Analiza zmogljivosti oblačnih storitev za hranjenje podatkov

Aleksandar Gogov, Staš Hvala, Matic Repše

16. maj 2016

Kazalo

1 Predstavitev ideje

Lokalni prostor izgublja na pomembnosti, ker se uporabniki vedno pogosteje poslužujejo oblačnega hranjenja podatkov. Osnovne oz. minimalne storitve so po večini brezplačne, možno pa je tudi nadgraditi prostor in uporabljati plačljive storitve, ki uporabniku omogočajo uporabo dodatnih funkcionalnosti. Namen naše raziskovalne naloge je testirati najpopularnejše ponudnike (Google Cloud Storage, Amazon S3, Microsoft Azure) in bralcu predstaviti rezultate in ugotovitve testiranj. Naša želja je tudi odkriti, kdo ima zanesljivejšo ter zmogljivejšo performanco oblaka.

2 Rešitev problema

Testiranja smo se najprej lotili z CloudHarmony benchmark-om (glej vir 1), da lahko postavimo začetne hipoteze in dobimo okvirno sliko rezultatov. Ker pa nam je konec koncev najpomembnejša neposredna izkušnja uporabnika, smo se odločili, da preverimo kakovost storitev tudi z lastnimi testi in tako pridemo do pristnejših rezultatov vsakdanje uporabe. Prav tako nam dovoljujejo tudi večjo manipulacijo z bremenom (tip, velikost, število).

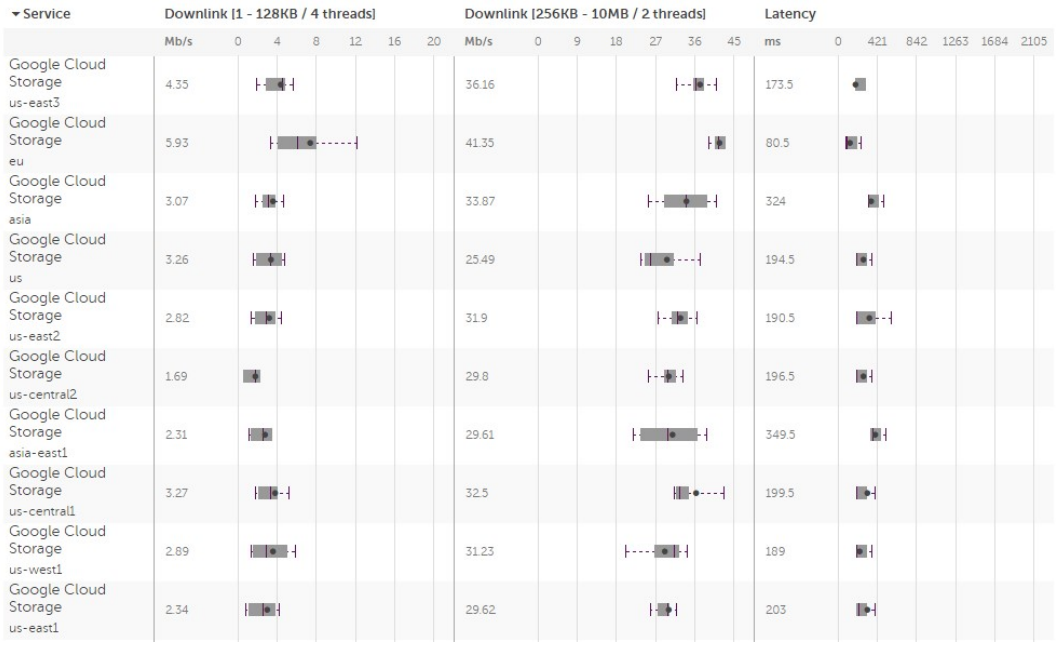
3 CloudHarmony Benchmark

CloudHarmony zagotavlja objektivno, nepristransko in zanesljivo analizo uspešnosti za primerjavo storitev računalništva v oblaku. Testirali smo Google Cloud Storage, Amazon S3 in Microsoft Azure v različnih regijah.
Osredotočili smo se na dva različna testa:

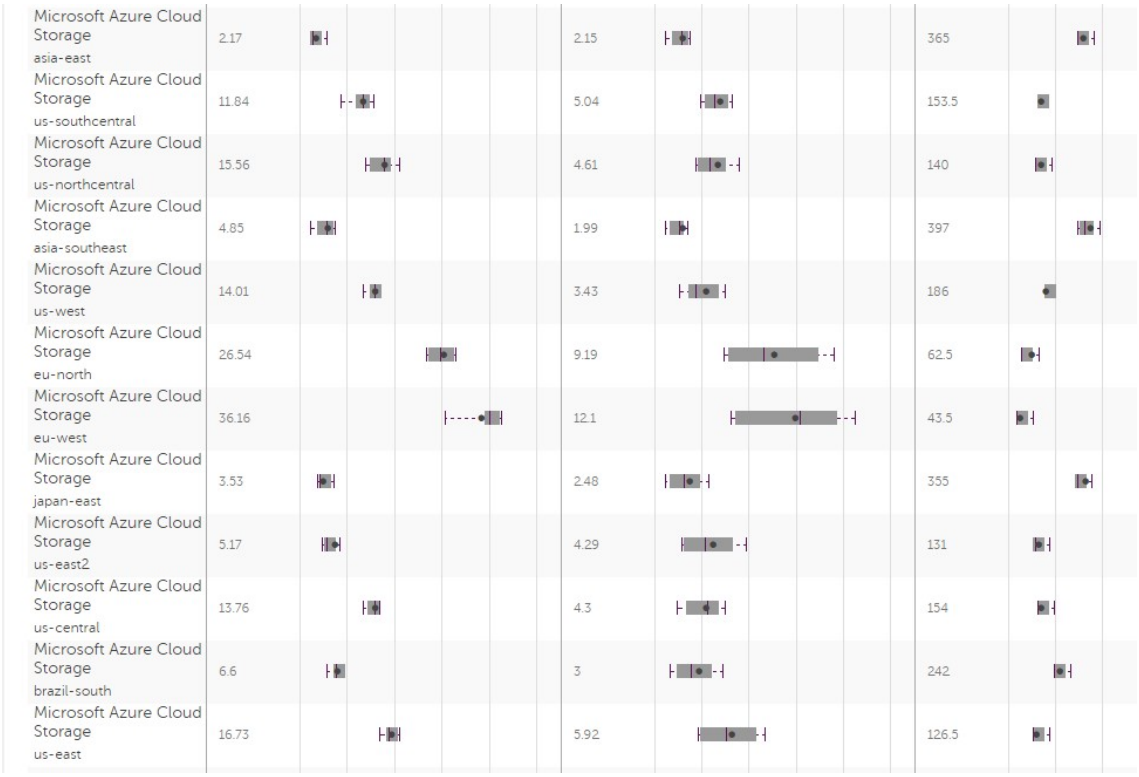
- Downlink
  - Downlink [256KB - 10MB / 2 threads],
  - Downlink [1 - 128KB / 4 threads],
- Latency.

Na vsako uro smo pognali 6 testov na HTTP protokolu, ker je hitrejši brez dodatnih varnostnih rokovanj. Kot je bilo pričakovano glede na naš geografski položaj, naši rezultati kažejo najboljšo odzivnost in hitrost prenosa pri evropskih strežnikih. Naše meritve so bile opravljene na 33 različnih točkah, ki so bile porazdeljene na treh kontinentah (US, EU, ASIA).

