Analiza zmogljivosti oblačnih in strežniških storitev

Uredil prof. dr. Miha Mraz

Maj 2017

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 2016/2017 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 2017

Poglavje 1

Analiza zmogljivosti oblačne storitve Cloud9

Žiga Kokelj, Tadej Hiti,
Miha Bizjak, Matej Kristan

1.1 Opis problema

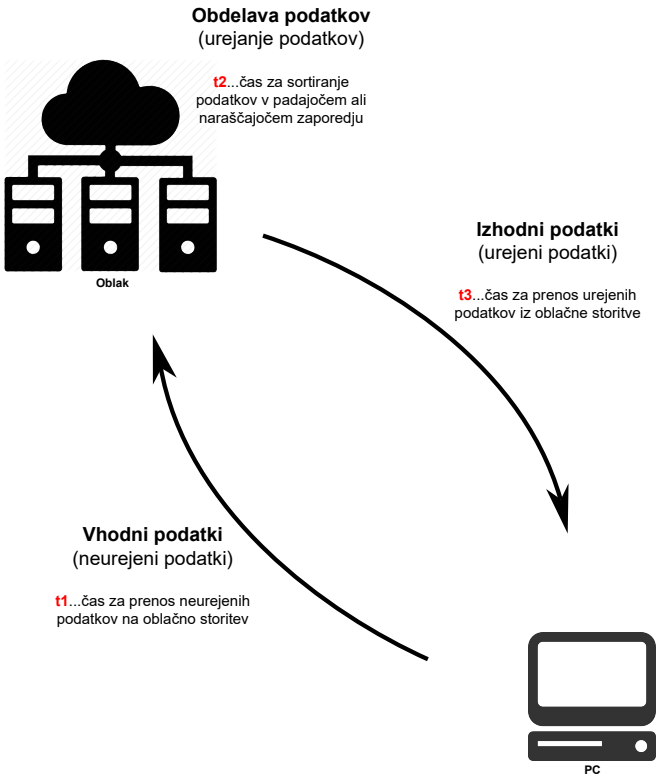
Današnje dni se uveljavljajo oblačne storitve, saj so s stališča uporabnika najenostavnejše za uporabo. Naša naloga je implementirati prenos datoteke na oz. z oblačne storitve in breme na oblačni storitvi, za katero smo si izbrali sortiranje numeričnih podatkov. Na sliki 1.1 je grafičen prikaz opisanega problema.

1.2 Namen

Naše testiranje bo obsegalo merjenje različnih izvajalnih časov na podlagi katerih bi prišli do podatkov o zmogljivosti sistema. Breme sistema bodo različni algoritmi sortiranja podatkov. Namen naše naloge bo ugotoviti zmogljivost zastonjske ponudbe z vidika različnih metrik zmogljivosti.

1.3 Izbira ponudnikov

Zaradi predhodnih izkušenj z oblačno storitvijo Cloud9 smo se odločili za njihovo zastonjsko ponudbo. Cloud9 ponuja razvojno okolje in operacijski sistem Ubuntu v katerem lahko pišemo ali pa izvajamo različne programe.



Slika 1.1: Shema delovanja sistema.

1.3.1 Cloud 9

Cloud9 je delavno orodje, ki je namenjeno programiranju v brskalniku. Torej vpišemo URL v brskalnik in že imamo vse pripravljeno za pisanje prvega programa. Pri zastonski narocnini nam dajo 512MB primarnega pomnilnika, in 2GB prostora na disku.

1.4 Izbira tehnologij

V tem razdelku so na kratko opisane vse izbrane tehnologije, ki smo jih realizirali sami za našo analizo.

1.4.1 Tehnologija v oblaku

V oblačni storitvi smo implementirali strežnik, ki je napisan v jeziku javascript z uporabo knjižnice Node.js [1], ki v odvisnosti od URL zahteve posreduje temu primerno datoteko na odjemalca. Strežnik poskrbi za prenašanje datotek in zagon potrebnega programa za sortiranje podatkov na strežniku.

1.4.2 Tehnologija za avtomatizacijo odjemalcev

Zaradi avtomatskega testiranja smo napisali tudi skripto v programskem jeziku python [2], ki omogoča avtomatsko pošiljanje datoteke in URL zahteve na strežnik, ter kot odgovor prejme urejeno datoteko z urejenimi podatki. Seveda pa ob tem še zabeležimo čas pred pošiljanjem zahteve in čas po prejetju urejene datoteke, da dobimo izvajalni čas celotne procedure. Ker je odjemalcev lahko večje število, smo ta problem rešili z nitmi, kjer vsaka nit predstavlja enega odjemalca in pošilja zahteve na strežnik.

1.5 Testiranje oblačne storitve

V tem razdelku je opisan način testiranja naše implementacije.

