2001:4860:1:1:0:51e5:0:1
11    *     419 ms  145 ms 2001:4860:0:101b::1
12    *     144 ms  428 ms 2001:4860:0:1::1f49
```

```
13    *        *        *     Request timed out.
14   346 ms   39 ms  118 ms mil04s26-in-x0a.1e100.net
    [2a00:1450:4002:808::200a]
```

Listing 1.12: Primer klica tracert za strežnik Mega iz območja fakultete.

```
 1    6 ms    2 ms    3 ms 2001:1470:ffef:fe01::1
 2    3 ms    1 ms    4 ms 2001:1470:ffff:1900::2
 3    2 ms    3 ms    2 ms 2001:1470:ffff:1::1
 4    4 ms    2 ms    1 ms larnes7-v6-v829.arnes.si
     [2001:1470:3004:1::1]
 5    2 ms    1 ms    1 ms 2001:1470:a:16::a
 6    1 ms    2 ms    2 ms 2001:1470:a:60::a
 7    1 ms    2 ms    1 ms arnes-ias-geant-gw.lju.si.geant.net
     [2001:798:1::11]
 8   27 ms   26 ms   26 ms ae2.mx1.vie.at.geant.net
     [2001:798:cc:2d01:1b01::9]
 9   26 ms   26 ms   29 ms ae6.2.mx1.fra.de.geant.net
     [2001:798:cc:1::5a]
10   27 ms   27 ms   26 ms ae1.mx1.ams.nl.geant.net
    [2001:798:cc:1401:2201::a]
11   32 ms   26 ms   28 ms nordunet-ias-nordunet-gw.ams.nl.geant.net
    [2001:798:1::126]
12   51 ms   55 ms   79 ms ch-gva.nordu.net [2001:948:1:e::1]
13   51 ms   76 ms   51 ms de-ffm.nordu.net [2001:7f8::a2b:0:1]
14   36 ms   37 ms   36 ms ae-10.r03.frnkge03.de.bb.gin.ntt.net
    [2001:7f8::b62:0:1]
15   44 ms   40 ms   41 ms ae-1.r00.lxmblc01.lu.bb.gin.ntt.net
    [2001:728:0:2000::1e2]
16   40 ms   40 ms   42 ms 2001:728:0:5000::33a
17   40 ms   40 ms   40 ms lu1.api.mega.nz [2001:67c:1998:202::184]
```

Povzetek rezultatov izvedbe klica ping iz različnih lokacij smo zapisali v tabeli 1.5

Ponudniki	*Povprečni čas iz Ribnice*	*Povprečni čas iz Ljubljane*	*Povprečni čas iz Vodic*	*Povprečni čas s fakultete*
Dropbox	38 ms	22 ms	122 ms	129 ms
Mega	50 ms	28 ms	34 ms	65 ms
Google Drive	37 ms		39 ms	42 ms

Tabela 1.5: Podatki o lokaciji ponudnikov.

1.4 Rezultati meritev

Pri meritvah smo si izbrali 3 tipične velikosti datotek in sicer datoteko velikosti 200kB, ki predstavlja primer velikosti Wordovega dokumenta, datoteko velikosti

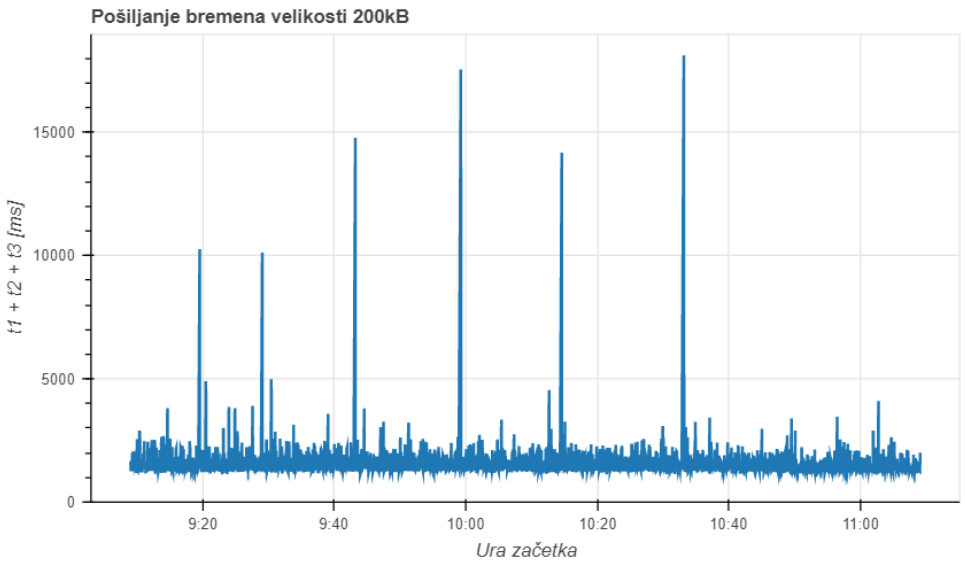
5MB, ki predstavlja velikost digitalne fotografije ter datoteko velikosti 20MB, ki ponazarja kratek videoposnetek. Pri meritvah smo uporabljali dva načina pošiljanja datotek. Za nekatere meritve smo uporabljali vsak svoj uporabniški račun, kjer pa je bilo to potrebno, pa smo vsi pošiljali datoteke na isti uporabniški račun.

1.4.1 Vpliv lokacije na čas prenosa

Da bi ugotovili kako lokacija ter pasovna širina pošiljatelja vplivata na rezultate, smo najprej izbrali samo enega ponudnika (Dropbox) in vsi izvedli enake meritve na svojem domu. Vsak izmed nas je uporabil lasten uporabniški račun. Pred vsakim pošiljanjem se je generirala nova datoteka z vnaprej podano velikostjo. Novo pošiljanje se začne takoj zatem, ko dobimo potrditev, da je prejšnja datoteka uspešno shranjena v oblaku. Te meritve smo izvajali 120 minut za vsako izbrano velikost datoteke. Pričakujemo, da se bodo med meritvami pojavljale razlike v izmerjenih časih, predvsem zaradi razlike v pasovni širini povezav.

Meritve iz Ribnice

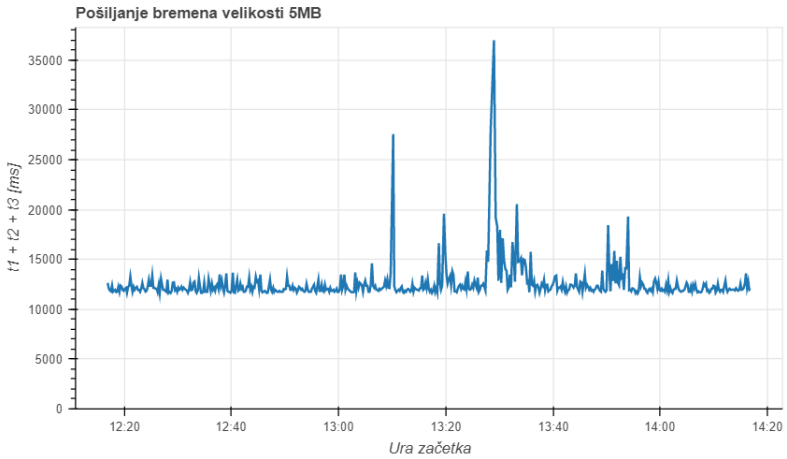
Vse meritve, ki so potekale v Ribnici, so uporabljale povezavo 16/4 Mbps.



Slika 1.2: Rezultati pošiljanja datoteke velikosti 200kB iz Ribnice.

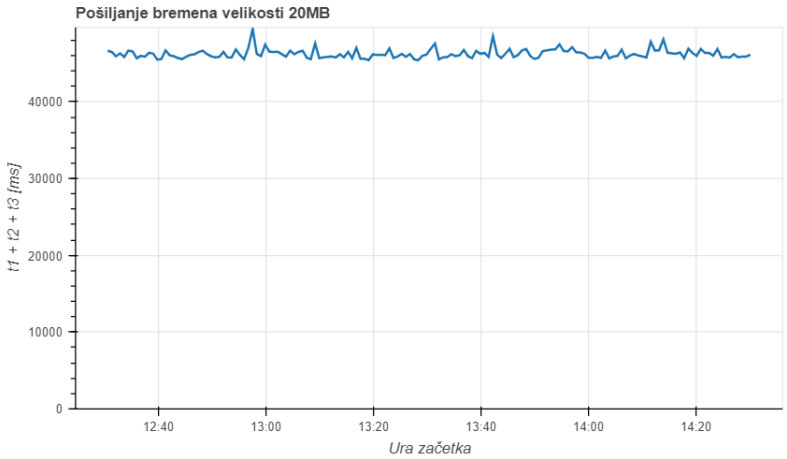
Meritve, ki jih lahko vidimo na sliki 1.2 so potekale v petek 6.4. v času od 09:09 do 11:09 po poletnem srednjeevropskem času (UTC+2) iz Ribnice. V dveh urah je bilo skupno prenešenih 4.643 datotek, povprečni čas prenosa pa je znašal

1.547 ms. Razen nekaj izrazitih izstopanj, ki jih lahko opazimo v rezultatih na sliki 1.2, so časi meritev dokaj konstantni.



Slika 1.3: Rezultati pošiljanja datoteke velikosti 5MB iz Ribnice.

Test z velikostjo datoteke 5 MB na sliki 1.3 se je izvajal 7.4., od 12:16 do 14:16 (UTC+2). Podobno kot na sliki 1.2 tudi tukaj opazimo, da časi večinoma ne odstopajo preveč od povprečja, pojavi pa se nekaj krajših obdobij v katerih so časi daljši od povprečja. Povprečni čas prenosa v tem primeru znaša 11.582 ms, skupno pa je bilo poslanih 575 datotek.

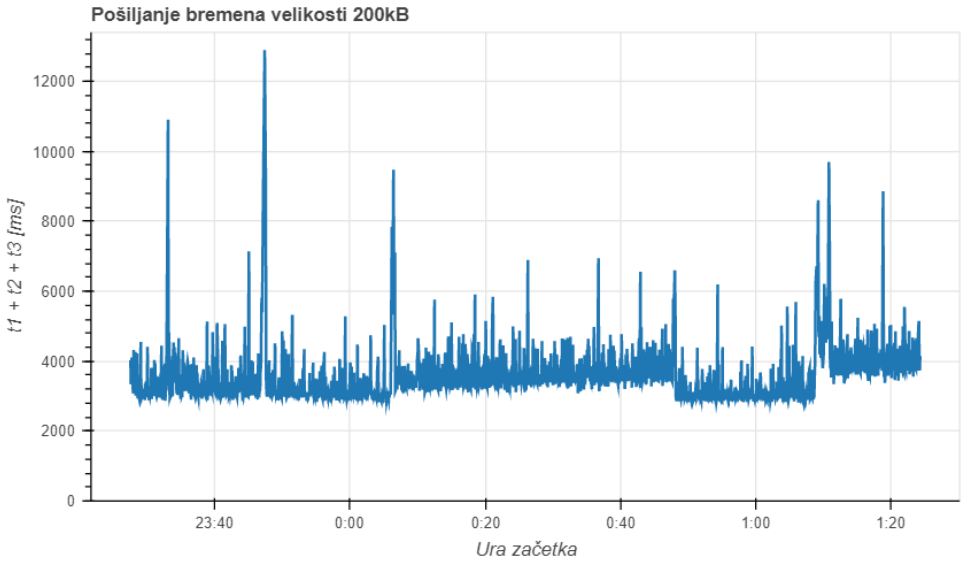


Slika 1.4: Rezultati pošiljanja datoteke velikosti 20MB iz Ribnice.

Meritve datotek velikosti 20 MB, na sliki 1.4, so tekle 23.5. od 12:29 do 14:29 (UTC+2). Povprečen prenos datoteke je trajal 46.231 ms, skupno pa je bilo v 2 urah poslanih 156 datotek velikosti 20 MB. Za razliko od prejšnjih meritev predstavljenih na slikah 1.2 in 1.3 lahko ob rezultatih na sliki 1.4 opazimo, da tokrat ni prišlo do večjih odstopanj in se čas prenosa za posamezno datoteko med merjenjem skoraj ni spreminjal.

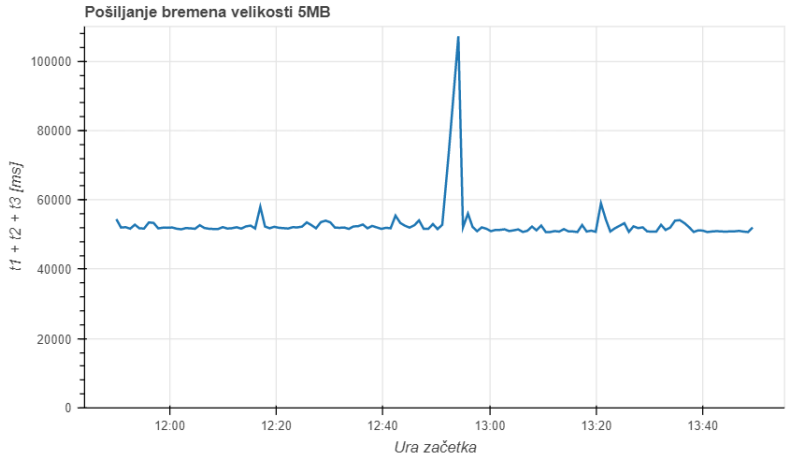
Meritve v Vodicah

Pasovna širina povezave, ki se je uporaljala za pošiljanje datotek iz Vodic, je bila 17/1 Mbps.



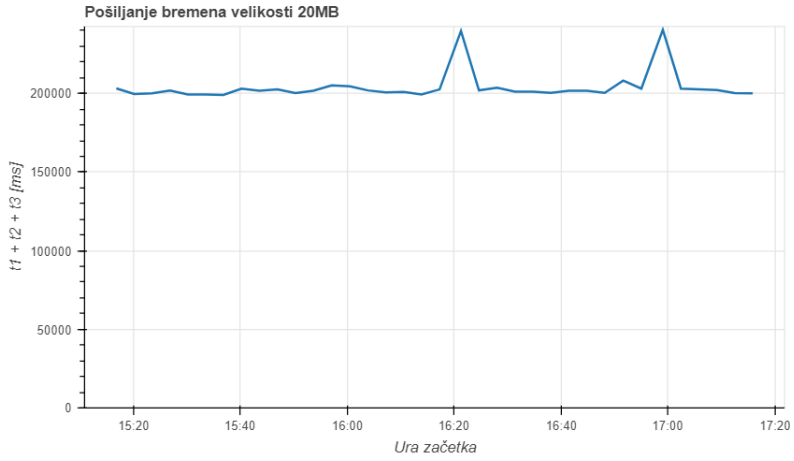
Slika 1.5: Rezultati pošiljanja datoteke velikosti 200kB iz Vodic.

Meritve, ki jih lahko vidimo na sliki 1.5 so potekale v petek, 6.4. v času od 23:27 do 7.4, 01:24 po poletnem srednjeevropskem času (UTC+2). Skupno je bilo prenešenih 1.999 datotek, povprečni čas prenosa pa je znašal 3.501 ms.



Slika 1.6: Rezultati pošiljanja datoteke velikosti 5MB iz Vodic.

Test z velikostjo datoteke 5 MB 1.6 se je izvajal 7.4., od 11:49 do 13:49 (UTC+2). Podobno kot na sliki 1.5 tudi tukaj opazimo da časi večinoma ne odstopajo preveč od povprečja, z izjemo večjega skoka pri 12:54, ki je trajal kar 107.204 ms. Povprečni čas prenosa v tem primeru znaša 52.638 ms, skupno pa je bilo poslanih 137 datotek.

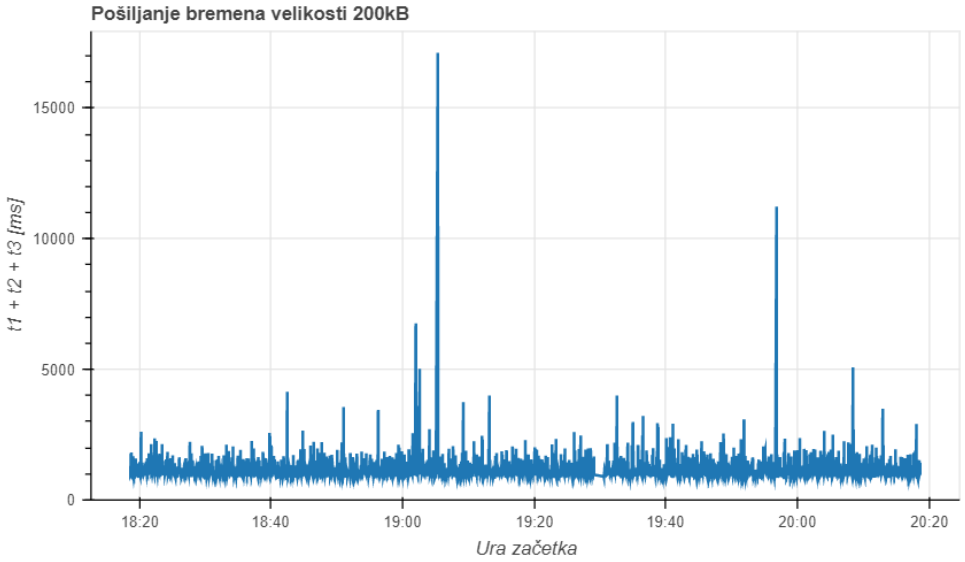


Slika 1.7: Rezultati pošiljanja datoteke velikosti 20MB iz Vodic.

Meritve datotek velikosti 20 MB, na sliki 1.7, so tekle 7.4. od 15:13 do 17:15 (UTC+2). Povprečen čas shranjevanja datoteke znaša 203.840 ms, skupno pa je bilo poslanih 36 datotek velikosti 20 MB. V primerjavi s prejšnimi meritvami so bili časi najbolj konstantni, brez večjih skokov.

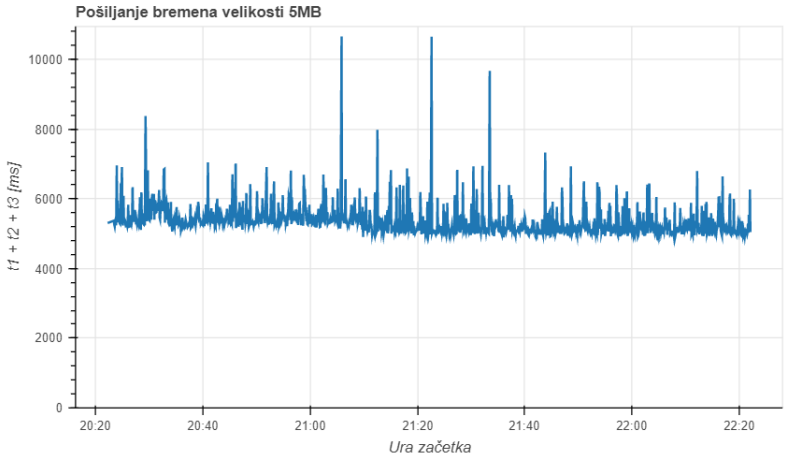
Meritve v Ljubljani

Meritve v Ljubljani so potekale s hitrostjo povezave 40/10 Mbps.



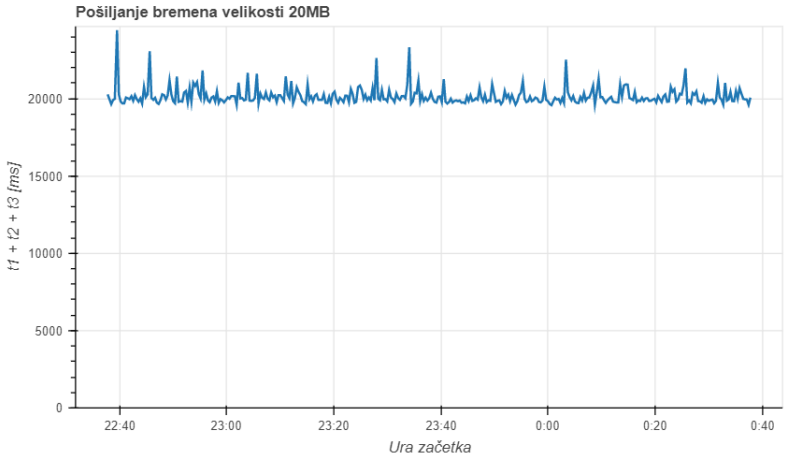
Slika 1.8: Rezultati pošiljanja datoteke velikosti 200kB iz Ljubljane.

Meritve, ki jih lahko vidimo na sliki 1.8 so potekale v petek 7.4. v času od 18:18 do 20:18 po poletnem srednjeevropskem času (UTC+2). V dveh urah je bilo skupno prenešenih 6.526 datotek, povprečni čas prenosa pa je znašal 1.095 ms. Razen nekaj izrazitih izstopanj, ki jih lahko opazimo v rezultatih na sliki 1.8, so časi meritev dokaj konstantni. Pomembno je omeniti, da sta daljša izmerjena časa ob 19:03 in 19:56 posledica ročnega sproščanja pomnilnega prostora na Dropbox-u, ki je bilo potrebno zaradi višje hitrosti pošiljanja in posledično tudi večjega števila shranjenih datotek v oblaku. Maksimalni izmerjen čas 88.594 ms smo zaradi preglednosti izpustili.



Slika 1.9: Rezultati pošiljanja datoteke velikosti 5MB iz Ljubljane.

Meritve, ki jih lahko vidimo na sliki 1.9, so potekale v petek, 7.4. v času od 20:22 do 22:22 po poletnem srednjeevropskem času (UTC+2). V dveh urah je bilo skupno prenešenih 1.330 datotek, povprečni čas prenosa pa je znašal 5.386 ms. Razen treh večjih izstopanj, ki jih lahko opazimo v rezultatih 1.9, so časi meritev dokaj konstantni. Okoli 21:10 se je povprečni čas pošiljanja malenkost zmanjšal. Ponovno lahko nekatere daljše meritve označimo kot posledico sproščanja pomnilnega prostora na testiranem ponudniku. Maksimalni izmerjeni čas 62.827 ms smo zaradi preglednosti izpustili.



Slika 1.10: Rezultati pošiljanja datoteke velikosti 20MB iz Ljubljane.

Meritve, ki jih lahko vidimo na sliki 1.10, so potekale v petek, 7.4. v času od 22:37 do 00:37 po poletnem srednjeevropskem času (UTC+2) iz Ljubljane.

V dveh urah je bilo skupno prenešenih 355 datotek, povprečni čas prenosa pa je znašal 20.140 ms. Opazimo lahko konstante izmerjene čase brez večjih odstopanj.

Povzetek vpliva lokacije

Velikost bremena	*Minimalni čas*	*Povprečje*	*Maksimalen čas*
200 kB	1.116 ms	1.547 ms	18.126 ms
5 MB	11.582 ms	12.484 ms	36.997 ms
20 MB	45.391 ms	46.231 ms	49.464 ms

Tabela 1.6: Minimalni, povprečni in maksimalni časi pošiljanja bremena na Dropbox, s povezavo 16/4 Mbps iz Ribnice.

Velikost bremena	*Minimalni čas*	*Povprečje*	*Maksimalen čas*
200 kB	2.815 ms	3.501 ms	12.897 ms
5 MB	50.735 ms	52.638 ms	107.204 ms
20 MB	199.080 ms	203.840 ms	240.494 ms

Tabela 1.7: Minimalni, povprečni in maksimalni časi pošiljanja bremena na Dropbox, s povezavo 17/1 Mbps iz Vodic.

Velikost bremena	*Minimalni čas*	*Povprečje*	*Maksimalen čas*
200 kB	799 ms	1.095 ms	88.594 ms
5 MB	4.883 ms	5.386 ms	62.827 ms
20 MB	19.575 ms	20.140 ms	24.420 ms

Tabela 1.8: Minimalni, povprečni in maksimalni časi pošiljanja bremena na Dropbox, s povezavo 40/10 Mbps iz Ljubljane.

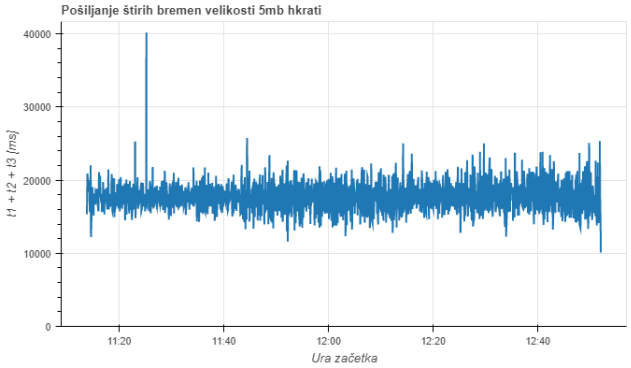
Iz prvotnih meritev, opravljenih na različnih lokacijah opazimo, da se pojavijo odstopanja, odvisna od lokacije uporabnika ter tudi njegove pasovne širine. Tako so najkrajši časi pošiljanja doseženi v Ljubljani v tabeli 1.8, najdaljši pa v Vodicah v tabeli 1.7. Odstopanja pri manjših datotekah še niso tako visoka. Med Ribnico in Ljubljano je pri velikosti 200 kB razlika v časih pošiljanja bremena le nekaj manj kot 500 ms, vendar večja kot je datoteka, višje so razlike v časih pošiljanja. Tako je čas pošiljanja datoteke velikosti 20 MB v Ribnici več kot dvakrat daljši v primerjavi z Ljubljano.

Zaradi omejitve Dropboxa na 2 GB prostora za brezplačne uporabniške račune, smo se že v tem poglavju soočili z nekaj težavami, ko smo med svojimi meritvami zasedli ves dodeljen prostor. Ker smo v nadaljnih meritvah pričakovali povečanje števila datotek smo se v orodje Benchcloud odločili vključiti še dodatno nit, ki periodično (na 30 sekund) preverja stanje zasedonosti pomnilnika in v primeru, da na njem ostane manj kot 500 MB prostora, vse shranjene

datoteke v oblaku izbriše in s tem sprosti dodaten prostor za testiranje. V vseh naslednjih meritvah je bila ta funkcionalnost že uporabljena, tako pa smo se izognili prekinjanju meritev, ki bi ga lahko povzročilo pomanjkanje prostora v oblaku.

1.4.2 Vpliv pošiljanja več datotek hkrati na čas prenosa

Želeli smo ugotoviti kako pošiljanje več različnih datotek hkrati vpliva na čas pošiljanja. Meritve smo izvajali v Ljubljani s povezavo 40/10 Mbps. Poslali smo 1330 datotek, kolikor smo jih v 2 urah uspeli poslati v prejšnjem poskusu predstavljenemu na sliki 1.9, velikosti 5 MB, pošiljanje pa so opravljale 4 niti vzporedno. Vsaka nit je datoteke pošiljala neodvisno od ostalih.



Slika 1.11: Rezultati pošiljanja štirih datotek velikosti 5MB hkrati iz Ljubljane.

Vrsta pošiljanja	*Minimalni čas*	*Povprečje*	*Maksimalen čas*	*Celoten čas*
4 niti hkrati	10.104 ms	17.727 ms	40.184 ms	98,43 min
1 nit	4.883 ms	5.386 ms	62.827 ms	120,00 min

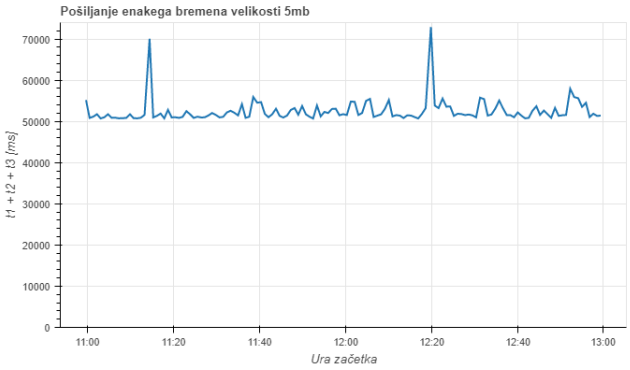
Tabela 1.9: Minimalni, povprečni, maksimalni časi in celoten čas pošiljanja bremena na Dropbox, s povezavo 40/10 Mbps iz Ljubljane.

Pošiljanje istega bremena s štirimi datotekami hkrati, kot prikazuje slika 1.11, je skupno trajalo 98,43 minut, kot vidimo v tabeli 1.9, kar je približno 18% hitreje kot pošiljanje vsake datoteke posebej, povprečen čas za posamezno datoteko pa je bil približno 3-krat daljši. Taki rezultati ustrezajo tudi ugotovitvam iz dela [4], ki pravijo da se v primeru večnitnega pošiljanja bremen skupni čas za enako število datotek zmanjša.

1.4.3 Vpliv pošiljanja enakih datotek na čas prenosa

Želeli smo ugotoviti, če bi bilo pošiljanje enakih datotek hitrejše, kot pošiljanje različnih datotek enake velikosti. Zanimalo nas je, če bi se sčasoma skrajšal

čas prenosa. Meritve smo izvajali v Vodicah s povezavo 17/1 Mbps. Pošiljali smo 120 minut datoteke velikosti 5 MB z enako vsebino in kasneje primerjali rezultate s poskusom na sliki 1.6.



Slika 1.12: Rezultati pošiljanja enakih datotek velikosti 5MB iz Vodic.

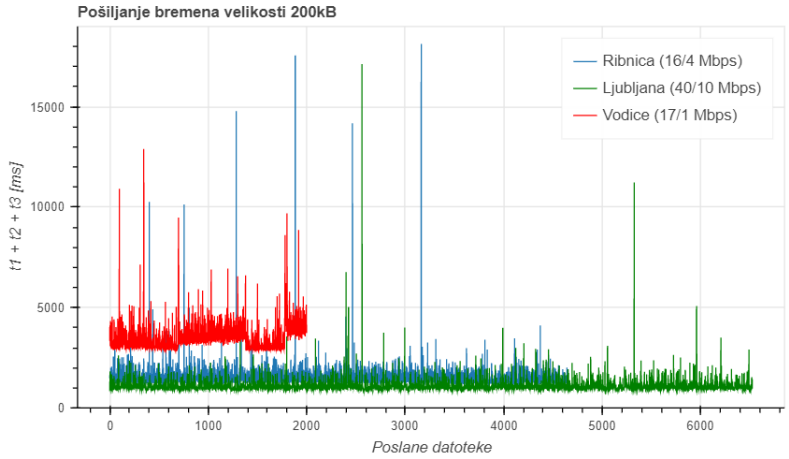
Datoteke	*Minimalni čas*	*Povprečje*	*Maksimalen čas*	*Št. pošiljanj*
Enake	50.683 ms	52.443 ms	72.868 ms	138
Različne	50.735 ms	52.638 ms	107.204 ms	137

Tabela 1.10: Minimalni, povprečni in maksimalni časi pošiljanja različnega in enakega bremena velikosti 5 MB, s povezavo 17/1 Mbps iz Vodic.

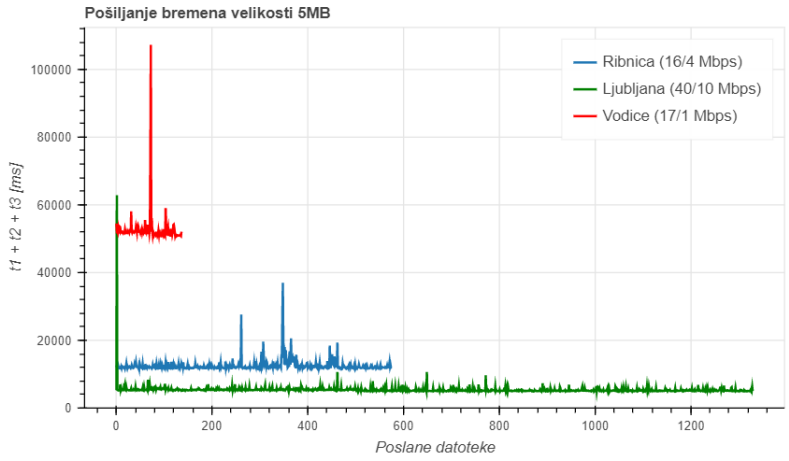
