Analiza zmogljivosti oblačnih in strežniških storitev

Uredil prof. dr. Miha Mraz

Maj 2016

Kazalo

Predgovor

Pričujoče delo je razdeljeno v devet poglavij, ki predstavljajo različne analize zmogljivosti nekaterih oblačnih izvedenk računalniških sistemov in njihovih storitev. Avtorji posameznih poglavij so slušatelji predmeta *Zanesljivost in zmogljivost računalniških sistemov*, ki se je v štud.letu 2015/2016 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 2016

Poglavje 1

Analiza zmogljivosti oblačnega sistema za hrambo datotek

Jure Jesenšek, Luka Prijatelj, Tine Šubic

1.1 Predstavitev ideje

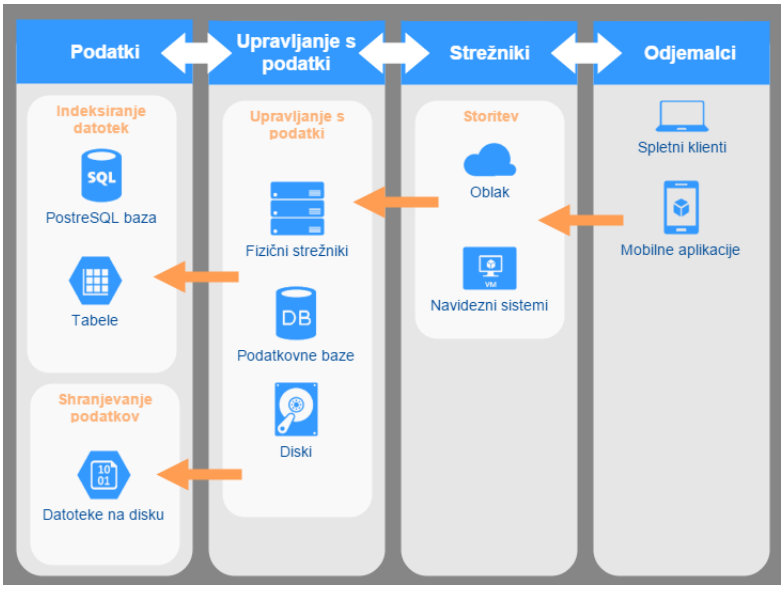
Razvili smo aplikacijo za hrambo datotek v oblaku, ki omogoča prenos uporabniških datotek na strežnik in s strežnika nazaj na lokalni sistem. Možen je tudi prenos večih datotek hkrati po principu storitve Google Drive, ki ob takem zahtevku datoteke arhivira kot ZIP arhiv.

Interakcija odjemalca in strežnika je implementirana po principu REST s HTTP zahtevki. Ob prejemu POST zahtevka za nalaganje datoteke strežnik prenese datoteko na disk in ji v bazi dodeli unikaten ključ. Če prejme GET zahtevek s ključem za eno samo datoteko, poišče ime datoteke v bazi in najdeno datoteko vrne odjemalcu, če pa je v zahtevku datotečnih identifikatorjev več, strežnik datoteke najprej arhivira v en sam arhiv in ga odpošlje odjemalcu.

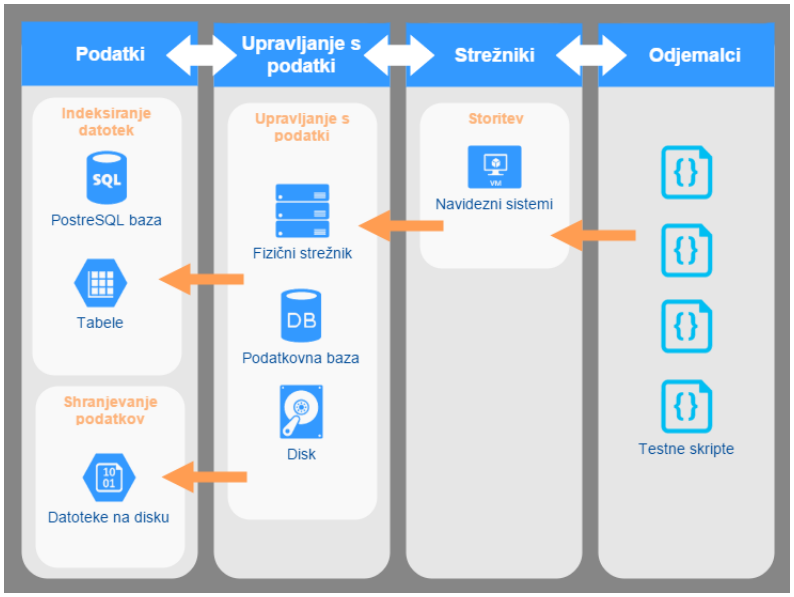
Na sliki 1.1 je prikazan diagram uporabe realnega sistema, na sliki 1.2 pa diagram simulacijskega okolja.