Slika 1: Amazon S3 CloudHarmony benchmark rezultat.



Slika 2: Google Cloud Storage CloudHarmony benchmark rezultat.



Slika 3: Microsoft Azure CloudHarmony benchmark rezultat.

Pri Amazon S3 je iz škatel z brki (angl. *box plot*) razvidno, da je downlink najhitrejši na eu-central strežniku. Ista zgodba je tudi pri latenci, najnižja je v centralni Evropi (glej sliko 1).

Google Cloud Storage ima v Evropi samo en strežnik, in sicer na zahodu. Tudi ta je najhitrejši po odzivnosti in prenosu podatkov glede na ostale kontinente (glej sliko 2).

Zadnji je še Microsoft Azure, pri katerem ponovno prevladuje evropski strežnik, je bilo pa velika podobnost med Severno in Zahodno Evropo. Po večjem številu testov je prevladal zahod (glej sliko 3).

Za vse tri ponudnike je očitno, da imajo najhitrejše in najodzivnejše strežnike v Evropi. Pričakovali smo, da pri vseh treh do tega ne bo prišlo, ker se ponavadi pretakajo informacije po celemu svetu, a očitno ima geografska razdalja večji vpliv kot smo mislili. Ob kratkem pregledu na internetu smo ugotovili (glej vir 5), da je odvisno kje je postavljen naš ISP (angl. *Internet service provider*) oz. v tem primeru CloudHarmony strežnik, ki izvaja teste. Očitno je CloudHarmony testni strežnik lociran nekje v Evropi in ker smo geografsko tudi mi najbližji ponudnikom v Evropi, smo se dokončno odločili

za zgoraj navedene lokacije.

Ob kategoričnem pregledu rezultatov je Microsoft Azure z najkrajšo latenco in največjim downlinkom zmagovalec na izbranem bencharku. Drugo mesto je zasedel Google. Njegov downlink se zelo približa Microsoftovemu, a je latenca podvojena. Amazonova latenca je malo večja od Azure-a, vendar je downlink približno dvakrat manjši kot pri ostalih dveh ponudnikih.

Uplinka in DNS-ja kljub možnosti izbire ni bilo mogoče testirati, saj očitno CloudHarmony pri teh storitvah tega ne podpira. Predpostavili pa smo, da so hitrosti za uplink glede na lokacijo podobne downlinku.

4 Lastni testi

Naši lastni testi so usmerjeni v čas upload-a in download-a posamezne storitve. Opravili smo več različnih testov, vsakega s svojim določenim tipom bremena, kjer smo za vsak test ločili bremena glede na posamezno lastnost (velikost datoteke, tip datoteke, število datotek, itd.)
Testirali smo torej upload/download čas:

- posamezne datoteke,
- večih datotek,
- datotek različnih velikosti,
- datoteke pred vstavitvijo in po vstavitvi v predpomnilniku na oblaku,
- ...

Za dostop in testiranje storitev smo uporabljali unix-ov terminal, v katerem se avtomatizirano izvaja *bash* skripta.
Za posamezno storitev smo uporabili naslednja bash orodja:

- `azure cli` za Microsoft Azure (glej vir 2),
- `gsutil` za Google Cloud Storage (glej vir 3),
- `aws cli` za Amazon S3 (glej vir 4).

4.1 Breme

Z določitvijo bremena smo testirali latenco in odzivnost ponudnika - za sekvenčno ter paralelno prenašanje. Glede na pridobljene rezultate smo poskušali ugotoviti ali uporabljajo *cache* (za majhne datoteke) in kompresijo (za določen tip datoteke, velike datoteke,

itd.). Poskušali smo se izogniti zunanjim vplivom in pridobiti čimbolj pristne rezultate. Meritve smo zato izvedli večkrat - vsako uro, tekom enega dneva - a smo bili s številom testiranj omejeni, ker se poslužujemo brezplačne verzije pri vseh ponudnikih. Testiranje smo izvajali na sledečih bremenih:

- Posamezne datoteke:
  - 1kB,
  - 10kB,
  - 100kB,
  - 1MB,
  - 10MB,
  - 100MB,
  - 500MB,
  - 1GB.
- Več datotek:
  - 1kB x 2, 5, 10, 50, 100,
  - 10kB x 2, 5, 10, 50, 100,
  - 100kB x 2, 5, 10, 50, 100,
  - 1MB x 2, 5, 10, 50, 100,
  - 10MB x 2, 5, 10, 50, 100,
  - 100MB x 2, 5, 10,
  - 500MB x 2, 5,
  - 1GB x 2.

Testne datoteke so končnice *.txt*, napolnjene z naključnimi podatki. Za testiranje download/upload pri načinu "več datotek" potrebujemo pripravljeno okolje in sicer smo zgradili drevesno strukturo datotek. V korenskem imeniku se nahajajo sinovi, ki so razdeljeni po velikosti datotek, ki jih vsebujejo. Znotraj so datoteke istih velikosti razdeljene še po kvantiteti, torej vsak imenik vsebuje število datotek, ki se bo paralelno preneslo. Za ustvarjanje in kopiranje datotek smo uporabili skripto 1.

```
#!/bin/bash

SIZE=$1
NUMBER=$2
PREFIX=$3
DESTINATION=$4

# Ustvarjanje tekstovne datoteke poljubne velikosti
base64 /dev/urandom | head -c $SIZE  > $PREFIX.txt

# Kopiranje datoteke poljubno krat
for file in `seq 1 $NUMBER`; do
    filename=${PREFIX}_${file}
    cp $PREFIX.txt $DESTINATION/$filename.txt
done
```

```
# Odstranitev originalne datoteke
rm $PREFIX.txt
```

Listing 1: Skripta za ustvarjanje in kopiranje datotek.

4.2 Samodejno izvajanje testov

Ker se izogibamo ročnemu poganjanju in beleženju rezultatov, smo napisali skripto 2, ki bo to opravljala namesto nas. Skripte smo poganjali na *eduroam* omrežju na serverju *XeonPhi* (z dovoljenjem as. uni. dipl. ing. Davor Sluga), saj server teče 24/7, sploh pa bo zaradi 24-ih jeder in 64GB *RAM*-a prenašanje datotek nemoteno. V primeru plačljivih verzij, bi za zanesljivejše rezultate testirali še na drugih omrežjih. Pri brezplačnemu načinu pa to ni možno, ker nam omejujejo število operacij, več računov pa ne moremo ustvariti, ker za registracijo zahtevajo številko kreditne kartice.

```
#!/bin/bash

printf "%s\n" "arguments:"
printf "%s\n" "-m ----> multiple files"
printf "%s\n" "-t ----> followed by int time in hours \
        (execute every H hours)"
printf "%s\n" "-n ----> execute N times"

#default values
SOURCE=single_file # default src FOLDER
H=1
N=24
S3Bucket=s3://zzrs2016/test
GSUtilBucket=gs://zzrs/test

while [ "$#" -gt 0 ]
do
        case $1 in
        -m)
                SOURCE=multiple_files
                shift 1
                continue
                ;;
  -t)
```

```
    H=$2
    shift 2
    continue
    ;;
  -n)
    N=$2
    shift 2
    continue
    ;;
        *)
                ;;
        esac
        shift
done

SleepTime=$[ $H*3600 ] # Set SleepTime

# SleepTime=1 # comment this line when executing for real
while [ "$N" -gt 0 ]
do
  for file in "$SOURCE"/*
  do
    ./amazonS3.sh -s $file -d S3Bucket # upload
    ./amazonS3.sh -d $file -s S3Bucket -D # download

    ./gcs.sh -s $file -d GSUtilBucker # upload
    ./gcs.sh -s $file -d GSUtilBucker -D # download
  done
  N=$[$N-1]
        echo "-----------------" >> "rezultati.txt"
        sleep $SleepTime
done
```

Listing 2: Skripta za samodejno izvajanje testov.