Kot vidimo v tabeli 1.10 in na sliki 1.12 je povprečje pošiljanja skoraj enako. Tudi število pošiljanj se skoraj ne razlikuje. Ugotovili smo, da pošiljanje enakih datotek ne vpliva na čas prenosa.

1.4.4 Pošiljanje iz 3 različnih lokacij hkrati

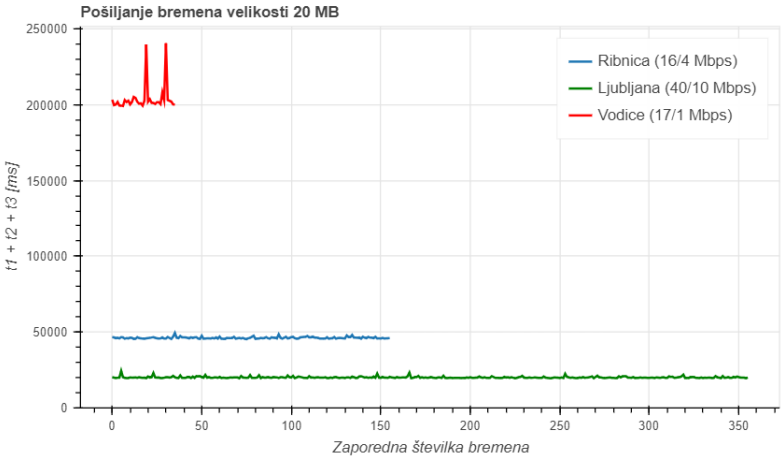
Da bi se prepričali, kako se ponudnik odziva na prejemanje bremen več različnih pošiljateljev, smo se odločili za hkratno izvajanje meritev iz 3 različnih lokacij hkrati. Za lažjo primerjavo smo za vsako velikost meritev, ki smo jih opravili v razdelku 1.4.1 posamično, združili na en graf v slikah 1.13, 1.14 in 1.15. Te meritve je vsak izvajal posamično skupno 2 uri in datoteke pošiljal na svoj lastni uporabniški račun pri oblačnem ponudniku. Ker smo te meritve izvajali 2 uri, se je število prenesenih datotek v tem obdobju razlikovalo med lokacijami, zaradi tega pa na grafih vidimo tudi različne dolžine krivulj.



Slika 1.13: Združeni rezultati posamičnega pošiljanja datotek v velikosti 200kB.



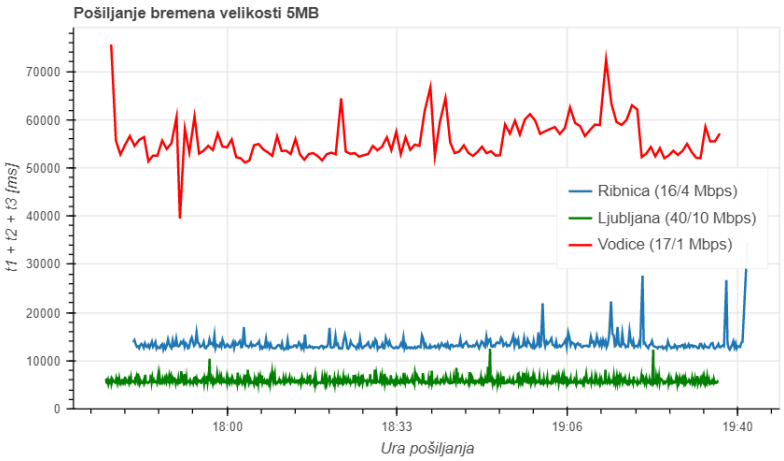
Slika 1.14: Združeni rezultati posamičnega pošiljanja datotek v velikosti 5MB.



Slika 1.15: Združeni rezultati posamičnega pošiljanja datotek v velikosti 20MB.

Vzporedno pošiljanje datotek velikosti 5 MB

Vsi trije smo v istem trenutku začeli meritve vsak iz svojega računalnika na različnih lokacijah ter ob tem uporabljali isti uporabniški račun. Pri tem smo dobili rezultate na slikah 1.16 in 1.17.



Slika 1.16: Rezultati vzporednega pošiljanja datotek v velikosti 5MB.

Meritve na sliki 1.16 so potekale 12.4. od v času od 17:36 do 19:36 (GMT +2). Spet lahko opazimo razlike v izmerjenih časih zaradi razlike v hitrostih povezav. Meritve, ki potekajo v Ljubljani so najhitrejše, medtem ko so meritve v Vodicah dale najdaljše čase in tudi največja nihanja pri prenašanju datotek.

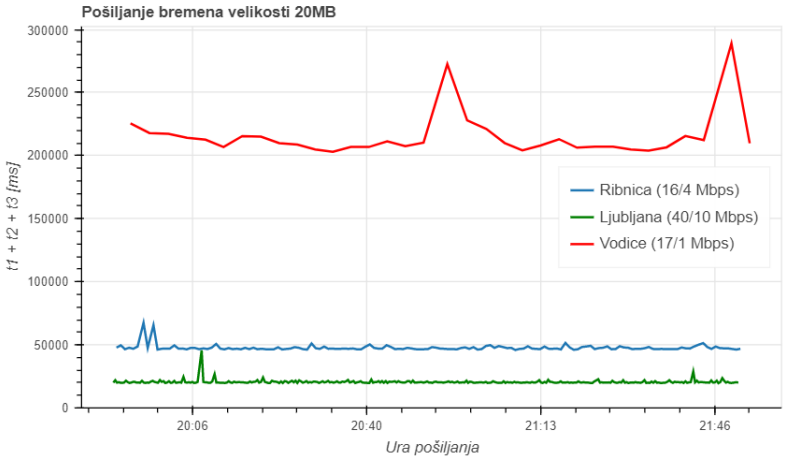
Lokacija	*Povprečni čas posamično*	*Povprečni čas skupinsko*
Ribnica	12.484 ms	13.424 ms
Vodice	52.683 ms	55.589 ms
Ljubljana	5.386 ms	5.806 ms

Tabela 1.11: Primerjava povprečnih časov pošiljanja ob posamičnih meritvah ter povprečnih časov skupinskega pošiljanja datotek velikosti 5 MB.

S pomočjo podatkov v tabeli 1.11 lahko razberemo, da so povprečni časi v primerjavi s posamičnimi meritvami narasli. Največjo spremembo opažamo v Vodicah (približno 3 s), najmanjšo pa v Ljubljani (okrog 500 ms).

Vzporedno pošiljanje datotek velikosti 20 MB

Meritve so na vseh treh lokacijah potekale 12.4. v času od 19:51 do 20:51 (GMT +2).



Slika 1.17: Rezultati vzporednega pošiljanja datotek v velikosti 20MB.

Lokacija	*Povprečni čas posamično*	*Povprečni čas skupinsko*
Ribnica	45.713 ms	47.682 ms
Vodice	203.480 ms	209.804 ms
Ljubljana	20.140 ms	20.575 ms

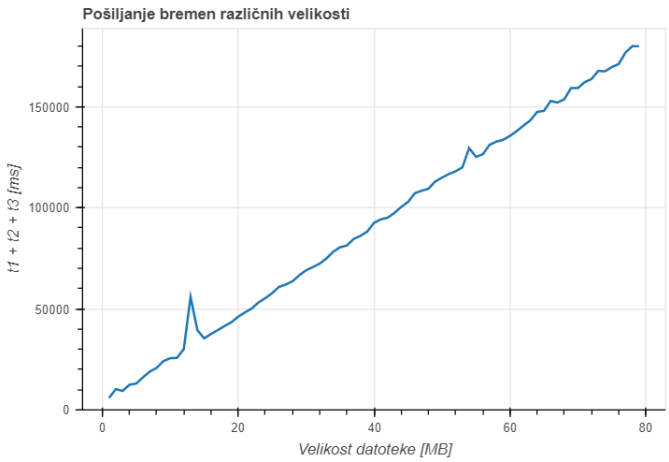
Tabela 1.12: Primerjava povprečnih časov pošiljanja ob posamičnih meritvah ter povprečnih časov skupinskega pošiljanja datotek velikosti 20 MB.

Tudi v primeru bremena velikosti 20 MB povprečni časi narastejo ob hkratnemu pošiljanju iz treh lokacij na sliki 1.17. To lahko vidimo tudi v tabeli 1.12, kjer se pojavljajo podobna odstopanja kot pri meritvah v tabeli 1.11. V

Ljubljani je čas pošiljanja daljši za približno 400 ms, v Ribnici za skoraj 2 s ter v Vodicah za približno 6,5 s v primerjavi z meritvami, ki smo jih izvajali ločeno vsak zase.

1.4.5 Povečevanja bremena med pošiljanjem

Zanimalo nas je kako se povečuje čas ob povečevanju velikosti bremena, ki ga pošiljamo v oblak. V ta namen smo izvedli meritve tako, da smo velikost bremena nastavili na 1 MB, potem pa velikost generirane datoteke vsakič povečali za 1 MB. Meritve so potekale iz Ribnice, 13.4. od 8:35 do 10:35, s povezavo 16/4 Mbps. Vidimo jih lahko na sliki 1.18.

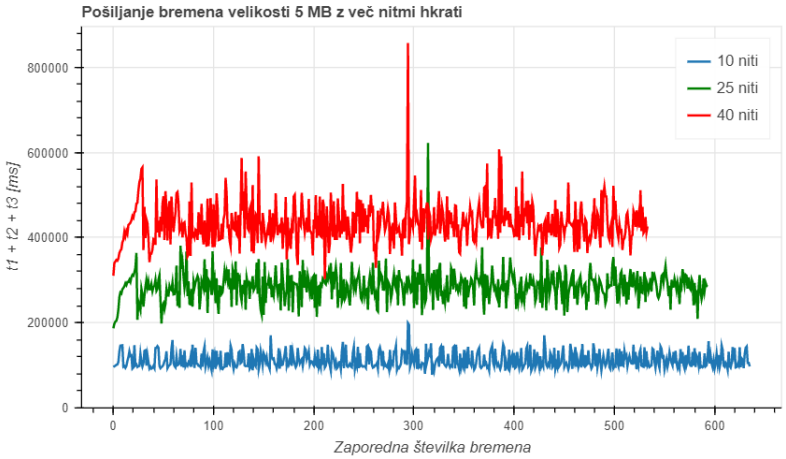


Slika 1.18: Pošiljanje datotek z različnimi velikostmi.

Na sliki 1.18 lahko opazimo, da smo s povečevanjem bremena pričakovano dobili dokaj konstantne čase, ki se linearno povečujejo s povečevanjem velikosti bremena. Ker smo z brezplačnim uporabniškim računom omejeni le na datoteke manjše od 2 GB, ne moremo doseči eksponentnega naraščanja časa.

1.4.6 Vpliv različnega števila niti

Meritve za pošiljanje več datotek hkrati smo izvedli še v Ribnici, s povezavo 16/4 Mbps in pri tem skušali preveriti kakšne razlike se pojavijo pri različnem številu vzporednih niti, ki hkrati pošiljajo datoteke. Velikost datotek, ki smo jih pošiljali je bila 5 MB, posamezne meritve za določeno število niti pa so trajale točno 120 minut.



Slika 1.19: Rezultati pošiljanja datotek velikosti 5 MB z različnim številom niti iz Ribnice.

Število niti	*Povprečni čas*	*Št. vseh datotek*	*Št. neuspelih pošiljanj*	*Odstotek neuspelih pošiljanj*
10	112.067 ms	636	0	0 %
25	285.421 ms	622	28	4,5 %
40	433.673 ms	697	163	23,4 %

Tabela 1.13: Vpliv različnega števila niti na čase pošiljanja.

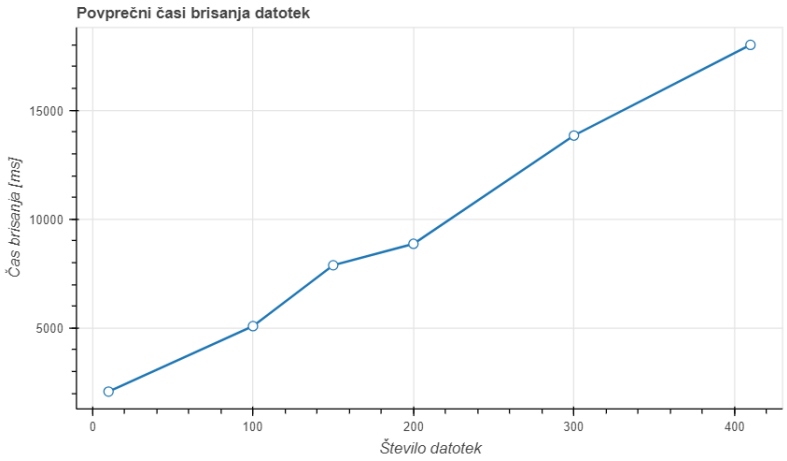
Iz rezultatov na sliki 1.19 in v tabeli 1.13 lahko vidimo da povečanje števila niti, ki hkrati vzporedno pošiljajo datoteke v oblak povzroči, da se povprečen čas prenosa ene datoteke podaljša, pri tem pa pride tudi do več neuspelih prenosov, ki pa so verjetno bolj posledica nizke hitrosti povezave, ki se še dodatno porazdeli med posamezne niti. Namesto potrditev so se ob neuspelih poskusih pojavila sporočila *Read timeout* in *Write timeout*.

1.4.7 Čas brisanja datotek

Ker se shranjevanje in brisanje datotek na disk hkrati ne more izvajati, nas ob večnitnem pošiljanju zanima kako dolgo traja brisanje datotek, s katerim zagotavljamo, da se meritve izvajajo poljubno dolgo in niso omejene z dodeljenim pomnilnikom brezplačnega računa. Meritve smo zastavili tako, da smo najprej v oblak poslali 10 datotek velikosti 5 MB ter nato zagnali le funkcijo, ki skrbi za brisanje datotek in izmerili čas od klica funkcije do prejema potrditve uspešnega brisanja. Da smo dobili bolj točne rezultate, smo preko spletnega vmesnika za Dropbox obnovili izbrisane datoteke (Dropbox ponuja funkcijo, ki uporabniku za določen čas omogoča obnavljanje že izbrisanih datotek) in ta čas izmerili večkrat. Ker je sam klic funkcije pri Dropboxu asinhron, smo za potrebe

merjenja časa priredili metodo brisanja v programu Benchcloud tako, da takoj po začetku brisanja začne z izvajanjem zanke v kateri ob vsaki iteraciji preveri ali je brisanje že zaključeno in iz nje izstopi šele ko dobimo odgovor, da je bilo brisanje končano.

Meritve smo izvedli za brisanje 10, 100, 150, 200, 300 in 410 (polno zaseden pomnilnik) datotek velikosti 5 MB.



Slika 1.20: Povprečni časi brisanja različnega števila datotek.

Število datotek	Povprečni čas brisanja
10	2.080 ms
100	5.084 ms
150	7.883 ms
200	8.866 ms
300	13.840 ms
410	18.003 ms

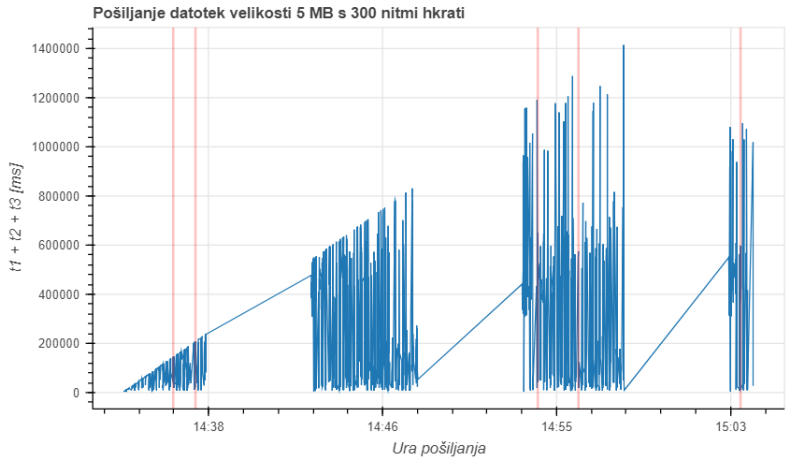
Tabela 1.14: Povprečni časi brisanja datotek.

Na sliki 1.20 lahko vidimo, da čas potreben za brisanje sorazmerno narašča s številom datotek, ki jih želimo izbrisati, točno izmerjeni časi teh meritev pa se nahajajo v tabeli 1.14. Ker je čas brisanja polno zasedenega oblaka dokaj visok, lahko predvidevamo, da ta lahko vpliva na shranjevanje datotek, ki ga opravljajo druge niti.

1.4.8 Poskus preobremenitve oblaka Dropbox

Meritve smo zaradi večje hitrosti povezave (245/690 Mbps) opravili na fakulteti. Opravili smo trideset minutne meritve in z različnim številom niti pošiljali 5 MB velike datoteke. Vsaka nit čaka na potrditev o uspešnem shranjevanju in takoj

zatem nadaljuje z generiranjem in pošiljanjem nove datoteke, pri tem pa deluje neodvisno od vseh ostalih niti. Pričakujemo, da se bo vrsta datotek, ki čakajo na shranjevanje v oblak povečevala, s tem pa bo vedno daljši tudi čas prenosa. Z navpičnimi rdečimi črtami smo skušali ponazoriti pri katerih datotekah je prišlo do napake. To pomeni, da datoteka ni bila uspešno shranjena v oblak, v takem primeru pa namesto potrditve o shranjeni datoteki, v Benchcloudu prejmemo sporočilo, da je prišlo do napake, porabljen čas od začetka pošiljanja do napake pa se v končnem rezultatu ne shrani.



Slika 1.21: Pošiljanje datotek s 300 nitmi hkrati.

V 30 minutni meritvi s 300 nitmi na sliki 1.21 je opaziti naraščanje časa potrebnega za shranjevanje posamezne datoteke. Prav tako lahko na grafu vidimo trend do katerega je prišlo tudi ob večkratni ponovitvi meritev, in sicer trikrat je prišlo do daljšega čakanja niti na potrditve uspešnega shranjevanja datoteke. Med prejemom zadnje potrditve in nove potrditve je čakanje vsakokrat trajalo skoraj točno 5 minut. Prvič med 14:38 in 14:43, drugič med 14:48 in in 14:53 in zadnjič med 14:58 in 15:03. Zaradi takšnega pojava pa tudi na sliki 1.21 opazimo 5 minutne premore v katerih so vse niti čakale na potrditev.

Število niti	*Povprečni čas*	*Št. uspelih datotek*	*Št. neuspelih pošiljanj*	*Odstotek neuspelih pošiljanj*
300	236.099 ms	1200	5	0,4 %

Tabela 1.15: Vpliv različnega števila niti na čase pošiljanja.

Povzetek meritev lahko vidimo v tabeli 1.15.

1.5 Zaključek

V tem poglavju smo izvedli različne teste za analizo zmogljivosti pomnjenja oblačnega ponudnika. Pri tem smo ugotovili, da je najpomembnejši dejavnik pri shranjevanju datoteke hitrost ter lokacija uporabnika. Pri ponudniku Dropbox skozi naše meritve nismo opazili uporabe mehanizma, ki bi ob pošiljanju že prepoznal datoteke z enako vsebino shranjene v oblaku in s tem ne bi zahteval ponovnega prenašanja celotne datoteke. Pošiljanje iz 3 različnih lokacij hkrati na isti uporabniški račun Dropboxu ne predstavlja večjih težav, saj so časi za posamezno lokacijo dokaj konstantni. Zaradi omejitve pomnilnika 2 GB smo v naše meritve ob večnitnem pošiljanju morali vključiti tudi operacijo brisanja, ki pa glede na meritve v poglavju 1.4.7 lahko povzroči dodatno zakasnitev pri shranjevanju datoteke. Pri uporabi 300 niti sicer pride le do nekaj neuspešnih pošiljanj, vendar pa lahko opazimo naraščanje čase prenosa za posamezno datoteko, na grafu pa so lepo vidna tudi obdobja v katerih daljši čas čakamo na prejem potrditev.

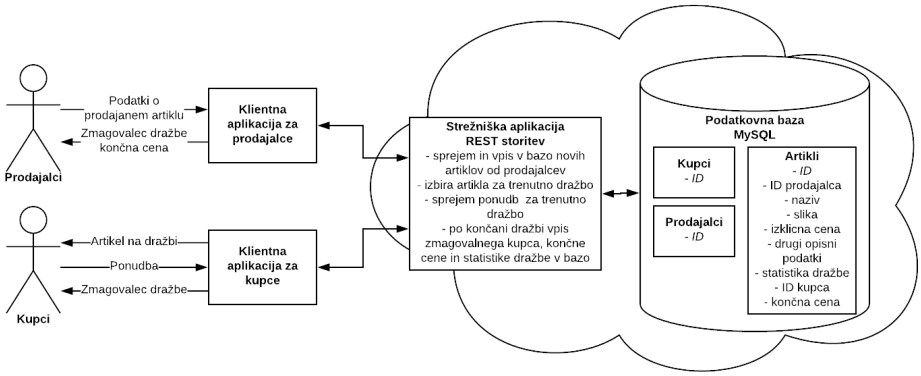
Poglavje 2

Testiranje oblačne strežbe dražb v realnem času

Jurij Bešter, Juš Debelak, Andrej Orehek

2.1 Opis problema

Realizirali bomo storitev strežbe dražbe v realnem času z beleženjem statistike. V oblačni strežnik se bosta povezovala dva tipa klientov in sicer manj številčni prodajalci, ki bodo v dražbo ponudili artikel z izklicno ceno, ter številčnejši kupci, ki bodo v omejenem času oddali ponudbe za nakup artikla. Strežnik bo moral posredovati ponudbo prodajalca trenutno prijavljenim kupcem, nato pa od kupcev sprejeti njihove pravilne (oddane v definiranem omejem času) ponudbe. Ponudbe bo moral v omejenem času sprocesirati, torej najti najvišjo ponudbo ter povezati zmagovalnega kupca in prodajalca, ter izdelati in shraniti statistiko posamezne dražbe. Shema delovanja je prikazana na sliki 2.1. Sorodne rešitve že obstajajo, bodisi namenske rešitve dražbenih hiš, ali pa spletne platforme za izvajanje živih dražb [9].



Slika 2.1: Shema delovanja dražbene storitve.

2.2 Opis ponudnikov in tehnologij

Za ponudnika oblačnih storitev smo si izbrali Amazonov AWS (*Amazon Web Services*) [10]. To je zbirka storitev, ki olajša razvoj spletnih storitev. Omogoča razvoj aplikacijske logike v različnih okoljih (framework) in programskih jezikih. Za hranjenje podatkov nudi široko paleto različnih podatkovnih baz.

Aplikacijska logika opisane rešitve je implementirana v okolju node.js [11] in programskim jezikom JavaScript [12]. Za hranjenje podatkov je uporabljena relacijska podatkovna baza MySQL [13]. Komunikacija med odjemalci in strežnikom je realizirana po REST [14] pristopu. REST pristop predvideva uporabo HTTP zahtevkov za manipulacijo (POST, DELETE) in pridobivanje podatkov (GET).

Uporabljene storitve v rešitvi, ki jih AWS ponuja:

- **Amazon API Gateway [15]**: Vstopna točka za zahtevke odjemalcev, ki preusmerja zahtevke na določene Lambda funkcije glede na tip;
- **Amazon Lambda [16]**: Lambda funkcije vsebujejo aplikacijsko logiko, ki obdelujejo posamezne zahtevke in so povezane z relacijsko podatkovno bazo (RDS);
- **Amazon RDS [17]**: Omogoča izbiro in uporabo različnih relacijskih podatkovnih baz za hranjenje podatkov. V opisani rešitvi je uporabljena MySQL relacijska podatkovna baza;

2.3 Opis izdelane rešitve

Rešitev vsebuje naslednje dostopne točke:

- /**user**: z zahtevkom POST ustvarimo novega uporabnika. Strežnik ob uspešnem vnosu vrne id uporabnika. Primer POST zahtevka:

```
{
    name: "User name"
}
```

- /**user/{id}**: z zahtevkom POST posodobimo podatke za uporabnika z izbranim **id**-jem. Oblika POST zahtevka je enaka kot pri vnosu novega uporabnika. Z zahtevkom DELETE pa uporabnika izbrišemo.

- /**item**: z zahtevkom GET pridobimo seznam vseh predmetov na dražbi, z zahtevkom POST pa pošljemo na dražbo nov predmet. Primer POST zahtevka:

```
{
    "name":"Test item",
    "description":"Test item description",
    "starting_price":1000,
    "user_id":4,
    "bidding_time":3600
}
```

- /**item/{id}**: z zahtevkom POST posodobimo predmet z izbranim **id**-jem. Oblika POST zahtevka je enaka kot pri vnosu novega predmeta. Z zahtevkom DELETE pa predmet izbrišemo.

- /**bid/{id}**: z zahtevkom POST podamo novo ponudbo za predmet z izbranim **id**-jem, z zahtevkom GET pa pridobimo že oddane ponudbe za ta predmet. Primer POST zahtevka:

```
{
    "user_id": 5,
    "bid": 2000
}
```

- /**bid/{id}/max**: z zahtevkom GET pridobimo trenutno najvišjo ceno za predmet z izbranim **id**-jem;

Na vse POST zahtevke strežnik odgovori z podatki o uspešnosti operacije. Primer odgovora:

```
{
    "fieldCount": 0,
    "affectedRows": 1,
    "insertId": 0,
    "serverStatus": 2,
    "warningCount": 0,
    "message": "(Rows matched: 1 Changed: 0 Warnings: 0",
    "protocol41": true,
    "changedRows": 0
}
```

Za lokacijo naših storitev smo izbrali *us-west-2*, ki se nahaja v državi Oregon, saj je to lokacija, ki jo ponudnik predlaga za brezplačno raven uporabe.

2.4 Analiza lokacije

Lokacijo *us-west-2* smo analizirali s 24-urnim testom, med katerim smo vsako minuto izvedli PING [18], s katerim smo izmerili čas med klientom in strežnikom (angl. *rount-trip time* - *RTT*). Hkrati smo s TRACEROUTE [19] beležili pot paketa med klientom in strežnikom. Test smo izvajali 1. maja 2018.

Test smo hkrati izvajali iz treh različnih lokacij:

- 1. lokacija: Ljubljana [Telemach - kabelski internet - 150Mbps/4Mbps]
- 2. lokacija: Šentvid pri Stični [Telekom - ADSL - 10Mbps/1Mbps]
- 3. lokacija: Falkenstein-DE [Hetzner - optika - 1Gbps/1Gbps] (Najet namenski strežnik pri ponudniku gostovanja strežnikov Hetzner. Ponudnik oglašuje privatne povezave v omrežja večjih ponudnikov spletnih vsebin, med drugimi tudi Amazon.)

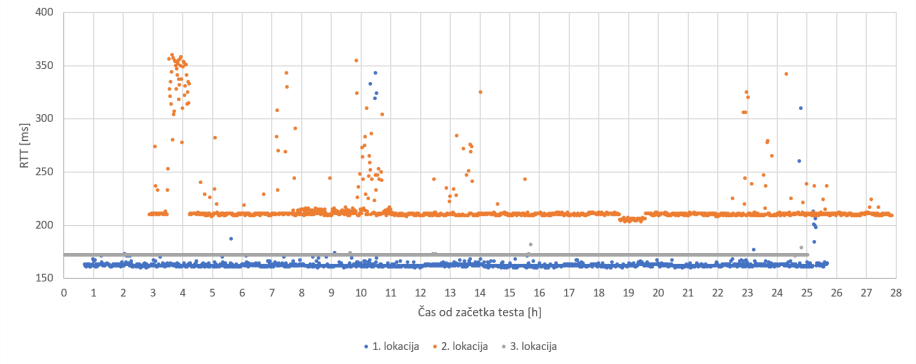
Na sliki 2.2 lahko vidimo, da so RTT časi in njihova konsistentnost med testnimi lokacijami različni. Izračunali smo povprečno RTT vrednost, in standardni odklon RTT vrednosti za posamezne lokacije:

- 1. lokacija: Povprečen RTT: 164,0ms, Standardni odklon RTT: 10,2ms
- 2. lokacija: Povprečen RTT: 217,5ms, Standardni odklon RTT: 24,5ms
- 3. lokacija: Povprečen RTT: 173,0ms, Standardni odklon RTT: 0,3ms

Na izpisih 2.1, 2.2 in 2.3 so vidne začetne poti med klienti in strežnikom. Poti smo natančneje pregledali v časovnih obdobjih ko so RTT rezultati za 1. in 2. lokacijo bistveno odstopali od njunih siceršnjih povprečij. Ugotovili smo, da je na 1. in 2. lokaciji do povečanih časov prišlo predvsem zaradi zakasnitev na poti znotraj domene naših ISPjev. Na lokaciji 3., ki je na komercialni internetni povezavi so bili RTT časi tekom testa zelo podobni. Poti so se na vseh lokacijah tekom testa rahlo spreminjale, a med različnimi potmi ni bilo bistvenih časovnih razlik.

Kljub temu, da je 3. lokacija na komercialni internetni povezavi z oglaševanimi privatnimi povezavami pa je bil povprečni RTT 3. lokacije višji od 1. lokacije, ki se nahaja na potrošniški internetni povezavi. Iz TRACEROUTE izpisov lahko razberemo, da je bil prihod v ZDA iz 3. lokacije (116,4ms) počasnejši od tistega iz 1. lokacije (102,3ms). Pot preko Atlantskega oceana iz 1. lokacije je potekala Francija - New York (70,3ms), iz 2. lokacije pa Švedska - Chicago (24,3ms). Kljub temu da je pot preko Atlantskega oceana Švedska - Chicago skoraj 3x hitrejša, pa je celotna pot daljša zaradi potratnega skoka iz Velike Britanije v Švedsko, ki znaša kar 66,1ms.

Razlog za veliko višji povprečni RTT iz 2. lokacije pa je bila prav dolga pot do vzhodne obale ZDA, ki je znašala kar 181,4ms. Točna pot 2. povezave preko Atlantskega oceana ni razvidna.



Slika 2.2: Rezultati PING testa iz vseh treh testnih lokacij.

Listing 2.1: TRACEROUTE izpis iz 1. lokacije

```
traceroute to 54.213.245.230 (54.213.245.230), 30 hops max, 60 byte packets
 1  router.asus.com (192.168.26.1)  0.731 ms
 2  10.224.128.1 (10.224.128.1)  7.901 ms
 3  217-72-74-225.ipv4.telemach.net (217.72.74.225) 13.377 ms
 4  185.66.148.235.ipv4.telemach.net (185.66.148.235)  13.438 ms
 5  185.66.148.235.ipv4.telemach.net (185.66.148.235)  12.655 ms
 6  185.66.148.219.ipv4.telemach.net (185.66.148.219)  19.456 ms
 7  185.66.148.219.ipv4.telemach.net (185.66.148.219)  18.805 ms - (Slovenija)
 8  10ge1-5.core1.vie1.he.net (216.66.82.73)  16.948 ms - (Avstrija)
 9  100ge13-1.core1.par2.he.net (184.105.65.5)  32.024 ms - (Francija)
10  100ge14-1.core1.nyc4.he.net (184.105.81.77)  102.326 ms - (ZDA, NY)
11  100ge14-1.core1.tor1.he.net (184.105.80.10)  122.009 ms
12  100ge6-1.core1.ywg1.he.net (184.105.64.102) 133.523 ms
13  100ge10-1.core1.yyc1.he.net (184.105.222.98)  146.205 ms
14  100ge10-2.core1.yvr1.he.net (184.105.64.113)  161.939 ms
15  100ge10-2.core1.sea1.he.net (184.105.64.109)  158.615 ms
16  paix01-sea4.amazon.com (198.32.134.41) 158.612 ms
17  52.95.52.120 (52.95.52.120)  157.282 ms
18  52.95.52.97 (52.95.52.97)  156.636 ms
19  *
20  54.239.42.223 (54.239.42.223)  167.441 ms
21  54.239.43.136 (54.239.43.136)  167.747 ms
22  *
23  52.93.13.4 (52.93.13.4)  169.798 ms
24  52.93.12.37 (52.93.12.37)  166.918 ms
25  52.93.12.63 (52.93.12.63)  166.653 ms
26  52.93.240.19 (52.93.240.19)  164.178 ms
```

Listing 2.2: TRACEROUTE izpis iz 2. lokacije

```
traceroute to 54.213.245.230 (54.213.245.230), 30 hops max, 60 byte packets
 1  TP-LINK (192.168.16.1) 0.563 ms
 2  95.176.255.241 (95.176.255.241)  14.349 ms
 3  95.176.241.151 (95.176.241.151)  13.821 ms
 4  212.162.28.5 (212.162.28.5)  14.237 ms - (Slovenija)
 5  *
 6  4.16.146.90 (4.16.146.90)  181.369 ms - (ZDA, NY)
 7  *
 8  *
 9  *
10  *
11  *
12  *
13  52.93.15.10 (52.93.15.10)  214.913 ms
14  *
15  52.93.14.142 (52.93.14.142)  219.543 ms
16  52.93.240.47 (52.93.240.47)  212.191 ms
```

Listing 2.3: TRACEROUTE izpis iz 3. lokacije

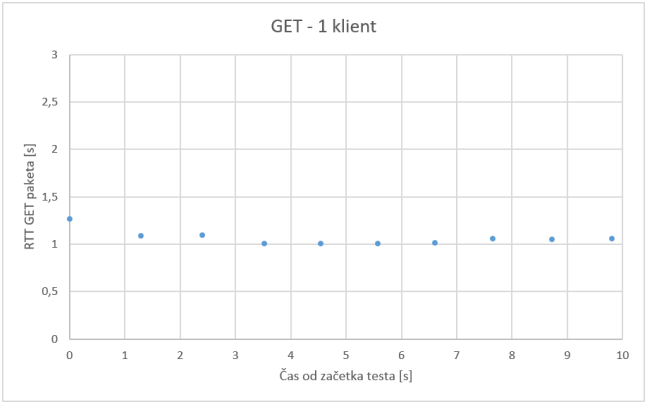
```
traceroute to 54.213.245.230 (54.213.245.230), 30 hops max, 60 byte packets
 1  static.65.57.243.136.clients.your-server.de (136.243.57.65)  0.369 ms
 2  core24.fsn1.hetzner.com (213.239.203.193)  0.333 ms
 3  core11.nbg1.hetzner.com (213.239.245.225)  2.761 ms
 4  juniper4.dc2.nbg1.hetzner.com (213.239.203.138)  2.808 ms
 5  hbg-b1-link.telia.net (213.248.70.0)  16.184 ms
 6  hbg-bb4-link.telia.net (213.155.135.86)  16.125 ms - (Nemcija)
 7  ldn-bb3-link.telia.net (80.91.249.10)  26.039 ms - (UK)
 8  nyk-bb4-link.telia.net (62.115.136.185)  92.127 ms - (Svedska)
 9  chi-b21-link.telia.net (80.91.246.162)  116.386 ms - (ZDA, Chicago)
10  sea-b2-link.telia.net (62.115.117.48)  170.325 ms
11  sea-b1-link.telia.net (62.115.115.2)  164.815 ms
12  amazon-ic-307562-sea-b1.c.telia.net (213.248.92.242) 157.532 ms
13  *
14  *
15  *
16  *
17  *
18  *
19  *
20  *
21  *
22  *
23  54.239.48.183 (54.239.48.183)  185.700 ms
```

2.5 Metrike

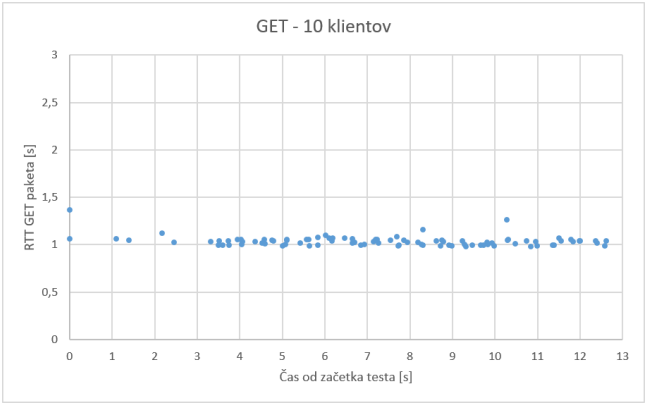
2.5.1 Mrežne metrike

Za testiranje delovanja smo opravili dva testa. Pri obeh smo teste izvajali iz 3. lokacije (gl. poglavje 1.3), 13. aprila 2018 ob 12. uri.

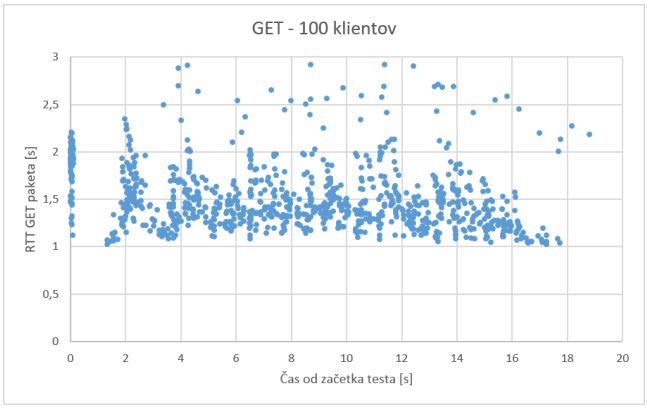
V 1. testu so klienti od strežnika 10 krat zapored pridobili seznam vseh predmetov na dražbi z metodo GET /**item** . Merili smo RTT čas GET paketa, od pošiljanja zahteve do prejetja odgovora. Test smo ponovili z 1, 10 in 100 klienti.



Slika 2.3: 1. test - 1 klient: povprečni čas = 1,067s, trajanje testa = 9,8s.

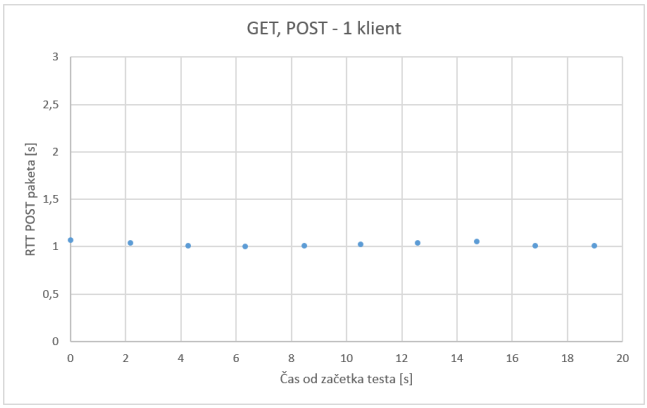


Slika 2.4: 1. test - 10 klientov: povprečni čas = 1,258s, trajanje testa = 12,6s.

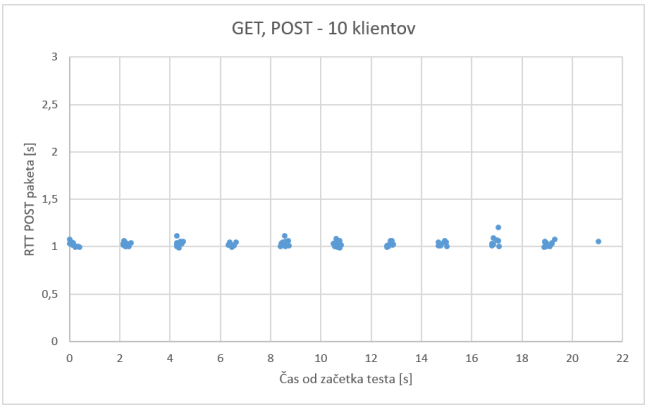


Slika 2.5: 1. test - 100 klientov: povprečni čas = 1,550s, trajanje testa = 18,8s.

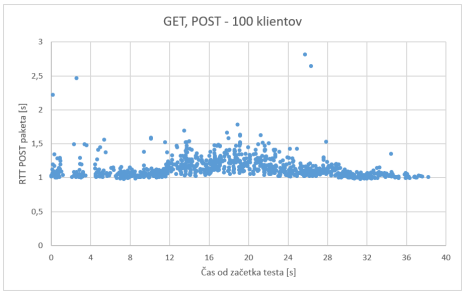
V 2. testu so klienti od strežnika 10 krat zapored pridobili trenutno najvišjo ceno za predmet z metodo GET **/bid/{id}/max**, ter nato podali novo ponudbo za predmet z metodo POST **/bid/{id}**. Merili smo RTT čas POST paketa, od pošiljanja zahteve do prejetja odgovora. Test smo ponovili z 1, 10 in 100 klienti.



Slika 2.6: 2. test - 1 klient: povprečni čas = 1,028s, trajanje testa = 19,0s.



Slika 2.7: 2. test - 10 klientov: povprečni čas = 1,033s, trajanje testa = 21,0s.



Slika 2.8: 2. test - 100 klientov: povprečni čas = 1,325s, trajanje testa = 38,2s.

2.5.2 Metrike stresnih testov storitve v oblaku

Za izvajanje stresnih testov storitve smo uporabili orodje ab [20] (Apache HTTP server benchmarking tool), ki omogoča izvajanje HTTP zahtev z različnimi parametri.

Orodje ab [20] je primerno za izvajanje stresnih testov opisane storitve, ker je vmesnik storitve realiziran po REST [14] pristopu, ki sloni na HTTP protokolu. Orodje omogoča določanje število sočasnih procesov, ki generirajo zahtevke. Prav tako orodje omogoča generiranje različnih tipov HTTP REST zahtevkov (GET, POST, DELETE, ...). Orodje, poleg opisanih opcij, ponuja nastavljanje še drugih parametrov, ki pa za potrebe izvajanja stresnih testov niso ključni. V stresnih testih smo povečevali število sočasnih procesov, ki generirajo zahtevke, posamezne meritve pa izvajali 1 minuto.

Realizirana storitev opisana ponuja več vstopnih točk (poglavje: 2.3). Ker je storitev namenjena dražbi produktov lahko pričakujemo, da bo najbolj obremenjena vstopna točka **/bid/{id}** za podajanje novih ponudb (cen) za posamezne predmete z zahtevkom POST. Storitev vsako podano ponudbo zapiše v bazo in vrne rezultat ali je ponudba uspešno obdelana.

Cilj podanih stresnih testov je bil poiskati točko, ko storitev ni bila več zmožna obdelovati vseh zahtevkov. Oddajali smo enake ponudbe za določen izdelek (identični paketi) in s tem preprečili, da bi različna velikost paketa vplivala na zmogljivost omrežja in rezultate meritev.

Listing 2.4: Izhodni podatki oroda AB

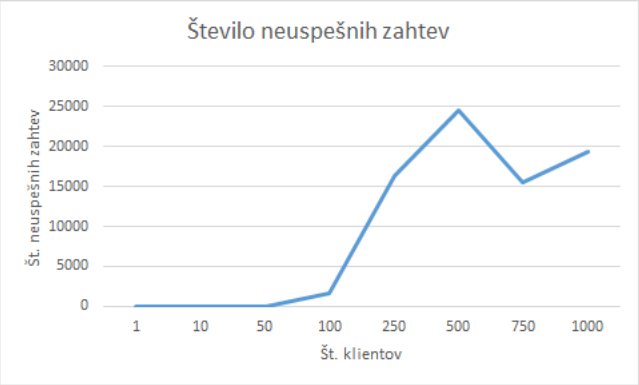
```
##########################################################