1.2 Izbira gostiteljskega sistema

Za testiranje aplikacije smo pripravili tri platforme:



Slika 1.1: Diagram realnega sistema.



Slika 1.2: Diagram simulacijskega okolja.

- **Digital Ocean Instance** [1] **:** manjši virtualni strežnik, 1x večjedrni procesor (do 3.3 GHz), 512MB pomnilnika, Ubuntu Server Linux OS,
- **Domači računalnik običajnih zmogljivosti:** 1x 4-jedrni procesor (do 3GHz), 12GB pomnilnika, Windows Server OS,
- **Razvojni strežnik Instituta Jožef Stefan:** 4x 8-jedrni procesor (do 3GHz), 512GB pomnilnika, Windows Server OS.

1.3 Izbira tehnologij

Izbrali smo dve tehnologiji za razvoj strežnika, podatkovne baze in odjemalcev.

1.3.1 NodeJS

Strežniška aplikacija je implementirana na NodeJS [2] ogrodju. Ta zaradi vgrajenih orodij in velikega števila dodatnih paketov funkcionalnosti omogoča hiter razvoj spletnih aplikacij, ki potrebujejo skalabilnost in asinhrono izvajanje večjega števila operacij.

Tudi odjemalci so napisani s pomočjo ogrodja NodeJS, saj nam je to omogočilo fleksibilnost in prenosljivost sistema.

1.3.2 PostgreSQL

Za podatkovno bazo v ozadju smo izbrali PostgreSQL [3], odprtokodno rešitev za SQL podatkovne baze, kjer so podatki shranjeni v tabelah. Paket **node-postgres** [4] omogoča hitro asinhrono povezavo med bazo in NodeJS strežnikom.

1.4 Implementacija sistema

1.4.1 Strežniška aplikacija

Pripravili smo strežniško aplikacijo na ogrodju NodeJS, ki implementira REST vmesnik s POST in GET metodami. Odjemalec lahko preko teh metod zahteva datoteke, ali jih nalaga na strežnik.

1.4.2 Nalaganje datotek na strežnik

Odjemalec pošlje HTTP POST zahtevek, ki vsebuje datoteko za prenos. Ob prejemu strežnik shrani datoteko na disk in ji v bazi dodeli identifikator in ga vrne odjemalcu v odgovoru na zahtevek. Ime datoteke na disku je konkatenacija dejanskega imena, prejetega v zahtevku in unikatnega identifikatorja, kar omogoča nalaganje večih datotek z istim imenom.

1.4.3 Prenos datotek s strežnika k odjemalcu

Odjemalec pošlje HTTP GET zahtevek strežniku. Zahtevek vsebuje seznam datotečnih identifikatorjev. Če je identifikator en sam, sistem poišče datoteko v bazi in jo pošlje odjemalcu. Če je identifikatorjev več, sistem najprej zgradi ZIP arhiv datotek in ga šele nato odpošlje odjemalcu. Če datoteke ne najde, odjemalec dobi temu ustrezno sporočilo.

Za arhiviranje datotek smo uporabili knjižnico **adm-zip** [5], ki omogoča sestavljanje ZIP arhivov iz več datotek na disku.

1.4.4 Podatkovna baza

Na strežniku smo postavili PostgreSQL podatkovno bazo, ki v tabeli hrani 2 polji - identifikatorje in imena datotek na disku. Strežnik bo z njo povezan s knjižnico node-postgres. Unikatni identifikatorji se ustvarijo s knjižnico **node-uuid** [6].

1.4.5 Odjemalec

Namen odjemalca je pošiljanje GET in POST zahtevkov z vsebino datotek ter merjenje časa, potrebnega za odgovor.

1.5 Meritve in testiranje

Meritve so se izvajale tako na strežniku, kot tudi na odjemalcih. Splošno dostopnost do strežnikov smo testirali tako, da smo v roku 24 ur periodično izvajali ping zahtevke ter si beležili dostopne čase. Iz teh smo lahko ocenili v katerem obdobju dneva so dostopni časi najkrajši. Na odjemalcih si tudi zapisujemo čase prenosov zahtevkov, saj nas predvsem zanima kakšen vpliv na odzivni čas strežnika imajo dostopi, ki zahtevajo kompresiranje datotek.