4.3 Amazon S3

Najprej smo na Amazonu ustvarili račun z osebnimi podatki in prestali avtorizacijo. Izbrali smo brezplačni paket, ki traja 12 mesecev. Ponudnik obljublja skalabilen in zanesljiv sistem z nizko latenco ter:

- 5GB prostora,
- 20.000 *GET* zahtev,
- 2.000 *PUT* zahtev.

Sledila je kreacija *bucketa* (odlagališče), dodelitev pravic (trenutno *public*) in zapis *credentionals* v `.bashrc` kot okoljske spremenljivke (`AWS_ACCESS_KEY_ID` in `AWS_SECRET_ACCESS_KEY`).
S pomočjo *AWS CLI* (angl. *Command Line Interface*) ukaza smo napisali skripto 3, ki obdela argumente in jih posreduje aws-ju, medtem pa meri čas izvajanja uploada ali downloada. Za prenos in pošiljanje se uporablja bash komande `cp`, kjer samo zamenjujemo izvor (angl. *source*) in cilj (angl. *destination*) za željeno operacijo (download/upload). S stikalom `--recursive` prenašamo celotne imenike, avtor *API*-ja (angl. *Application programming interface*) pa nam zagotavlja, da se prenašanje izvaja paralelno.

```
#!/bin/bash

printf "%s\n" "arguments:"
printf "%s\n" "-s ----> source file (path)"
printf "%s\n" "-d ----> destination file (path)"
printf "%s\n" "-D ----> download (upload is the default\
        action)"

# default vrednosti za gradnjo aws ukaza
DESTINATION=s3://zzrs2016/test
SOURCE=upload_test
UPLOAD=upload
start_measuring_time() {
  read s1 s2 < <(date +'%s %N')
  timestamp=$( date +"at: %d-%m-%Y %H:%M:%S" )
}
stop_measuring_time() {
  read e1 e2 < <(date +'%s %N')
}
show_elapsed_time() {
  if [[ $UPLOAD == "upload" ]]
  then
    echo "File: $SOURCE, Started $UPLOAD $timestamp, \
    lasted: $((e1-s1)) seconds, $((e2-s2)) nanoseconds" >>\
    "rezultatiAmazonS3.txt"
```

```
  else
    echo "File: $DESTINATION, Started $UPLOAD $timestamp, \
    lasted: $((e1-s1)) seconds, $((e2-s2)) nanoseconds" >>\
    "rezultatiAmazonS3.txt"
  fi
}
while [ "$#" -gt 0 ]
do
        case $1 in
        -s)
                SOURCE=$2
                shift 2
                continue
                ;;
        -d)
                DESTINATION=$2
                shift 2
                continue
                ;;
  -D)
    UPLOAD=download
    ;;
        *)
                ;;
        esac
        shift
done
if [ -d "$SOURCE" ]
then
   start_measuring_time
   aws s3 cp $SOURCE $DESTINATION --recursive
   stop_measuring_time
else
   start_measuring_time
   aws s3 cp $SOURCE $DESTINATION
   stop_measuring_time
fi
show_elapsed_time
```

Listing 3: Skripta za testiranje Amazon S3 storitve.

4.4 Google Cloud Storage

Pri GCS je brezplačna ponudba malo drugačna. Ob registraciji so nam dali **300$** virtualnega denarja, ki ga lahko mirne vesti zapravljamo na njihovi strani. Cene storitev, ki smo jih uporabljali, so sledeče:

- **$0.10** na 10.000 operacij za *GET* ali *PUT* ukaze,
- Standard storage (tip bucketa, najnižja latenca) **$0.026** na GB podatkov na mesec,
- *DELETE* je brezplačen.

Lokalni terminal smo z bucketom povezali preko ukaza `gsutil config`. Ta nam vrne *url* (angl. *Uniform Resource Locator*) naslov na katerem dobimo aktivacijsko kodo, ki jo nato vpišemo v terminal.

Skripta za prenašanje datotek je sila podobna Amazonovi, saj za prenos tudi uporablja ukaz `cp`. S kombinacijo stikal `-r` (*recursive*) prenašamo imenike, z `-m` pa paralelno prenašanje. Primer uporabe: `gsutil -m cp -r dir gs://zzrs`. Skripte zaradi podobnosti ne bomo prilagali.

4.5 Microsoft Azure

Azure CLI nam prav tako ponuja širok spekter ukazov s katerimi delamo na Azure platformi, kjer je velik poudarek na operiranju z *Blobi*. Za dostopanje do storitve smo najprej kreirali uporabniški račun in ob registraciji dobili **200$** virtualnega denarja, ki se troši s hranjenjem in prenašanjem datotek (blobov). Cene storitev, ki smo jih uporabljali, so sledeče:

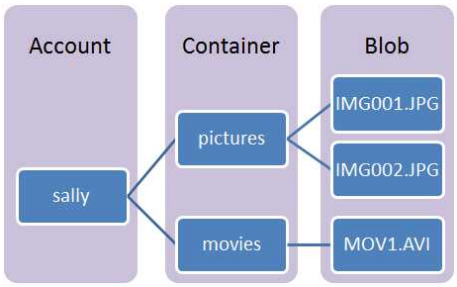
- za *Standard storage (Block blobs)* na mesec za prvi TB podatkov hranjenje stane **$0.024** na GB podatkov,
- na 100.000 transakcij (branje/pisanje) je **$0.0036** tarife,
- brisanje je brezplačno.

Že takoj opazimo, da je ta storitev cenovno najugodnejša in je ne glede na odzivnost storitve najboljša izbira za uporabnika, ki hoče koristiti brezplačno verzijo.

Po registraciji smo se preko terminala povezali na *container* tako, da smo se najprej indentificirali z vključitvijo `AZURE_STORAGE_ACCOUNT` in `AZURE_STORAGE_ACCESS_KEY` v datoteko `.bashrc`. Za sam prenos datotek nam je spet ponujen intuitiven *API*, a se od ostalih dveh ponudnikov ločuje z uporabo dveh različnih funkcij za upload in download. Primer uporabe:

`azure storage blob upload source_to_upload container_name blob_name.`
`azure storage blob download container_name blob_name destination_folder.`

Značilnost tega ponudnika je, da se podatki prenašajo v *blob-ih*. Azure blob je storitev za shranjevanje velike količine nestrukturiranih podatkov, kot so besedilo ali binarni podatki, do katerih lahko dostopamo prek HTTP ali HTTPS. Na sliki 4 je razvidno, kakšno arhitekturo uporablja Azure.



Slika 4: Primer Azure Blob storitve.

