

Uporaba benchmark orodij za določanje zmogljivosti oblačnih računalniških sistemov

Žiga Šebenik, Tomaž Mrežar

2020

Contents

Predgovor	iii
1 Benchmark orodja	1
1.1 Opis problema	1
1.2 Benchmark orodja	2
1.3 Brezplačna orodja	2
1.3.1 Ročno benchmark testiranje	2
1.3.2 PerfKit Benchmarker	3
1.4 Plačljiva orodja	3
1.4.1 SPEC Cloud® IaaS 2018	3
1.4.2 Cloud Spectator	4
1.4.3 Cloud Performance Benchmark	4
1.4.4 Technology Business Research, Inc.	4
1.5 Implementacija merilnega okolja	5
1.5.1 Tehnične specifikacije računalnika	5
1.6 Rezultati meritev	5
1.6.1 Geekbench 3	6
1.6.2 iPerf	7
1.6.3 ioPing	8
1.6.4 Fio	9
1.6.5 Lastnosti metrik testov	9
1.7 Zmogljivost omrežnih povezav	10
1.7.1 RTT	10
1.7.2 Ping	10
1.7.3 TraceRoute	10
1.7.4 Implementacija	10
1.7.5 Umetno breme	11
1.8 Zmogljivost diskovnega sistema	13
1.9 Zmogljivost procesorja	14
1.9.1 Opis programa	14
1.9.2 Algoritem za faktorizacijo	14
1.9.3 Algoritem za iskanje praštevil	15
1.9.4 Rezultati meritev	16
1.10 Zaključek	17

1.11 Viri	17
---------------------	----

Predgovor

Delo je razdeljeno v deset poglavij, ki predstavljajo uporabo benchmark orodij za določanje zmogljivosti oblčnih računalniških sistemov. Avtorji posameznih poglavij so poslušalci predmeta *Zanesljivost in zmogljivost računalniških sistemov*.

Chapter 1

Benchmark orodja

Žiga Šebenik, Tomaž Mrežar

1.1 Opis problema

Za določanje zmogljivosti računalniških sistemov je na voljo veliko orodij, tako odprtokodnih, zastojnih kot tudi plačljivih. V tem poglavju bova predstavila orodja in postopke, s katerimi sva midva testirala virtualni računalnik na Microsoftovem oblaku Azure, kjer sva imela odprt brezplačni račun. Na njem sva pognala različna bremena, kjer sva testirala omrežne povezave, diskovje in procesor. Nekaj od teh bremen sva ustvarila s pomočje zastojnih programov: Geekbench 3 za test procesorja, iPerf za test omrežja, ioPing za latenco diskovja in fio za test pasovne širine diskovja. Vsako orodje sva opisala in naštela prepoznavne značilnosti in prednosti pred ostalimi orodji. Ostala bremena sva ustvarila sama. Napisala sva namreč dva krata programa, enega za test omrežja in diska hkrati, ter enega za test moči procesorja. Ta dva programa sva pognala vsak na svojem računalniku, prav tako pa sva jih pognala tudi na virtualnem računalniku, da sva lahko primerjala zmogljivosti. Najina računalnika se razlikujeta v moči, saj je en računalnik prenosnik, drugi pa osebni računalnik, razlika pa je tudi v lokaciji, saj sva teste poganjala iz Kranja in Brezovice pri Ljubljani. Imava tudi različna ponudnika interneta, kar vse pride v poštev pri testih. Iz rezultatov testiranja sva nato poskušala sklepati o razlogih za razlike, ki so se pri teh testih pojavile.

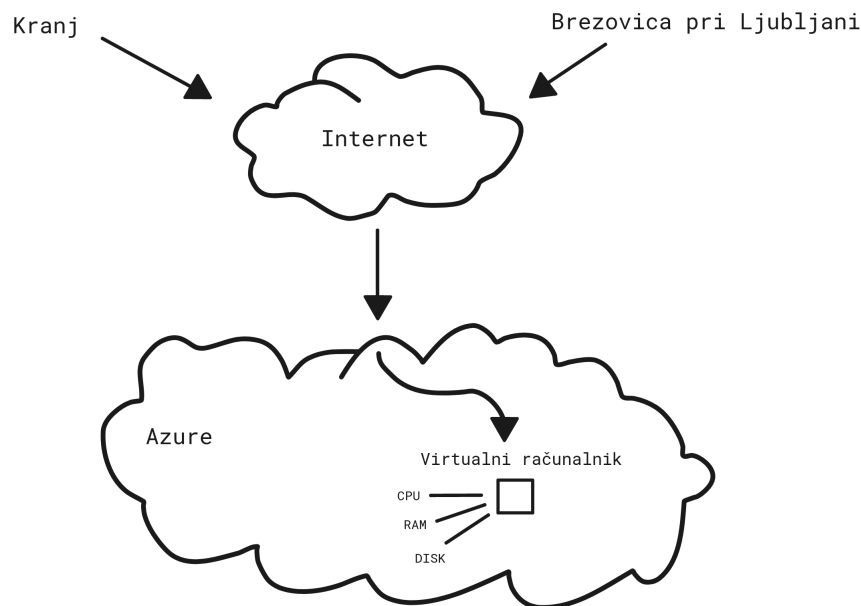


Figure 1.1: Celotni prikaz testiranja na Azure platformi.

1.2 Benchmark orodja

Obstaja več vrst benchmark orodij. Testi, ki jih benchmark orodja izvajajo se lahko razlikujejo med seboj vendar so osnovne funkcionalnosti testov skoraj povsod enake. Skorajda vsako orodje testira dosegljivost sistemov, upočasnitve delovanja, latenco sistema in prepustnost sistema. Orodja bova delila na dve skupini, prva skupina bodo orodja, ki so brezplačna, medtem ko bodo v drugo skupino spadala orodja, ki so plačljiva, brezplačna za določen čas ali pa imajo v brezplačni verziji omejene funkcionalnosti.

