

# RZNU MI 2019/2020

**Cache:**

**1.) Ako je u cache spremljen objekt i poslužitelj javi da objekt nije promijenjen, koji kod vraća:**

**a) poslužitelj -> cache**

304 Not Modified. (SOC\_03 slide 171)

**b) cache -> klijent**

200 OK

**2.) Ako je u cache spremljen objekt i poslužitelj javi da je objekt promijenjen, koji kod vraća:**

**a) poslužitelj -> cache**

200 OK. (SOC\_03 slide 172)

**b) cache -> klijent**

200 OK. (?)

**3.) Ako je u cache spremljen objekt i poslužitelj javi da je objekt izbrisan, koji kod vraća:**

**a) poslužitelj -> cache**

404 Not Found. (SOC\_03 slide 174)

**b) cache -> klijent**

Pretpostavljam isto 404 Not Found. (?) Nakon što cache dobije 404, on ga obriše iz memorije pa pretpostavljam da onda isto odgovori 404 Not Found.

**4.) Razlika između koda 204 i 304?**

204 se koristi kada nemamo ništa za poslati u body-ju, dok 304 kada imamo informaciju o stanju vezanu za resurs, ali klijent već ima najnoviju (most recent) verziju te reprezentacije.

**5.) Ako je sustav skalabilan i sa 100 jedinica obrađuje 2000 procesa, koliko će procesa obrađivati sa dodatnih 50 jedinica?**

100 jedinica = 100%

$100 + 50 = 100\% + 50\% = 1.5$

$3000 = (2000 * 1.5) ??$

**6.) Statelessness svojstvo smanjuje pritisak na poslužitelju, a povećava na klijentu.**

**7.) Dana je json poruka i pita koji je Content-Type.**

Content-Type: application/json.

**8.) Na kojem URL-u je stvoren objekt ako je poslan POST zahtjev na http://nesto/x/y?**

Stvoren je objekt na http://nesto/x/y/{id}.

**9.) Na kojem URL-u je stvoren objekt ako je poslan PUT zahtjev na <http://nesto/x/y>?**

Napravljen je update objekta na <http://nesto/x/y>.

**10.) Na kojem URL-u je stvoren objekt ako je poslan GET zahtjev na <http://nesto/x/y>?**

Nije stvoren objekt.

**11. a) Na poslužitelj se šalje PUT zahtjev na URL gdje nema resursa. Što će napraviti poslužitelj?**

PUT će stvoriti novi resurs (SOC\_03 slide 120).

**b) Koji kod vraća klijentu?**

Vraća kod 201 Created.

**12. a) Na poslužitelj se šalje PUT zahtjev ako je kolekcija na URL gdje ima resursa. Što će napraviti poslužitelj?**

Napravit će update trenutnog resursa.

**b) Koji kod vraća klijentu?**

Vraća kod 200 OK.

**13. Hit rate i trošak (8 bodova)**

Dan je sustav sa cache memorijom i poslužiteljem. Dano je nekoliko (npr. 18) veličina bodyja raznih zahtjeva u B. Odgovori nekih od tih zahtjeva su već u cache (podcrtani - u zadatku je bilo 18 brojeva, a njih 8 je bilo podcrtano. Ti podcrtani su značili cache hit.), a ostali se moraju dohvaćati sa poslužitelja. Dan je trošak obrade transakcije na poslužitelju (P) i trošak prijenosa 1000B preko mreže (M). Veličina headera zahtjeva je 500 B. Bili su razni brojevi od cca 200B do 200 000B.

**a) Izračunaj cache-hit ratio.**

$\text{cache\_hit\_ratio} = \text{num\_of\_cache\_hits} / \text{num\_of\_requests}$

$\text{num\_of\_cache\_hits} = \text{broj podcrtanih} = 8$

$\text{num\_of\_request} = 18$

(provjerite ovo pls -> Točan je postupak, 8/18)

**b) Izračunaj byte cache-hit ratio.**

Znam da su mi tu na uvidima rekli da ima neka caka sa headerima i njihovom veličinom.