Obstajajo tri vrste blob-ov: *block blobs*, *page blobs* in *append blobs*. Block blobs so namenjenji hranjenju tekstovnih in binarnih datotek (npr. dokumenti, medijske datoteke, itd.). Append blobi se uporabljajo za isti namen, razlika je le, da so sestavljeni iz blokov in so optimizirani za hitro dodajanje (uporablja se jih lahko npr. za logiranje). Page blobi so namenjeni za večje datoteke (do 1 TB) in so uporabni, kadar potrebujemo hitre dostope za branje/pisanje. Ker uporabljamo brezplačno verzijo in imamo omejene kredite, uporabljamo Block blobe, saj ne bomo prenašali ogromnih datotek, poleg tega pa so tudi tekstovne narave.

Struktura napisane skripte se je ohranila v primerjavi z ostalima dvema, razlikujejo se le ukazi za prenos datotek. Nažalost pa to orodje in ostale alternative ne ponujajo sočasnega prenašanja večjega števila datotek, zato smo pri Microsoft Azure testirali in primerjali samo rezultate za prenašanje ene datoteke naenkrat.

4.6 Primerjava rezultatov

Pri testiranju za eno datoteko naenkrat smo izvedli 24 iteracij za vse 3 ponudnike naenkrat in enournim razmakom med iteracijami. Pri testiranju za več datotek (paralelni prenos) naenkrat pa smo zaradi omejenih resursov izvedli 12 iteracij z dvournim razmakom samo za Google in Amazon, ker Azure ne podpira paralelnega prenosa. Prav tako je vredno omeniti, da smo teste izvajali čez vikend, ker je po analizi lanske 6. skupine

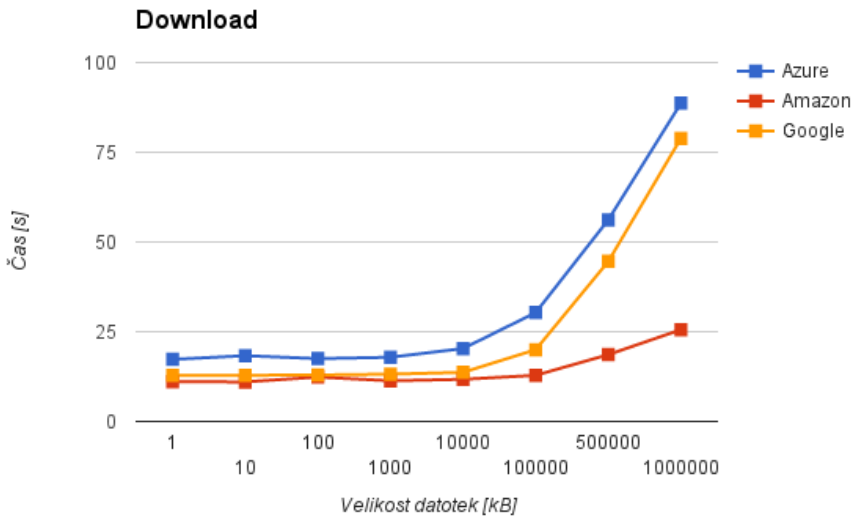
(glej vir 6) takrat obremenitev strežnika Amazon S3 v Evropi najmanjša. Enako smo predpostavili za ostala dva ponudnika.

Za vse iteracije smo izračunali povprečje prenosa, ker pa smo nekajkrat dobili iregularne rezultate, smo vključili v izračun tudi standardno deviacijo.

4.6.1 Testiranje enkratnega prenosa

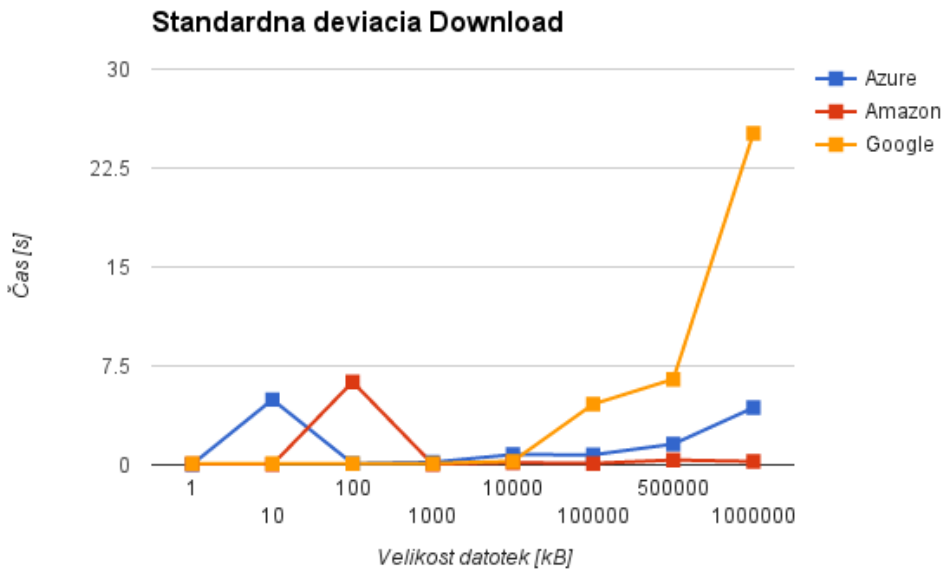
Na sliki 5 za download se takoj opazi, da so jasni hitrostni pasovi za vse tri ponudnike, ne glede na velikost datoteke. Najhitrejši je Amazon, drugi Google in tretji Azure. Uvrstitev je ravno obratna kot pri CloudHarmony (glej poglavje 3), a težko ugotovimo kje in zakaj je do tega prišlo, ker na njihovi strani ne opisujejo prav posebej kako in kakšne teste izvajajo. Je pa zanimivo, da je uvrstitev skladna z radodarnostjo ponudbe za brezplačno verzijo. Amazon nudi najmanj, a to opravi z največjo hitrostjo, Google ima srednjo drago storitev ter srednjo hitrost, Azure pa ponuja najcenejšo storitev, a se to odraža tudi v hitrosti prenosa.

Pri vseh treh ponudnikih je za downloadiranje vidno, da je čas prenosa do velikosti 10000kB skoraj zanemarljiv in do izraza pride le latenca, ki pa med ponudniki raste v enakem zaporedju kot sam download. Za velike datoteke (nad 100000kB) čas prenosa raste precej linearno, torej nobeden ne uporablja kompresije ali pa paralelnega prenosa po koščkih. Googlova krivulja se po obliki prilagaja Azurovi in za velike datoteke obe strmo naraščata, medtem, ko pri Amazonu narašča z manjšim naklonom in za 1GB veliko datoteko prinaša že do 50 sekund razlike v prenosu.



Slika 5: Graf povprečnega downloada ene datoteke.