1.3 Brezplačna orodja

Brezplačni orodji sta sledeči:

- ročno benchmark testiranje;
- PerfKit Benchmarker;

1.3.1 Ročno benchmark testiranje

Zelo preprosta izbira, ki nam je na voljo, je da preprosto sami testiramo zmogljivost oblačnih storitev s pomočjo več različnih orodij, pri čemer je vsako namenjeno specifičnemu delu sistema. Na voljo nam je veliko zastojnih orodij, precej

jih je tudi odprtokodnih, znani med njimi pa so ping, Geekbench, fio, iPerf... Testiramo lahko zmogljivost posameznega strežnika, gruče ali pa celotnega oblaka. Njihova prednost je, da so preprosta in fleksibilna, vendar pa moramo več dela opraviti sami.

1.3.2 PerfKit Benchmarker

PerfKit Benchmarker je odprtokodno orodje uporabljamo za meritve in primerjave oblačnih performans. Podpira več večjih oblačnih ponudnikov, kot sta Google Cloud Platform in Amazon Web Services, pa tudi mnoge druge. PerfKit Benchmarker meri končni čas za zagotavljanje virov v oblaku in tudi vse osnovne oblačne meritve našete v predhodnem razdelku. PerfKit Benchmarker zmanjšuje kompleksnost v zaganjanju testov na oblačnih ponudnikih z enotnimi in preprostimi ukazi. Vsebuje tudi množice javnih testov za uporabo. Vsi testi se zaženejo z privzeto konfiguracijo, ki ni nastavljena v prid nobenemu ponudniku oblačnih storitev. To ponuja možnost testiranja na več različnih oblačnih platformah. V bistvu je Perfkit Benchmarker le orodje, ki avtomatizira zagon ostalih, ozko nameskih orodij za test posameznih metrik platforme. Vsa orodja, ki jih uporablja so odprtokodna in bi jih lahko pognali sami, vendar nam Perfkit Benchmarker, zmanjša količino dela, saj ima prilagojene skripte za vse večje ponudnike.

1.4 Plačljiva orodja

Plačljiva orodja so sledeča:

- SPEC Cloud® IaaS 2018;
- Cloud Spectator;
- Cloud Performance Benchmark;
- Technology Business Research, Inc.;

1.4.1 SPEC Cloud® IaaS 2018

SPEC Cloud® IaaS 2018 testira delovanje infrastrukture kot storitev oblačnih implementacij. Podpira testiranje javnih in zasebnih oblakov. Orodje deluje nad strežbo storitve, kot tudi nad izvajanjem storitve oblaka z uporabo vhodno izhodnih in CPE intenzivnih del. Vsako delo se zažene kot distribuirana aplikacija narejena iz 6 ali 7 instanc, ki obremenijo oblakove resource (CPE, diski in omrežje). Delo bo teklo dokler testi ne naredijo več kakovosti storitve. Administrator lahko tudi omeji število aplikacij kreiranih med izvedbo. Orodje nam omogoča obremeniti računsko zmogljivost, shrambo in omrežje oblaka. Pri tem pa ne potrebuje hypervizorja ali virtualizacijske plasti in uporablja delovne obremenitve, ki spominjajo na tiste, ki običajno delujejo v oblaku, kot so aplikacije za socialne medije in velika analiza podatkov. SPEC Cloud izdaja poročila, ki ne grejo tako v detajle posameznega dela platforme ter izdajajo ločene metrike za vsak del

platforme, ampak testirajo zmogljivost ponudnikove platforme kot celote. Merijo čas stvaritve, konfiguracije in zagona instanc oz. virtualnih strojev, latenco vstavitve oz. branja iz baze na postavljeni virtualki, prepustnost, skalabilnost. Vse metrike so merjene v sekundah oz. operacijah na sekundo.

1.4.2 Cloud Spectator

Cloud Spectator sicer ni orodje, ampak podjetje, ki ponuja benchmarking in konzultacijo glede oblačnih storitev. Podjetjem pomaga z analizo različnih ponudnikov oblačnih storitev in testira zmogljivost njihove infrastrukture ter svetuje pri ekonomskih odločitvah. Namenjen je tako primerjavi ponudnikov oblačnih storitev, kot tudi ponudnikom samim, da lahko analizirajo zmogljivost svoje infrastrukture. Nudili naj bi sposobnost izbire pravega ponudnika, kjer stranka postavi zahteve svoje aplikacije, Cloud spectator pa s kombinacijo zahtevosti strankine aplikacije, zmogljivosti infrastrukture različnih ponudnikov in njihovih cenikov, izbere pravega ponudnika. Poročilo vrača rezultate v obliki VM Performance Score in CloudSpecs Score. Nobeden od njiju nima posebne merske enote, saj je VM Performance Score le povprečje točk, ki jih vrne Geekbench 4 in fio, tako da imata oba enak prispevek k točkam. CloudSpecs Score se izračuna kot VM Performance Score, ki je utežen s ceno, tako da dobimo zmogljivost na ceno, ki naj bi strankam omogočala lažjo izbiro pravega ponudnika platforme.

1.4.3 Cloud Performance Benchmark

Cloud Performance Benchmark je poročilo o največjih petih ponudnikih Amazon Web Services, Google Cloud Platform, Microsoft Azure, IBM Cloud in Alibaba Cloud. Zagotavljalo naj bi nepristransko strokovno poročilo, ki je podprto z raznimi metrikami. Poročilo primerja infrastrukturo posameznega ponudnika in pokaže kako ta infrastruktura njihovega vpliva na zmogljivost ter jih seveda primerja med seboj. Prav tako se dotakne geografskih razlik in njihov vpliv na zmogljivost. Poročilo ne vsebuje le kvantitativnih podatkov o posameznih platformah, temveč tudi veliko več kvantitativnih in statističnih podatkov o predvidljivosti posameznih metrik poleg razlag arhitektur vsake platforme. Poročilo je zelo poglobljeno, kvantitativne metrike pa se tičejo predvsem omrežja, saj je velik poudarek na latenci, tako izven kot znotraj platforme, ter izgubi paketov. Veliko podatkov najdemo tudi o vplivu geografskih pozicij na kakovost omrežja vsake platforme, vse podprto z statistično analizo obeh metrik.