**Može netko objasniti kako se ovo računalo sa tim headerima?**

$\text{byte\_hit\_ratio} = \text{size\_of\_bytes\_delivered\_from\_cache} / \text{total\_size\_of\_responses}$

$\text{byte\_hit\_ratio} = (\text{suma\_veličina\_podcrtanih} + \text{broj\_podcrtanih} * \text{header}) /$

$(\text{suma\_veličina\_svih} + \text{ukupan\_broj} * \text{header})$

**c) P = 0, M = 100. Izračunaj trošak.**

$\text{broj\_NEpodcrtanih} = 10$   
 $\text{trošak} = \text{suma byteova svih NEpodcrtanih} + \text{broj\_NEpodcrtanih} * \text{header (ili header} * 2 \text{ nisam siguran)} * M / 1000$

vjerujem da ti ide  $\text{header} * 2$  zato što imaš request i response.

**d)  $P = 0$ ,  $M = 100$ . Koji od ratia daje bolju procjenu troška?**

Bolji je  $\text{byte\_hit\_ratio}$ , treba objasniti zašto (dobije se 0 bodova ako ne objasnis koji), mislim da je bolji  $\text{byte}$  jer se  $\text{cache hit}$  uopće ne uzima u obzir.

**e)  $P = 100$ ,  $M = 0$ . Izračunaj trošak.**

$\text{trošak} = \text{broj\_NEpodcrtanih} * P$

**f)  $P = 100$ ,  $M = 0$ . Koji od ratia daje bolju procjenu troška?**

$\text{Cache\_hit\_ratio}$ . Isto kao d) samo obrnuto

**g)  $P = 100$ ,  $M = 100$ . Izračunaj trošak.**

$\text{trošak iz c)} + \text{trošak iz e)}$

**h)  $P = 100$ ,  $M = 100$ . Koji od ratia daje bolju procjenu troška?**

Bolji je onaj koji je veći jer ima puno veći utjecaj na cijenu od drugog (ne sjećam se napamet koji je bio veći).

**14. Bilo je zadano napisati URL-ove za sve predmete, sve profesore, određen predmet, određenog nastavnika, određenog profesora na predmetu i određeni predmet koji predaje nastavnik.**

Nešto u stilu [www.example.com/nastavnici](http://www.example.com/nastavnici) (JAKO BITNO da su nastavnici, ne nastavnik skidali su bodove ako je bilo u jednini.)

**NAPOMENA:** Bilo je zadano da se pojedinačni predmeti označavaju kao  $p1, p2, p3...$ , a pojedinačni nastavnici kao  $n1, n2, n3...$ . Tu nemojte "nasjesti" pa u URL-u označiti sve predmete kao  $p$ , a sve nastavnike kao  $n$ , nego koristite pune riječi u množini kao što je sugerirano iznad.

[www.example.com/predmeti](http://www.example.com/predmeti)

[www.example.com/profesori](http://www.example.com/profesori)

[www.example.com/predmeti/p1](http://www.example.com/predmeti/p1)

[www.example.com/nastavnici/n3](http://www.example.com/nastavnici/n3)

[www.example.com/nastavnici/n3/predmeti/p1](http://www.example.com/nastavnici/n3/predmeti/p1)

[www.example.com/predmeti/p1/nastavnici/n3](http://www.example.com/predmeti/p1/nastavnici/n3)

15. Konstruirati tablicu za http metode, u kojoj su označena njihova svojstva s obzirom na sigurnost(oni su rekli neškodljivost) i idempotentnost. (+ znači da je, - znači da nije)

**Neškodljivost - ako se pozivom dogodi isto kao da se nije ništa dogodilo**

**Idempotencija - ako više poziva ima isti učinak kao i jedan**

	Neškodljivost	Idempotencija
POST	-	-
GET	+	+
PUT	-	+
HEAD	+	+
OPTIONS	+	+
DELETE	-	+

16. Kojom metodom je preporučljivo provjeravati dostupnost URL-a?

Pretpostavljam HEAD ili OPTIONS. Može netko provjeriti?

Poprilično sam siguran da je HEAD jer nam OPTIONS vraća metode...

Da, HEAD je točno.