Informacijo o razliki med časom ping in časom obdelave zahtevka bomo prvenstveno uporabili pri dinamičnemu nalaganju večih datotek. S to informacijo bomo lahko nadzorovali izbiranje naslednje datoteke, ki naj se naloži, saj se lahko zgodi da je pri obdelavi datotek čas ping majhen, čas obdelave zahtevka pa velik, saj se mora stežnik še ukvarjati s stiskanjem datotek. Testirali bomo tudi razliko med različnima načinoma nalaganja datotek.

Prvi način bo linearen način, ki bo začel z nalaganjem testnih primerov, ki imajo najmanjše velikosti, nato pa bo te velikosti linearno stopnjeval. Drugi način pa je bolj dinamičen, saj bo prav tako začel z nalaganjem testnih primerov z najmanjšo velikostjo, nato pa bo velikost datotek stopnjeval glede na odzivni čas strežnika. Če bo ta čas majhen, potem bo odjemalec povečal velikost primerkov, če pa bo čas velik, pa bo zmanjšal velikost. Prav tako bo tudi nastavljal

hitrost pošiljanja zahtevkov na strežnik.

1.5.1 Strežniški del

Strežnik med delovanjem periodično beleži zasedenost lastnega procesorja in pomnilnika. Prav tako ob vsakem API zahtevku zabeleži čas, ki ga je porabil za obdelavo zahtevka (čas od prejema zahtevka do zaključitve odgovora). Njegovo delo je, da shrani poslano datoteko na disk, ter zanjo generira unikaten identifikator in ga shrani v podatkovno bazo. Identifikator je vrnjen odjemalcu v odgovoru na zahtevek. V primeru, da odjemalec želi prenesti določeno datoteko, ki jo je predhodno že naložil na strežnik, potem potrebuje njen identifikator.

1.5.2 Odjemalec

Odjemalec beleži čas, ki je potreben za potek celotnega zahtevka, poleg tega pa ima še naslednje osnovne funkcije:

- ping strežnika - merjenje odzivnega časa strežnika,
- spremljanje časa odgovora strežnika od trenutka, ko odjemalec pošlje API zahtevek,
- nalaganje ene binarne datoteke na strežnik (JPG slika, TXT besedilo, BIN format...),
- zahteva za prenos ene binarne datoteke s strežnika (JPG slika, TXT besedilo, BIN format...),
- zahteva za prenos več datotek v ZIP formatu s strežnika.

1.5.3 Bremena

Prenose na strežnik simuliramo z bremeni velikosti od 10 kB do 5MB. To so datoteke različnih formatov: TXT, PNG, ZIP in PDF. Za vsak velikostni razred smo naključno generirali 10 datotek vsakega tipa, izmed katerih ob izvedbi testiranja za vsak zahtevek naključno izberemo breme. Na ta način se izognemo anomalijam robnih primerov, predvsem v primeru testiranja kompresije, da dobimo reprezentativen vzorec bremen.

1.6 Rezultati

1.6.1 Testi dostopnosti

Pri testih dostopnosti smo poskušali ugotoviti ali za dan gostiteljski sistem obstaja časovni interval, ki bi bil s stališča dostopnega časa neobremenjenega strežnika najugodnejši.

Teste smo vršili tako, da smo v intervalih dolžine 15 sekund pošiljali PING zahtevke na ustrezen strežnik in beležili čase, potrebne za prejem odgovora. To smo počeli 24 ur in s tem dobili 5760 rezultatov. Tak test smo za vsak strežnik ponovili trikrat in nato rezultate povprečili da smo se znebili anomalij.