1.4.4 Technology Business Research, Inc.

Technology Business Research, Inc. je podjetje, ki prav tako nudi storitve tako ponudnikom oblačnih storitev, kot tudi njihovim strankam. Ponudnikom nudijo podatke o trgu, finančne podatke o ponudnikih programske opreme, napovedi, strategije prodaje, itd. Ponudnikom oblačnih storitev pa nudijo podatke o ponudnikih le teh storitev, zmogljivosti ponudnikove infrastrukture za strankin primer uporabe ter tudi prihajajoče trende, ki se bodo posluževali in vplivali

na zmogljivost oblačnih sistemov. TBR Inc. je bolj usmerjeno v svetovanje situaciji na trgu, kot pa poglobljeni analizi metrik in zmogljivosti. Sledijo trendom in priložnostim na trgu, napovedujejo nove trende in sledijo finančnim podatkom ponudnikov platform. Njihov glavni cilj je direktni stik in osebno svetovanje strankam, zato nisem uspel najti nobenih uporabnih podatkov o njihovi metodologiji oz. metrikah.

1.5 Implementacija merilnega okolja

Na oblaku Microsoft Azure sva ustvarila račun in na njem postavila virtualni stroj, ki poganja Ubuntu 18.04. Na tej virtualki sva ročno pognala več benchmark testov, saj googlov odprtokodni Perfkit Benchmarker ni deloval. Le ta se namreč zanaša na avtomatsko ustvarjanje virtualnih strojev, zastonjski račun na Azure pa to omejuje. Pognala sva odprtokodna orodja:

- Geekbench 3 = uporabljen predvsem za test CPE zmogljivosti;
- test pasovne širine omrežja in dostop do interneta;
- latenca diskovja;
- zmogljivost diskovja;

1.5.1 Tehnične specifikacije računalnika

Specifikacije računalnika na katerem teče virtualni stroj so sledeče:

- OS: Ubuntu 18.04.4 LTS 5.0.0-1032-azure x86_64
- CPE: Intel Xeon Platinum 8168 @ 2.69 GHz 1 processor, 2 threads
- RAM: 4 GB
- Disk: 32 GB SSD

Ker se računalnik nahaja nekje v Microsoftovem strežniškem centru in ker uporablja zastonjski račun na Azure, nimava na razpolago celotnega računalnika, saj na njem verjetno teče tudi kakšna druga virtualka, kar zna vplivati na rezultate meritev. Omenjene specifikacije računalnika so le te, ki jih imava na voljo na virtualki.

1.6 Rezultati meritev

V naslednjih razdelkih bova predstavila testna orodja uporabljena na virtualnem računalniku in dobljene rezultate.

1.6.1 Geekbench 3

Najprej sva pognala Geekbench 3, benchmark, ki se uporablja za testiranje CPE zmogljivosti. Imel naj bi to prednost pred klasičnimi testi CPE, da simulira tako breme na procesorju, ki dobro ponazarja produkcijsko okolje med izvajanjem realnih programov, in ne samo sintetični breme. Prav tako Geekbench dodobra obremeni računalnik, da lahko vidimo zmogljivost ob velikem stresu. Še ena dobra stvar kar se tiče Geekbench-a je, da je zelo razširjen, kar pomeni da lahko najdemo veliko različnih rezultatov testov za različne konfiguracije računalnikov, vendar pa to ne pomeni da je zanesljiv. Obstajajo namreč primeri, kjer ima strežniški CPE slabšo oceno kot nek mobilni CPE, saj Geekbench ne testira termalnih zmogljivosti, prav tako pa so razlike med rezultati na različnih operacijskih sistemih. Čeprav Geekbench uporabi več različnih testov, iz povzetka vseh teh testov vrne dve glavni številki: Single-core točke in Multi-core točke, ki sami po sebi nič ne pomenita in nimata merskih enot, velja pa višje je, bolje je. Šele ko ju primerjamo z ostalimi sistemi, dobimo neko sliko zmogljivosti.

Na sliki 1.2 so predstavljeni rezultati večih algoritmov za performanse celih števil. Algoritmi se izvajajo na enem jedru procesorja in na večih.

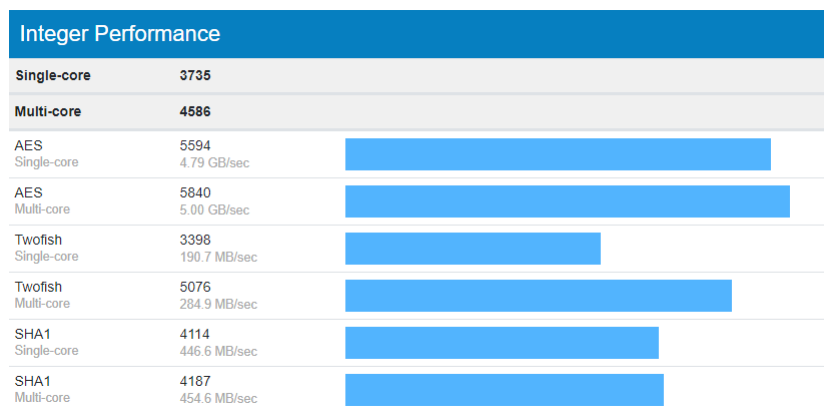


Figure 1.2: Cela števila.

Na sliki 1.3 so predstavljeni rezultati večih algoritmov za performanse števil v plavajoči vejici. Algoritmi se izvajajo na enem jedru procesorja in na večih.

