Elektrotehnički fakultet,
Univerzitet u Beogradu

04.07.2012

Performanse računarskih sistema (SI4PRS)

domaći zadatak, jun/jul 2012

Student:
Aleksandar Abu-Samra, 252/08

Uvod

Ovaj izveštaj predstavlja deo izrade domaćeg zadatka iz predmeta SI4PRS i zamišljen je kao propratna dokumentacija koja detaljno opisuje metode simulacije i analitičkog rešavanja problema korišćenih u programskom delu izvođenja ovog zadatka.
Za zatvorenu mrežu modeliranog računarskog sistema potrebno je odrediti parametre kao što su iskorišćenje resursa, protok kroz resurse, prosečan broj poslova na čekanju na svakom resursu, i vreme odaziva sistema. Metode određivanja traženih parametara su pokretanje projektovane simulacije i analitička metoda Bjuzenovim algoritmom, nakon čega je potrebno odrediti relativno odstupanje između dobijenih rezultata.

Startovanje programa, klasa *AnalizaPerformansi*

Ovo je glavna, koordinišuća klasa sistema. Nakon preuzimanja ulaznog parametra i određivanja stepena multiprogramiranja, ovde se pozivaju klase za simulaciju, Bjuzenovu metodu i određivanje odstupanja.
Po zahtevu zadatka, metode klasa se pozivaju za više hardverskih postavki, konkretno za verzija od K korisničkih diskova, gde K iterira od 2 do 8.

Metod rada simulacije

Glavna logika metode rada simulatora se nalazi u klasi Simulator, izvedenoj iz apstraktne klase Analiza.
Pri kreiranju objekta klase se setuje stepen multiprogramiranja i određuje log fajl u kojem će biti ispisani rezultati. Dalje je odgovornost glavnog programa da šalje hardverske scenarije i pokreće obradu na simulatoru.
Algoritam bi mogao da se opiše u nekoliko koraka:

- Poziv metode novaAnaliza(int), sa brojem korisničkih diskova kao argumentom
    - Simulacija se resetuje na početno stanje
- Poziv metode izvrsiAnalizu(), bez argumenata. Ovo je apstraktna metoda koju klasa Simulator implementira na sledeći način:
    - Prave se prazne liste procesa na čekanju
    - Iteracija od vremenskog trenutka 0 do zadatog vremena trajanja simulacije
        - Za svaku komponentu sistema, poziva se tick() metoda koja prati protok vremena i statističke podatke na komponenti
        - U slučaju da je komponenta završila ciklus obrade, uzima se jedan proces iz njenog reda i stavlja na listu čekanja za sledeću iteraciju
        - Nakon prolaska kroz sve komponente, prazne se liste čekanja tako što se slučajnom raspodelom stavljaju u redove komponenti gde čekaju sledeću obradu
- U posebnoj matrici se čuva stanje svih komponenti nakon izvršenog koraka simulacije. Ovo će biti korišćeno pri određivanju relativnog odstupanja rezultata
- Poziv metode snimiLog(int) koja je zadužena za ispisivanje završenog računa u log fajl

Bjuzenova metoda

Klasna struktura korišćena pri izvršavanju Bjuzenovog algoritma je analogna klasnoj strukturi korišćenoj kod Simulatora, pa je u daljem tekstu akcenat stavljen na suštinske razlike.

Programiranje analitičke metode određivanja performansi sistema je podrazumevalo korišćenje dobro poznatih Gordon-Newellovih i Bjuzenovih algoritama.

U ovoj konkretnoj implementaciji, Gordon-Newellovov algoritam je izračunat van koda ovog programa i hardkodovan u njegovu strukturu. Pošto dotični algoritam zahteva rešavanje sistema jednačina, ovakva odluka značajno ubrzava vreme razvoja, a pritom nimalo ne remeti fleksibilnost zadatka. Tako smo dobili vrednosti niza x[] koji dalje koristi Bjuzenov algoritam.

Dalji postupak predstavlja programabilno korišćenje dobro poznatih Bjuzenovih formula, koje zbog svoje kompleksnosti i teškoće razumevanja van konteksta neće biti opisane ovde. Dokument sa detaljnim opisom algoritma se može naći na veb straici http://rti.etf.rs/rti/prs/materijali/.