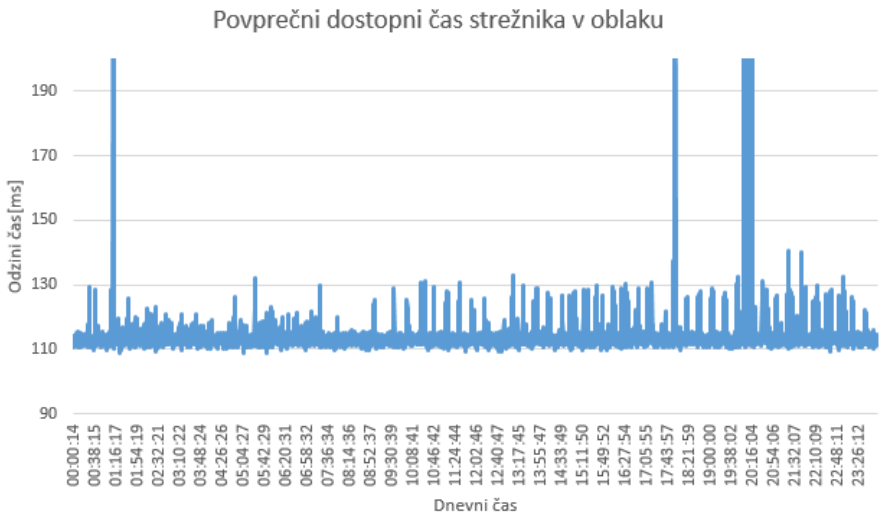
Prvi test je bil izveden z virtualnim strežnikom podjetja Digital Ocean, lociranem v strežniškem centru v New Yorku. Odjemalec se je nahajal v Kranju, za dostop pa je bila uporabljena optična povezava, meritev pa je bila izvedena trikrat. Rezultati meritve so vidni na sliki 1.3. Postavili smo si hipotezo, da bo tu odzivni čas višji kot pri ostalih sistemih zaradi lokacije strežnika, pa tudi da bomo opazili višja nihanja odzivnih časov med dnevnim časom evropskih časovnih pasov.

Rezultati potrjujejo naši hipotezi. Povprečni odzivni časi se gibajo med 105 in 115 milisekund z občasnimi skoki do 135 milisekund. Iz grafa 1.3 je razvidno da se število konic z visokimi odzivnimi časi močno poveča okrog devete ure zjutraj (po lokalnem času, sicer 7:00 po UTC). Ta trend se nadaljuje do polnoči (po lokalnem času, sicer 22:00 UTC), nato pa se število konic zmanjša. Gre za del dneva, v katerem je na internetu aktivna večina populacije, tako v okviru službenih dolžnosti kot prostočasnih aktivnosti, kar povzroča večjo gostoto prometa.

V drugem testu je bil strežnik postavljen na razvojnem sistemu Inšituta Jožef Stefan v Ljubljani. Metodologije meritev je bila tu enaka ko pri prvem testu, prav tako tudi lokacija odjemalca. Naša hipoteza je bila, da bomo zaradi bližje geografske lokacije opazili znatno nižje dostopne čase kot pri oddaljenem virtualnem strežniku.

Tudi v tem primeru so rezultati meritev (Vidni na sliki 1.4)potrdili našo hipotezo. Povprečni dostopni časi so za cel velikostni red manjši od časov, izmerjenih pri oddaljenem strežniku, in sicer med 1 in 10 milisekund, z občasnimi nihanji do 40 milisekund. Tudi tu so večja nihanja lokalizirana v dnevnem, predvsem popoldanskem času. Razlika je tu še posebej očitna zaradi že tako nižjih dostopnih časov. V eni od meritev smo opazili smo tudi močen skok med osmo in deveto uro (nekaj 1000 milisekund), za kar predvidevamo da je posledica povečane aktivnosti ali motnje v omrežju odjemalca, saj se podoben skok pojavlja ob istem času tudi na grafih drugih dveh testov, izvedenih tisti dan.

Zadnji strežnik je bil postavljen na običajnem namiznem računalniku v Ljubljani in povezan na omrežje z optično povezavo. Tu smo predvidevali, da bodo rezultati zelo podobni kot pri strežniku IJS, saj sta si sistema geografsko zelo blizu, na test pa strojna oprema (razen omrežne povezave) ne vpliva. Rezultati so vidni na sliki 1.5. Povprečni dostopni čas se giba med 1 in 5 milisekund, s skoki do 25 milisekund. Malenkost višje vrednosti v prejšnjem eksperimentu pripisujemo predvsem omrežju IJS, ki vsebuje notranje požarne zidove in filtriranje mrežnega prometa, kar najverjetneje povzroči nekaj milisekundni zamik pri odzivu. Na grafu 1.5 ponovno opazimo nekaj zalo velikih skokov, ki so verjetno



Slika 1.3: Testiranje odziva strežnika, Digital Ocean Virtualni strežnik.

posledica aktivnosti v odjemalčevem omrežju.