# RZNU ZI 2019/2020

**1. Preporučeni način za izvedbu potpore udomljavanju web sredstava na zajedničkoj poslužiteljskoj infrastrukturi (engl. Virtual web hosting ) je:**

- A. Raspoznavanje usluga po IP adresi
- B. Raspoznavanje usluga po izlaznom portu na klijentu
- C. Raspoznavanje usluga po pristupnom portu na poslužitelju
- D. Raspoznavanje usluga po Host zaglavlju u HTTP zahtjevu**
- E. Raspoznavanje usluga po Host zaglavlju u HTTP odgovoru

**2. Cjevovodni režim rada HTTP protokola (engl. HTTP pipelining) omogućava**

- A. Slanje velikih HTTP poruka u više mrežnih paketa
- B. Istodobno posluživanje više različitih korisnika
- C. Prijenos zvuka i videa HTTP protokolom
- D. Korištenje HTTP protokola za prijenos poruka drugih protokola
- E. Upućivanje HTTP zahtjeva prema poslužitelju bez čekanja HTTP odgovora na prethodno upućeni zahtjev**

**3. Ispravno oblikovanim REST sučeljem (engl. REST api) smatra se ono koje je klijentima izloženo na način**

- A. GET /deleteUser/1234
- B. GET /deleteUser?id=1234
- C. DELETE /deleteUser/1234
- D. DELETE /users/1234**
- E. POST /users/1234/delete

**4. Oblikovanje URL adresa za dva sredstva X i Y koji su u arhitekturnom stilu REST na istoj razini hijerarhije preporučljivo je izvesti na način**

- A. http://api.example.com/x za sredstvo X, http://api.example.com/x/y za sredstvo Y
- B. http://api.example.com/x za sredstvo X, http://api.example.com/y/x za sredstvo Y
- C. http://api.example.com/x/y za sredstvo X, http://api.example.com/y za sredstvo Y
- D. http://api.example.com/y/x za sredstvo X, http://api.example.com/y za sredstvo Y
- E. http://api.example.com/x za sredstvo X, http://api.example.com/y za sredstvo Y**

**5. X na višoj, Y na nižoj razini**

- A. http://api.example.com/x za sredstvo X, http://api.example.com/x/y za sredstvo Y**
- B. http://api.example.com/x za sredstvo X, http://api.example.com/y/x za sredstvo Y
- C. http://api.example.com/x/y za sredstvo X, http://api.example.com/y za sredstvo Y
- D. http://api.example.com/y/x za sredstvo X, http://api.example.com/y za sredstvo Y
- E. http://api.example.com/x za sredstvo X, http://api.example.com/y za sredstvo Y

**6. U sustavu s pravilnim periodičnim promjenama gdje je period promjene dugačak (primjerice svakih nekoliko sati), a količina korisnih podataka značajno veća od zaglavlja HTTP protokola, promjene s poslužitelja na klijente preporučljivo je izvesti**