Concurrency Level:      1
Time taken for tests:   60.239 seconds
Complete requests:      201
Failed requests:        0
Requests per second:    3.34 [#/sec] (mean)
Time per request:       299.698 [ms] (mean)

Connection Times (ms)
              min  mean[+/-sd] median max
Connect:        0    3  39.2      0     555
Processing:   276  297  10.3    297     361
Waiting:      276  297  10.3    297     361
Total:        276  300  44.7    297     917

##########################################################

Concurrency Level:      10
Time taken for tests:   60.058 seconds
Complete requests:      1625
Failed requests:        0
Requests per second:    27.06 [#/sec] (mean)
Time per request:       369.590 [ms] (mean)

Connection Times (ms)
              min  mean[+/-sd] median max
Connect:        0    3  42.5      0     560
Processing:   243  365  93.5    324     886
Waiting:      243  365  93.5    324     886
Total:        243  368 100.4    326     886

##########################################################

Concurrency Level:      50
Time taken for tests:   60.013 seconds
Complete requests:      14884
Failed requests:        0
Requests per second:    248.01 [#/sec] (mean)
Time per request:       201.603 [ms] (mean)

Connection Times (ms)
              min  mean[+/-sd] median max
Connect:        0    2  31.5      0     564
Processing:   186  199  13.1    198    1218
Waiting:      186  199  13.1    198    1218
Total:        186  201  38.6    198    1218

##########################################################

Concurrency Level:      100
Time taken for tests:   60.001 seconds
Complete requests:      27666
Failed requests:        1603
Requests per second:    461.09 [#/sec] (mean)
Time per request:       216.876 [ms] (mean)

Connection Times (ms)
              min  mean[+/-sd] median max
Connect:        0    2  33.2      0     582
Processing:   185  214  64.8    198    1209
Waiting:      185  214  64.8    198    1209
Total:        185  216  72.9    198    1209

##########################################################

Concurrency Level:      250
Time taken for tests:   60.002 seconds
Complete requests:      49038
Failed requests:        16395
Requests per second:    817.27 [#/sec] (mean)
Time per request:       305.895 [ms] (mean)

Connection Times (ms)
              min  mean[+/-sd] median max
Connect:        0    3  41.6      0     641
Processing:   185  302 156.7    203     986
Waiting:      185  302 156.7    203     986
Total:        185  305 160.9    204    1303
```

```
##################################################

Concurrency Level:      500
Time taken for tests:   60.015 seconds
Complete requests:      63445
Failed requests:        24632
Requests per second:    1057.15 [#/sec] (mean)
Time per request:       472.971 [ms] (mean)

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    5  56.6      0     742
Processing:   186  433 821.6    205   10831
Waiting:      186  433 821.6    205   10831
Total:        186  438 823.0    205   10831

##################################################

Concurrency Level:      750
Time taken for tests:   60.001 seconds
Complete requests:      34292
Failed requests:        15547
Requests per second:    571.52 [#/sec] (mean)
Time per request:       1312.284 [ms] (mean)

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   21 137.8      0    1111
Processing:   182 1228 2436.2    207   10878
Waiting:      182 1228 2436.2    207   10878
Total:        182 1249 2460.6    208   11922

##################################################

Concurrency Level:      1000
Time taken for tests:   60.004 seconds
Complete requests:      33366
Failed requests:        19440
Requests per second:    556.06 [#/sec] (mean)
Time per request:       1798.366 [ms] (mean)

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   23 133.8      0     978
Processing:   180 1661 3066.8    206   10933
Waiting:      180 1661 3066.8    206   10933
Total:        180 1684 3070.8    206   11740