1.6.2 Testi nalaganja datotek na strežnik

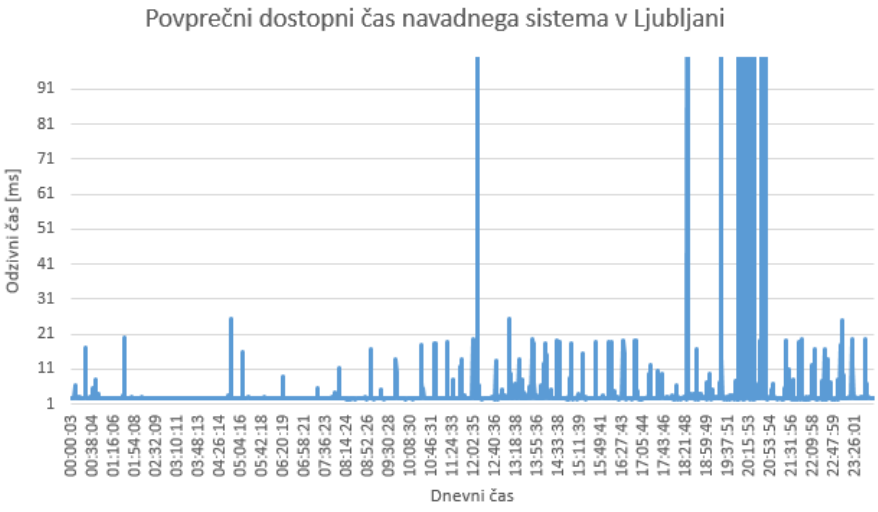
Tu smo testirali kako se spreminja čas, potreben za obdelavo zahtevka v odvisnosti od velikosti datoteke, ki jo je odjemalec poslal na strežnik. Uporabili smo zaporedno pošiljanje datotek s 50 zahtevki v vsakem velikostnem razredu (50kB, 100kB, 500kB, 1000kB, 2500kB, 5000kB). Testi so bili izvedeni preko optične povezave na sistemih standardne zmogljivosti, ki predstavljajo povprečnega odjemalca take storitve. Test vsakega sistema je bil izveden trikrat, rezultati pa nato povprečeni da smo se znebili anomalij.

Pred začetkom testiranja smo postavili nekaj osnovnih hipotez. Predpostavili smo, da se bo najslabše odrezal virtualni strežnik, saj je po strojnih specifikacijah najslabši, hkrati pa je geografsko zelo oddaljen od odjemalcev. Nadalje smo predvidevali, da se bo najbolje odrezal IJS strežnik zaradi velike strojne zmogljivosti, navadni PC pa se bo umestil nekje vmes.

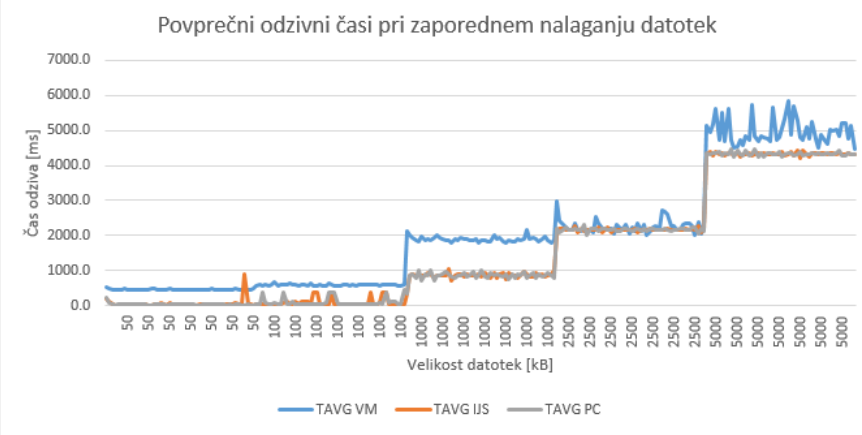
Rezultati so vidni na grafu 1.6.

Naša predvidevanja se niso uresničila popolnoma. Iz grafa 1.6 je razvidno da smo potrdili svojo prvo hipotezo, saj je virtualni strežnik res pokazal najslabše rezultate. Poleg tega, da je bil povprečni čas procesiranja za vsak velikostni razred precej višji kot pri ostalih sistemih, so se pri večjih velikostih pojavljala

Slika 1.4: Testiranje odziva strežnika, razvojni strežnik IJS.



Slika 1.5: Testiranje odziva strežnika, navadni PC.



Slika 1.6: Testiranje odziva na zaporedne zahtevke za nalaganje datotek, vsi sistemi.

tudi velika nihanja odzivnega časa, kar je posledica same narave virtualnega strežnika v oblaku.