Nakon analitičkog izračunavanja svih potrebnih parametara, oni se ubacuju u “Dummy” komponente, koje služe isključivo za smisleno držanje relevantnih parametara, lakše prikazivanje i korišćenje već gotovih klasa za ispisivanje u log fajlove.

Po završetku, i ovde se stanje svih komponenti čuva u posebnoj matrici, a rezultati smeštaju u log.

Određivanje relativnog odstupanja

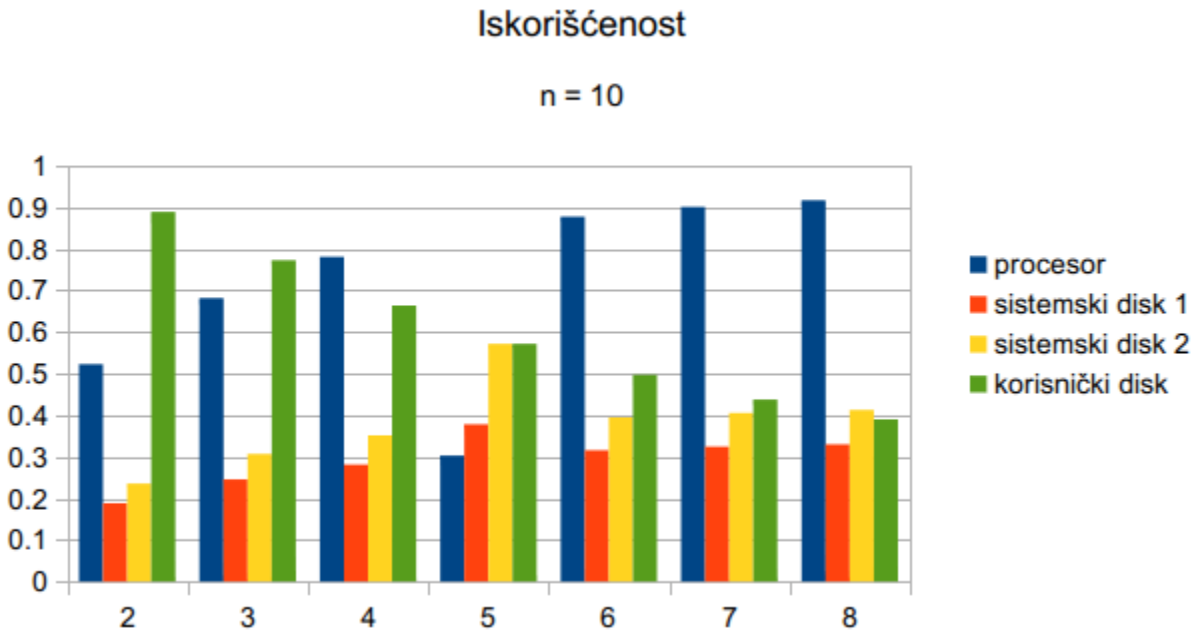
Pri određivanju relativnog odstupanja rezultata između korišćenih metoda analize, koriste se matrice komponenata koje su u tim procesima bile formirane. I ovaj metod koristi već postojeći set klasa i koristi klasu Komponenta da u nju smešta svoje rezultate.