**A. Tehnikom prozivanja(polling)**

B. tehnikom blokirajućeg prozivanja(long polling)

C. protokolom WebSocket

D. tehnologijom Server-Sent Events

E. izbor dojavne tehnike nema utjecaja na radna svojstva sustava

**7. Nakon početne uspostave veze, tehnologija SSE omogućava komunikaciju klijenta i poslužitelja koja je**

A. Jednosmjerna od klijenta prema poslužitelju

**B. Jednosmjerna od poslužitelja prema klijentu**

C. Dvosmjerna sinkrona

D. Dvosmjerna asinkrona

E. Nije omogućena izravna komunikacija klijenta i poslužitelja, već isključivo putem posrednika

**8. Nakon početne uspostave veze, protocol WebSocket omogućava komunikaciju klijenta i poslužitelja koja je**

A. Jednosmjerna od klijenta prema poslužitelju

B. Jednosmjerna od poslužitelja prema klijentu

C. Dvosmjerna sinkrona

**D. Dvosmjerna asinkrona**

E. Nije omogućena izravna komunikacija klijenta i poslužitelja, već isključivo putem posrednika

**9. Protokolom WebSocket moguće je prenositi**

A. Isključivo tekstualne poruke

B. Isključivo binarne poruke

**C. Tekstualne i binarne poruke**

D. Binarne izravno, tekstualne samo ako su prethodno kodirane u Base64 format

E. Tekstualne izravno, binarne samo ako su prethodno kodirane u Base64 format

**10. Protokolom SSE moguće je prenositi**

A. Isključivo tekstualne poruke

B. Isključivo binarne poruke

C. Tekstualne i binarne poruke

D. Binarne izravno, tekstualne samo ako su prethodno kodirane u Base64 format

**E. Tekstualne izravno, binarne samo ako su prethodno kodirane u Base64 format**

**11. U okviru standardnog WebSocket programskog sučelja čitanje poruka na prijemnoj strani izvodi se**

- A. Registracijom ugrađene rutine Web preglednika na događaj onmessage koja se poziva mehanizmom povratnog poziva
- B. Registracijom ugrađene rutine Web preglednika na događaj onmessage koja se poziva u petlji
- C. Registracijom korisničke funkcije na događaj onmessage koja se poziva mehanizmom povratnog poziva**
- D. Registracijom korisničke funkcije na događaj onmessage koja se poziva u petlji
- E. Automatskim pozivanjem ugrađene rutine Web preglednika, nakon čega se podaci čitaju iz varijable msg

```
/* Open a new WebSocket connection */
var ws = new WebSocket('ws://example.com/socket');

-----

/* A callback function invoked for each new message from the server */
ws.onmessage = function(msg) {
  if (msg.data instanceof Blob) {
    processBlob(msg.data);
  } else {
    processText(msg.data);
  }
}

-----

/* Client-initiated message to the server */
ws.send("Hello server! This is a text message for you.");
```

**12. Mehanizam integriranih usluga (IntServ) zasniva se na**

- A. Posluživanju po redu prispjeća
- B. Prioritetnom posluživanju s obzirom na vrstu mrežnog prometa
- C. Prioritetnom posluživanju s obzirom na identitet klijenta
- D. Prioritetnom posluživanju s obzirom na identitet poslužitelja
- E. Unaprijednoj rezervaciji sredstava**

**13. Mehanizam razlikovnih usluga (DiffServ) zasniva se na**

- A. Posluživanju po redu prispjeća
- B. Prioritetnom posluživanju s obzirom na vrstu mrežnog prometa**
- C. Prioritetnom posluživanju s obzirom na identitet klijenta
- D. Prioritetnom posluživanju s obzirom na identitet poslužitelja
- E. Unaprijednoj rezervaciji sredstava

**14. Glavna zadaća RPC sustava je**

- A. Pretvorba asinkronog poziva udaljene procedure u sinkroni poziv
- B. Vrednovanje semantike poziva funkcija u različitim adresnim prostorima
- C. Omogućavanje učinkovite serijalizacije u različitim adresnim prostorima
- D. Opisivanje semantike argumenata procedure doristeći IDL
- E. Normalizacija semantike poziva funkcija u istim i različitim adresnim prostorima**

**15. RPC tehnologija zasnovana na jeziku XML bez službenog IDL prevoditelja je: XML-RPC**

Bili su još ponuđeni CORBA (sigurno jer sam to krivo zaokružio :D) i (sigurno) Java RMI, SunRPC (možda) i još nešto sa XML-om u odgovoru.

**16.**

<pre>var p={   "firstName":"John",   "lastName":"Smith" };  var r=p.firstName;</pre>	<pre>var p='{   "firstName":"John",   "lastName":"Smith" }';  var r=p.firstName;</pre>
--	--

**a) Objasniti razliku između lijevog i desnog koda. (mislim 3 boda)**

Lijevi kod predstavlja Javascript objekt (koji može sadržavati i atribut

Objects are variables too. But objects can contain many values.

[https://www.w3schools.com/js/js\\_objects.asp](https://www.w3schools.com/js/js_objects.asp)), dok je desni kod samo jedan dugački string.

Lijevi kod će normalno raditi bez ikakvog parsiranja (radio bi i za var r = p["firstName"]), a desni kod neće raditi jer je potrebno parsiranje u JSON objekt.

**b) Uvesti minimalne potrebne dopune na kod/kodove (to nisu napisali u ispitu da se može dodati i na samo jedan da nije potrebno na oba dodati promjenu) kako bi se isječci kodova izvršili ispravno na Javascript interpretatoru (2 boda).**

Kako je napisano u a) dijelu, u lijevi dio koda nije potrebno ništa dodavati, dok je u desni dio koda potrebno dodati iznad linije `var r = p.firstName;` `p = JSON.parse(p);`

([https://www.w3schools.com/js/js\\_json\\_parse.asp](https://www.w3schools.com/js/js_json_parse.asp)).