Poleg tega da je bil izmed uporabljenih sistemov le-ta najmanj zmogljiv, si mora tak strežnik deliti strojne in omrežne vire ostalimi virtualnimi sistemi na gostitelju. To privede do ozkih grl, med drugim tudi pri bralnih dostopih do diska in podatkovne baze za katere predvidevamo, da so bili poleg geografske lokacije drugi kritični vzrok nihanja odzivnih časov.

Na naše presenečenje so si bili rezultati na standardnem PCju in strežniku IJS zelo podobni. Vzrok tega je naš način testiranja. Ker smo izvajali enostaven test z enim klientom in zaporednimi prenosi, dejanske zmogljivosti niso posebej vplivale na ta dva sistema.

Testi vzporednih nalaganj datotek na strežnik

Pri tem testu smo ugotavljali, kako se spreminja čas, potreben za prenos datotek na strežnik če sočasno poženemo večje število odjemalcev. Vsak od odjemalcev je uporabljal zaporedno pošiljanje s 50 zahtevki (poslanimi datotekami) za vsak velikostni razred (50kB, 100kB, 500kB, 1000kB, 2500kB, 5000kB). Odjemalske skripte so bile porazdeljene na večih fizičnih sistemih v različnih omrežjih, ki so bili na omrežje povezani z optično povezavo. Test za vsak sistem je bil izveden trikrat, rezultati pa nato povprečeni za čimboljšo reprezentativnost podatkov.

Postavili smo si naslednje hipoteze: Sistemi ki se zanašanjo na trde diske (HDD) se bodo odrezali slabše kot tisti s solid state diski (SSD). Ker je NodeJS strežnik enonitna aplikacija, smo domnevali tudi da bodo sistemi s hitrejšimi CPU jedri lažje servirali večje število zahtevkov.

Rezultati so vidni na grafu 1.7.

Slika 1.7: Testiranje odziva vzporednega nalaganja datotek na strežnik, vsi sistemi.

Našo prvo hipotezo smo delno potrdili. Naš testni PC in virtualni strežnik sta oba uporabljala SSD, medtem ko je IJS strežnik Mustang za hrambo podatkov uporabljal HDD. Glede na to da se je pri večjih datotekah najslabše odrezal virtualni strežnik v oblaku, lahko domnevamo da je na ta čas še vedno močno vplivala latenca čezoceanske povezave in nizka frekvenca procesorja.

Bolj primerljiva pa sta rezultata PCja in IJS strežnika, saj se sistema nahajata na približno enako razdalji od odjemalcev (oz. je razlika razdalje zanemarljiva glede na hitrost povezave). Za večje datoteke se je PC odrezal precej bolje, kar je najverjetneje posledica razlike v hitrosti med SSD in HDD.

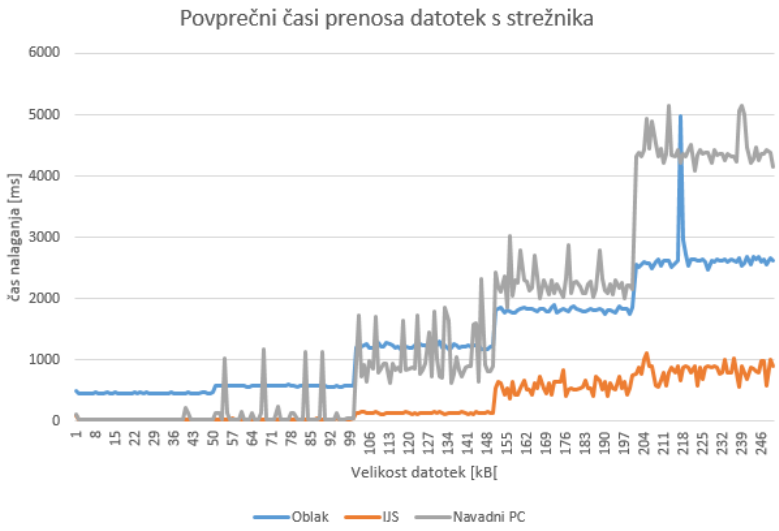
1.6.3 Testi prenosa datotek s strežnika

Tu smo testirali kako se spreminja čas, potreben za obdelavo zahtevka v odvisnosti od velikosti datoteke, ki jo je odjemalec zahteval od strežnika. V testu odjemalec najprej od strežnika pridobi seznam datotek na strežniku - za testne namene smo uporabili po 10 naključno generiranih datotek za vsak velikostni red (50kB, 100kB, 500kB, 1000kB, 2500kB, 5000kB). Odjemalec nato zaporedno zahteva po eno naključno izbrano datoteko ter to stori po petdesetkrat za vsako kategorijo velikosti. Z naključnim izbiranjem poskušamo čimbolj izničiti vpliv robnih primerov naključnih datotek in predpomnjenja. Za vsako datoteko si odjemalec zabeleži njeno velikost, čas za odziv in prenos s strežnika. Vsi testi so bili izvedeni z enega odjemalca z optično povezavo, ponovljeni trikrat, rezultati

pa nato povprečeni za čimboljšo reprezentativnost podatkov.