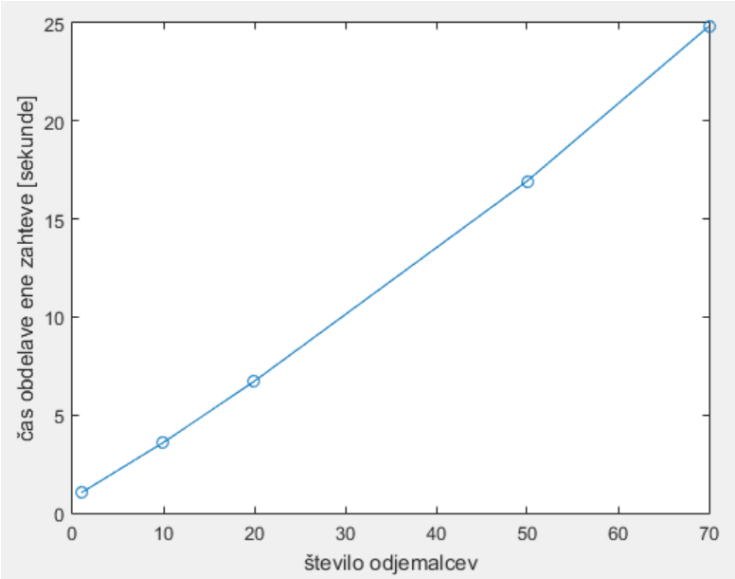
1.5.1 Opis načina testiranja

Kot smo opisali že zgoraj v razdelku 1.1 bomo pošiljali na strežnik datoteke z neurejenimi podatki in čakali na prejem datotek iz strežnika z urejenimi podatki. Testirali bomo z različnimi:

- Števili odjemalcev
- Števili podatkov za urejanje
- Algoritmi urejanja podatkov
- Tipi podatkov, ki jih urejamo
- Strategija pošiljanja(čakaj na odgovor/ne čakaj na odgovor)

Število odjemalcev	Čas obdelave
1	1.041
10	3.597
20	6.721
50	16.930
70	24.832

Slika 1.2: Tabela časov obdelav datotek z 10 tisoč integer števili.



Slika 1.3: Graf časa v odvisnosti od števila odjemalcev.

1.5.2 Testiranje I.

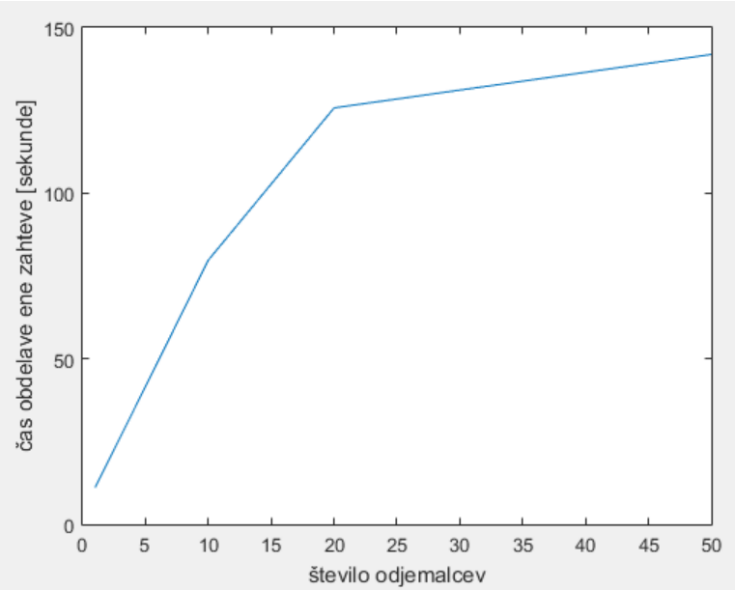
Naključno smo generirali nekaj datotek z 10000 integer števili, ki jih je nato 1/10/20/50/70 odjemalcev pošiljalo na strežnik eno za drugo istočasno, in preden pošlje novo počaka, da dobi urejeno datoteko kot odgovor. Izmerili smo čase potrebne za prejetje urejene datoteke in prišli do povrečnih časov. Testiranje je bilo izvedeno okoli popoldneva. Vse meritve so prikazane na sliki 1.3 in v tabeli 1.2.

1.5.3 Testiranje II.

Naključno smo generirali nekaj datotek z 50000 integer števili, ki jih je nato 1/10/20/50/70 odjemalcev pošiljalo na strežnik eno za drugo istočasno, in preden pošlje novo počaka, da dobi urejeno datoteko kot odgovor. Izmerili smo čase

Število odjemalcev	Čas obdelave
1	11.118
10	79.759
20	125.738
50	141.873
70	ERROR

Slika 1.4: Tabela časov obdelav datotek z 50 tisoč integer števili.



Slika 1.5: Graf časa v odvisnosti od števila odjemalcev.

potrebne za prejetje urejene datoteke in prišli do povrečnih časov. Vse meritve so prikazane na sliki 1.5 in v tabeli 1.4. Testiranje je bilo izvedeno okoli popoldneva. Pri testu z 70 odjemalci je prihajalo do predolge obdelave podatkov na strežniku in je python skripta javila napako, da se strežnik ne odziva.

Literatura

[1] "Node.js." Dosegljivo: https://nodejs.org/en/.

[2] "Python." Dosegljivo: https://www.python.org/.