**17. Svaki dio 1 bod**

**a) Navesti barem 3 binarna protokola za RPC.**

Ja sam na ispitu napisao CORBA, SunRPC i Java RMI (dobio sam sve bodove)

Kolega Trooper na fer2 je dodao još i **Apache Thrift, DCOM, CORBA**.

U prezama stoji da su binarne : protobuffs, Thrift, Java RMI



**b) Navesti barem 3 tekstualna protokola za RPC**

Opet na ispitu sam napisao XML, SOAP i JSON (i HTTP), ali HTTP je kriv nemojte to pisat uzimaju bodove!! Prva 3 su sigurno točna provjerio sam na uvidima.

Kolega Trooper na fer2 napisao Apache Thrift, XML RPC, gRPC.

**18. Opisati oblik i nazivlje u zapisu semantičkog verzioniranja te objasniti što znači promjena pojedinačne stavke u zapisu. (4 boda)**

Major.Minor.Patch (ovo je vrijedilo 1 bod)

Major

- Prva brojka
- Velike promjene s obzirom na prijašnju verziju
- Najčešće nije kompatibilno sa prijašnjom verzijom

Minor

- Druga brojka
- Uvođenje novih funkcionalnosti
- Kompatibilno sa prijašnjom verzijom

Patch

- Treća brojka
- Minor bug fixes trenutnih funkcionalnosti
- Kompatibilno sa prijašnjom verzijom

**19. Navedite prednosti i nedostatke postavljanja svake mikrousluge na zaseban fizički poslužitelj. (4 boda)** Asistent davao negativne bodove ako si nešto krivo napisao. Na kraju uzeo kao broj bodova  $\max(0, \text{broj\_bodova})$ , ali ako si imao 1 točnu stvar i dvije krive to je kao -1 bod, ali ti je dao 0. Uglavnom želim reći da ti kriva stvar poništi točnu tako da se mora na to paziti. Svaka točna prednost nosila je 1 bod, a svaka točna mana 0.5 boda.

Prednosti

- Nema single point of failure
- Easier to scale (skalabilno)
- Alternativne deployment tehnike

Mane

- Cijena
- Additional complexity

# Ostalo

Materijali sa starih ZI → <https://www.fer3.net/threads/72766/post-2822200>

16. – 25. (5 bodova ukupno, 0.5 po podzadatku) U nadzornom sustavu tunela poslužitelj prikuplja informacije o odvijanju prometa i periodički ih dojavljuje klijentima. Svaka dojava obavlja se posebno oblikovanom tekstualnom porukom u koju su ugrađene prometne informacije. Struktura poruke unaprijed je poznata klijentima i poslužitelju i uvijek je iste veličine, a iznosi 1000 okteta. Prijenos poruka s prometnim informacijama ostvaren je tehnologijom *Server-Sent Events*. Operater tunela dobio je ponudu za modernizaciju dojavnog dijela sustava nadogradnjom na suvremeniji *WebSocket* protokol, dok bi svi ostali parametri rada sustava ostali isti. On je sada u dilemi isplati li mu se ulaganje u modernizaciju pa je zatražio konzultantske usluge FER-a. Konzultantima na FER-u je rekao kako mu je cilj optimirati rad sustava s obzirom na količinu ukupno generiranog mrežnog prometa i želi znati pod kojim uvjetima je moguće postići uštedu od 1 GB ( $2^{30}$  okteta) mrežnog prometa. Odredite okolnosti u kojima se to postiže s obzirom na parametre rada sustava zadane tablicom. Pretpostavite da je veličina zaglavlja svakog HTTP zahtjeva i svakog HTTP odgovora po 500 okteta zajedno s praznim retkom na kraju zaglavlja. Oznaka kraja retka sama za sebe zahtijeva 2 okteta (`\r\n`). U trenutnoj inačici sustava koja koristi *Server-Sent Events* koristi se najjednostavniji oblik dojave koja sadrži samo redak s oznakom `data`: nakon čega odmah slijede korisni podaci (bez razmaka). Protokol *WebSocket* koristio bi standardom definiranu strukturu *WebSocket* okvira (*WebSocket frame*).