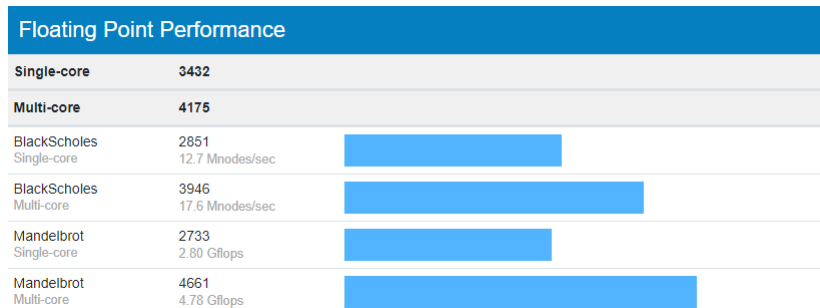


Figure 1.3: Plavajoča vejica.

Na sliki 1.4 so predstavljeni rezultati za testiranje spomina. Testira se z kopiranjem na večih jedrih in na enem jedru procesorja.

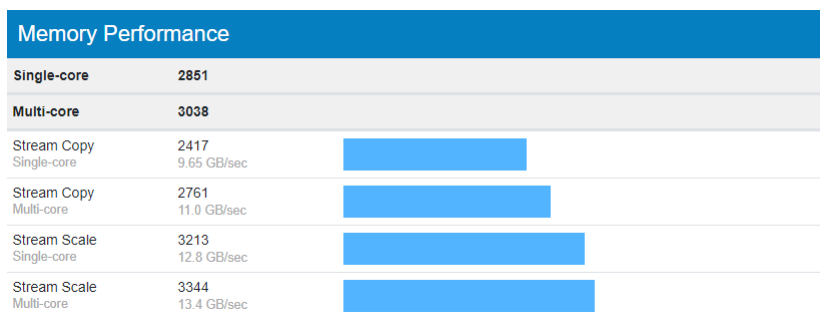


Figure 1.4: Performanse spomina.

1.6.2 iPerf

Naslednji benchmark, ki sva ga pognala je iperf, ki meri podatke o omrežju kot sta upload in download pasovno širino. Meri jih v bajt/sekunda, kjer prilagodi prefix od bajta glede na hitrost omrežja. Test sva pognala nad omrežjem med nama in virtualnim strojem. Čeprav bi lahko v našem primeru podatke popačila hitrost ponudnika interneta na naši strani, pa temu verjetno ni tako, saj sva testirala na omrežju z 12 MB/s prenosa. Nisva prepričana, ali isto velja tudi za upload hitrost, latence pa nisva testirala, saj se strežnik nahaja na nizozemskem in latenca zaradi geografske lokacije pač je kakršna je. Iperf nama vrne povprečno hitrost prenosa podatkov 2,71 MB/s in 2,53 hitrost uploada podatkov, kjer velja, da večje številke pomenijo boljše performanse. Na sliki 1.5 so predstavljeni rezultati testa iPerf, kjer lahko vidimo hitrosti prenosa na intervalih.

```

ziga@ziga-desktop:~$ iperf3 -c 13.81.58.48 -f M
Connecting to host 13.81.58.48, port 5201
[ 5] local 192.168.0.101 port 50250 connected to 13.81.58.48 port 5201
[ ID] Interval      Transfer    Bitrate      Retr    Cwnd
[ 5]  0.00-1.00    sec  4.35 MBytes  4.35 MBytes/sec  462    65.5 KBytes
[ 5]  1.00-2.00    sec  2.94 MBytes  2.94 MBytes/sec   9    62.8 KBytes
[ 5]  2.00-3.00    sec  2.27 MBytes  2.27 MBytes/sec   8    55.8 KBytes
[ 5]  3.00-4.00    sec  2.33 MBytes  2.33 MBytes/sec   4    82.3 KBytes
[ 5]  4.00-5.00    sec  2.27 MBytes  2.27 MBytes/sec  11    78.1 KBytes
[ 5]  5.00-6.00    sec  3.19 MBytes  3.19 MBytes/sec  11    73.9 KBytes
[ 5]  6.00-7.00    sec  2.27 MBytes  2.27 MBytes/sec  10    69.7 KBytes
[ 5]  7.00-8.00    sec  2.21 MBytes  2.21 MBytes/sec  12    65.5 KBytes
[ 5]  8.00-9.00    sec  2.94 MBytes  2.94 MBytes/sec  12    62.8 KBytes
[ 5]  9.00-10.00   sec  2.33 MBytes  2.33 MBytes/sec  10    61.4 KBytes
-----
[ ID] Interval      Transfer    Bitrate      Retr
[ 5]  0.00-10.00   sec  27.1 MBytes  2.71 MBytes/sec  549
[ 5]  0.00-10.00   sec  25.3 MBytes  2.53 MBytes/sec
sender
receiver
iperf Done.

```

Figure 1.5: test iPerf.