Pred testiranjem smo si postavili naslednje hipoteze: domnevali smo, da se bo ponovno najslabše odrezal virtualni strežnik zaradi manjše strojne zmogljivosti in da bomo pri tem opazili tudi največja nihanja časa obdelave podatkov za večje datoteke. Predvidevali smo tudi, da se bo IJS strežnik odrezal najbolje, navadni PC pa nekje vmes.

Rezultati so vidni na grafu 1.8.



Slika 1.8: Testiranje odziva na zaporedne zahtevke za nalaganje datotek, vsi sistemi.

Naša predvidevanja se niso uresničila popolnoma. Iz grafa 1.8 je razvidno, da smo potrdili hipotezo o strežniku IJS. Zaradi geografske lokacije in večje strojne zmogljivosti je ta sistem deloval najbolje. Pri majhnih datotekah je bilo delovanje podobno navadnemu PCju, a se je očitna razlika pokazala pri večjih datotekah.

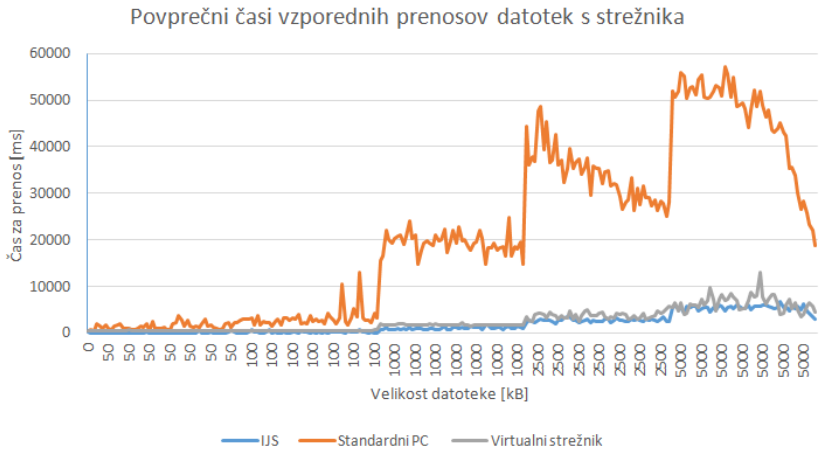
Virtualni strežnik se je odrezal presenetljivo dobro. Pri manjših datotekah so bili zaradi oddaljenosti sicer odzivni časi višji kot pri ostalih sistemih, a pri velikih datotekah se je ta strežnik odzival hitreje kot navadni PC, hkrati pa so bili časi bolj stabilni, z manjšimi nihanji kot ostala dva sistema. Vzrok za stabilnost je najverjetneje uporaba hitrih SSD diskov v virtualnih sistemih podjetja Digital Ocean, medtem ko sta IJS Mustang in navadni PC uporabljala navadne trde diske za hrambo datotek.

Odzivi navadnega PCja so bili slabši kot smo pričakovali. Kljub nizkim odzivnim časom za majhne datoteke je sistem deloval slabo pri večjih velikostih. Na grafu so vidna tudi večja nihanja kot pri ostalih sistemih.

Testi vzporednih prenosov datotek iz strežnika

Podobno kot pri testu nalaganja datotek na strežnik, smo podoben test izvedli tudi z vzporednimi prenosi datotek iz strežnika k klientom. Testirali smo kako večje število sočasnih zahtevkov vpliva na čase prenosov. Vsak od klientov je uporabljal zaporedno pošiljanje 50 zahtevkov za vsak velikostni razred (50kB, 100kB, 500kB, 1000kB, 2500kB, 5000kB). Funkcijo skupno 30 odjemalcev so vršili trije sistemi na različnih lokacijah - vsi pa so bili povezani z optično povezavo. Testi so bili izvedeni trikrat, ob različnih časih, rezultati meritev pa so povprečeni.

Podobno kot pri vzporednem nalaganju na strežnik, smo tudi tukaj predpostavili da se bodo strežniki s trdimi diski slabše odrezali kot tisti, ki vsebujejo diske SSD. Poleg tega smo predvidevali da bo na čas vplivala tudi hitrost internetne povezave pri internetnemu ponudniku strežnika (upload hitrost strežnika).