Bit	+0..7		+8..15		+16..23	+24..31
0	FIN		Opcode	Mask	Length	Extended length (0–8 bytes) ...
32	...					
64	...					Masking key (0–4 bytes) ...
96	...					Payload ...
...	...					

Zadatak	Broj klijenata		Period slanja poruke	Vrijeme do isplativosti (u satima)		
16.	1 000		1 s	A) 0,0596 D) 59,65	B) 298,26 E) 0,5965	C) 59 652
17.	100 000		1 s	A) 0,0596 D) 59,65	B) 2,98 E) 0,5965	C) 59 652
18.	1 000 000		1 s	A) 0,0596 D) 59,65	B) 0,298 E) 0,5965	C) 59 652
19.	1 000		10 s	A) 596,52 D) 0,596	B) 5,965 E) 2982,61	C) 596 523
20.	100 000		10 s	A) 596,52 D) 0,596	B) 5,965 E) 29,82	C) 596 523
21.	1 000 000		10 s	A) 596,52 D) 0,596	B) 5,965 E) 2,982	C) 596 523
22.	1 000		60 s	A) 3 579 139 D) 3,579	B) 3 579,139 E) 17 895,69	C) 35,79
23.	100 000		60 s	A) 3 579 139 D) 3,579	B) 3 579,139 E) 178,95	C) 35,79
24.	1 000 000		60 s	A) 3 579 139 D) 3,579	B) 3 579,139 E) 17,89	C) 35,79
25.	A) 62 138 C) 24 855	B) 124 276 D) 24 856 E) 41 426	10 s	24		

Postupak:

I za WebSocket i za SSE prvo ide jedan HTTP request - response par poruka za inicijalizaciju - tu nema razlike u količini podataka.

Svaki event SSE zapravo se appenda na tijelo odgovora HTTP responsea, a izgleda ovako "data:payload\r\n\r\n", što čini 1009 B (data: je 5, payload je 1000, a par newlineova (<https://streamdata.io/blog/server-sent-events/>) je još 4 B).

WebSocket kad šalje odgovor prema klijentu ima obavezna 2 B zastavica i budući da je duljine između  $126$  i  $2^{16}-1$  ima još 2 B duljine poruke, a ne koristi masku (koju inače koristi klijent kad šalje serveru), payload je 1000 B i sveukupno je poruka 1004 B.

Dakle, za svaki event SSE šalje 5 B više.

To znači da je throughput kod SSE-a za  $(5 \cdot \#klijenata) / (\text{period\_slanja\_poruke})$  veći, a u primjeru 16. to je 5000 B/s. Vrijeme potrebno da se pošalje 1 GB više koristeći SSE nego WebSocket je  $2^{30} / 5000 = 214748,3648$  s, odnosno 59,65 h.

Istom analogijom dobivamo:

17. E) 0,5965 h

18. A) 0,05965 h

19. A) 596,52 h

20. B) 5,965 h

21. D) 0,596 h

22. B) 3579,139 h

23. C) 35,79 h

24. D) 3,579

25. U 24 h za 1 GB moramo imati  $(2^{30} / (24 \cdot 3600))$  12427 B/s veći throughput kod SSE-a, a on je jednak  $\#korisnika \cdot 5 / 10$ . Odatle dobivamo  $\#korisnika = 24855,13481$ , što zaokružujemo na veći broj, jer bi s 24855 korisnika trebalo nešto više od 24 h za 1 GB uštede (24,00013 h), a s 24856 nešto manje od 24 h (23,999165 h), pa za točno 24 h sigurno imamo 1 GB uštede s 24856 korisnika.

Zato D) 24856

Ovo je navodno postupak, mogu potvrdit za veličinu websocket poruke, ali nisam siguran za veličinu SSE poruke.

Mislim da veličina SSE poruke je 5 (data:) + 1000 (payload) + 4 ( $\backslash\r\n\r\n$ ) = 1009, kao što piše.

HTTP/1.1 200 OK  
Content-Type: text/plain  
Content-Length: 3  
Expires: [vrijednost iz tablice 1]

[troznamenkasta numerička vrijednost sredstva]

Tablica 1.