##################################################
```

Slika 2.9: Povprečni čas strežbe zahteve glede na število hkrati poslanih zahtev.

Slika 2.10: Povprečno število zahtev na sekundo glede na število hkrati poslanih zahtev.

Slika 2.11: Število uspešno obdelanih zahtev glede na število hkrati poslanih zahtev.



Slika 2.12: Število neuspešno obdelanih zahtev glede na število hkrati poslanih zahtev.

Iz izpisa 2.4 in grafov na slikah 2.9, 2.10, 2.11 in 2.12 lahko opazimo dve ključni točki:

Prva ključna točka je pri 100 hkrati poslanih zahtevah (461 zahtev na sekundo). Do te točke je storitev na strežniku lahko uspešno obdelovala zahteve. Ko je število hkrati generiranih zahtev preseglo 100, se je začelo povečevati število neuspešno obdelanih zahtev. Storitev se še vedno odziva v normalnem času pod 400ms.

Druga ključna točka je pri 500 hkrati poslanih zahtevah (1057 zahtev na sekundo), kjer lahko opazimo občutno poslabšanje odzivanja storitve. To točko lahko označimo kot točko preloma.

Po analizi dogajanja med obema točkama smo ugotovili, da je število uspešno procesiranih zahtev upadalo zaradi zmogljivosti podatkovne baze. Aplikacijska logika storitve je lahko procesirala zahtevke do druge ključne točke. Odziv storitve je bil v pričakovanem času vendar se posredovani podatki niso zapisali v podatkovno bazo. Po tej točki je odzivnost celotne storitve upadla, kar pomeni da tudi aplikacijska logika ni bila več zmožna obdelovati vseh zahtevkov.

Po rezultatih zgornje analize bi bilo potrebno najprej izboljšati zmogljivost podatkovne baze.

2.6 Zaključek

Med samim razvojem naše aplikacije smo opazili, da je dokumentacija AWS zelo pomanjkljiva. Kasneje nam je to tudi oteževalo delo pri testiranju.

V testih nismo ugotovili profiliranja internetnega prometa iz same strani AWS, so pa v rezultatih vidni vplivi različnih specifikacij internetne povezave in različnih globalnih internetnih poti na delovanje oz. uporabo storitve.

Rezultati na sliki 2.9 lahko kažejo na to, da AWS dinamično prilagaja resurse za izvajanje strežbe. Vidimo da se povprečni časi strežb zahtev z zviševanjem

števila hkrati poslanih zahtev sprva višajo, nato pa pri 50 hkrati poslanih zahtevah, ko je obremenitev višja, padejo nižje kot so bili sprva.

Ugotavljamo, da naše storitve ne bi mogli gostovati na osnovnih paketih AWS. Razpoložljivi resursi so namreč prenizki za omogočanje dostopa do storitve vsem potencialnim klientom. Vprašljivi so tudi visoki dostopni časi. Za primerno delovanje bi potrebovali strežnike, ki so lokacijsko blizu klientom.

Poglavje 3

Analiza zmogljivosti oblačnih storitev

Edi Čebokli, Rok Grmek, Tadej Škapin

3.1 Opis problema

Za izvedbo analize zmogljivosti oblačnih storitev smo si izbrali problem računanja števila π na oddaljenem strežniku. Računamo ga na dva načina - po Leibnizovi [21] formuli:

$$\frac{\pi}{4} = 1 + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \frac{1}{9} + ... \tag{3.1}$$

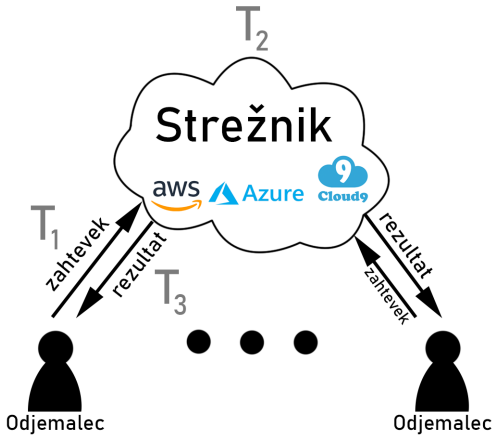
in po Bellardovi [22] formuli:

$$\pi = \frac{1}{2^6} \sum_{n=0}^{\infty} \frac{(-1)^n}{2^{10n}} (-\frac{2^5}{4n+1} - \frac{1}{4n+3} - \frac{2^8}{10n+1} - \frac{2^6}{10n+3} - \frac{2^2}{10n+5} - \frac{2^2}{10n+7} - \frac{1}{10n+9}). \tag{3.2}$$

V obeh primerih je število π predstavljeno kot vsota neskončne vrste, seveda pa v praksi seštejemo le končno mnogo členov. Pri tem se vsota večjega števila členov izraža v bolj natančnem izračunu, vendar pa s tem narašča časovna zahtevnost. Želeli smo, da oba algoritma izračunata število π s podobno natančnostjo, to pa smo dosegli s seštevanjem prvih 24.473.399 členov Leibnizove formule in s seštevanjem le prvih dveh členov Bellardove formule. V obeh primerih naš izračun odstopa od dejanske vrednosti števila π za približno 10^{-7}.

Oba algoritma smo implementirali v Python-u in C-ju, aplikacijo strežnika pa smo namestili pri treh različnih ponudnikih gostovanja (ti so podrobneje predstavljeni v razdelku 3.4). V okviru analize zmogljivosti smo sistem obremenili tako, da smo z večkratnim pošiljanjem zahtev simulirali večje število odjemalcev

(natančnejši opis bremena je v razdelku 3.5). To prikazuje tudi shema na sliki 3.1, kjer T1 označuje čas prenosa zahtevka od odjemalca do strežnika, T2 označuje čas potreben za izračun na strani strežnika, T3 pa čas prenosa rezultatov od strežnika do odjemalca.



Slika 3.1: Shema pošiljanja zahtevkov na strežnik.

3.2 Pregled sorodnih virov

Računanje števila π je numeričen problem, ki ga rešujemo s številnimi zaporednimi aritmetičnimi operacijami nad števili v plavajoči vejici (v dvojni natančnosti). Zato smo že pred samim računanjem želeli preveriti, kako hitre so takšne operacije v primeru oblačnega računanja. V članku "A Performance Analysis of EC2 Cloud Computing Services for Scientific Computing" [23] je analizirana zmogljivost računanja pri ponudniku gostovanja Amazon Web Services. Med drugim so v članku predstavljeni tudi rezultati meritev, kjer je razvidno, kakšno število operacij (seštevanj/množenj v plavajoči vejici, v dvojni natančnosti) lahko strežnik izvede na sekundo. Analizirani so sicer le plačljivi paketi, za te pa je pričakovana nekolika višja zmogljivost v primerjavi z brezplačnim paketom t2.micro, ki smo ga uporabili sami. V povprečju so v analizi namerili približno 0.6 GOPS (angl. *giga (billions) operations per second*). Glede na način izvajanja meritev, bomo lahko za računanje števila π po Leibnizovi formuli, implementirano v C-ju na koncu še sami izračunali povprečno število operacij na sekundo in ga primerjali z rezultatom iz prej omenjene analize zmogljivosti.

3.3 Izbira tehnologij

Pri izbiri programskih jezikov smo izbrali Python in C, ki sta različna po načinu izvajanja programov. Python je interpreterski jezik, ki hitreje analizira izvorno kodo, vendar počasneje izvaja program v primerjavi z eksekucijskim jezikom C. Strežnik in odjemalec sta implementirana v Python-u, algoritma za računanje π pa v C-ju in Python-u.

3.4 Ponudniki gostovanja

Aplikacijo strežnika za oddaljeno računanje števila π smo namestili pri treh različnih ponudnikih gostovanja - Amazon Web Services [24], Microsoft Azure [25] in Cloud9 [26]. Za večino testov smo uporabili Amazon Web Services, v enem od testov pa so ponudniki neposredno primerjani med sabo.

Pri ponudniku Microsoft Azure smo izbrali paket A1 standard, pri AWS pa paket t2.micro. To so paketi, ki jih lahko izberemo v brezplačni poskusni dobi uporabe storitve. Specifikacija sistemov, ki jih ponujajo zgornji trije ponudniki, je navedena v tabeli 3.1.

Ponudnik	CPE	Št. jeder	RAM	HDD	Lokacija
AWS	Intel Xeon E5-2676 v3	1	1GB	8GB	Zahodna Evropa
Azure	Intel Xeon E5-2630 v3	1	1.5GB	30GB	Južna Azija
Cloud9	Intel Xeon @2.50GHz	8	1GB	2GB	ZDA

Tabela 3.1: Tabela ponudnikov in njihovih specifikacij.

3.5 Definicija bremena storitve

Obremenjenost našega strežnika, ki ponuja oddaljeno računanje števila π, je odvisna od izbrane metode računanja, od implementacije te metode in od intenzivnosti prejemanja zahtevkov za računanje. Zato smo strežnik obremenili tako, da smo generirali večje število zaporednih zahtevkov za enega od načinov računanja, pri tem pa določili interval čakanja med dvema zahtevkoma.

Interval je v grobem lahko deterministične (točno določen), ali pa nedeterministične (psevdo-naključno generiran in porazdeljen eksponentno). Prva možnost omogoča lažjo analizo rezultatov zaradi konstantnega intervala, druga pa boljše opiše realno breme.

Pomembna je tudi delitev glede na to, kakšen je interval čakanja v primerjavi s časom računanja. Velika razlika v obremenitvi sistema se namreč pojavi, če interval med dvema zahtevkoma zmanjšamo do te mere, da ta postane manjši od povprečnega časa računanja in zahtevki za računanje prihajajo pogosteje, kot jih lahko strežnik obdela.

3.6 Definicija metrik in orodij za meritve

Za posamezno računanje na oddaljenem strežniku dobimo kot rezultat poleg števila π tudi čas, ki ga je strežnik potreboval za računanje (T2), obenem pa merimo še celoten čas od pošiljanja zahtevka za računanje do prejetja rezultata (T1 + T2 + T3). Pri eksperimentih nato opazujemo čas računanja na strežniku (T2) in čas, potreben za komunikacijo s strežnikom (T1 + T3).

3.7 Rezultati meritev

3.7.1 Primerjava različnih načinov računanja

Opis eksperimenta

Prve meritve smo opravili le pri ponudniku gostovanja AWS. Za vsako od štirih različic izračuna smo strežniku 1 uro, iz enega klienta, na vsake 3 sekunde (deterministično) pošiljali zahtevke in ob tem merili čase. S tem eksperimentom smo želeli preveriti delovanje sistema in primerjati štiri različne načine računanja med sabo.

Hipoteza

Pričakujemo lahko, da bodo časi računanja (T2) za vse 4 načine računanja skozi celoten čas testiranja približno konstantni. Strežnik bo namreč vedno obdeloval le eno zahtevo na enkrat, ker nobeno računanje naj ne bi trajalo več kot 3 sekunde.

Računanje implementirano v Python-u bo najverjetneje trajalo dlje od tistega v C-ju (za isto metodo računanja), za vsako od implementacij pa bo Leibnizova metoda verjetno zahtevala več časa od Bellardove. Kljub temu pa težko ocenimo, ali se bo iz časovnega vidika boljše odrezala Leibnizova metoda implementirana v C-ju ali Bellardova metoda implementirana v Python-u.

Čas komunikacije (T1 + T3) ni odvisen od načina računanja, zato sklepamo, da bo ta približno enak za vse štiri. Ta čas bo najverjetneje približno konstanten tudi skozi celoten čas testiranja, saj ne pričakujemo večjih sprememb v zasedenosti omrežja med testiranjem.

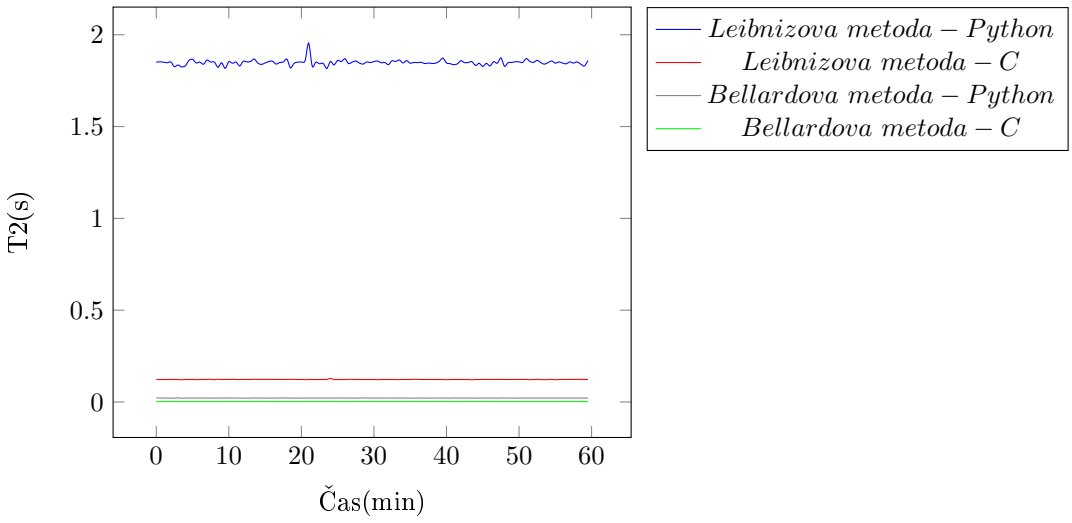
Okoliščine

Meritve so bile opravljene v Ljubljani, s povezavo 100 Mbps / 10 Mbps. Bile so opravljene med 18. in 19. uro v sredo 18. 4. 2018. Vsaka meritev se je izvajala na svojem strežniku, vsi strežniki pa imajo identično konfiguracijo in lokacijo.

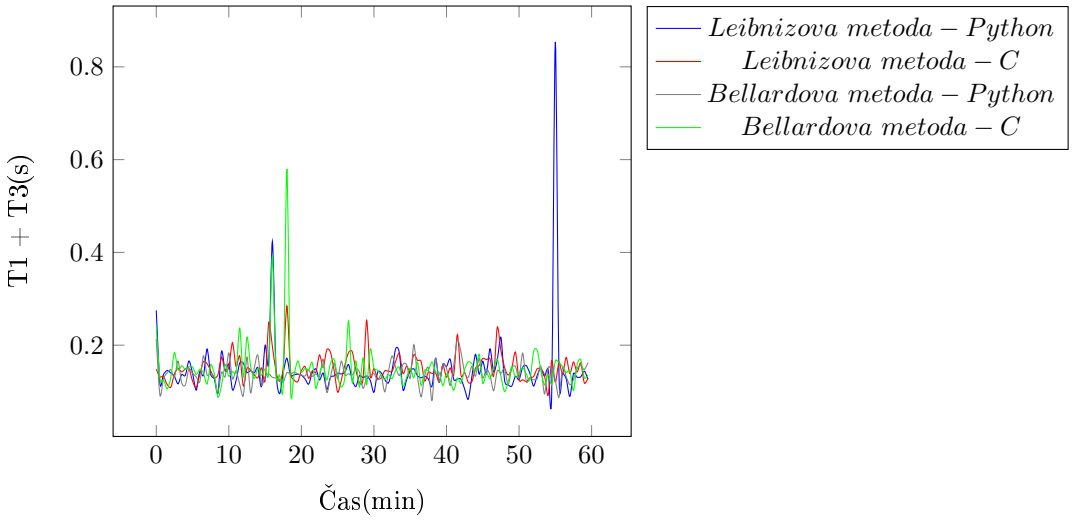
Rezultati

Čas računanja na strežniku (T2) v odvisnosti od časa testiranja je prikazan na sliki 3.2, celoten čas komunikacije (T1 + T3) spet v odvisnosti od časa testiranja pa je prikazan na sliki 3.3.

Hipoteza je bila očitno smiselna, iz rezultatov pa je vidno, da računanje po Bellardovi metodi zahteva manj časa v primerjavi z Leibnizovo metodo, ne glede na tehnologijo implementacije.



Slika 3.2: Čas računanja na strežniku v odvisnosti od časa testiranja.



Slika 3.3: Celoten čas komunikacije v odvisnosti od časa testiranja.

3.7.2 Primerjava ponudnikov gostovanja

Opis eksperimenta

Za ta eksperiment smo aplikacijo strežnika namestili pri treh različnih ponudnikih gostovanja (ti so predstavljeni v razdelku 3.4). Zahtevke smo spet pošiljali 1 uro, na vsake 3 sekunde (deterministično), vendar tokrat do treh različnih strežnikov. Na vsakem od strežnikov smo testirali le računanje z Leibnizovo metodo, implementirano v Python-u.

Hipoteza

Glede na specifikacije sistemov ponudnikov (tabela 3.1), pričakujemo podobne čase računanja (T2) za AWS in Azure, nekoliko krajše čase pa za Cloud9.

Časi komunikacije (T1 + T3) se bodo najverjetneje razlikovali zaradi različnih lokacij strežnikov. Najkrajši čas pričakujemo za AWS, saj je strežnik lociran v Evropi, najdaljši čas pa bo verjetno potreben za dostop do strežnika v Južni Aziji, ki ga ponuja Azure.

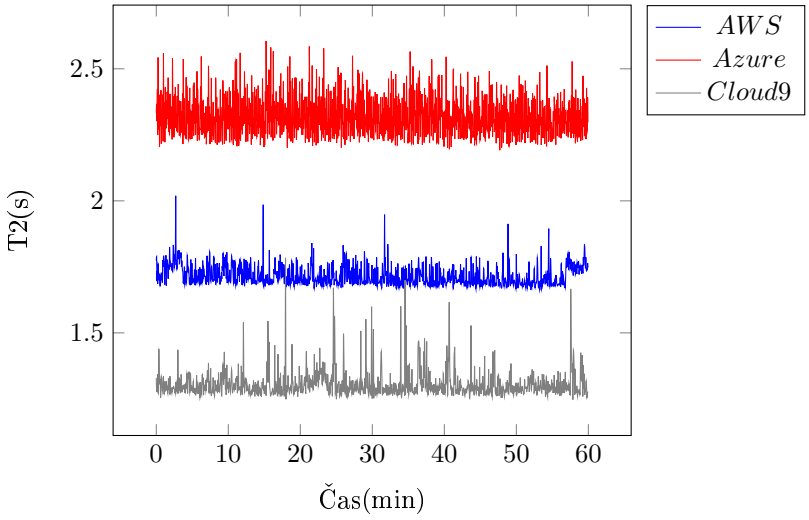
Okoliščine

Meritve so bile opravljene v Ilirski Bistrici, s povezavo 20 Mbps / 5 Mbps. Bile so opravljene med 14. in 15. uro v sredo 2. 5. 2018.

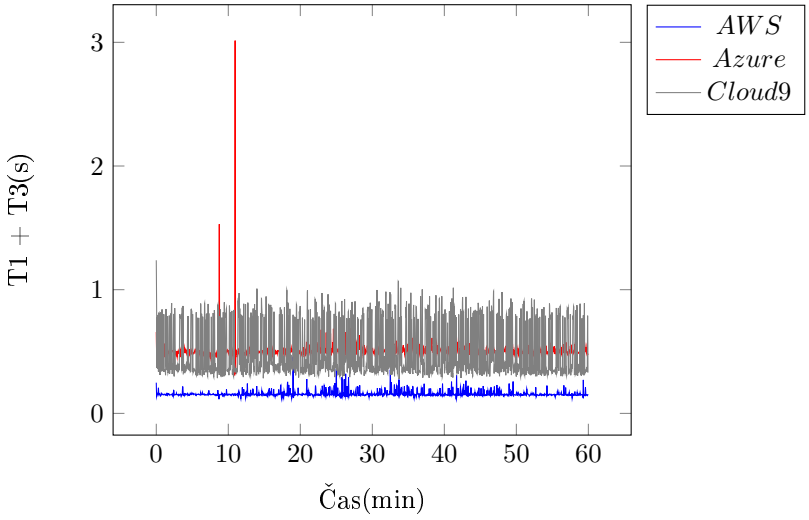
Rezultati

Čas računanja na strežniku (T2) v odvisnosti od časa testiranja je prikazan na sliki 3.4, celoten čas komunikacije (T1 + T3) spet v odvisnosti od časa testiranja pa je prikazan na sliki 3.5.

Rezultati se do neke mere ujemajo z dano hipotezo. Nekoliko so nas presenetili časi računanja na strežniku, ki je bil nameščen pri ponudniku gostovanja Azure, saj so ti očitno precej višji od časov računanja na strežniku, nameščenem pri AWS. Poleg tega, smo slabo predvideli tudi čase komunikacije s strežnikom, nameščenim pri Cloud9. Ti so, za razliko od časov pri ostalih dveh ponudnikih, razpršeni na bistveno širšem intervalu in celo presegajo čase komunikacije s strežnikom, lociranim v Južni Aziji (Azure).



Slika 3.4: Čas računanja na strežniku v odvisnosti od časa testiranja.



Slika 3.5: Celoten čas komunikacije v odvisnosti od časa testiranja.

3.7.3 Nedeterministično pošiljanje zahtevkov

Opis eksperimenta

V prvih dveh eksperimentih smo zahtevke pošiljali deterministično, točno na vsake 3 sekunde. Ker strežniki običajno niso obremenjeni na tak način, smo tokrat generirali zahtevke po Poissonovi porazdelitvi [27], saj tak model boljše

ponazori običajen promet. Časovni interval med dvema zahtevkoma je v takem primeru porazdeljen eksponentno s parametrom λ, pričakovana vrednost intervala pa je enaka $\frac{1}{\lambda}$. V našem primeru smo želeli v povprečju ohraniti 3 sekundni interval, zato smo uporabili parameter porazdelive ali intenzivnosti porajanja zahtev $\lambda = \frac{1}{3}$ zahteve na sekundo. Eksperiment je tudi tokrat trajal 1 uro, meritve pa smo opravili le pri ponudniku gostovanja AWS in sicer za računanje po Leibnizovi metodi, implementirano v Python-u.

Hipoteza

Čas računanja (T2) bo v najboljšem primeru enak tistemu iz eksperimenta z determinističnim intervalom pošiljanja zahtevkov, lahko pa pričakujemo, da se bodo nekateri zahtevki obdelovali nekoliko dlje. Zaradi naključnosti modela, se bo namreč zelo verjetno zgodilo to, da bo nek zahtevek poslan še preden bo prejšnji obdelan, kar pa bo prineslo daljši čas računanja.

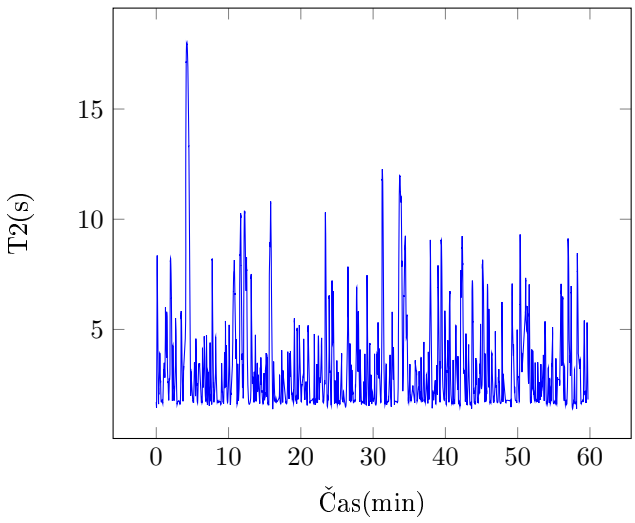
Sprememba modela za generiranje zahtevkov najverjetneje ne bo vplivala na čas komunikacije (T1 + T3), zato predvidevamo, da bo ta čas ostal nespremenjen.

Okoliščine

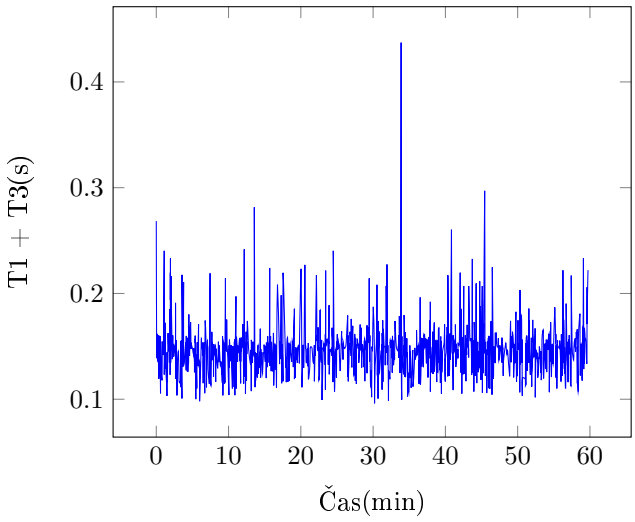
Meritve so bile opravljene v Ilirski Bistrici, s povezavo 20 Mbps / 5 Mbps. Bile so opravljene med 11. in 12. uro v ponedeljek 30. 4. 2018.

Rezultati

Čas računanja na strežniku (T2) v odvisnosti od časa testiranja je prikazan na sliki 3.6, celoten čas komunikacije (T1 + T3) spet v odvisnosti od časa testiranja pa je prikazan na sliki 3.7. Ti rezultati popolnoma potrjujejo dano hipotezo.



Slika 3.6: Čas računanja na strežniku v odvisnosti od časa testiranja.



Slika 3.7: Celoten čas komunikacije v odvisnosti od časa testiranja.

3.7.4 Ogrevanje sistema (postopno večanje obremenitve)

Opis eksperimenta

V predhodnih eksperimentih nismo nikoli pošiljali zahtevkov bolj pogosto, kot jih je strežnik lahko obdelal (v povprečju), zato smo pripravili eksperiment, kjer postopoma višamo frekvenco pošiljanja zahtevkov. Najprej smo 5 minut

pošiljali zahtevke na vsake 3 sekunde, naslednjih 5 minut na 2 sekundi, nato 5 minut vsako sekundo, zadnjih 5 minut pa vsake 0,5 sekunde (vsi intervali čakanja so bili generirani deterministično). Računali smo z Leibnizovo metodo, implementirano v Pythonu, strežnik pa smo namestili pri ponudniku gostovanja AWS.

Hipoteza

Prvih 10 minut lahko pričakujemo konstanten čas računanja (T2), saj še vedno pošiljamo zahtevke dovolj počasi. Interval med dvema zahtevkoma je namreč v tem delu testiranja najprej enak 3 in nato 2 sekundi, kar je več od povprečnega časa trajanja enega izračuna. Naslednjih 5 minut bo čas računanja najverjetneje začel linearno naraščati, saj se program izvaja več kot 1 sekundo, kar pomeni, da bodo novi zahtevki prihajali hitreje kot se lahko obdelajo in bo hkrati v obdelavi vedno več zahtevkov. Enako velja za naslednjih 5 minut, vendar pričakujemo še hitrejše naraščanje časa računanja.

Čas komunikacije (T1 + T3) bo najverjetneje konstanten skozi vse 4 faze, saj ta eksperiment predstavlja večjo obremenitev predvsem za strežnik, ne pa tudi za omrežje. Zahtevke namreč pošiljamo kvečjemu 2-krat na sekundo, paketi pa so ranga velikosti nekaj KB.

Okoliščine

Meritve so bile opravljene v Ilirski Bistrici, s povezavo 20 Mbps / 5 Mbps. Bile so opravljene med 14. in 15. uro v ponedeljek 30. 4. 2018.

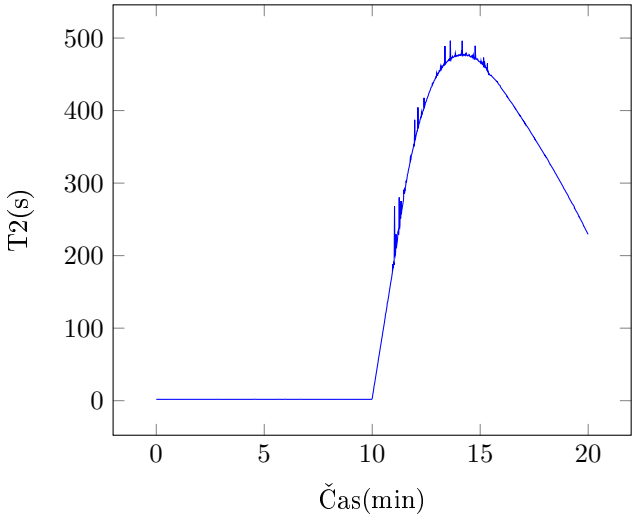
Rezultati

Čas računanja na strežniku (T2) v odvisnosti od časa testiranja je prikazan na sliki 3.8, celoten čas komunikacije (T1 + T3) spet v odvisnosti od časa testiranja pa je prikazan na sliki 3.9.

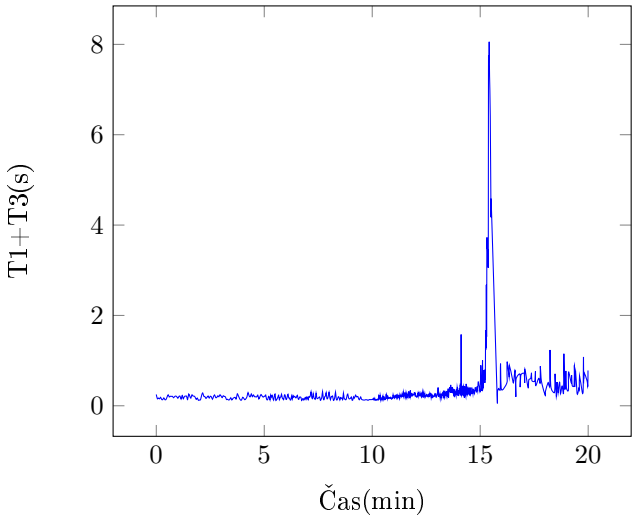
Pri postavljanju hipoteze smo tokrat naredili napako. Skozi prvi dve fazi so časi računanja res konstantni in nato nekaj časa naraščajo, vendar se malo pred koncem tretje faze naraščanje upočasni, kot bi se časi približevali neki zgornji meji, nato pa ti časi celo padajo. Približevanje zgornji meji je smiselno, saj strežnik ne more hkrati obdelovati poljubnega števila zahtevkov (zaradi končnega pomnilnika) in zato ta v neki točki začne spuščati zahtevke (od strežnika smo namreč prejeli le 692 odgovorov za 1150 poslanih zahtevkov). Padanje časov je najverjetneje povezano s koncem eksperimenta. Tisti zadnji zahtevki se (za razliko od predhodnih) obdelujejo še po koncu ekpserimenta, takrat pa novi zahtevki ne prihajajo več na strežnik. Nekaj časa na koncu se torej manjša število zahtevkov, ki so hkrati v obdelavi, to pa pomeni krajše čase računanja.