1.6.3 ioPing

Tretji benchmark, ki sva ga izbrala je ioPing. Prednost ioPing-a naj bi bila njegova preprostost, saj je zelo podoben znanemu ukazu ping, le da je namenjen testu diskovja v računalniku. Ni namenjen testiranju diskovja pod naporom, zanima ga le latenca zahtev za pisanje/branje. Le ta je izredno pomembna za podatkovne baze, ki niso pod velikim stresom zaradi števila zahtev, vendar bi njihov opazen zamik pri odzivu negativno vplival na uporabniško izkušnjo. Latenca nima velikega pomena pri večih asinhronih operacijah pisanja, saj je to veliko bolj odvisno od prepustnosti diska. Zato sva pognala testa latence pri sinhronih oz. zaporednih operacijah pisanja in latenci pri asinhronih operacijah branja. ioPing poda rezultate latence v obliki sekund, seveda pa prilagodi predpono velikosti latence. SSD, do katerega dostopa virtualka, je precej hiter in ima latenco v rangi 200 mikrosekund, kjer sta pomembni še minimum in maksimum vrednosti latence. Standardni odklon (mdev) nam pove kakšen razpon latenc lahko pričakujemo pri večini operacij. Na sliki 1.6 so predstavljeni rezultati testa ioPing, kjer lahko vidimo odzivne čase.

```

ziga@test:~$ ioping -c 10 .
4 KiB from . (ext4 /dev/sda1): request=1 time=405 us
4 KiB from . (ext4 /dev/sda1): request=2 time=112 us
4 KiB from . (ext4 /dev/sda1): request=3 time=110 us
4 KiB from . (ext4 /dev/sda1): request=4 time=224 us
4 KiB from . (ext4 /dev/sda1): request=5 time=187 us
4 KiB from . (ext4 /dev/sda1): request=6 time=124 us
4 KiB from . (ext4 /dev/sda1): request=7 time=136 us
4 KiB from . (ext4 /dev/sda1): request=8 time=188 us
4 KiB from . (ext4 /dev/sda1): request=9 time=124 us
4 KiB from . (ext4 /dev/sda1): request=10 time=240 us

--- . (ext4 /dev/sda1) ioping statistics ---
10 requests completed in 9.00 s, 5.41 k iops, 21.1 MiB/s
min/avg/max/mdev = 110 us / 185 us / 405 us / 85 us

```

Figure 1.6: test ioPing.

1.6.4 Fio

Četrty benchmark, ki sva ga pognala je program fio, kar pomeni "flexible I/O". Fio je precej fleksibilen in torej bolj kompleksen program za testiranje diskovja, omogoča več specifičnih testov. Uporabljajo ga razvijalci in administratorji, za testiranje delovanje diskovja in datotečnih sistemov. Midva sva ga uporabila za test pasovne širine pisanja/branja, samo pisanja ali samo branja na disk. V povprečju neka standardna podatkovna baza dobi 3 bralne operacije za vsako pisalno operacijo, torej je razmerje med read in write operacijami 3:1. To razmerje sva uporabila pri testu kombinacije pisanja in branja. Na sliki 1.7 so predstavljeni rezultati testa Fio za read in write.

```

randrw: (groupid=0, jobs=1): err= 0: pid=77126: Sat Mar 21 11:25:56 2020
read: IOPS=1790, BW=7162KiB/s (7334kB/s)(3070MiB/438951msec)
bw (  KiB/s): min= 16, max=14032, per=100.00%, avg=8507.62, stdev=4830.79, samples=739
iops      : min=  4, max= 3508, avg=2126.88, stdev=1207.69, samples=739
write: IOPS=598, BW=2393KiB/s (2451kB/s)(1026MiB/438951msec)
bw (  KiB/s): min=  8, max= 5240, per=100.00%, avg=2838.93, stdev=1634.56, samples=740
iops      : min=  2, max= 1310, avg=709.71, stdev=408.64, samples=740
cpu        : usr=0.50%, sys=1.91%, ctx=389703, majf=0, minf=8
IO depths  : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.1%, 32=0.1%, >=64=100.0%
submit     : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
complete   : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.1%, >=64=0.0%
issued rwt: total=785920,262656,0, short=0,0,0, dropped=0,0,0
latency    : target=0, window=0, percentile=100.00%, depth=64

Run status group 0 (all jobs):
  READ: bw=7162KiB/s (7334kB/s), 7162KiB/s-7162KiB/s (7334kB/s-7334kB/s), io=3070MiB (3219MB), run=438951-438951msec
  WRITE: bw=2393KiB/s (2451kB/s), 2393KiB/s-2393KiB/s (2451kB/s-2451kB/s), io=1026MiB (1076MB), run=438951-438951msec

Disk stats (read/write):
  sda: ios=787458/263388, merge=0/687, ticks=5953973/22909741, in_queue=28364920, util=16.35%

```

Figure 1.7: test Fio, R/W.

1.6.5 Lastnosti metrik testov

Tabela prikazuje metrike uporabljenih testov. Njihove slabosti in prednosti pred drugimi.

Metrike	Geekbench 3	iPerf	ioPing	Fio
linearnost	NE	DA	NE	NE
zanesljivost	NE	DA	NE	DA
ponovljivost	DA	NE	DA	NE
enostavnost	NE	DA	DA	DA
konsistentnost	DA	DA	NE	DA
neodvisnost	NE	DA	DA	DA

Table 1.1: Metrike testov.

1.7 Zmogljivost omrežnih povezav

Merjenje zmogljivosti povezav do virtualnega računalnika sva izmerila z uporabo orodja PING in Traceroute.

1.7.1 RTT

Pri naslednjih razdelki se bo uporabljala beseda RTT, katero bom definiral tukaj. RTT (angl. Round Trip Time) ali po slovensko čas povratnega potovanja. Z besedo opišemo čas, ki je potreben za pošiljanje in prihod paketa do destinacije + čas potovanja potrditve do izvora.

1.7.2 Ping

Ping je administracijsko programsko orodje, katerega se uporablja za testiranje dosegljivosti in latence nekega omrežja preko IP protokola. Ping meri RTT sporočil poslanih od izvora do destinacije. Deluje na protokolu ICMP, kateri pošlje zahtevo in počaka na odziv.

1.7.3 TraceRoute

TraceRoute je orodje, ki se uporablja za prikaz poti od izvora do destinacije preko protokola IP. Zgodovina skokov paketa je shranjena kot RTT paketov prejetih od vsakega naslednjega vozlišča. Vsota povprečnih časov vsakega skoka je meritev celotnega časa potrebnega za vzpostavitev povezave.