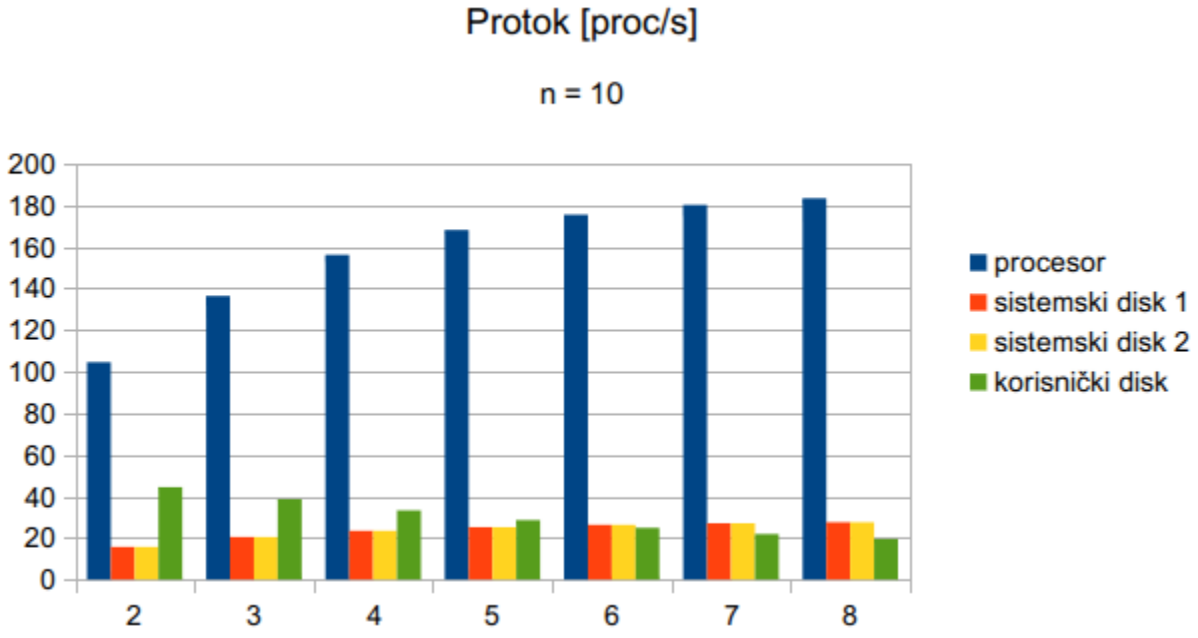
Za svaki podatak ponaosob, određuje se procenat odstupanja rezultata Bjuzenove metode od reultata simulacije. Ti podaci se smeštaju u nove komponente, a one se zatim u "Dummy" formi ubacuju u objekat klase RelativnoOdstupanje i odatle standardno snimaju u log fajlu.

Primer generisanog log fajla sa koeficijentima relativnog odstupanja

```
▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫ TEST 1 (K = 2) ▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫
stepen multiprogramiranja: 20
vreme odaziva sitema: 0.0520 ms

              komponenta |    iskorišćenje | protok [proc/s] |  prosek poslova
=========================================================================================
*                Procesor |          0.1880 |          0.0494 |          1.2213
         Sistemski disk 0 |          0.0080 |          0.0491 |          0.1436
         Sistemski disk 1 |          0.0172 |          0.0503 |          0.2120
        Korisnički disk 0 |          0.0256 |          0.0500 |          0.0476
        Korisnički disk 1 |          0.0242 |          0.0486 |          0.0341

▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫ TEST 2 (K = 3) ▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫▫
stepen multiprogramiranja: 20
vreme odaziva sitema: 0.0930 ms

              komponenta |    iskorišćenje | protok [proc/s] |  prosek poslova
=========================================================================================
*                Procesor |          0.1436 |          0.0851 |          2.0817
         Sistemski disk 0 |          0.0449 |          0.0845 |          0.1393
         Sistemski disk 1 |          0.0195 |          0.0847 |          0.2204
        Korisnički disk 0 |          0.0620 |          0.0855 |          0.1307
        Korisnički disk 1 |          0.0615 |          0.0850 |          0.1173
        Korisnički disk 2 |          0.0613 |          0.0848 |          0.1235
```