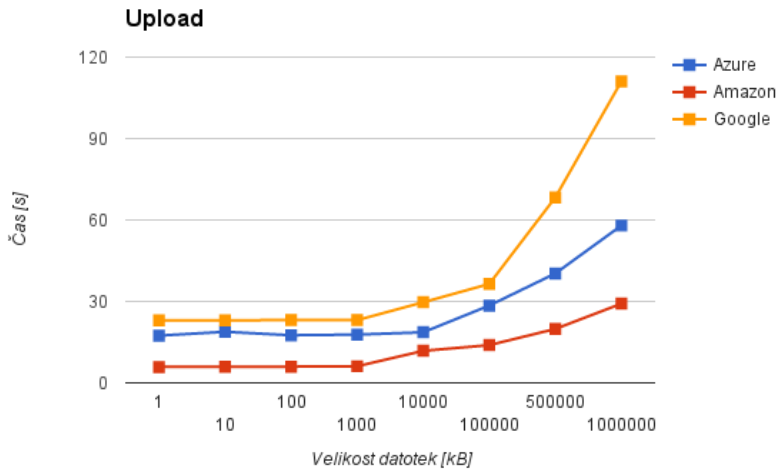
Do odstopanj pri Azure-u in Amazonu (glej sliko 6) pride le za manjše datoteke, pri večjih pa se je čas prenosa večinoma držal povprečja. Pri Googleu pa je bilo ravno obratno, za manjše datoteke je bila standardna deviacija izredno majhna, medtem ko je za velike datoteke (nad 10000kB) eksponentno naraščala. To nakazuje na izredno nestabilno storitev, a mogoče smo imeli le smolo in naleteli na obdobje vzdrževalnega dela na serverju.



Slika 6: Graf standardne deviacije downloada ene datoteke.

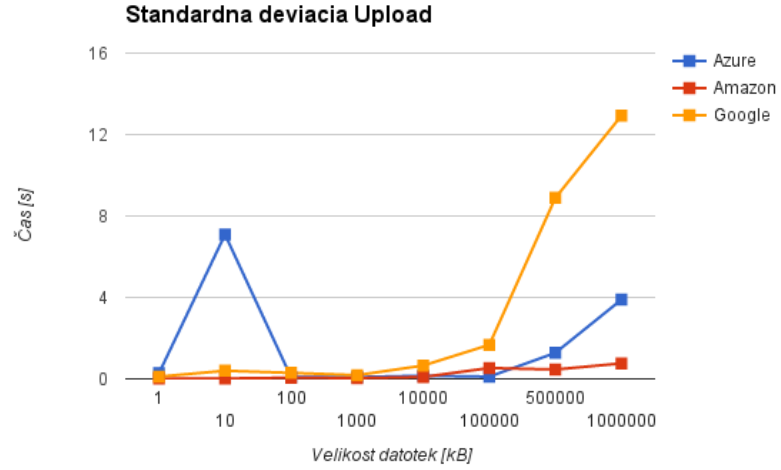
Za upload je s slike 7 razvidna drugačna razvrstitev. Na prvem mestu je še vedno Amazon, sta se pa zamenjala Google in Azure. Med ponudniki in njihovimi krivuljami uploada glede na velikost datoteke ni več očitnih podobnosti, pač pa vsaka narašča s svojim naklonom. Prav tako je razlika med ponudniki večja za upload kot za download, torej imata Azure in Google pred sabo še precej optimizacije za nalaganje datotek.

Prag latence za Azure ostaja pri do 10000kB, pri Googleu in Amazonu pa so večje časovne razlike opazne že za datoteke večje od 1000kB. Tudi tukaj je razvidno, da sta hitrost uploada in višina latence povezana, torej nobena storitev ne potrati preveč časa za rokovanje (angl. *handshake*).



Slika 7: Graf povprečnega uploada ene datoteke.

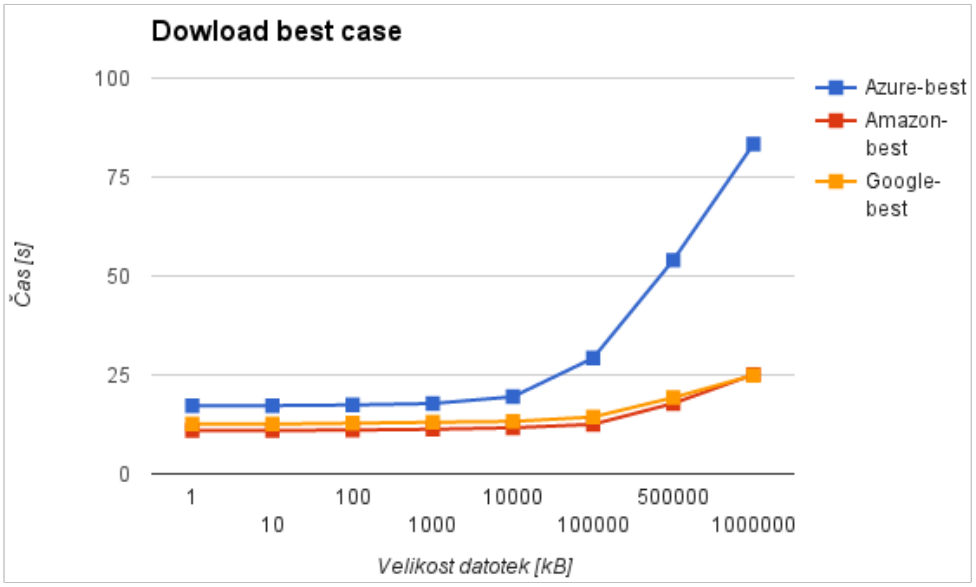
Pri standardnem odklonu za upload (glej sliko 8) se zgodba ponovi, nekaj manjših razlik je prišlo pri Azureu, Google pa je ponovno variiral za velike datoteke (nad 100000kB). Konsistenca je pri prenašanju lahko izredno pomembna in se je vsekakor splačalo izvesti teste večkrat, ker smo le tako dobili bolj točno oceno ponudnikov. Če bi imeli neomejeno resursov, bi seveda testirali večkrat v enem tednu in mogoče dobili lepše rezultate (manj deviacije), ampak razvrstitev ponudnikov bi verjetno ostala kar ista.



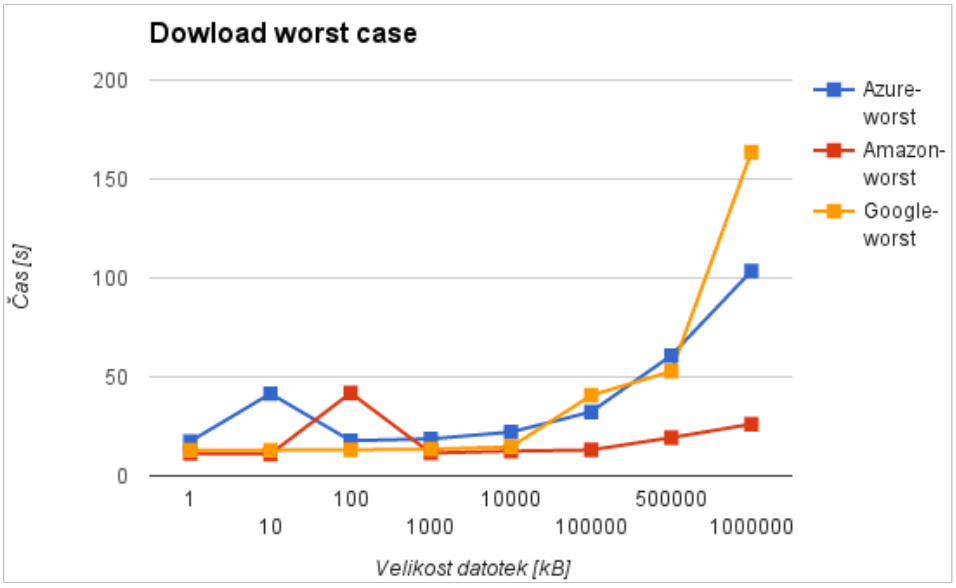
Slika 8: Graf standardne deviacije uploada ene datoteke.