1.7.4 Implementacija

Na virtualnem računalniku je bilo potrebno nastaviti pravilo, da se računalnik odziva na ICMP pakete katere prejme. Neodzivanje na ICMP pakete se lepo vidi iz slike 1.8, kjer * predstavljajo vse usmerjevalnike na poti kateri se niso odzvali. Na sliki 1.8 se vidi testiranje iz lokacije Kranja, medtem ko na sliki 1.9 lahko vidimo testiranje iz lokacije Brezovica pri Ljubljani. Virtualni računalnik se nahaja na Nizozemskem zato je zadnjih nekaj skokov skoraj enakih, razlikujejo se le v številkah serverja, skozi katerega so paketi potovali. Na lokaciji Brezovica pri Ljubljani je prihodna/odhodna hitrost 100/10, na lokaciji Kranj pa 50/5.


```

$ sudo traceroute -I 13.81.58.48
traceroute to 13.81.58.48 (13.81.58.48), 30 hops max, 60 byte packets
 1  DD-WRT (192.168.1.1) 1.298 ms 2.192 ms 2.274 ms
 2  89-212-0-1.gw.t-2.net (89.212.0.1) 5.049 ms 5.088 ms 5.429 ms
 3  89-212-88-69.dynamic.dsl.t-2.net (89.212.88.69) 4.634 ms 4.750 ms 4.753 ms
 4  84-255-250-5.core.t-2.net (84.255.250.5) 5.467 ms 5.815 ms 5.877 ms
 5  t2-d.o.o.zag30-96cbe-la.ntwk.msn.net (207.46.219.166) 9.574 ms 9.589 ms 9.929 ms
 6  ae35-0.vie-96cbe-la.ntwk.msn.net (104.44.233.238) 14.702 ms 11.657 ms 11.418 ms
 7  be-33-0.ibr02.vie.ntwk.msn.net (104.44.20.87) 30.004 ms 29.805 ms 29.829 ms
 8  be-6-0.ibr04.fra30.ntwk.msn.net (104.44.18.250) 30.182 ms 30.574 ms 30.603 ms
 9  be-3-0.ibr02.fra21.ntwk.msn.net (104.44.18.244) 31.577 ms 31.605 ms 31.937 ms
10  be-9-0.ibr02.ams30.ntwk.msn.net (104.44.19.236) 31.939 ms 32.295 ms 32.300 ms
11  be-3-0.ibr04.ams06.ntwk.msn.net (104.44.18.187) 33.717 ms 33.722 ms 33.707 ms
12  ae120-0.icr01.ams06.ntwk.msn.net (104.44.21.188) 32.276 ms 29.152 ms 29.407 ms
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
21  * * *
22  * * *
23  * * *
24  * * *
25  * * *
26  13.81.58.48 (13.81.58.48) 30.707 ms 30.748 ms 31.088 ms

```

Figure 1.8: Prikaz poti iz Kranja do virtualnega računalnika.

```

Tracing route to 13.81.58.48 over a maximum of 30 hops
 0  <1 ms <1 ms <1 ms 192.168.0.1
 1  <1 ms <1 ms <1 ms Gateway.Home [192.168.1.1]
 2  3 ms 2 ms 2 ms bsn-access.dynamic.siol.net [213.250.19.90]
 3  6 ms 2 ms 4 ms 95.176.242.54
 4  18 ms 18 ms 17 ms msft-decix-01-fra.ntwk.msn.net [80.81.194.52]
 5  21 ms 22 ms 23 ms ae26-0.icr01.fra21.ntwk.msn.net [104.44.232.127]
 6  * * 25 ms be-100-0.ibr01.fra21.ntwk.msn.net [104.44.23.103]
 7  25 ms 26 ms 25 ms be-8-0.ibr01.ams30.ntwk.msn.net [104.44.19.234]
 8  25 ms 25 ms 26 ms be-4-0.ibr03.ams06.ntwk.msn.net [104.44.18.185]
 9  25 ms 25 ms 25 ms ae141-0.icr03.ams06.ntwk.msn.net [104.44.21.176]
10  * * * Request timed out.
11  * * * Request timed out.
12  * * * Request timed out.
13  * * * Request timed out.
14  * * * Request timed out.
15  * * * Request timed out.
16  * * * Request timed out.
17  * * * Request timed out.
18  * * * Request timed out.
19  * * * Request timed out.
20  * * * Request timed out.
21  * * * Request timed out.
22  * * * Request timed out.
23  * * * Request timed out.
24  25 ms 25 ms 25 ms 13.81.58.48

```

Figure 1.9: Prikaz poti iz Brezovice pri Ljubljani do virtualnega računalnika.

1.7.5 Umetno breme

Za bolj podrobno testiranje omrežja sva pripravila 2 programa. Prvi program iz lokalnega računalnika pošilja 2MB veliko datoteko, v tem programu lahko tudi določimo interval pošiljanja datoteke. Drugi program pa je treba pognati na najinem virtualnem računalniku na katerem sprejema datoteko in po uspešnem sprejemu pošlje čas prejetja, katerega prvi program izpiše. Cilj programov je testirati hitrost procesiranja datoteke brez zapisovanja na disk. Na sliki 1.10 lahko vidimo čase pošiljanja na 5 sekund z zelo malo nihanja. Zelo malo nihanja je zato, ker ima virtualni računalnik, ki prejema datoteke dovolj časa da stvari procesira.

```
C:\Users\zigas\Desktop\ZZRS test>node send.js 5000
ID: 0 | Travel+Procesing time: 1156 ms
ID: 1 | Travel+Procesing time: 1624 ms
ID: 2 | Travel+Procesing time: 1605 ms
ID: 3 | Travel+Procesing time: 1254 ms
ID: 4 | Travel+Procesing time: 1608 ms
ID: 5 | Travel+Procesing time: 1646 ms
ID: 6 | Travel+Procesing time: 1575 ms
ID: 7 | Travel+Procesing time: 1485 ms
ID: 8 | Travel+Procesing time: 1724 ms
ID: 9 | Travel+Procesing time: 1732 ms
ID: 10 | Travel+Procesing time: 1809 ms
```

Figure 1.10: Časi pošiljanja na 5 sekund.