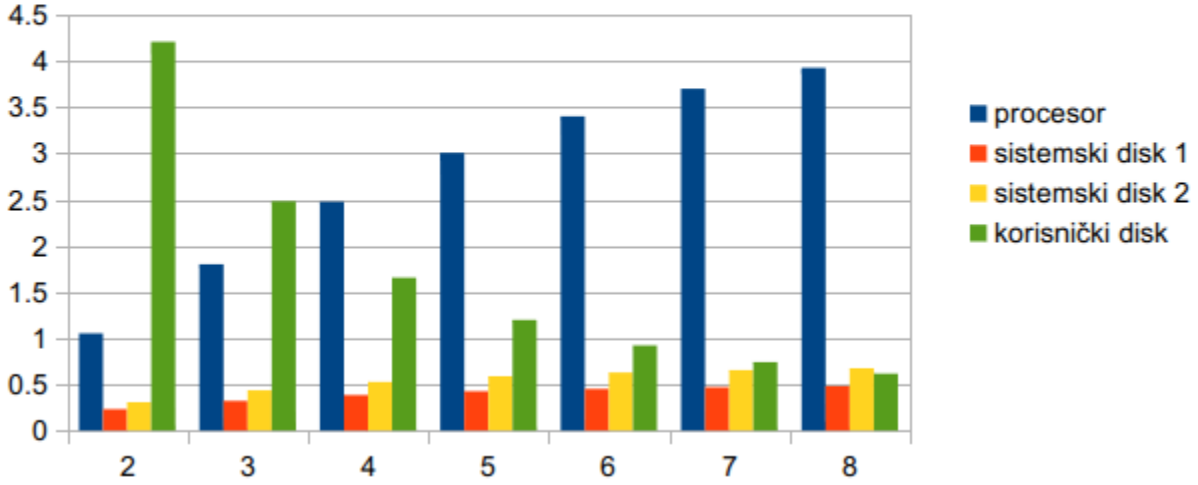
Grafički prikaz performansi (histogrami)