Če pri download operaciji pogledamo najboljše primere testiranja za npr. 1GB datoteke (glej sliko 9), lahko opazimo, da je Amazonov praktično enak kot povprečje, kar je precej presenetljivo, ker sta se za to breme ostala dva uporabnika izkazala precej slabše. Pri Azure je najboljši primer le za nekaj sekund hitrejši od povprečja. Googlov najboljši primer pa se zelo razlikuje od povprečja, in sicer kar za dobrih 50 sekund. To je znova dokaz Googlove nekonsistentnosti, ki se izraža tudi pri najslabšem primeru.

Za najslabše primere download testiranja za 1GB velike datoteke na sliki 10 opazimo, da je Amazon znova tik ob povprečju, kar glede na najboljši primer ni presenečenje. Azure je nekoliko slabši, ampak še vseeno ni tako daleč od povprečja (približno 10-15 sekund). Google je znova precej zgrešil povprečje. Ker smo teste opravljali čez vikend (zaradi manj prometa - ugotovitev lanske 6. skupine (glej vir 6), težko določimo razlog za tako veliko odstopanje. Mogoče so takrat opravljali kakšna vzdrževalna dela na strežniku, ali pa je to posledica kakšnega drugega dogodka na katerega nismo računali. Dodatna možna razlaga je, da Google enostavno ponuja nekonsistenten sistem za svoje brezplačne storitve in je mogoče stanje pri plačljivi verziji drugačno. Standardna deviacija močno potrjuje naš sklep o Googlovi nezanesljivosti.



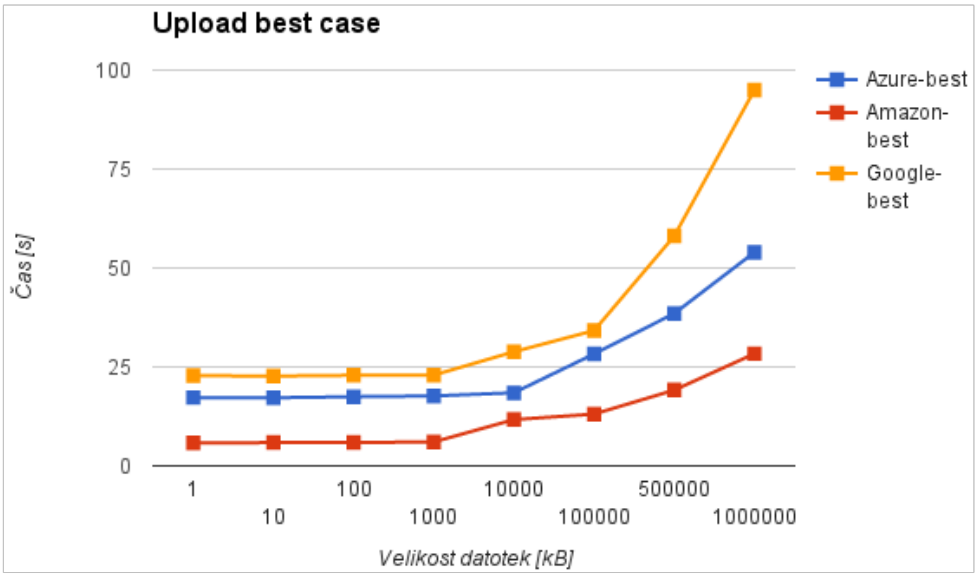
Slika 9: Graf najboljših primerov download-a.



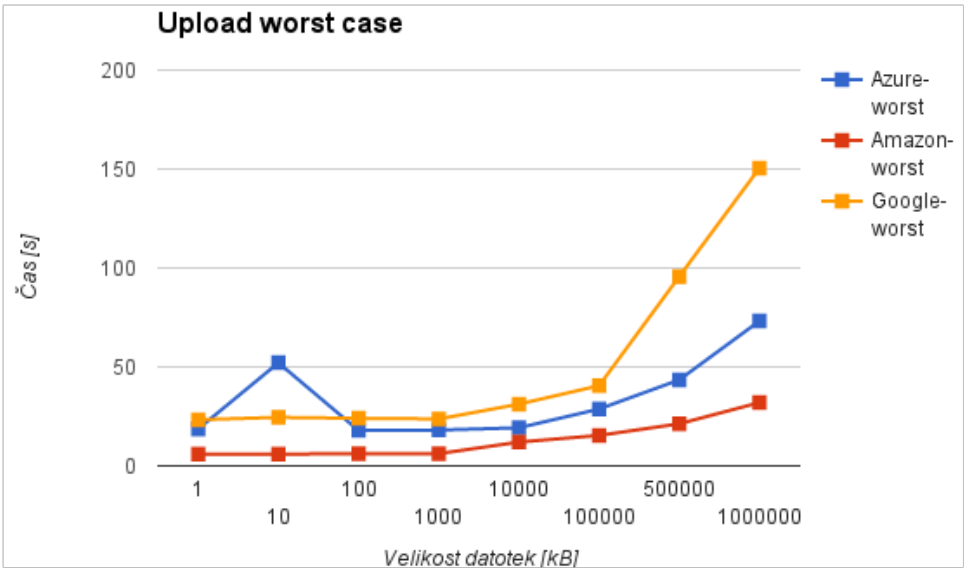
Slika 10: Graf najslabših primerov download-a.

Pri najboljših primerih upload operacije (glej sliko 11) za datoteke velike 1GB, lahko pri vseh potegnemo enake ugotovitve kot pri najboljših primerih download operacije. Trend se že ves čas ponavlja. Amazon zelo konsistenten, Azure malo manj, Google zelo nekonsistenten.

Za najslabše primere upload operacije (glej sliko 12) lahko prav tako opazimo podobnost z našo analizo najslabših primerov download operacije. Tukaj lahko potegnemo končno črto, da Googlova storitev enostavno ne ponuja zanesljivosti, saj je enako kot pri download operaciji tudi tukaj standardna deviacija ogromna v primerjavi z Azure in Amazon. To nam pokažeta tudi najboljši in najslabši čas Googlovega uploada, saj se razlikujeta za dobrih 50 sekund.



Slika 11: Graf najboljših primerov upload-a.