Slika 1.9: Rezultati testiranja odziva na vzporedne zahtevke, vsi sistemi.

Drugo hipotezo smo potrdili, saj se je najslabše odrezal standardni PC (slika 1.9) s standardno optično povezavo (100Mb/s download, 10Mb/s upload), kjer se je za ozko grlo izkazala nižja hitrost do strežnika proti odjemalcem. Pri virtualnem strežniku smo pričakovali dobre rezultate, saj so sistemi na katerih smo gostili naš strežnik, v večji meri namenjeni pošiljanju datotek proti odjemalcem. Podobno predivdevamo za strežnik IJS Mustang, pri katerem sta verjetno hitrosti povezave visoki v obe smeri.

Pri standardnem PCju druga hipoteza glede tipa diskov ne pride v poštev, saj je največja razlika nastala zaradi počasnejše internetne povezave. Pri ostalih dveh strežnikih pa tudi ne zaznamo večjega odstopanja na ta račun, saj bi lahko nekoliko daljše čase odziva virtualnega strežnika pripisali geografski oddaljenosti strežnika (ZDA).

Pri grafu na sliki 1.9 bi radi še omenili, da je vzrok za padajočo krivuljo pri standardnem PCju ta, da je strežnik nekaterim odjemalcem že odgovoril na vse njihove zahtevke in je z njimi prekinil povezavo, posledično se je odzivni čas za ostale odjemalce zmanjšal.

1.7 Zaključek

Sistem ki smo ga razvili deluje zanesljivo in je dovolj skalabilna za širšo uporabo. V pripravah na teste je sistem vzporedno serviral 50 različnih odjemalcev (višjega števila odjemalcev nismo uporabili zaradi finančnih omejitev), brez kritičnih težav. Med testiranjem smo odkrili tri glavna ozka grla sistema, ki so omrežna povezava, hitrost diskov in frekvenca CPU jeder. PostgreSQL podatkovna baza je ves čas testiranja delovala brez večjih sprememb v porabi sistemskih virov, saj so vsi testi uporabljali zgolj nekaj tisoč vrstic tabele, kar je za bazo minimalna obremenitev.

V realnem produkcijskem okolju bi te težave lahko rešili na več načinov. V kolikor bi aplikacija tekla v oblaku lahko običajno privzamemo da ima podatkovni center oblačnega ponudnika zadostno pasovno širino za naše potrebe oz. lahko za zadostno pasovno širino doplačamo, medtem ko smo bili mi omejeni na zmogljivost ki so nam jo nudili naši ponudniki interneta. V idealnem primeru bi naš sistem tekel v večjem številu strežniških centrov po svetu, saj smo videli da geografska lokacije močno vpliva na prenosne čase in bi na ta način znižali latenco. Večina ponudnikov virtualnih stežnikov ponuja izbiro regije kjer uporabnik želi gostiti aplikacijo. Za večje število uporabnikov bi uporabili t.i. load balancing [7], sisteme ki so sposobni optimalno porazdeliti promet aplikacije da en sam sistem ni preobremenjen. Uporabiti je mogoče tudi samodejno skaliranje sistemov in tako zagotoviti visoko zanesljivost in dostopnost sistemov iz katerekoli regije. S povečanjem števila instanc aplikacije se znebimo tudi potrebe po sistemih z visokimi frekvencami CPU jeder. Za najboljše delovanje bi morali biti podatki shranjeni na SSD diskih ali na klasičnih trdih diskih, povezanih v RAID 0 [8] načinu.

Med testiranjem smo uporabili vse kredite za storitev Digital Ocean ki smo jih imeli na voljo in doplačali 4 USD.

Literatura

[1] "Digital Ocean." https://www.digitalocean.com/, April 2016.

[2] "NodeJS." https://nodejs.org/, March 2016.

[3] "PostgreSQL." http://www.postgresql.org/, March 2016.

[4] "Node-Postgres library." https://github.com/brianc/node-postgres, March 2016.

[5] "ADM Zip." https://github.com/cthackers/adm-zip, March 2016.

[6] "UUID Generator." https://github.com/broofa/node-uuid, March 2016.

[7] "Load balancing." https://en.wikipedia.org/wiki/Load_balancing_(computing), May 2016.

[8] "Raid 0." https://en.wikipedia.org/wiki/Standard_RAID_levels#RAID_0, May 2016.