Tudi časi komunikacije se nekoliko razlikujejo od pričakovanih, vendar to odstopanje povezujemo z napako pri merjenju časa računanja (T2) v primeru, ko strežnik obdeluje veliko število zahtevkov hkrati. Lahko se zgodi, da zaradi zelo velikega števila niti, ki obdelujejo posamezne zahtevke, glavna nit, ki sprejema

zahtevke in meri čas računanja, nekaj časa ne pride na vrsto in posledično zamudi pri beleženju časa, ko je zahtevek prišel do strežnika. Ta napaka se odraža kot nekaj sekundni pribitek času komunikacije in enako dolg odbitek času računanja, žal pa je odvisna od operacijskega sistema, ki razvršča niti in je ne moremo odpraviti.



Slika 3.8: Čas računanja na strežniku v odvisnosti od časa testiranja.



Slika 3.9: Celoten čas komunikacije v odvisnosti od časa testiranja.

3.7.5 Konstantna visoka obremenitev

Opis eksperimenta

Pri prejšnjem eksperimentu so nas rezultati rahlo presenetili, zato smo želeli našo interpretacijo rezultatov preveriti še z dodatnim eksperimentom. V ta namen smo tokrat izvedli meritve le za tretjo fazo prejšnjega eksperimenta (deterministično pošiljanje zahtevkov na vsako sekundo), vendar smo eksperiment izvajali več časa - 1 uro.

Hipoteza

Pričakujemo, da bodo časi računanja (T2) najprej naraščali dokler ne dosežejo zgornje meje, nato bodo večji del testiranja ti časi približno konstantni, pred koncem pa se bodo časi spet zmanjšali.

Čas komunikacije (T1 + T3) bo najverjetneje spet višji kot bi bilo smiselno, vendar bo to zaradi napake merjenja časov, omenjene pri prejšnjem eksperimentu. Če so višji izmerjeni časi komunikacije res posledica napake, potem bodo ti časi tudi nekoliko korelirani s časi računanja. Časi računanja namreč dobro kažejo na obremenjenost sistema z velikim številom hkrati aktivnih niti, velikost napake pa je odvisna ravno od števila niti, ki si z glavno nitjo delijo računske vire.

Okoliščine

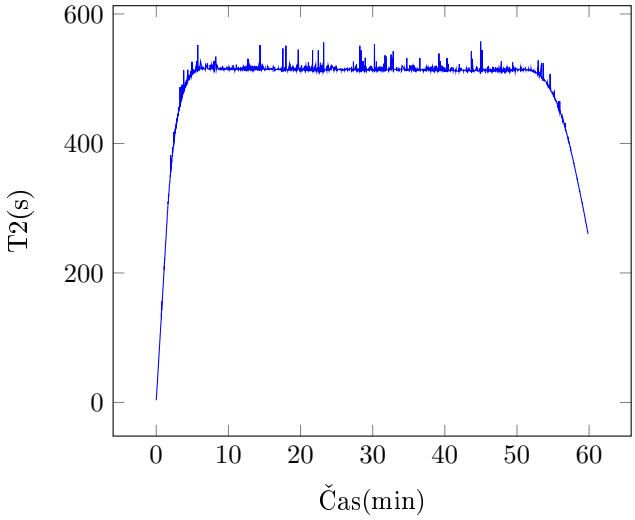
Meritve so bile opravljene v Ilirski Bistrici, s povezavo 20 Mbps / 5 Mbps. Bile so opravljene med 15. in 16. uro v ponedeljek 30. 4. 2018.

Rezultati

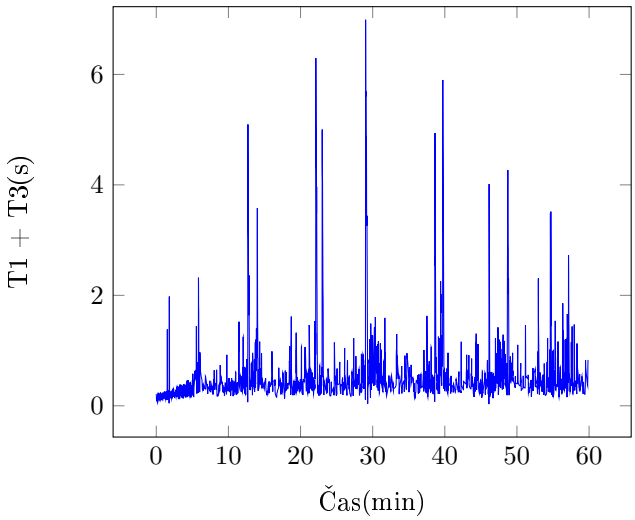
Čas računanja na strežniku (T2) v odvisnosti od časa testiranja je prikazan na sliki 3.10, celoten čas komunikacije (T1 + T3) spet v odvisnosti od časa testiranja pa je prikazan na sliki 3.11.

Rezultati se ujemajo z dano hipotezo, to pa potrjuje tudi celotno interpretacijo rezultatov iz prejšnjega eksperimenta.

Zanimivo je še pogledati število zahtevkov, ki so bili na strežniku spuščeni. Od 3600 poslanih zahtevkov, smo odgovor dobili le za 2049 zahtevkov. V eni uri je strežnik torej 2049-krat uspešno izračunal število π, kar pomeni, da je za posamezen izračun efektivno potreboval le 1,76 s ($\frac{3600s}{2049}$), ta čas je pa zelo podoben času, ki ga tak strežnik potrebuje za en izračun, ko sploh ni preobremenjen. Iz tega sledi, da strežnik v primeru preobremenitve spusti praktično najmanjše možno število zahtevkov. Če bi želeli spuščanje zahtevkov popolnoma preprečiti, bi te lahko generirali kvečjemu na vsake 1,76 s (interval je torej potrebno prilagoditi času, ki ga strežnik potrebuje za en izračun).



Slika 3.10: Čas računanja na strežniku v odvisnosti od časa testiranja.



Slika 3.11: Celoten čas komunikacije v odvisnosti od časa testiranja.

3.7.6 Primerjava različno intenzivnih preobremenitev

Opis eksperimenta

Pri prejšnjem eksperimentu smo zahtevke pošiljali pogosteje kot jih lahko strežnik obdela in opazili, da se v neki točki časi računanja ustalijo. To se zgodi, ko je dosežena zgornja meja za število zahtevkov, ki se lahko hkrati obdelujejo

(tisti zahtevki, ki bi to mejo presegli pa so izpuščeni in nanje ne dobimo odgovora). S tem eksperimentom bi radi preverili, do kakšne spremembe pride pri časih računanja, če zahtevke pošiljamo nekoliko pogosteje ali pa nekoliko redkeje (vendar še vedno bolj pogosto, kot jih lahko strežnik obdela). Zato smo tokrat 20 minut deterministično pošiljali zahtevke, na vsake 0,8 s, na vsako sekundo ter na vsake 1,2 s do strežnika, nameščenega pri ponudniku AWS. Eksperimenti so potekali istočasno, vsak na svoji instanci. Vse instance so identične.

Hipoteza

Čas računanja je v našem primeru odvisen od števila zahtevkov, ki se hkrati obdelujejo, zato pričakujemo, da bodo časi računanja naraščali hitreje v primeru, kjer pogosteje pošiljamo zahtevke in obratno. Poleg tega, lahko pričakujemo, da se bodo v vseh treh primerih časi ustalili pri približno enaki vrednosti, saj bo meja, pri kateri strežnik spušča zahtevke enaka v vseh treh primerih, ne glede na to, kako pogosto pošiljamo zahtevke.

Pri časih komunikacije ne pričakujemo večjih razlik, saj noben od treh primerov pošiljanja zahtevkov ne predstavlja resnejše obremenitve za omrežje. Po drugi strani pa v vseh treh primerih vseeno pričakujemo prisotnost že omenjene napake pri merjenju časov, saj bo sistem obremenjen s precej visokim številom niti.

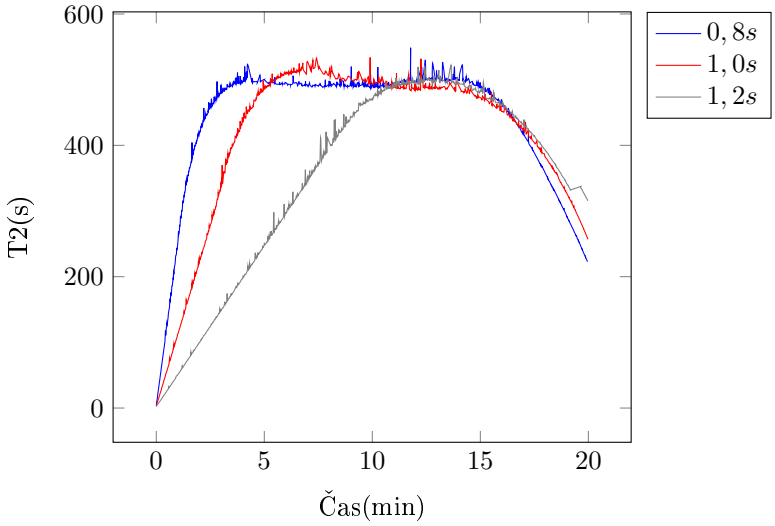
Okoliščine

Meritve so bile opravljene v Sežani, s povezavo 10 Mbps / 10 Mbps. Bile so opravljene med 7. in 8. uro v soboto 19. 5. 2018.

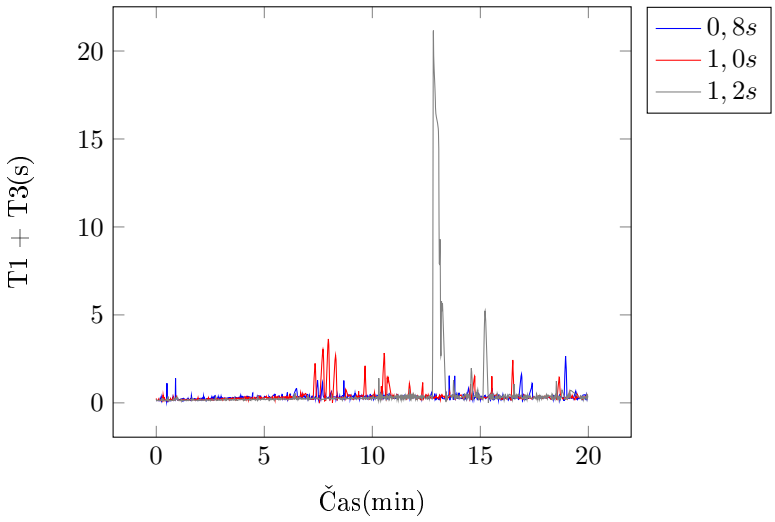
Rezultati

Čas računanja na strežniku (T2) v odvisnosti od časa testiranja je prikazan na sliki 3.12, celoten čas komunikacije (T1 + T3) spet v odvisnosti od časa testiranja pa je prikazan na sliki 3.13.

Hipoteza se je izkazala za pravilno, saj časi računanja najprej naraščajo intervalu primerno, nato pa se v vseh treh primerih ustalijo na približno 500 s. Tudi časi komunikacije ustrezajo hipotezi, vključno z omenjeno napako, ki je še posebej vidna v primeru 1,2 s intervala.



Slika 3.12: Čas računanja na strežniku v odvisnosti od časa testiranja.



Slika 3.13: Celoten čas komunikacije v odvisnosti od časa testiranja.

3.7.7 Eksperimentalno določanje meje preobremenitve

Opis eksperimenta

V enem od prejšnjih eksperimentov (razdelek 3.7.5) je že bilo govora o največji možni obremenitvi strežnika, pri kateri se zahtevki pravočasno obdelujejo in se novi zahtevki ne kopičijo v vrsti. Predvideli smo, da je interval pošiljanja

potrebno prilagoditi povprečnemu času računanja posameznega zahtevka, s tem eksperimentom pa bi to želeli še dodatno preveriti. V našem primeru je za strežnik, nameščen pri ponudniku gostovanja AWS ta čas enak približno 1,76 s. Zato smo tokrat za gostovanje spet uporabili ponudnika AWS, zahtevke pa smo 20 minut pošiljali nedeterministično z intervali 1,70 s, 1,75 s ter 1,80 s. Eksperimenti so potekali istočasno, vsak na svoji instanci. Vse instance so identične.

Hipoteza

Najverjetneje bodo časi računanja v obeh primerih, kjer je interval generiranja zahtevkov nižji od povprečnega časa računanja nekoliko naraščali skozi čas, v primeru višjega intervala pa pričakujemo, da bo čas računanja približno konstanten. Seveda ne smemo pozabiti, da se tokrat zahtevki generirajo nedeterministično, kar pomeni, da bo pri vseh treh primerih prisotno tudi nekakšno nihanje v časih računanja - zgornje trditve se torej nanašajo bolj na trend, kot pa na dejansko gibanje vrednosti skozi čas.

Pri časih komunikacije v tem eksperimentu ne pričakujemo nobenih večjih posebnosti.

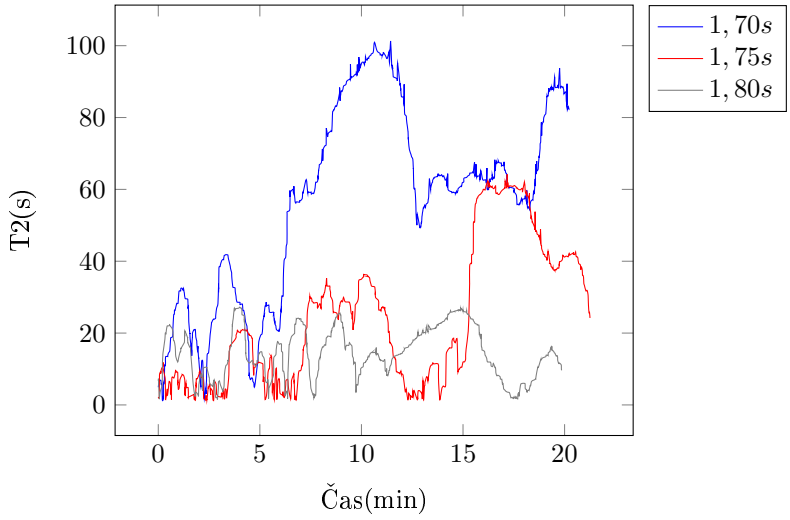
Okoliščine

Meritve so bile opravljene v Sežani, s povezavo 10 Mbps / 10 Mbps. Bile so opravljene med 13. in 14. uro v soboto 19. 5. 2018.

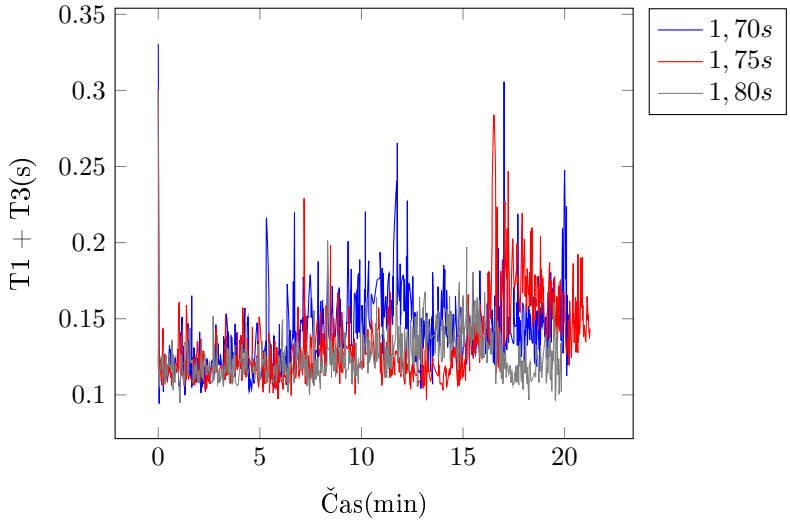
Rezultati

Čas računanja na strežniku (T2) v odvisnosti od časa testiranja je prikazan na sliki 3.14, celoten čas komunikacije (T1 + T3) spet v odvisnosti od časa testiranja pa je prikazan na sliki 3.15.

Na prvi pogled izgledajo časi računanja v vseh treh primerih precej podobno, vendar lahko opazimo, da se v primeru 1,80 s intervala časi računanja po naraščanju vedno vrnejo do običajnega nivoja, v ostalih dveh primerih pa časi postopoma naraščajo intervalu primerno in s tem kažejo na predpostavljeno preobremenitev. S časi komunikacije v tem eksperimentu res ni bilo posebnosti. Izkaže se, da ti sploh niso korelirani z intervalom pošiljanja zahtevkov, kar je tudi smiselno za tako majhne razlike v obremenitvi omrežja.



Slika 3.14: Čas računanja na strežniku v odvisnosti od časa testiranja.



Slika 3.15: Celoten čas komunikacije v odvisnosti od časa testiranja.

3.7.8 24-urni eksperiment

Opis eksperimenta

S prejšnjim eksperimentom smo potrdili, da mejni interval generiranja zahtevkov, preden sistem preobremenimo, leži nekje med 1,75 s ter 1,80 s (za ponudnika gostovanja AWS). Kot zadnji eksperiment pa smo želeli pripraviti še testiranje,

kjer sistem obremenimo na približno dve tretjini mejne obremenitve, vendar takšno obremenitev vzdržujemo celih 24 ur. Zahtevke smo torej pošiljali 24 ur do strežnika, nameščenega pri ponudniku gostovanja AWS, nedeterministično z intervalom 2,65 s.

Hipoteza

Ta eksperiment simulira promet, pri katerem uporabniki občiajno že opazijo, da sistem deluje rahlo počasneje kot sicer. Pričakujemo torej nekoliko višje čase računanja (v primerjavi s časom, ki je potreben za obdelavo enega samega zahtevka).

Pri časih komunikacije tudi tokrat ne pričakujemo večjih posebnosti. Pozorni bomo lahko na razlike skozi različne ure dneva, vendar nismo prepričani, če bodo te razlike dovolj očitne.

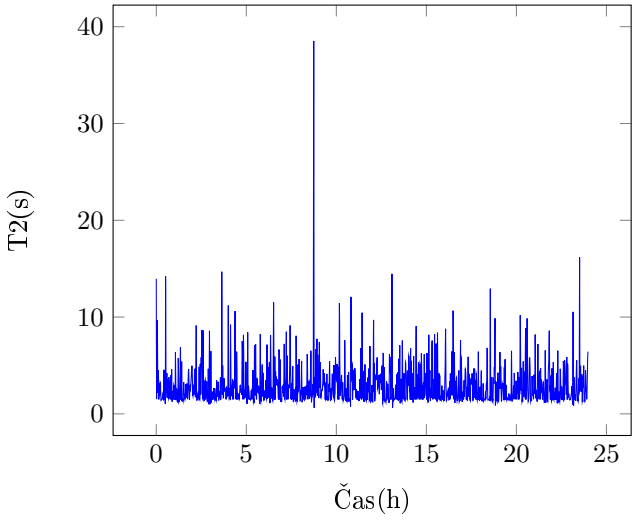
Okoliščine

Meritve so bile opravljene v Sežani, s povezavo 10 Mbps / 10 Mbps. Bile so opravljene med 14. uro v soboto 19. 5. 2018 in 14. uro v nedeljo 20. 5. 2018.

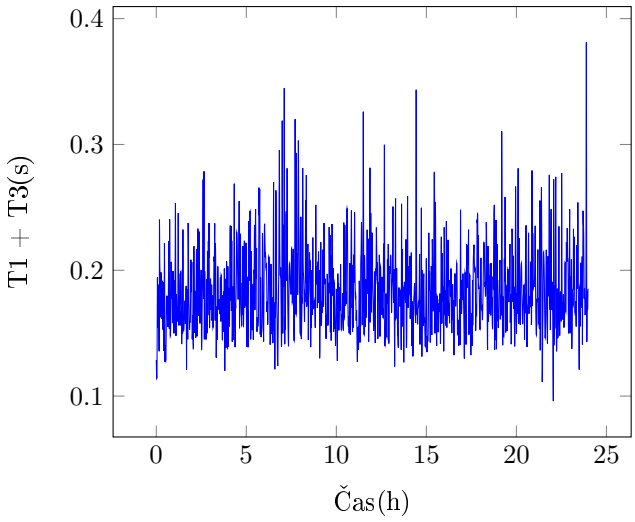
Rezultati

Čas računanja na strežniku (T2) v odvisnosti od časa testiranja je prikazan na sliki 3.16, celoten čas komunikacije (T1 + T3) spet v odvisnosti od časa testiranja pa je prikazan na sliki 3.17.

Rezultati v celoti potrjujejo hipotezo. V najslabšem primeru je računanje trajalo skoraj 40 s (sicer pa za obdelavo enega samega zahtevka v povprečju preteče približno 1,76 s). Iz časov komunikacije žal ne razberemo odvisnosti od ure dneva.



Slika 3.16: Čas računanja na strežniku v odvisnosti od časa testiranja.



Slika 3.17: Celoten čas komunikacije v odvisnosti od časa testiranja.

3.8 Zaključek

Z zgornjimi eksperimenti smo ovrednotili časovno zahtevnost štirih različnih načinov računanja števila π in zmogljivost strežnikov pri treh različnih ponudnikih gostovanja. Iz rezultatov lahko razberemo, da je Bellardova metoda konstantno hitrejša od Leibnizove ne glede na implementacijo. Prav tako lahko opazimo, da

je računanje pri ponudniku Cloud9 najhitrejše, pri Azure najpočasnejše, AWS pa pade nekje vmes. Drugačno razporeditev vidimo pri času komunikacije, kjer je pri ponudniku AWS najkrajši, zaradi nam ugodne lokacije strežnikov. Azure in Cloud9 imata podoben čas komunikacije, vendar ima slednji veliko večjo varianco.

S stopnjevanim deterministčnim pošiljanjem zahtevkov smo prišli do ugotovitve, da strežnik pri preveliki obremenitvi začne spuščati zahtevke. Na podlagi te ugotovitve, smo z dodatnim testom določili zgornjo mejo obremenitve (1,76 s), pri kateri strežnik še obdela vse poslane zahtevke (ne glede na čas testiranja). V eksperimentu 1.6.6 smo opazovali tudi kako različni intervali pošiljanja zahtevkov vplivajo na hitrost polnjenja vrste, pri tem pa smo tudi določili zgornjo mejo časa izračuna posameznega zahtevka (v primeru maksimalnega števila zahtevkov v hkratni obdelavi). Opazimo tudi, da proti koncu testiranja strežnik potrebuje vse manj časa za obdelavo posameznega zahtevka, saj se po zadnjem poslanem zahtevku, hkrati obdeluje vedno manjše število zahtevkov. To potrdimo še z dodatnim eksperimentom, kjer zahtevke eno uro deterministično pošiljamo vsako sekundo.

Ko smo določili zgornjo mejo frekvence deterministčnega pošiljanja zahtevkov, smo strežnik preizkusili tudi na način ki simulira realno okolje. Zahtevke smo večkrat nedeterministično pošiljali z različnimi frekvencami, blizu predvidene meje preobremenitve. Opazimo, da se zahtevki na strežniku počasi kopičijo in včasih nastanejo daljše čakalne vrste, kljub temu pa v primeru, ko pošiljamo z nekoliko manjšo frekvenco od predvidene meje preobremenitve, ne izgubimo nobenega zahtevka. Za konec na strežnik nedeterministično pošiljamo zahtevke pri $\frac{2}{3}$ sposobnosti delovanja strežnika, saj je to meja pri kateri uporabnik občuti zakasnitve. Čas obdelovanja zahtev je v tem primeru podoben kot v prvem eksperimentu, kjer strežnik ni bil tako intenzivno obremenjen, le da občasno pride do nekoliko večjih časov računanja.

V primerjavi z meritvami v članku "A Performance Analysis of EC2 Cloud Computing Services for Scientific Computing" [23], kjer so namereli 0.6 GOPS je naš rezultat z brezplačnim paketom t2.micro približno 0.42 GOPS. To smo izračunali s pomočjo Leibnizove metode implementirane v c-ju. Celoten čas izračuna je približno 0,12 s, za izračun je pa potrebnih približno 25 milijonov seštevanj in deljenj. V enem izračunu se torej zgodi približno 50 milijonov operacij, opravimo pa lahko 8,3 izračunov na sekundo kar pomeni da dosežemo 0,416 GOPS. Teoretična maksimalna zmogljivost procesorja je 4,4 GOPS [23], ponudnik pa nam da na razpolago 10 odstotkov tega [28], torej 0,44 GOPS. To pomeni, da je bilo v času našega testiranja izkoriščenih 94,5% računske moči.

Poglavje 4

Analiza zmogljivosti ponudnika PythonAnywhere

Žiga Babnik, Mitja Maren, Matej Horvat, Gašper Jelovčan

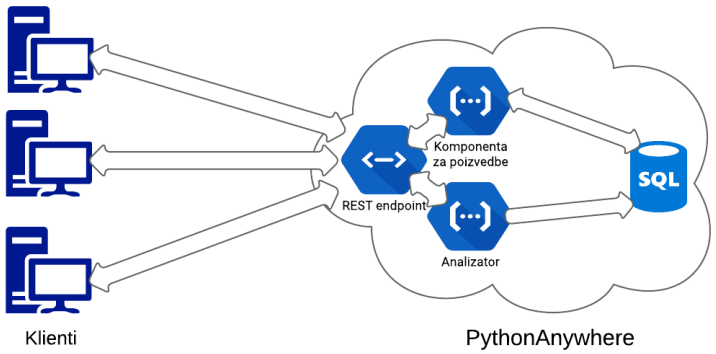
4.1 Opis oblačne storitve

V pričujočem poglavju predstavimo zanesljivost in zmogljivost storitve oblačnega ponudnika PythonAnywhere. Analizirali jo bomo z aplikacijo štetja frekvenc besed v tekstovnih datotekah. Odjemalci bodo na strežnik poslali neko besedilo, strežnik pa bo izračunal število pojavitev posameznih besed v njem, rezultat pojavitev pa se bo shranjeval v podatkovno bazo, iz katere se bo dalo poizvedovati rezultate. Ta problem, tako imenovani text mining [29], se pojavlja veliko po svetu. Primer takšne aplikacije opisuje članek [30], kjer je opisana aplikacija TextArc.

Strežnik bo vseboval dve komponenti: analizator besedil ter komponento za poizvedovanje. Analizator bo iz zahtevka prebral besedilo, preštel frekvence besed ter jih shranil v podatkovno bazo. Primer dobrega analizatorja besedil najdemo v članku [31]. Komponenta za poizvedbe bo v zahtevku pobrala poizvedbo ter vrnila rezultate poizvedbe iz podatkovne baze. Kot zanimivost: članek [32] opisuje zakon pojavitve besed v besedilih.

Aplikacija sicer ni široko uporabna, vendar menimo, da se z njo da dobro preveriti procesorsko moč, hitrost omrežne povezave in hitrost delovanja podatkovne baze strežnika. Slika 4.1 prikazuje zgradbo aplikacije.

4.2 Izbira tehnologij

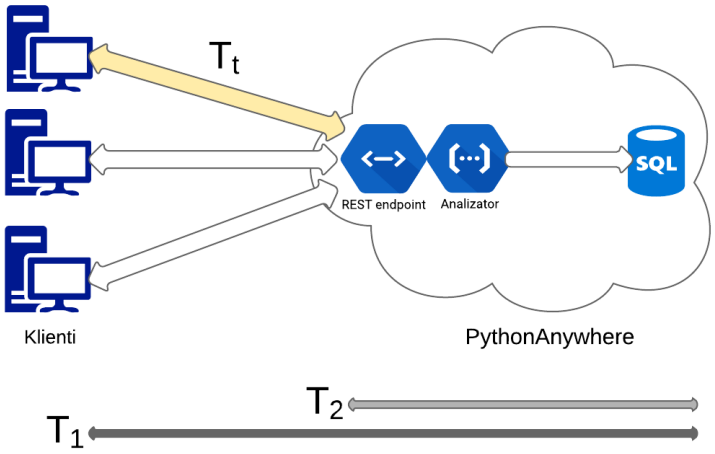


Slika 4.1: Shema delovanja aplikacije.

Uporabili smo ponudnika storitev PythonAnywhere, ki je specializiran za gostovanje aplikacij, napisanih v jeziku Python. V ozadju za svoje delovanje uporablja Amazon EC2 [33] in ga je mogoče z nekaj omejitvami uporabljati zastonj. Te omejitve so sledeče:

- Na voljo imamo le 512 MB prostora na disku.

- Kodo lahko izvajamo s polno hitrostjo le 100 procesorskih sekund na dan. Procesorska sekunda pomeni eno sekundo procesiranja, pri katerem je procesor v celoti zaseden [34]. Več niti lahko porabi več procesorskih sekund v eni realni sekundi, vendar ni dokumentirano, koliko jeder je na voljo. Ko porabimo vse procesorske sekunde, se naša koda izvaja z najnižjo prioriteto (relativno na druge uporabnike strežnika) [35].

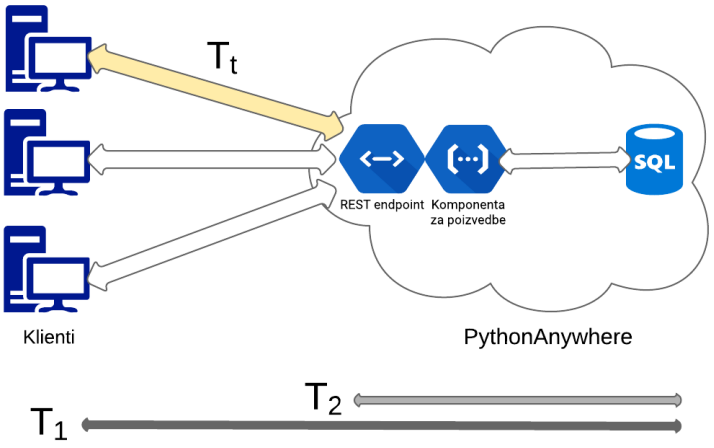


Slika 4.2: Shema delovanja aplikacije z vidika analizatorja.

Slika 4.2 prikazuje tok datotek oziroma zahtev preko aplikacije z vidika analizatorja. Na skici so označeni tudi časi, ki jih bomo merili oziroma izračunali. T_1 predstavlja celoten čas, potreben za zahtevo. Meritev zanj se v celoti izvede na strani klienta. Čas začetka in konca pošiljanja odčitamo s klicem *time.time*(), končni čas T_1 pa izračunamo kot razliko med časom konca in začetka pošiljanja, kot prikazuje izpis 4.1. T_2 predstavlja čas delovanja analizatorja. Ta se izmeri na strani oblačne storitve. Meritev se začne, ko oblačna storitev prejeme zahtevo, zaključi pa, ko analizator zaključi s preštevanjem besed. T_t je čas potovanja zahteve po omrežju, ki ga lahko izračunamo na podlagi razlike izmerjenih časov T_1 in T_2. Za meritve, povezane z analizatorjem, je pomembno še dejstvo, da se med vsako posamezno meritvijo počisti baza na oblačni storitvi. Tako zagotovimo večjo neodvisnost posameznih meritev in poskrbimo, da ne zapolnimo prostora na disku. Čas, porabljen za čiščenje, ni štet v meritvah.

Listing 4.1: Prikaz kode za merjenje časa T_1.