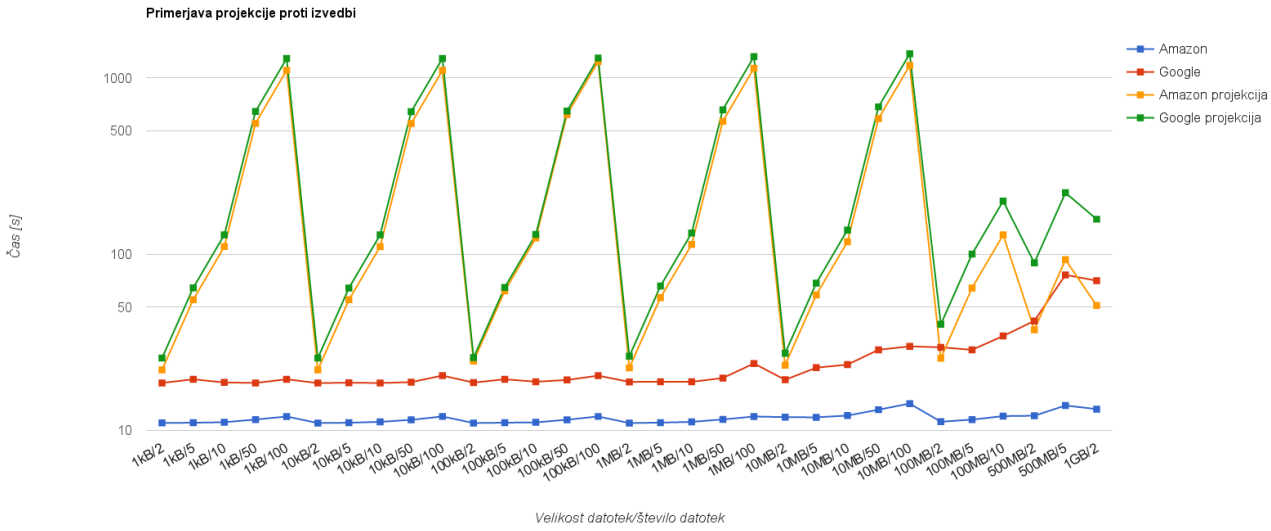


Slika 12: Graf najslabših primerov upload-a.

4.6.2 Testiranje paralelnega prenosa

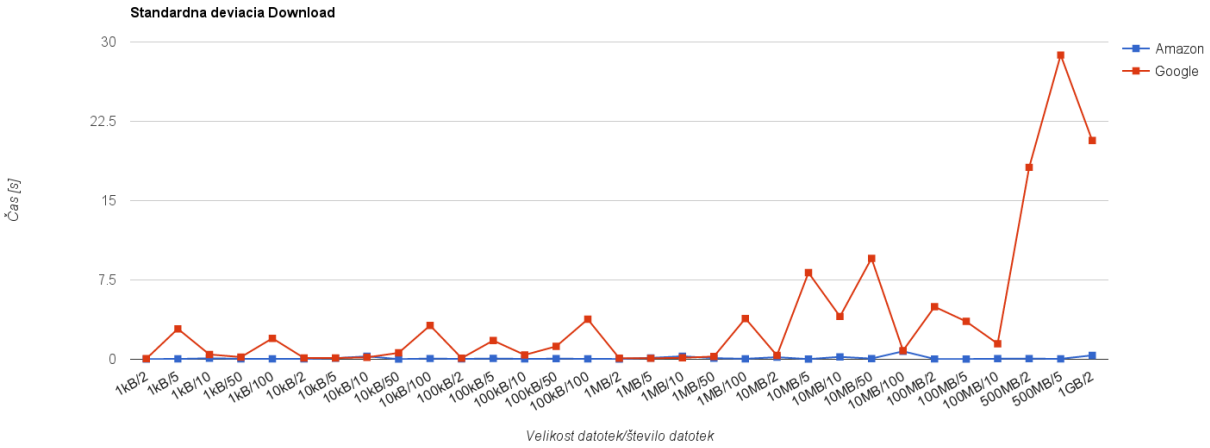
Najprej smo izvedli testiranje na eni datoteki naenkrat, zato smo ob pričetku testiranja paralelnega prenosa postavili hipotezo, da bo Amazon ponovno prevzel vodstvo in smo predvidevali da bo čas prenosa linearno naraščal s številom datotek. Za konec smo naredili še analizo vseh paralelnih testov skupaj, ker nas je zanimalo, koliko bi to vplivalo na časovno skalo. Tudi pri paralelnemu prenosu se je izkazalo, da je Amazon hitrejši od Googla, kar pomeni da smo potrdili prvi del hipoteze.

Pri povprečnem downloadu večih datotek (glej sliko 13), nas je Amazon presenetil z skoraj vodoravno krivuljo ne glede na velikost. Tukaj je najverjetneje problem pri *AWS CLI* orodju in ukazu `sync` (glej vir 7).



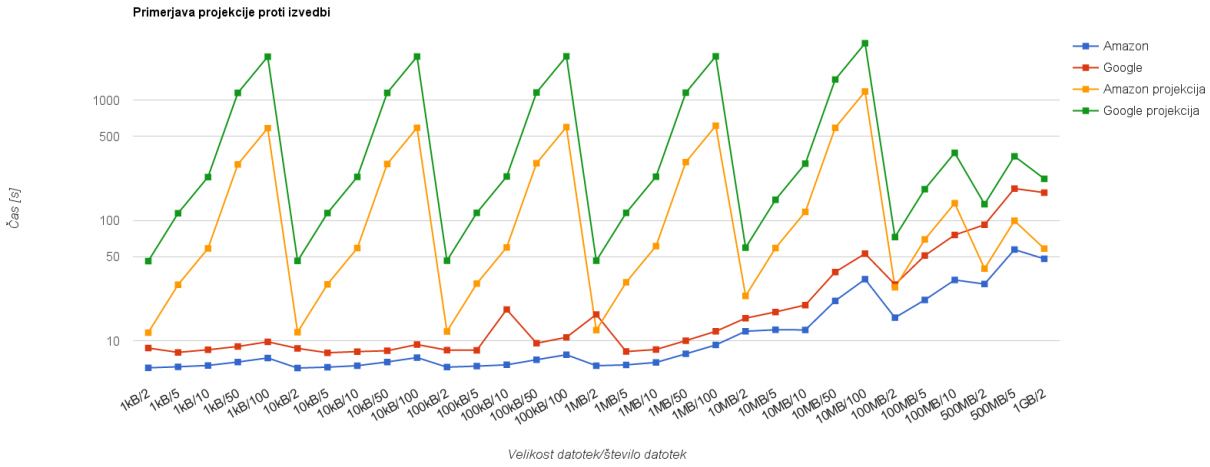
Slika 13: Graf povprečnega downloada večih datotek naenkrat in napoved rezultatov.

Pri standardni deviaciji za download večih datotek (glej sliko 14), ponovno pride do izraza nekonsistentnost download časov Googla, podobno kot pri testih enkratnega prenosa. Medtem, ko lahko pri Amazonu opazimo kako enovit je bil download čas ukaza `sync` in odstopanja praktično ni.



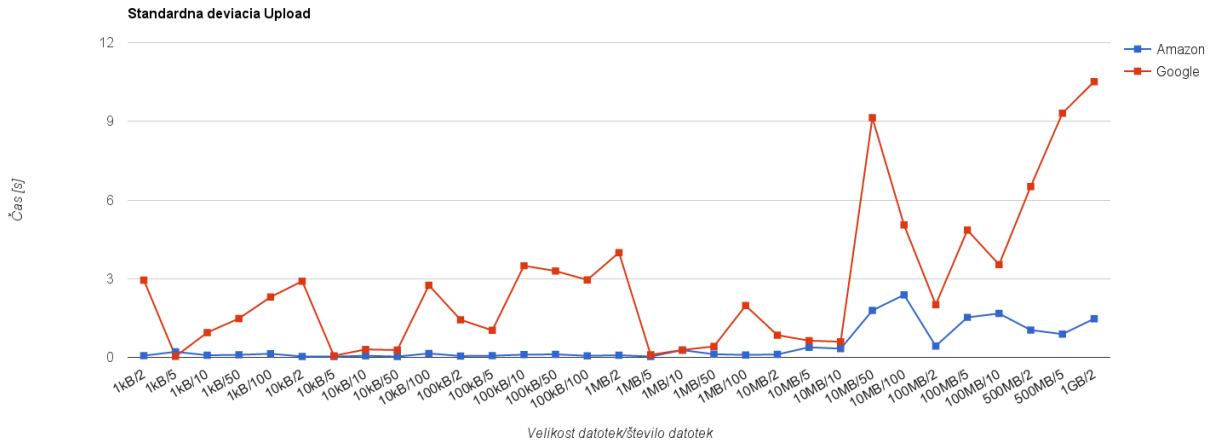
Slika 14: Graf standardne deviacije downloada večih datoteke naenkrat.