Če zmanjšamo interval pošiljanja na 0.5 sekunde se časi povečajo, saj virtualni računalnik nima dovolj hitrega procesiranja datotek in zaradi tega prihaja do zamud. To lahko vidimo tudi na sliki 1.15, kjer se je čas prejemanja iz povprečno 1.2 sekunde povečal na povprečno 7 sekund.

```
C:\Users\zigas\Desktop\ZZRS test>node send.js 500
ID: 0 | Travel+Procesing time: 2954 ms
ID: 1 | Travel+Procesing time: 6467 ms
ID: 2 | Travel+Procesing time: 4500 ms
ID: 3 | Travel+Procesing time: 6401 ms
ID: 4 | Travel+Procesing time: 8504 ms
ID: 5 | Travel+Procesing time: 6795 ms
ID: 6 | Travel+Procesing time: 8093 ms
ID: 7 | Travel+Procesing time: 7090 ms
ID: 8 | Travel+Procesing time: 14529 ms
ID: 9 | Travel+Procesing time: 15556 ms
ID: 10 | Travel+Procesing time: 10622 ms
```

Figure 1.11: Časi pošiljanja na 0.5 sekunde.

Na zadnji sliki 1.12 lahko vidimo da je idealen čas pošiljanja datotek na približno 900 milisekund ali 0.9 sekunde. Idealen je zaradi tega, ker se datoteke pošiljajo z najhitrejšim intervalom z majhnim povprečnim časom.

```
C:\Users\zigas\Desktop\ZZRS test>node send.js 900
ID: 0 | Travel+Procesing time: 1656 ms
ID: 1 | Travel+Procesing time: 1703 ms
ID: 2 | Travel+Procesing time: 2141 ms
ID: 3 | Travel+Procesing time: 3873 ms
ID: 4 | Travel+Procesing time: 3193 ms
ID: 5 | Travel+Procesing time: 1793 ms
ID: 6 | Travel+Procesing time: 1734 ms
ID: 7 | Travel+Procesing time: 1799 ms
ID: 8 | Travel+Procesing time: 6253 ms
ID: 9 | Travel+Procesing time: 1810 ms
ID: 10 | Travel+Procesing time: 2847 ms
```

Figure 1.12: Časi pošiljanja na 0.9 sekunde.

1.8 Zmogljivost diskovnega sistema

Po sprejemu datoteke od pošiljatelja, sprejemnik pošlje svoj odgovor, datoteka pa ostane le v pomnilniku in se ne zapiše na disk. Da bi testirala tudi hitrost pisanja datotek na disk, sva ustvarila več datotek različnih velikosti, vsako od njih večkrat poslala na virtualni računalnik, ter jo po sprejetju tudi zapisala na disk. Merila sva čas od začetka zapisovanja do konca zapisovanja vsake datoteke, potrebno pa je upoštevati, da je disk, ki ga ima virtualni računalnik na voljo, precej hiter SSD.

Čas pošiljanja	Min [ms]	Povprečje [ms]	Max [ms]
1 MB	1751	2273	2945
2 MB	3377	3830	4589
5 MB	3918	4012	4277
10 MB	8208	8356	8532
20 MB	17019	17160	17371
40 MB	34819	35094	35439

Table 1.2: Metrike testov.

Najprej sva za vsako prejeto datoteko v pomnilniku na virtualnem računalniku ustvarila novo datoteko na disku. Povprečni časi zapisovanja so precej nizki zaradi SSD diska, odstopanja pa so razmeroma velika.

Čas shranjevanja	Min [ms]	Povprečje [ms]	Max [ms]
1 MB	1	1.3	4
2 MB	1	1.73	2
5 MB	3	3.36	4
10 MB	6	6.8	8
20 MB	12	14.7	22
40 MB	27	30.3	34

Table 1.3: Metrike testov.

Zanimivi so rezultati, ki sva jih dobila, ko sva vsako prejeto datoteko v pomnilniku zapisala v isto datoteko na disku. Seveda sva prepisala vsebino prejšnje prejete datoteke. Povprečni čas se je namreč močno podaljšal, minimum pa je ostal podoben prejšnjemu. Ni nama povsem jasno, zakaj se to zgodi, vendar je odstopanje od pisanja vsakič v novo datoteko preveliko, da bi ga lahko zanemarili.

Čas shranjevanja	Min [ms]	Povprečje [ms]	Max [ms]
1 MB	1	1.3	2
2 MB	2	2.3	5
5 MB	4	24.6	91
10 MB	9	63.3	112
20 MB	18	192.3	225
40 MB	80	255.9	348

Table 1.4: Metrike testov.

1.9 Zmogljivost procesorja

V tem razdelku bova testirala zmogljivost procesorja na virtualnem računalniku in na domačih lokacijah.

1.9.1 Opis programa

Program, ki bo testiral zmogljivost sva napisala sama. Program deluje na enem jedru procesorja. ne uporablja diskovnih pogonov. Osnovna naloga programa je, da z matematično intenzivnim problemom testira hitrost procesorja za kar sva v najinem programu pripravila 2 taka problema. Oba problema uporabljata zelo malo dinamičnega pomnilnika in oba sta implementirana z algoritmoma, ki imata časovno zahtevnost $O(n)$. Vsak algoritem se kliče večkrat, zaradi tega je časovna zahtevnost celotnega programa $O(n^2)$. Ko se program zaključi se izpiše čas izvajanja enega in drugega algoritma, nato pa izpiše še skupni čas izvajanja obeh problemov.

1.9.2 Algoritem za faktorizacijo

Prvi algoritem za vsa števila na intervalu $[10050000, 10050500]$ izračuna faktoriteto (npr. $5! = 120$). Algoritem ne uporablja nobenih izboljšav s katerimi se lahko pohitri njegovo delovanje. Implementacijo algoritma lahko vidite spodaj.