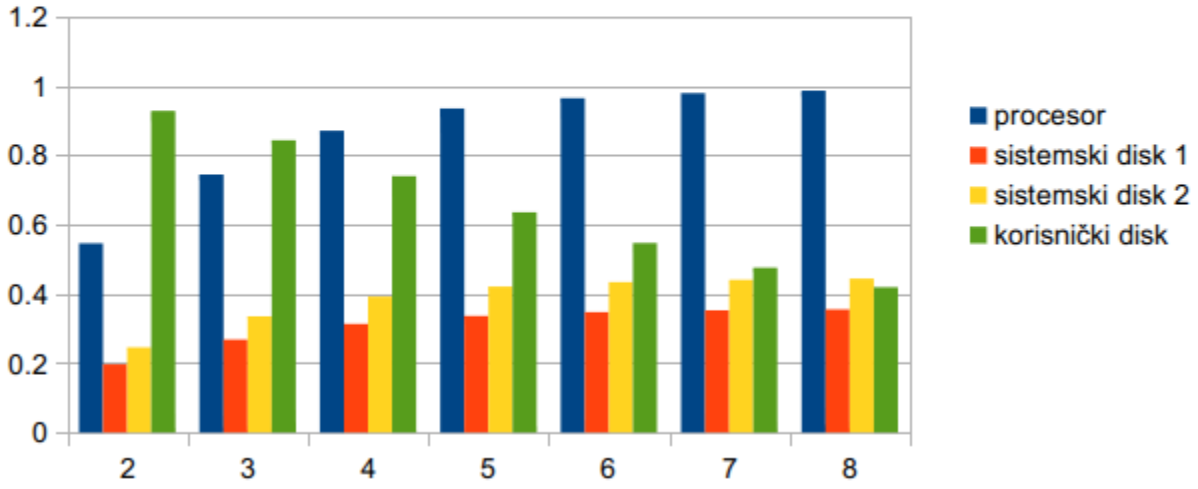
Prosečni broj poslova

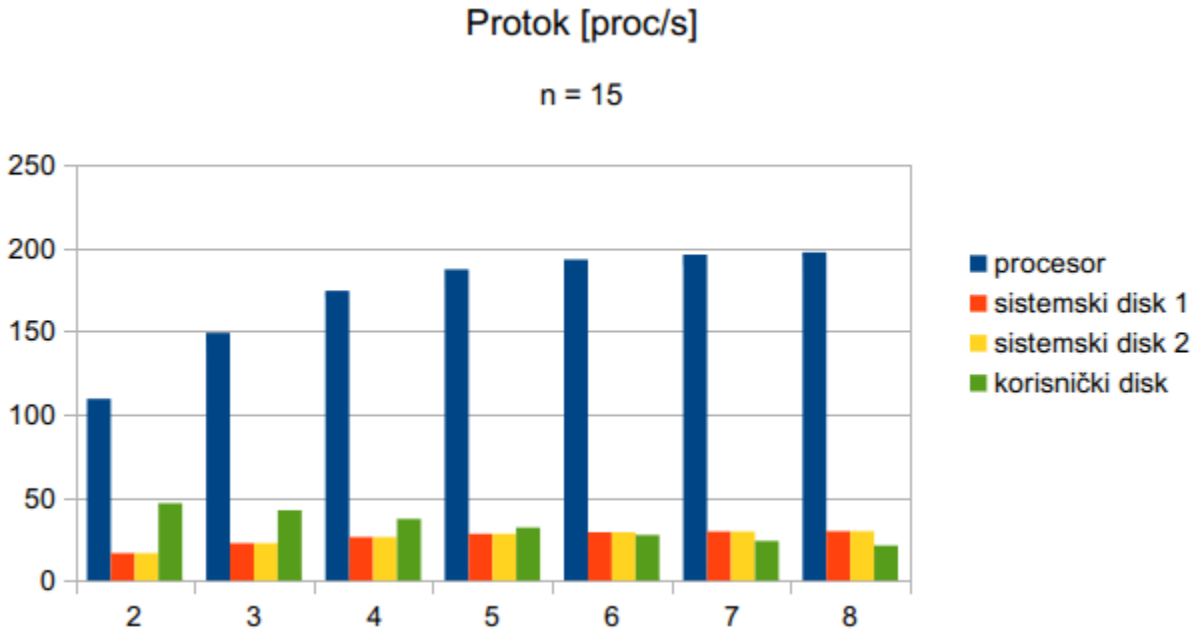
n = 10

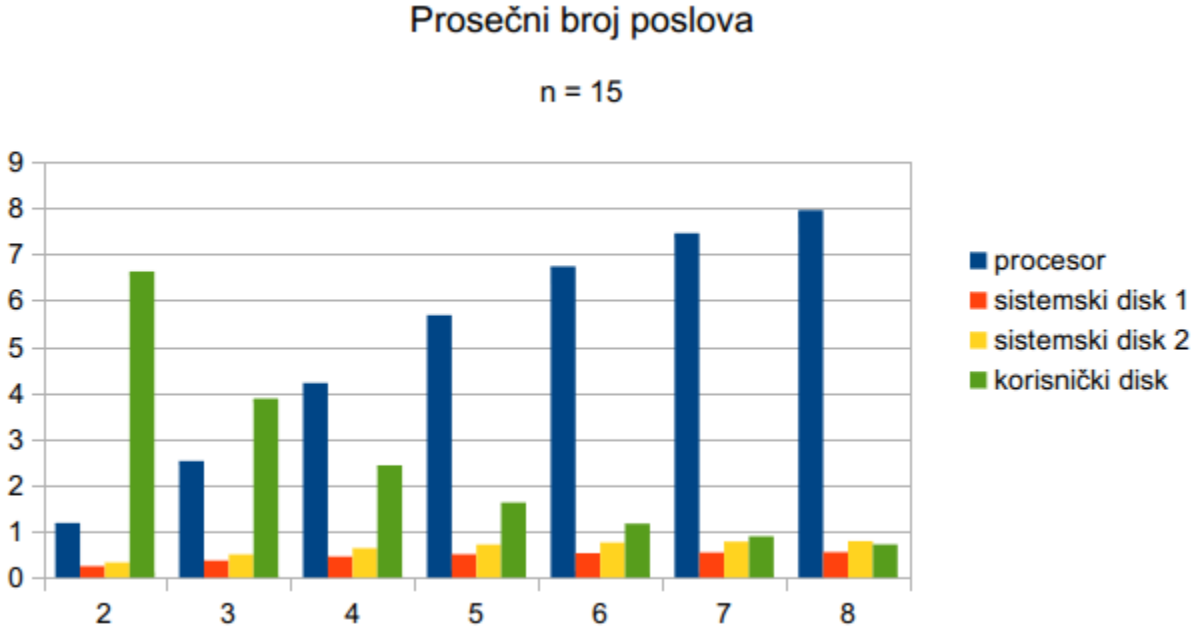


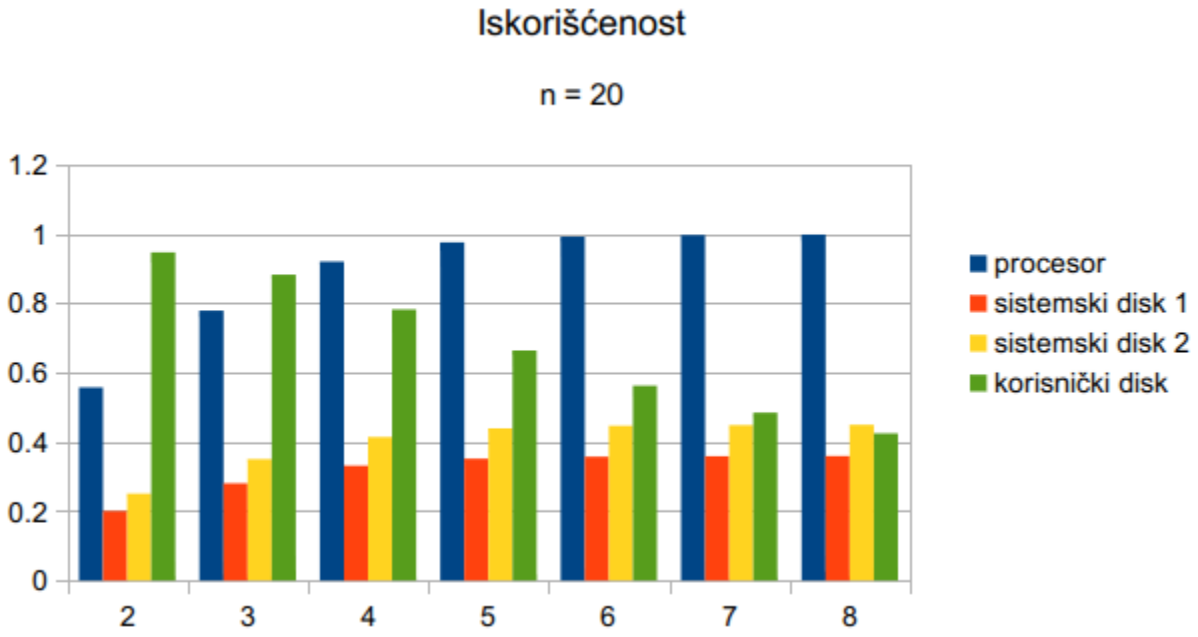
Iskorišćenost

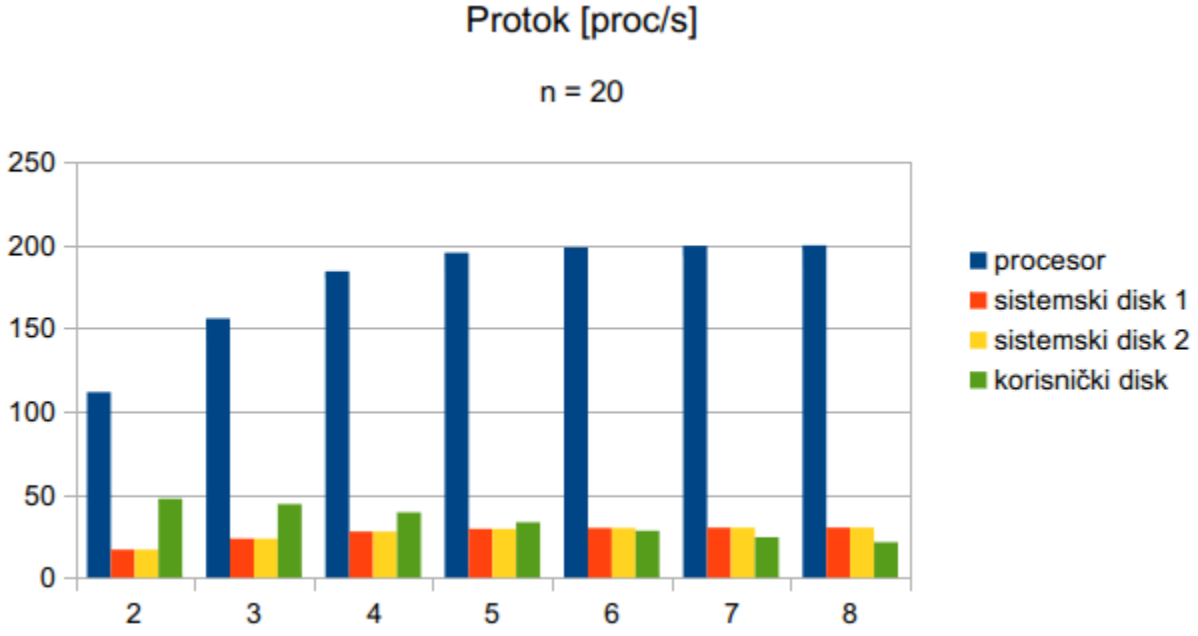
n = 15

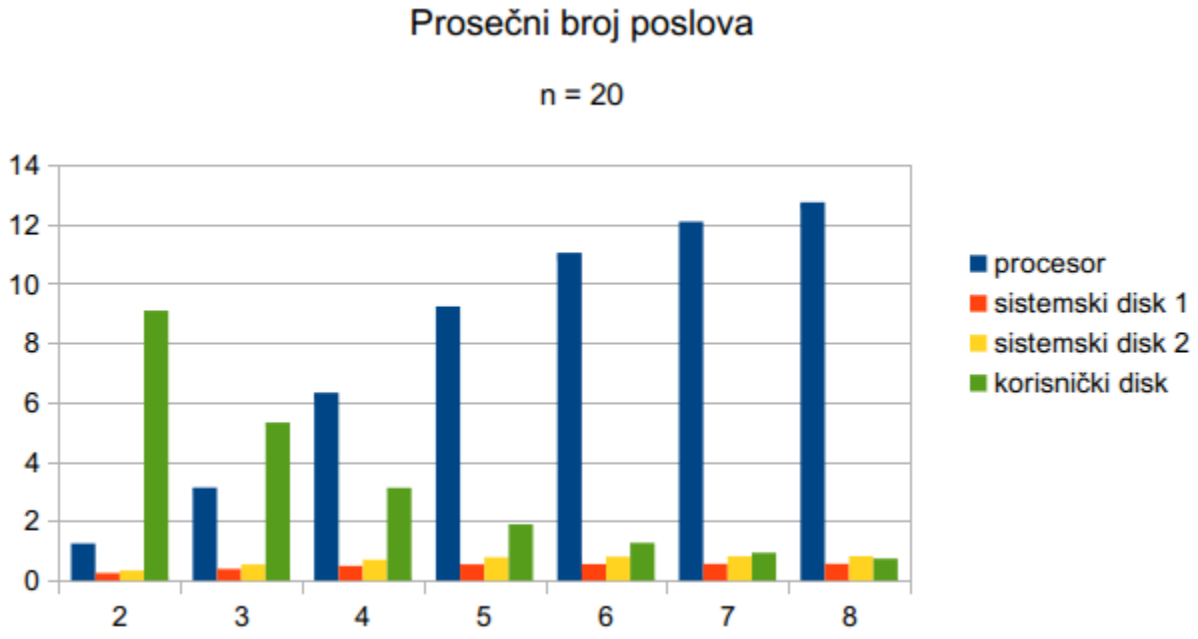



Protok [proc/s]
n = 15
250
200
150
100
50
0
2
3
4
5
6
7
8
procesor
sistemski disk 1
sistemski disk 2
korisnički disk


Prosečni broj poslova
n = 15
9
8
7
6
5
4
3
2
1
0
2
3
4
5
6
7
8
procesor
sistemski disk 1
sistemski disk 2
korisnički disk


Iskorišćenost
n = 20
1.2
1
0.8
0.6
0.4
0.2
0
2
3
4
5
6
7
8
procesor
sistemski disk 1
sistemski disk 2
korisnički disk


Protok [proc/s]
n = 20
250
200
150
100
50
0
2
3
4
5
6
7
8
procesor
sistemski disk 1
sistemski disk 2
korisnički disk


Prosečni broj poslova
n = 20
14
12
10
8
6
4
2
0
2
3
4
5
6
7
8
procesor
sistemski disk 1
sistemski disk 2
korisnički disk