```
zacetniCas = time.time()
response = requests.post(restResource, data=data)
celotniCas = time.time() - zacetniCas
```



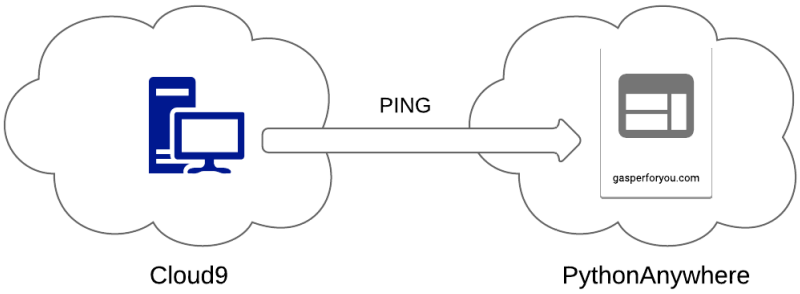
Slika 4.3: Shema delovanja aplikacije z vidika komponente za poizvedbe.

Slika 4.3 prikazuje tok datotek oziroma zahtev preko aplikacije z vidika komponente za poizvedbe. Na skici so označeni tudi časi, ki jih bomo merili oziroma lahko izračunali. T_1 predstavlja celoten čas, potreben za zahtevo, T_2 predstavlja čas delovanja analizatorja, čas T_t pa je čas potovanja, ki ga lahko izračunamo na podlagi izmerjenih časov T_1 in T_2. Samo merjenje tu poteka drugače kot pri analizatorju, saj je znotraj zahteve potrebno čakati na odgovor baze, pri čemer to ni potrebno v primeru analizatorja.

4.3 Definicija bremena

Breme predstavljajo tekstovne datoteke knjig, pridobljene s spletne strani Project Gutenberg [36] s pomočjo krajše skripte. Te uporabljamo na zaključnih testih, ki predstavijo odziv aplikacije v realni situaciji in potrdijo naše razumevanje, pridobljeno s pošiljanjem raznih sintetičnih bremen. Ta so podrobneje predstavljena pri posameznih poizkusih, v splošnem pa so to tekstovne datoteke z nizi znakov, kjer se spreminja povprečna dolžina nizov, število besed in druge lastnosti, ki nas zanimajo.

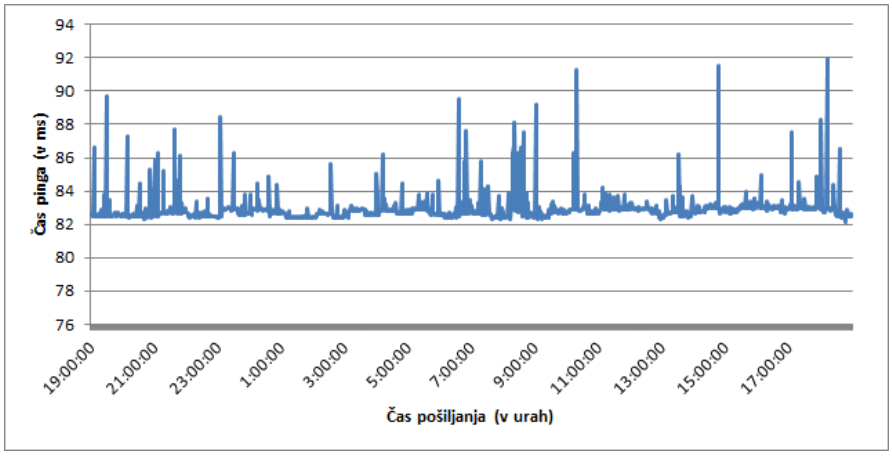
4.4 Klienti za testiranje analizatorja



Slika 4.4: Slika ponazarja potek testiranja povezave.

Klienti, ki testirajo delovanje in zmogljivost oblačne aplikacije, podrobneje teste povezane z analizatorjem, tečejo na oblačni storitvi Cloud9 [37], ki je del Amazonovega oblaka AWS (Amazon Web Services). Ta ponuja virtualko z operacijskim sistemom Linux kot rešitev programiranja v oblaku. Specifikacije virtualke, na kateri tečejo klienti, ki jih je možno dobiti, so sledeče:

- velikost pomnilnika (RAM): 1 GB,
- velikost diska (HDD): 5 GB,
- število procesorjev: 1.



Slika 4.5: Graf pingov oblačne storitve Cloud9 na našo oblačno storitev.

Slika 4.5 prikazuje rezultate testiranja spletne povezave med klienti na oblačni storitvi Cloud9 in našo oblačno storitvijo. Testiranje je potekalo, kot prikazuje

slika 4.4. Samo pošiljanje pingov je potekalo od ponedeljka, 16. 4. 2018 od 19:00:00 (+0) do torka, 17. 4. 2018 19:00:00 (+0), ping pa se je vršil vsako minuto. Tako dobimo rezultate, ki nakazujejo, da povezava med klienti in oblačno storitvijo sicer ni zelo hitra (v povprečju je čas malce nad 82 ms), ampak dokaj konstantna, kar je razvidno tudi iz grafa. Prav tako ne zaznamo nobenih globalnih trendov iz rezultatov, ki bi nakazovali, da se tekom nekega dela dneva hitrost povezave spremeni.

Želimo izvedeti tudi kaj več o sami poti med klienti, torej oblačno storitvijo Cloud9 in našo oblačno storitvijo. Zato na virtualki, na kateri tečejo tudi klienti za testiranje analizatorja, poženemo ukaz tcptraceroute, ki nam po korakih prikaže skoke paketov na poti med klienti in aplikacijo na oblačni storitvi PythonAnywhere.

Listing 4.2: Rezultat analize omrežne poti med klienti za testiranje analizatorja in našo oblačno storitvijo.