Pri testiranju drugega dela hipoteze (linearna povezanost med časom in številom datotek) se je izkazalo, da oba ponudnika izkoriščata paralelni prenos. Iz slike 15 je razvidno da naša predpostavka o sekvenčnem prenosu ne drži, pravzaprav naši ponudniki svoje delo opravljajo mnogokrat hitreje kot smo napovedali.



Slika 15: Graf povprečnega uploada večih datotek naenkrat in napoved rezultatov.

Zgodba se pri standardni deviaciji uploada večih datotek ponovi (glej sliko 16), saj spet pride do nihanj pri Googlovi storitvi. Pri Amazonu je do malo večjega odstopanja prišlo le pri večjih datotekah, a le pri večjem številu uploadanih datotek.



Slika 16: Graf standardne deviacije uploada večih datoteke naenkrat.

4.6.3 Metrika zmogljivosti

Iz naših grafov in analiz je razvidno, da je ponudnik Microsoft Azure performančno najbolj povprečen ter cenovno najugodnejši, zato smo se odločili, da se za analizo metrike zmogljivosti osredetočimo le na tega ponudnika.

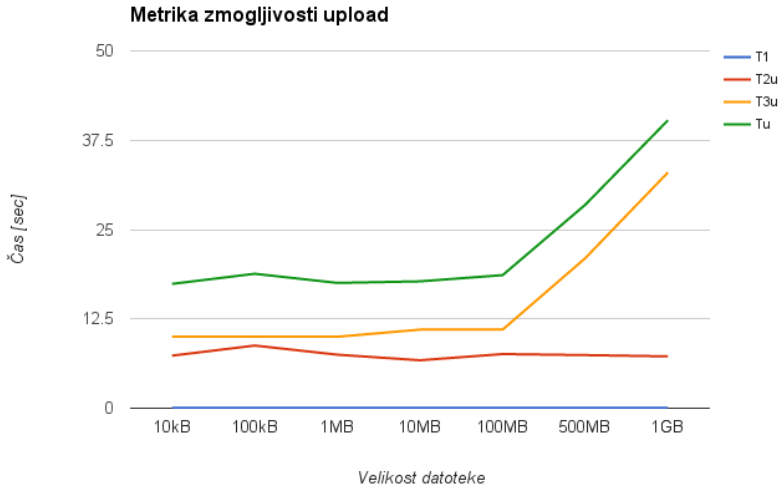
Merili smo čas prenosa datoteke med oblakom ter uporabnikom, ki ga lahko predstavimo z enačbo $T = T_1 + T_2 + T_3$. Sestavljajo jo sledeči parametri:

- T_1: čas dostopa do storitve,
- T_2: čas procesiranja zahteve,
- T_3: čas prenosa datoteke med oblakom in uporabnikom (upload / download).

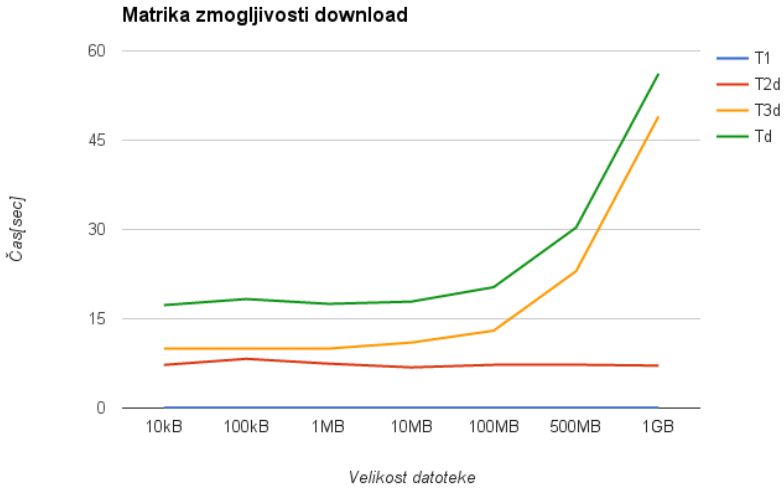
Za testna bremena smo vzeli enake datoteke kot pri ostalih testiranjih in za njih izmerili povprečni čas prenosa v obeh smereh.

T_1 smo v grobem ocenili glede na povprečno latenco, ki smo jo dobili iz Microsoft Azure CloudHarmony benchmark rezultatov (glej sliko 3). Rezultat je bila izredno majhna vrednost in sicer $T_1 = 43.5ms$. Bolj bi bilo zanesljivo, če bi čas latence pridobili sami, a Azure zaradi strožje varnosti ne odgovarja na *ICMP* (angl. *Internet Control Message Protocol*) pakete, zato smo tukaj prisiljeni zaupati CloudHarmony-ju.

Pri prenosu datotek nam ponudnik pošlje oceno časa T_3. Ker lahko merimo le skupni čas T_D (download) in T_U (upload), smo morali izračunati čas T_2 po enačbi $T_2 = T - (T_1 - T_3)$. Iz grafov na sliki 17 in 18 je razvidno, da sta časa T_1 in T_2 bolj ali ne konstantna, medtem, ko čas T_3 pri povečevanju velikost datotek (a šele pri 100MB) eksponentno narašča.



Slika 17: Graf metrike zmogljivosti za upload



Slika 18: Graf metrike zmogljivosti za download

5 Zaključek

Iz vseh rezultatov testiranja je jasno razvidno, da je Amazon vsekakor na prvem mestu, kar se tiče hitrosti. Za uporabnika, ki pri oblačnih storitvah za hranjenje podatkov potrebuje predvsem hitrost in časovno konsistentnost, denar pa ni ovira, je prava izbira Amazon. Če pa nam je vseeno, koliko časa se datoteke prenašajo, ampak nas zanima samo koliko časa lahko storitev uporabljamo brezplačno, potem bi izbrali Azure, ki je najbolj radodaren z brezplačno verzijo.

Vsekakor bi lahko našo raziskavo zelo izboljšali, če bi razširili število ponudnikov (DropBox, Mega, ...) in pognali več raznoličnih testov. Zanimivo bi pa bilo poizkusiti tudi plačljive verzije, ker vsi trije ponudniki ponujajo hitrejše opcije prenosa, a seveda z večjo tarifo.

6 Viri

1. https://cloudharmony.com/
2. https://azure.microsoft.com/en-us/documentation/articles/storage-azure-cli/
3. https://cloud.google.com/storage/docs/gsutil
4. https://aws.amazon.com/cli/
5. http://serverfault.com/questions/346906/effect-of-distance-from-server-on-page-load-speed
6. http://lrss.fri.uni-lj.si/sl/teaching/zzrs/lectures/Recenzije/Sk_6_8.pdf [Dostopno: 02.05.2016]
7. http://docs.aws.amazon.com/cli/latest/reference/s3/sync.html