Listing 1.1: Algoritem faktoritet

```
void factorialTEST()
{
    long N = 10050000;

    long iterations = 500;

    for(int i=0;i<iterations;i++)
    {
        long c, n = N, f = 1;
        for (c = 1; c <= n; c++)
            f = f * c;
    }
}
```

```
    n++;  
}  
}
```

Algoritem za faktorizacijo naredi 500 ponovitev. Pri vsaki ponovitvi se naredi $N+1$ primerjav, $C+2$ seštevanj in N množenj. Začetno število N za izračun faktoritete je 10050000. Število za izračun se pri vsaki ponovitvi poveča za 1. Algoritem uporablja podatkovne strukture tipa long, kar je pri vseh računalnikih na katerih se je testiral 64-bitov.

1.9.3 Algoritem za iskanje praštevil

Drugi algoritem za vsa števila na intervalu [10050000, 10055000] preveri če je število praštevilo. Algoritem ne uporablja nobenih izboljšav s katerimi se lahko pohitri njegovo delovanje. Implementacijo algoritma lahko vidite spodaj.

Listing 1.2: Algoritem faktoritet

```
bool isPrime(int n)  
{  
    if (n <= 1)  
        return false;  
  
    for (int i = 2; i < n; i++)  
        if (n % i == 0)  
            return false;  
  
    return true;  
}  
  
void primeTEST()  
{  
    long N = 10050000;  
  
    long iterations = 5000;  
  
    for(int i=N; i<N+iterations; i++)  
    {  
        isPrime(i);  
    }  
}
```

Algoritem za preverbo praštevil naredi 5000 ponovitev. Pri vsaki ponovitvi se naredi v najslabšem primeru N primerjav, $N+1$ seštevanj in N deljenj. Začetno število N za preverbo pripadnosti praštevilom je 10050000. Število za izračun se pri vsaki ponovitvi poveča za 1. Algoritem uporablja podatkovne strukture

tipa long in int, kar je pri vseh računalnikih na katerih se je testiral za tip long 64-bitov za tip int pa 32-bitov.

1.9.4 Rezultati meritev

Program sva pognala tako na najinih računalnikih, na prenosnem in osebem računalniku, kot seveda tudi na virtualnem računalniku na Azure platformi. Ročno sva za vsak slučaj preverila, da program uporablja dejansko samo eno CPU jedro, malo pomnilnika in ne piše na disk, kar sva storila z uporabo sistemskega monitorja. Rezultati meritev:

```
tomi@tomi-laptop:~/random/fri_3_letnik/zzrs/cputest$ ./a.out
Prime TEST elapsed time: 8.545718 seconds.
Factorial TEST elapsed time: 10.659676 seconds.
Program total elapsed time: 19.205493 seconds.
```

Figure 1.13: Časi izvajanja meritev na prenosnem računalniku s procesorjem Intel i5 7. generacije.

```
C:\Users\zigas\CLionProjects\untitled\cmake-build-debug\benchmark.exe
Prime TEST elapsed time: 10.352597 seconds.
Factorial TEST elapsed time: 14.628952 seconds.
Program total elapsed time: 24.983546 seconds.

Process finished with exit code 0
```

Figure 1.14: Časi izvajanja meritev na osebem računalniku s procesorjem Intel i5 3. generacije.

```
ziga@test:~/cputest$ ./a.out
Prime TEST elapsed time: 9.265168 seconds.
Factorial TEST elapsed time: 9.724968 seconds.
Program total elapsed time: 18.990237 seconds.
```

Figure 1.15: Časi izvajanja meritev na virtualnem računalniku na Azure platformi s procesorjem Intel Xeon Platinum 8. generacije.

Rezultati meritev so precej zanimivi. Čeprav je procesor v prenosnem računalniku (i5) 4 generacije novejši od procesorja v osebem računalniku (i5), je ta sodeč

po meritvah na spletu 20% počasnejši od procesorja v osebnem računalniku. Intuicija prav tako pravi, da lahko procesor v osebem računalniku porablja več elektrike in bi zato moral biti hitrejši, čeprav je starejše generacije. V najinih meritvah je procesor v prenosnem računalniku v vseh treh testih hitrejši od procesorja v osebem. Razlogov za to je lahko več:

- i5 ima kljub slabši splošni oceni boljšo oceno za eno jedro, kar je v teh dveh problemih relevantno
- i5 ima novejšo arhitekturo, ki ima določene optimizacije za ukaze, ki se uporabljajo v teh dveh problemih
- čeprav lahko procesor v prenosnem računalniku vleče manj elektrike, pa se testa dovolj hitro končata, da je razlika v temperaturi, ki je posledica večjega toka elektrike, minimalna in lahko tudi prenosi procesor drži polno hitrost
- razlika v predpomnilnikih obeh procesorjev
- nekaj drugega

Na to primerjavo se dobro nanaša tudi Xeon procesor na virtualnem računalniku, ki je zanimivo počasnejši v enem testu in hitrejši v drugem testu od prenosnega procesorja. Glede na to da ima Xeon boljšo oceno za eno jedro kot prenosni procesor, bi moral biti hitrejši v obeh testih. Iz primerjav najinih procesorjev in iz primerjav rezultatov na Azure platformi sklepava, da je razlika posledica arhitekturnih razlik med procesorji ali pa razlik v pomnilniku. Xeon je namreč 48 jedrni procesor in midva si ga seveda deliva z ostalimi virtualkami, ki tečejo na tem procesorju, kar lahko vpliva na delovanje predpomnilnika.

1.10 Zaključek

Tule bo zaključek.

1.11 Viri

- SPEC Cloud R IaaS 2018
- Cloud Spectator
- Cloud Performance Benchmark
- Technology Business Research, Inc.
- + dodaj tiste, za katere nisi prepričan ?

Bibliography