```
traceroute to gasperforyou.pythonanywhere.com (34.206.101.184), 30
    hops max, 44 byte packets
 1  172.17.0.1 (172.17.0.1) 0.065 ms 0.064 ms 0.130 ms
 2  108.170.235.156 (108.170.235.156) 82.480 ms 108.170.232.198
     (108.170.232.198) 82.541 ms 216.239.35.163 (216.239.35.163)
     82.632 ms
 3  108.170.246.13 (108.170.246.13) 82.507 ms 82.522 ms
     108.170.246.47 (108.170.246.47) 82.392 ms
 4  52.95.219.138 (52.95.219.138) 82.555 ms 82.782 ms 52.95.219.140
     (52.95.219.140) 83.263 ms
 5  * * *
 6  52.93.27.0 (52.93.27.0) 95.534 ms 52.93.27.30 (52.93.27.30)
     101.764 ms 52.93.27.2 (52.93.27.2) 94.269 ms
 7  52.93.27.41 (52.93.27.41) 103.926 ms 52.93.27.11 (52.93.27.11)
     100.927 ms 52.93.27.45 (52.93.27.45) 99.710 ms
 8  54.239.108.95 (54.239.108.95) 82.729 ms 52.93.25.118
     (52.93.25.118) 82.344 ms 54.239.109.9 (54.239.109.9) 82.646 ms
 9  52.93.24.140 (52.93.24.140) 122.872 ms * 111.595 ms
10  52.93.24.97 (52.93.24.97) 82.427 ms 52.93.24.121 (52.93.24.121)
    82.251 ms 52.93.24.137 (52.93.24.137) 82.542 ms
11  * * *
12  * * *
13  * * *
14  * * *
15  * * *
16  ec2-34-206-101-184.compute-1.amazonaws.com (34.206.101.184)
    <syn,ack> 82.806 ms 82.751 ms 82.879 ms
```

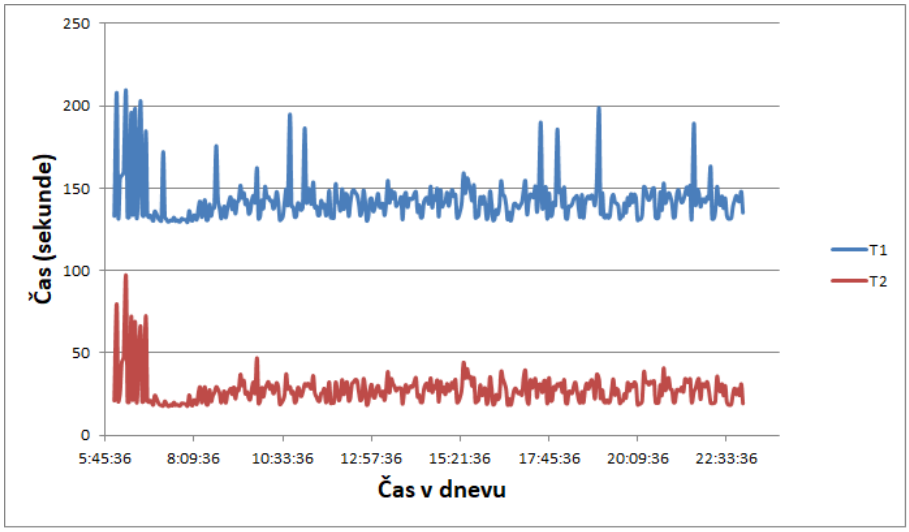
Izpis 4.2 prikazuje rezultate analize omrežne poti med klienti, pozicioniranimi na oblačni storitvi Cloud9, namenjeni testiranju analizatorja in našo oblačno storitvijo na oblačnem ponudniku PythonAnywhere. Razvidno je, da paketi naredijo šestnajst skokov na poti v omrežju, ter da paketi potrebujejo največ časa prav na prvem skoku. Podrobnejša analiza končnega naslova (vr-

stica 16 na sliki 4.2), ki jo izvajamo s pomočjo storitve **IpData** [38], pokaže, da se ta nahaja v Združenih državah Amerike, natančneje v mestu Ashburn v zvezni državi Virginiji.

Opazili smo, da se IP naslov naše oblačne storitve sčasoma spreminja, kar nakazuje, da se naša koda ne izvaja vedno na istem fizičnem strežniku, vendar nismo opazili, da bi se storitev kdaj premaknila na drugo geografsko lokacijo.

4.5 Analiza dnevnega delovanja aplikacije

Zanima nas, kako je delovanje naše oblačne storitve odvisno od ure v dnevu. Predpostavljamo, da obstajajo ure v dnevu, ko je zasedenost oblačnega ponudnika PythonAnywhere oziroma AWS večja. Pričakujemo, da bo takrat delovanje naše storitve upočasnjeno. Izvedeti želimo, ali so naše predpostavke pravilne in natančneje kako se hitrost naše storitve spreminja iz ure v uro. Meritve izvajamo v urah, v katerih tipično izvajamo vse naprej omenjene meritve. To je med 6:00 in 23:00. Proti oblačni storitvi pošiljamo večja bremena, saj bomo tako dobili večjo variacijo med hitrejšim in počasnejšim delovanjem oblačne aplikacije. Natančneje je to sintetično breme, ki vsebuje deset milijonov besed, vsaka od teh je dolga natanko pet znakov. Meritve se izvajajo neprekinjeno, torej ko klient dobi odgovor na trenutno poslano zahtevo, se proti oblačni aplikaciji takoj pošlje nova zahteva.

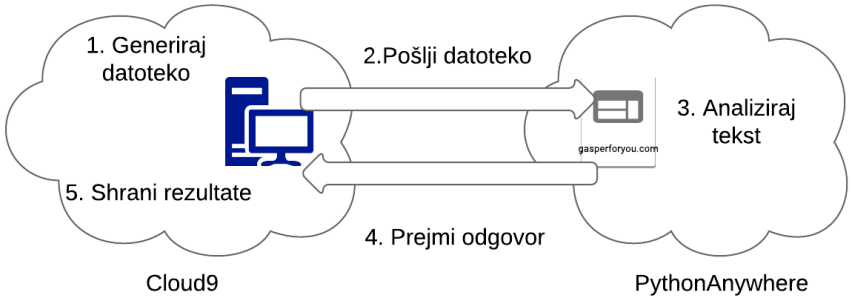


Slika 4.6: Slika ponazarja odvisnost časa delovanja aplikacije od časa pošiljanja.

Vse meritve na sliki 4.6 se vršijo na klientu, pozicioniranem na oblačni storitvi Cloud9, 22. 5. 2018 od 6:00:00 (+0), do 23:00:00 (+0).

Slika 4.6 prikazuje odvisnost celotnega časa zahteve T_1 in časa analize T_2 od samega časa pošiljanja zahteve v dnevu. Na sliki 4.6 lahko opazimo, da so rezultati čez dan več ali manj enakomerni. V začetni uri lahko opazimo več daljših časov T_1 in T_2. Skozi preostale ure lahko opazimo enakomerno gibanje časov T_1 in T_2, z občasnimi skoki časa T_1. Te lahko pojasnimo z zamašenostjo omrežja med oblačnima storitvama.

4.6 Rezultati meritev - analizator

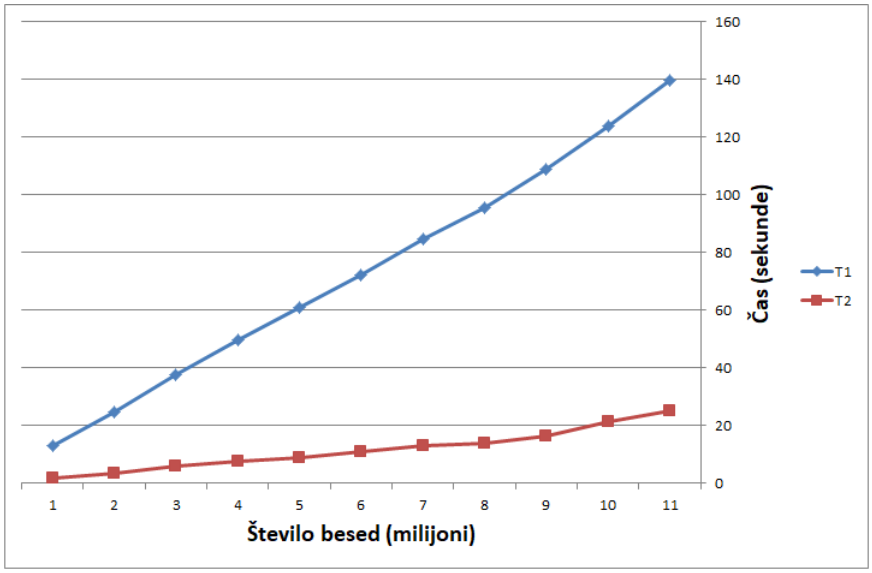


Slika 4.7: Slika ponazarja potek testiranja aplikacije z vidika analizatorja.

Vse meritve v tem razdelku potekajo po principu, nakazanem na sliki 4.7.

4.6.1 Odvisnost med številom besed v datoteki in merjenim časom

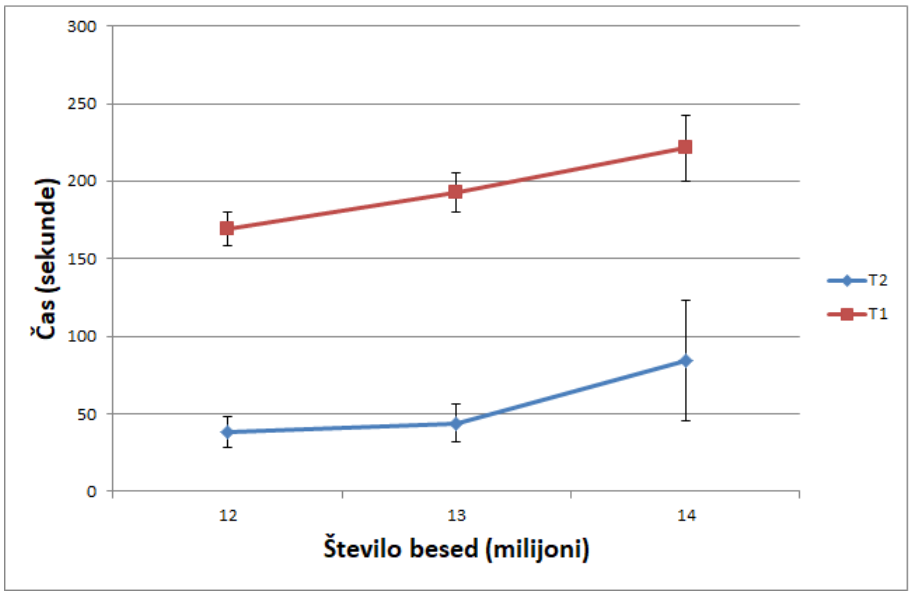
Oblačno storitev želimo preizkusiti z vidika strojne opreme. Konkretno nas v tem primeru zanima, kako se bo storitev odzvala, če pošiljamo vedno večje tekstovne datoteke proti oblačni storitvi. Tako testiramo pomnilniške zmožnosti in v manjši meri tudi procesne zmožnosti storitve. Konkretno uporabljamo tekstovne datoteke, ki vsebujejo med enim in enajstimi milijoni besed, pri čemer je dolžina vseh besed enaka pet znakov, kar nam omogoča generiranje ravno malo več kot enajst milijonov unikatnih besed. Upamo, da bo zgornja meja enajst milijonov besed že dovolj za večjo obremenitev storitve. V nasprotnem primeru potrebujemo nov sistem generiranja besed.



Slika 4.8: Graf odvisnosti časov T_1 in T_2 od števila besed.

Vse vrednosti na sliki 4.8 so izračunane kot povprečje tridesetih meritev. Meritve se vršijo na klientu, pozicioniranem na oblačni storitvi Cloud9, 9. 4. 2018 od 18:25:00 (+0), do istega dne 18:57:00 (+0).

Slika 4.8 prikazuje odvisnost celotnega časa pošiljanja T_1 in časa analize T_2 od števila besed v poslanih datotekah. Na žalost nismo opazili hudega porasta pri trenutnem povečevanju besed. To nakazuje, da moramo to število še drastično povečati, da bomo oblačno storitev dovolj obremenili do točke, ko bo začela odpovedovati. Ker smo v razdelku 4.5.2 ugotovili, da število unikatnih besed na čas pošiljanja T_1 ali analize T_2 nima vpliva, bomo to ugotovitev uporabili pri generiranju večjih datotek. Tako generiramo niz, sestavljen iz enajstih milijonov unikatnih besed, na koncu pa do potrebnega števila besed v niz dodajamo besede *aaaaa*. Ta pristop nam v praksi omogoča generiranje datotek s poljubno velikim številom besed.



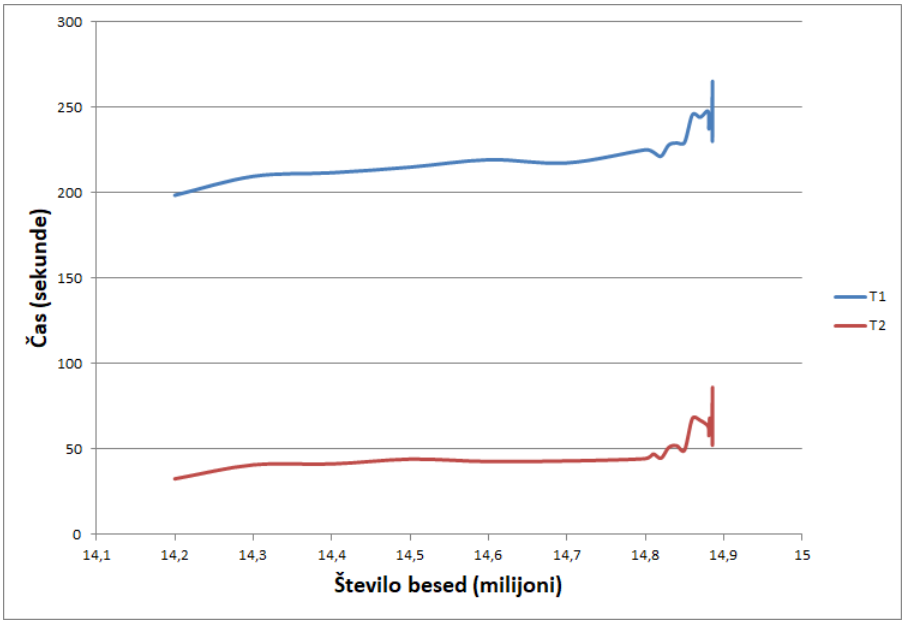
Slika 4.9: Graf odvisnosti časov T_1 in T_2 od števila besed.

Vse vrednosti na sliki 4.9 so izračunane kot povprečje desetih meritev. Meritve se vršijo na klientu, pozicioniranem na oblačni storitvi Cloud9, 20. 4. 2018 od 13:51:00 (+0), do istega dne 15:11:00 (+0).

Slika 4.9 prikazuje gibanje časov T_1 in T_2, v odvisnosti od števila besed v datoteki, ko število besed še nadaljno povečujemo na vsakem koraku za milijon besed. Pri petnajstih milijonih besed v poslani datoteki aplikacija ne vrne več odgovora. Velika deviacija rezultatov je vidna tudi pri trinajstih in štirinajstih milijonih besed, kjer postane čas analize zelo šumen. Na sliki 4.9 je označena z intervali pri posameznih velikostih, ki nakazujejo odmik za +/-1 standardne deviacije podatkov. Za boljšo predstavo velikosti omenjenih datotek lahko povemo, da so posamezne datoteke velike skoraj 100 MB.

Podrobno nas zanima stanje med štirinajstimi in petnajstimi milijoni besed, zato imamo pripravljeno strategijo pošiljanja, ki nam omogoča natančno analizo točke odpovedi. Predpostavimo, da je število besed v datoteki štirinajst milijonov in da se v vsakem koraku to število poveča za milijon besed. Torej v naslednjem koraku pošiljamo datoteko, ki vsebuje petnajst milijonov besed. Recimo, da v tej točki (petnajst milijonov besed v datoteki) aplikacija odpove. Ker klient od aplikacije ne bo dobil odgovora, se vrne na prejšnjo točko, kjer je pošiljal štirinajst milijonov besed v datoteki. To stori tako, da od števila besed, pri katerem je aplikacija odpovedala, odšteje število besed, za katero povečujemo število besed v datoteki. Nato klient zmanjša samo število, za katerega povečujemo število besed v datoteki. V našem primeru smo omenili, da je to število enako milijonu besed, in sicer to število zmanjša za faktor deset. Tako bo število besed, s katerim povečujemo datoteke, enako stotisočim besedam. Sedaj

lahko pošiljanje nadaljujemo, tako da štirinajstim milijonom besed prištejemo novo število povečevanja in tako dobimo datoteko s štirinajstimi milijoni besed. Meritve lahko zaključimo, ko bo število povečevanja besed v datoteki enako nič, saj imamo takrat že do besede natančno določeno točko odpovedi.



Slika 4.10: Graf odvisnosti časov T_1 in T_2 od števila besed.

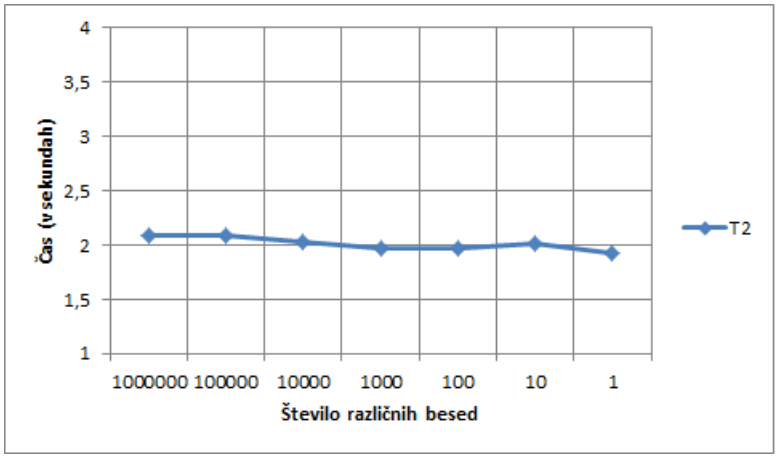
Vse vrednosti na sliki 4.10 so izračunane kot povprečje dvajsetih meritev. Meritve se vršijo na klientu, pozicioniranem na oblačni storitvi Cloud9. Meritve so bile izvedene 26. 4. 2018 med 6:27:00 (+0) in 19:13:00 (+0), 27. 4. 2018 med 5:23:00 (+0) in 21:40:00 (+0), 28.4.2018 med 7:10:00 in 23:07:00 ter 29.4.2018 med 9:50:00 (+0) in 17:04:00 (+0).

Slika 4.10 prikazuje gibanje časov T_1 in T_2 med štirinajstimi in petnajstimi milijoni besed, kar je za našo analizo najbolj zanimivo področje, saj se v tem območju nahaja točka odpovedi. S pomočjo daljšega pošiljanja smo uspeli to točko natančno definirati in sicer aplikacija odpove pri natanko štirinajst milijonov osemstopetdeset tisoč sedemstodevetinpetdeset besedah. Zanimivo je tudi gibanje časov v območju od štirinajst milijonov osemsto tisoč besed dalje. Tu lahko opazimo zelo velik porast časa T_2. To se odraža tudi na porastu časa T_1.

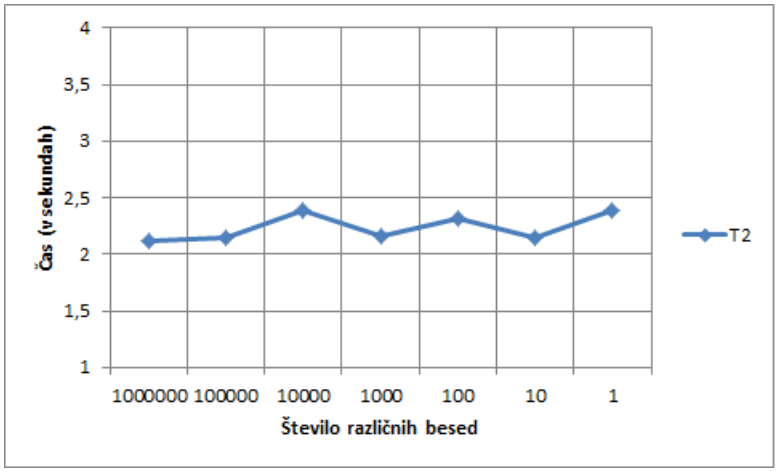
4.6.2 Odvisnost med številom unikatnih besed v datoteki in merjenimi časi

Zanima nas, kako se oblačna storitev odziva na datoteke, ki sicer vsebujejo enako število besed, vendar različno število duplikatov, zato imamo pripravljena dva tipa tekstovnih generatorjev, ki sta si sicer zelo podobna, razlikujeta pa se samo

v tem, da prvi generator zapisuje vse enake besede v datoteki skupaj eno za drugo; tako dobimo dolge verige enakih besed dolžin med eno in milijon besedami, medtem ko drugi generator zapisuje v verige tako rekoč unikatne besede, kjer se vsaka pojavi le enkrat; tako dobimo dolgo zaporedje verig, ki vsebujejo med eno in milijon unikatnih besed. Končne datoteke, ki jih generiramo, vse vsebujejo milijon besed, imamo pa za vsak generator posebej šest različnih datotek, ki vsebujejo med eno, deset, tisoč in tako dalje unikatnih besed.



Slika 4.11: Graf odvisnosti časa od števila unikatnih besed, za prvi generator.



Slika 4.12: Graf odvisnosti časa od števila unikatnih besed, za drugi generator.

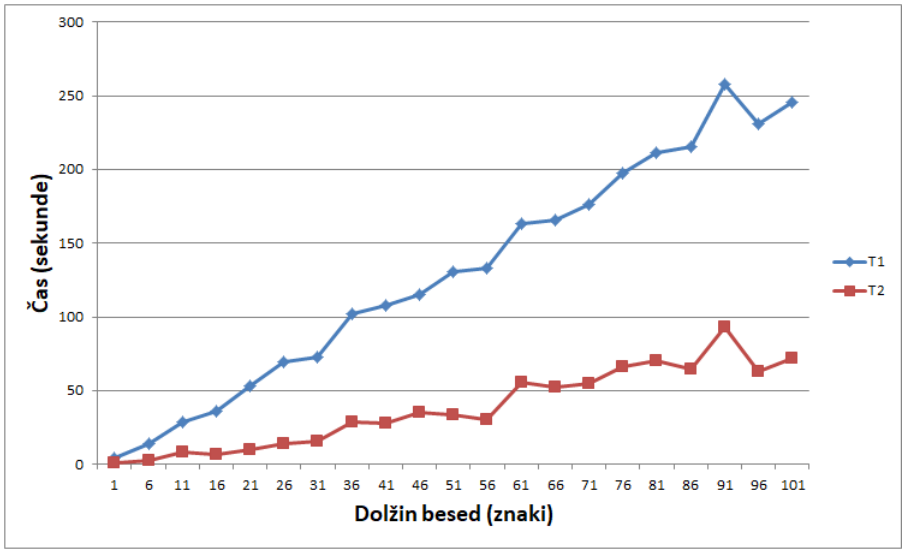
Vse vrednosti na slikah 4.11 in 4.12 so izračunane kot povprečje šestdesetih meritev. Meritve se vršijo na klientu, pozicioniranem na oblačni storitvi Cloud9, za prvi generator 11. 4. 2018 od 16:12:00 (+0), do istega dne 17:38:00 (+0), za

drugi generator pa 13. 4. 2018 od 13:17:00 (+0), do istega dne 14:42:00 (+0).

Iz slik 4.11 in 4.12 sklepamo, da število unikatnih besed v tekstu ne vpliva, oziroma vpliva minimalno na čas analize. Večino manjših sprememb na obeh grafih je možno pripisati zunanjim vplivom. Tako rekoč je edini pravilen sklep, da čas procesiranja ni odvisen od števila unikatnih besed v tekstu.

4.6.3 Odvisnost med dolžino besed v datoteki in merjenimi časi

Zanima nas, kako se oblačna storitev odziva na datoteke, ki sicer vsebujejo enako število besed, vendar je dolžina besed v datotekah različna. Vnaprej smo določili, da pošiljamo datoteke z natanko milijonom besed, spreminjamo pa njihovo dolžino šteto v znakih (beseda "aaaaa" vsebuje pet znakov). Tako začnemo s pošiljanjem datoteke, ki vsebuje besede, dolge natanko en znak, vsaka nadaljno poslana datoteka pa vsebuje besede, ki so za pet znakov daljše kot v prejšnji iteraciji. Torej v drugi iteraciji generiramo datoteko, ki vsebuje besede, dolge šest znakov, za pet znakov daljše od besed v prvi iteraciji. V tretji iteraciji generiramo datoteko, ki vsebuje besede, dolge enajst znakov. S tem postopkom nadaljujemo, dokler aplikacija ne odpove.

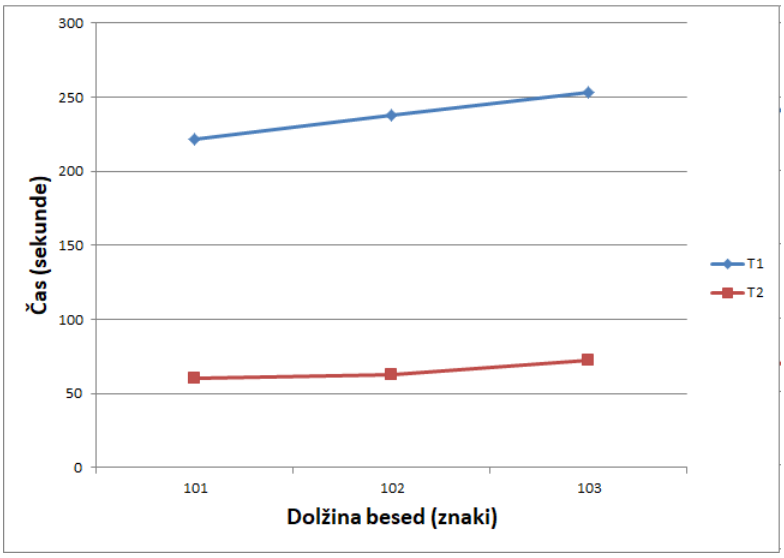


Slika 4.13: Graf odvisnosti časa od dolžine posameznih besed.

Vse vrednosti na sliki 4.13 so izračunane kot povprečje desetih meritev. Meritve se vršijo na klientu, pozicioniranem na oblačni storitvi Cloud9. Meritve so bile izvedene 29. 4. 2018, med 19:31:00 (+0) in 22:08:00 (+0) ter 30. 4. 2018, med 6:26:00 (+0) in 11:40:00 (+0).

Slika 4.13 prikazuje odvisnost med časom pošiljanja T_1 in časom analize T_2 ter dolžino besed v datotekah. Rezultati so občasno dokaj šumni. Kot primer

lahko vzamemo rezultat za dolžino besede enaindevetdeset znakov. Šum lahko pripišemo dokaj majhnemu številu ponovitev meritve za isto dolžino besed, ki je v tem primeru enak desetim ponovitvam. Kljub šumnim podatkom lahko globalno vidimo dokaj lepo linearno rast časov T_1 in T_2. Aplikacija se ne odziva več pri dolžini besed stošest znakov. Naš cilj je podrobneje raziskati, kaj se dogaja v intervalu od stoenega do stošestih znakov.



Slika 4.14: Graf odvisnosti časa od dolžine posameznih besed.

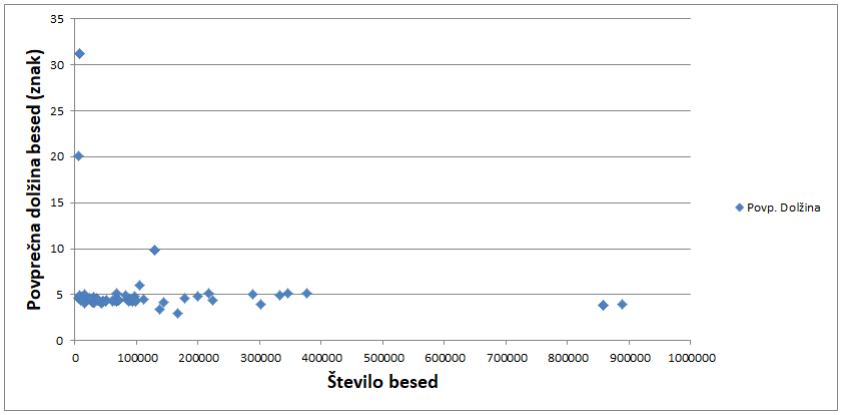
Vse vrednosti na sliki 4.14 so izračunane kot povprečje desetih meritev. Meritve se vršijo na klientu, pozicioniranem na oblačni storitvi Cloud9. Meritve so bile izvedene 30. 4. 2018, od 9:43:00 (+0) do 12:45:00 (+0).

Slika 4.14 prikazuje odvisnost časa pošiljanja T_1 in časa analize T_2 v odvisnosti od dolžine besed v poslani datoteki, natančneje za interval dolžine besed od stoenega do stošestih znakov, pri kateri je pri prejšnjih meritvah aplikacija že odpovedala. Iz dobljenih razultatov ugotovimo, da aplikacija odpove še pred mejo stošestih znakov in sicer pri dolžini stoštirih znakov. Tako smo dobili dolžino besede, pri kateri aplikacija odpove, vendar ker posamezna datoteka vsebuje milijon besed, ta meja ni natančno določena. Za natančno določitev bi bilo potrebno preplesti meritve iz podpoglavja 4.6.1, kjer je govora o vplivu števila besed na čas T_1 in T_2 in v meritve vnesti še različno število besed v poslanih datotekah.

4.6.4 Rezultati za realno breme

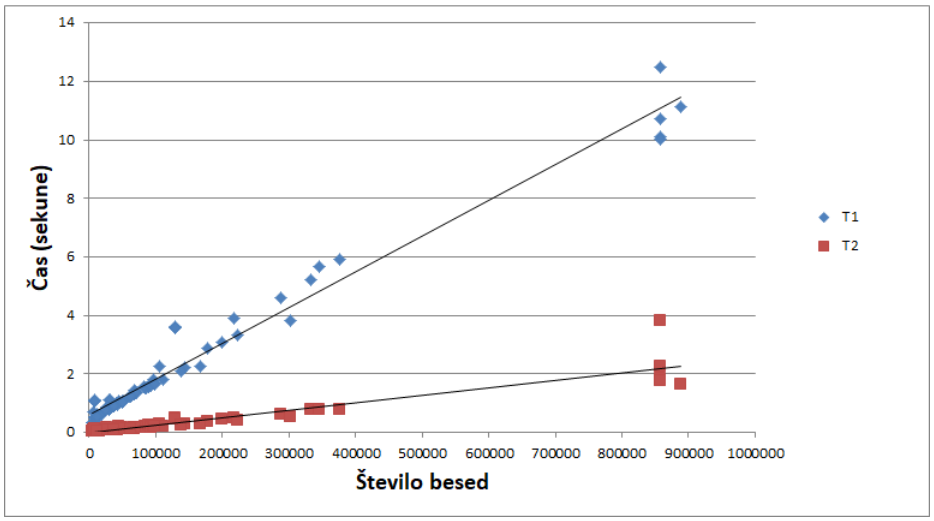
Zanima nas, kako se analizator odziva na realno breme, natančenje na knjige, pridobljene s strani Project Gutenberg. Zapisane so v navadnih tekstovnih datotekah, lahko pa vsebujejo znake, ki niso alfanumerični, zato že na strani klienta

izvedemo manjše predprocesiranje teksta, ki enostavno odstrani vse znake, ki niso alfanumerični. Pred samim pošiljanjem klient prav tako izvede preproste meritve in sicer nas zanima, koliko besed se nahaja v knjigi in kolikšna je povprečna dolžina besed.



Slika 4.15: Graf odvisnosti povprečne dolžine besed od števila besed v knjigi.

Slika 4.15 prikazuje osnovne lastnosti knjig, ki jih pošiljamo na stran oblačne storitve. Vseh knjig je devetdeset. Na žalost je težko pridobiti enakomerno porazdelitev knjig z vidika števila besed v knjigi, zato je večina knjig v intervalu od ene do dvesto tisoč besed. Opaziti je možno enakomernost povprečne dolžine besed v knjigah. Ta se giblje okrog pet znakov na besedo. Opazimo lahko tudi tri knjige, ki od tega odstopajo. Sam razlog za odstopanje nam ni znan. Za celoten vzorec lahko predpostavimo (saj je knjig, ki močno odstopajo, malo), da je povprečna dolžina besed v knjigah kar konstantna vrednost. Tako nas zanima kot lastnost knjig le število besed v posamezni knjigi.



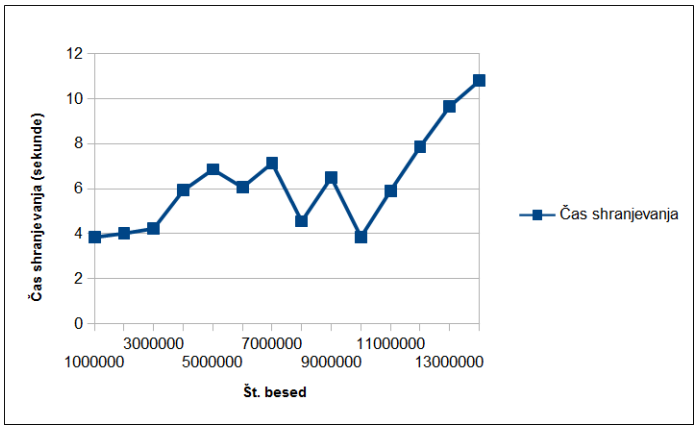
Slika 4.16: Graf odvisnosti časa od števila besed v knjigi.

Vse vrednosti na sliki 4.16 so izračunane kot povprečje desetih meritev. Meritve se vršijo na klientu pozicioniranem na oblačni storitvi Cloud9. Meritve so bile izvedene 9. 5. 2018, od 16:05:00 (+0) do 19:04:00 (+0).

Slika 4.16 prikazuje odvisnost celotnega časa pošiljanja knjig T_1 in časa analize knjig T_2 v odvisnosti od števila besed v knjigah. Seveda sta časa T_1 in T_2 odvisna tudi od povprečne dolžine besede v knjigi. Kot smo že omenili, jo lahko zaradi majhnega odstopanja odmislimo, oziroma označimo za konstanto.

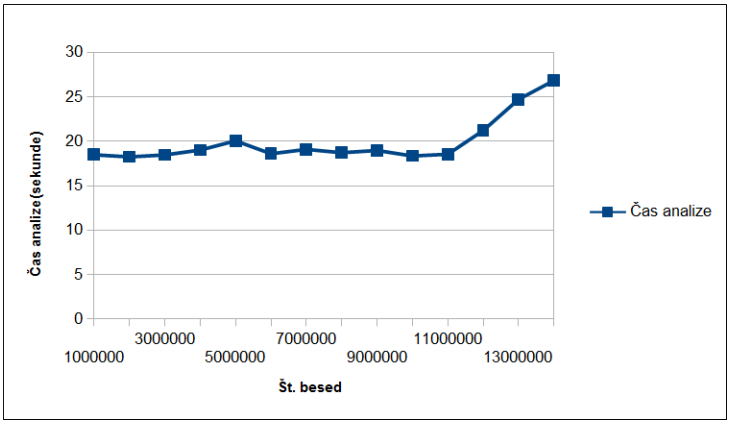
4.6.5 Meritve hitrosti podatkovne baze

Meritve smo naredili tako, da smo še enkrat zagnali analizator, ki po analizi shrani podatke v bazo. Merili smo čas T_2 kot prej, le da smo tokrat zabeležili še čas shranjevanja v bazo. Iz časa T_2 in časa shranjevanja v bazo smo izračunali še čas analize po formuli $Cas\_analize = T_2 - Cas\_shranjevanja$. To smo naredili za število besed od 1 milijona do 15 milijonov. Pri 15 milijonih meritev s strežnika nismo več dobili.



Slika 4.17: Graf časa shranjevanja v bazo v odvisnosti od števila besed.

Slika 4.17 prikazuje graf, kjer smo na strežnik pošiljali besede z dolžinami od 1 milijona do 14 milijonov. Graf pravzaprav prikazuje povprečje desetih poslanih datotek z istim številom besed. Meritve so bile izvedene 19.5.2018, od 20:30:00 (+0) in so se izvajale ponoči.



Slika 4.18: Graf časa analize v odvisnosti od števila besed.

Slika 4.18 prikazuje graf časa analize, ki je potreben za posamezno število besed v poslani datoteki. V resnici graf prikazuje povprečje desetih poslanih datotek z enakim številom besed. Meritve so bile izvedene 19.5.2018, od 20:30:00 (+0) in so se izvajale ponoči. Torej smo oboje merili istočasno.

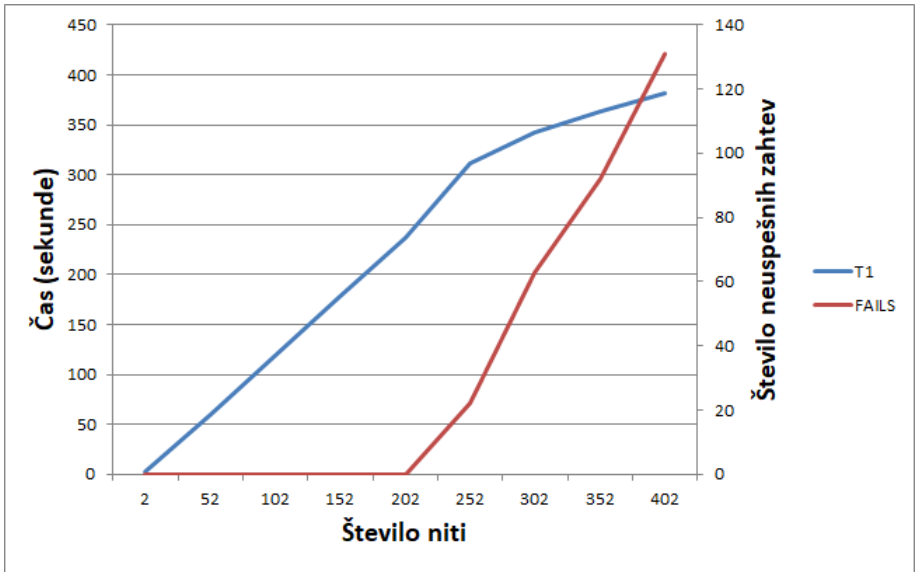
4.7 Rezultati za multipliciranje klientov

V razdelku 4.6 smo se ukvarjali predvsem z vplivom različnih lastnosti bremena, ki ga pošiljamo, na delovanje oblačne storitve. Zdaj pa nas zanima tudi

vpliv lastnosti klientov. Primarno nas tu zanima, kako se aplikacija odziva, ko povečujemo število klientov, ki oblačni storitvi istočasno pošiljajo zahteve.

4.8 Multipliciranje klientov z večnitnim izvajanjem

Zanima nas, kako povečevanje števila klientov vpliva na čas celotne zahteve T_1, ko meritve implementiramo z večnitnim izvajanjem. Tu lahko omenimo, da se časi T_2 ne spreminjajo z večanjem števila klientov. Zato v tem poglavju govorimo predvsem o času T_1, ki ponazarja celoten čas zahteve. Za meritve uporabljamo sintetično breme, ki vsebuje stodvajset tisoč besed, dolžine pet znakov. Te lastnosti so povprečja pridobljena iz vzorca, ki smo ga uporabljali v poglavju 4.6.4. Torej breme, ki ga uporabljamo, predstavlja realno sliko bremena, uporabljenega v realnem svetu. Meritve pričnemo z dvonitnim izvajanjem, na vsakem koraku pa število niti povečamo za petdeset. Vsak korak merjenja traja pet minut. To pomeni, da če od začetka izvajanja koraka še ni minilo pet minut, klienti lahko pošiljajo zahteve. Ob poteku petih minut pa se počaka na odgovor prej poslane zahteve, nato pa se izvajanje zaključi.



Slika 4.19: Graf odvisnosti časa od števila niti.

Vse vrednosti na sliki 4.19 so izračunane kot povprečje več meritev. Meritve se vršijo na klientu, pozicioniranem na oblačni storitvi Cloud9. Meritve so bile izvedene 17. 5. 2018, od 12:35:00 (+0) do 13:46:00 (+0).

Slika 4.19 prikazuje odvisnost med časom celotne zahteve T_1 in številom ne-

uspešnih zahtev $FAILS$ ter številom niti oziroma klientov, ki obremenjujejo oblačno storitev. Opazimo lahko čisto linearno naraščanje časa T_1 z večanjem števila niti vse do dvestodveh niti. Tu se začne višati število neuspešnih zahtev, ki posredno zmanjšajo čas T_1. Tako lahko vidimo hkratno večanje števila neuspešnih zahtev in vse počasnejšo rast časa zahteve T_1. Pri štiristodveh nitih je število neuspešnih zahtev že kar stoenaintrideset.

4.9 Zaključek

V tem poglavju smo izvedli analizo oblačne storitve PythonAnywhere, ki ponuja gostovanje Pythonovskih aplikacij v oblaku. Storitev teče v ozadju na Amazon EC2 brezplačno. Za analizo smo si izbrali aplikacijo analiziranja besed, ki prešteje frekvence na oblak poslanih besedil. Za breme smo si izbrali knjige iz spletne strani Project Gutenberg, za poglobitev analize pa smo uporabili tudi sintetična bremena.

Glavne meritve, ki smo jih opravljali, so bili časi, ki so potrebni za določeno opravilo. Merili smo čas, ki ga potrebuje analizator na strežniku za analizo besedil, ter čas od pošiljanja zahtevka do prejemanja odgovora. S pomočjo teh dveh časov smo izračunali še čas, ki je potreben, da zahteva in odgovor prepotujeta skozi omrežje. Poleg analiziranja besedil smo rezultate in besedila tudi shranili na strežniku. Izmerili smo tudi čas, potreben za shranjevanje in poizvedbo rezultatov. Zaradi praktičnosti smo bremena pošiljali iz storitve Cloud9, ki teče na oblačni storitvi AWS. Preverili smo omrežje s pošiljanjem paketov na Cloud9 in potem še iz Cloud9 na PythonAnywhere. Nato pa smo analizirali še analizator z različnih vidikov. Najprej smo preverili, kako se storitev obnaša, če povečujemo število poslanih besed. Sprva smo testirali velikost besedil do 11 milijonov besed, kar ni obremenilo storitve. Pozneje smo točko odpovedi natančneje določili na med štirinajstimi in petnajstimi milijoni besed. Nato smo testirali točke odpovedi od števila unikatnih besed. Zaključili smo, da število unikatnih besed ne vpliva pomembno na velikost merjenih časov. Ugotovili smo, da se časi povečujejo linearno, kar je posebej zanimivo. Točke odpovedi nismo natančno določili, ker bi bilo potrebno rezultate primerjati s prvim aspektom analize.

Nato smo analizirali še, kako velike besede aplikacija zmore analizirati. Dobivali smo precej šumne rezultate, ampak vseeno nam je na koncu uspelo zelo natančno določiti točko odpovedi na 104 znake pri milijonu besed v datoteki. Potem smo analizator preizkusili še na realnih podatkih s strani Project Gutenberg. Besedila smo najprej predprocesirali, da smo odstranili znake, ki niso alfanumerični, in pogledali, kako je s povprečno dolžino besed v primerjavi z dolžino besedil. Zaradi zelo podobnih povprečnih dolžin besed je odvisnost praktično konstantna. Potem smo naredili še analizo teh besedil in izrisali graf. Preveriti smo hoteli tudi, koliko časa aplikacija porabi za shranjevenje rezultatov. Zato smo izmerili še čas shranjevanja v bazo in narisali graf. Temu smo poleg narisali še graf s časom analize besed, kar da podrobnejši pogled na notranje delovanje naše aplikacije.

Na koncu smo želeli videti še, kako se aplikacija odziva na veliko število poslanih zahtev. Breme smo nastavili tako, da se pet minut pošiljajo zahteve z povprečnim besedilom. Izrisali smo graf, ki prikazuje število uspešnih in neuspešnih odgovorov in čas procesiranja v odvisnosti od števila niti. Ugotovili smo, da začnemo dobivati neuspešne odgovore pri 202 nitih. Potem pa se z naraščanjem števila neuspešnih odgovorov upočasni tudi rast časa uspešnih odgovorov.

Naša ugotovitev je, da je aplikacija (analizator) na oblačni storitvi precej zmogljiva v smislu enostavnega analiziranja besedil in shranjevanja. Analizirali smo več različnih možnih aspektov aplikacije in dobili smiselne rezultate.

Literatura

[1] "Dropbox." https://www.dropbox.com/.

[2] "Mega." https://mega.nz/.

[3] "Google drive." https://drive.google.com/.

[4] X. Wang, "Benchmarking cloud storage systems," Master's thesis, Norwegian University of Science and Technology, Department of Telematics, 2014.

[5] E. Bocchi, I. Drago, and M. Mellia, "Personal cloud storage benchmarks and comparison," *IEEE Transactions on Cloud Computing*, vol. 5, no. 4, pp. 751–764, 2015.

[6] "Benchcloud." https://github.com/zenja/benchmarking-cloud-storage-systems.

[7] "Wireshark." https://www.wireshark.org/.

[8] "Ipdata." https://ipdata.co/.

[9] "Easy live auction." https://www.easyliveauction.com/, Maj 2018.

[10] "Amazon web services." https://aws.amazon.com, Maj 2018.

[11] "node.js." https://nodejs.org/en/, Maj 2018.

[12] "Javascript." https://www.javascript.com/, Maj 2018.

[13] "Mysql." https://www.mysql.com/, Maj 2018.

[14] "Representational state transfer." https://en.wikipedia.org/wiki/Representational_state_transfer, Maj 2018.

[15] "Amazon api gateway." https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html, Maj 2018.

[16] "Amazon lambda." https://docs.aws.amazon.com/lambda/latest/dg/welcome.html, Maj 2018.

[17] "Amazaon rds." https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html, Maj 2018.

[18] "Ping." https://linux.die.net/man/8/ping, Maj 2018.

[19] "Traceroute." https://linux.die.net/man/8/traceroute, Maj 2018.

[20] "Apache http server benchmarking tool." https://httpd.apache.org/docs/2.4/programs/ab.html, Maj 2018.

[21] Wikipedia contributors, "Leibniz formula for pi — Wikipedia, the free encyclopedia," 2018. [Online; dostopano 2-April-2018].

[22] Wikipedia contributors, "Bellard's formula — Wikipedia, the free encyclopedia," 2018. [Online; dostopano 2-April-2018].

[23] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "A performance analysis of ec2 cloud computing services for scientific computing," in *International Conference on Cloud Computing*, pp. 115–131, Springer, 2009.

[24] "Amazon web services." https://aws.amazon.com/. [Online; dostopano 2-April-2018].

[25] "Microsoft azure." https://azure.microsoft.com/en-gb/. [Online; dostopano 2-April-2018].

[26] "Aws cloud9." https://aws.amazon.com/cloud9/?origin=c9io. [Online; dostopano 2-April-2018].

[27] M. Moškon and M. Mraz, *Modeliranje računalniških omrežij.* Založba FE in FRI, 2012.

[28] "Amazon elastic compute cloud." https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/t2-credits-baseline-concepts.html. [Online; dostopano 30-Maj-2018].

[29] A.-H. Tan *et al.*, "Text mining: The state of the art and the challenges," in *Proceedings of the PAKDD 1999 Workshop on Knowledge Disocovery from Advanced Databases*, vol. 8, pp. 65–70, sn, 1999.

[30] W. B. Paley, "Textarc: Showing word frequency and distribution in text," in *Poster presented at IEEE Symposium on Information Visualization*, vol. 2002, 2002.

[31] S. Stemler, "An overview of content analysis," *Practical assessment, research & evaluation*, vol. 7, no. 17, pp. 137–146, 2001.

[32] A. D. Booth, "A law of occurrences for words of low frequency," *Information and control*, vol. 10, no. 4, pp. 386–393, 1967.

[33] "Pythonanywhere." https://www.pythonanywhere.com/, Maj 2018.

[34] "What are cpu seconds?." https://help.pythonanywhere.com/pages/WhatAreCPUSeconds, Maj 2018.

[35] "What is a tarpit and why are my processes in it?." https://www.pythonanywhere.com/tarpit/, Maj 2018.

[36] "Free ebooks - project gutenberg." http://www.gutenberg.org/, Maj 2018.

[37] "Aws cloud9." https://aws.amazon.com/cloud9/?origin=c9io, Maj 2018.

[38] "Ip geolocation and threat data api." https://ipdata.co/, Maj 2018.