Trenutak promjene sredstva	0	7	13	28	31	37	52	58	93	107	110
Vrijednost sredstva	111	222	333	444	555	666	777	888	999	111	222
Vrijednost Expires zaglavljaja	10	17	23	38	41	47	62	68	103	117	120

Tablica 2.

Trenutak slanja zahtjeva s klijenta	2	11	16	19	38	42	55	56	57	60	61
Primljena vrijednost sredstva na klijentu	111	222	222	333	444	555	777	777	777	999	999
Način obrade zahtjeva na posredniku	CH	RH	PH	CH	CH	RM	RM	CH	CH	CH	CH

- 0,5 a) (1,5 bodova) Odrediti vrijednost sredstva koju klijent dobiva za svaki postavljeni zahtjev. Odgovore upisati u tablicu 2.
- 0 b) (1,5 bodova) Za svaki zahtjev odrediti način njegove obrade na posredniku. Odgovore upisati u tablicu 2 na način:  
CH=cache hit, CM=cache miss, RH=revalidate hit, RM=revalidate miss
- 0 c) Odrediti učinkovitost privremene memorije posrednika (engl. hit rate) pod pretpostavkom (odgovore upisati na crtu):  
(0,5 boda) da se pogotkom smatra samo CH 0,5 — 6/11  
(0,5 boda) da se promašajem smatra samo CM 0,5 — 10/11
- d) (2 boda) Komentirati ispravnost rada sustava. Ako sustav radi ispravno, obrazložiti odgovor. Ako sustav ne radi ispravno, obrazložiti problem i predložiti mjere za njegovo otklanjanje (odgovor upisati u prazno polje ispod teksta).

27. (4 boda) Objasniti na koji način granularnost podjele web usluge na skup sredstava utječe na skalabilnost web poslužitelja i mreže (odgovor upisati u prazno polje ispod teksta).

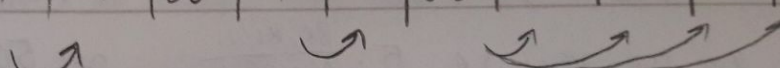
Mikroservisi imaju veću skalabilnost jer se mogu nezavisno deployati na različite poslužitelje dok se npr. monolitna aplikacija mora cijela



Evo slikana tablica.

2	11	16	19	38	42	55	56	57	60	61
111	222	222	333	666	666	777	777	777	777	777
CM	RM	CH	RM	RM	CH	RM	CH	CH	CH	CH

t SLANJA	2	11	16	19	38	42	55	56	57	60	61
PRIMIJEN VAZ	111	222	222	333	666	666	777	777	777	777	777
NAČIN OBR.	CM	RM	<u>CH</u>	RM	RM	<u>CH</u>	RM	<u>CH</u>	<u>CH</u>	<u>CH</u>	<u>CH</u>
VRJEDI DO	10	17	17	23	47	47	62	62	62	62	62

  
 CACHE MISLI  
 DA JE PODATAK  
 JOŠ UVIJEK OK

Najlakše je imati još jedan redak u kojem se pamti do kojeg trenutka podatak valja. Ako je trenutak slanja zahtjeva za podatkom manji od "vrijedi do" dohvaća se podatak koji je u cache-u, inače se dohvaća najnoviji podatak na poslužitelju. (ako je trenutak slanja 38 dohvaća se zadnji unešeni podatak prije 38, a to je 37. Nadam se da je sad sve jasno.

Najlakše je imati još jedan redak u kojem se pamti do kojeg trenutka podatak valja. Ako je trenutak slanja zahtjeva za podatkom manji od "vrijedi do" dohvaća se podatak koji je u cache-u, inače se dohvaća najnoviji podatak na poslužitelju. (ako je trenutak slanja 38 dohvaća se zadnji unešeni podatak prije 38, a to je 37.) Nadam se da je sad sve jasno.

Hvala kolega Kljknj odlično objašnjeno :D :)

c) broj CH / ukupan = 6/11

broj (CH + RM + RH) / ukupan = 10/11

d) sustav nije(?) ispravan jer se može dogoditi promjena podatka na poslužitelju, dok na cache-u je stari podatak sve dok ne istekne vrijeme valjanosti podatka. Također možda bi mogel biti problem s nedefiniranim područjem kao što je između 23 i 28?????

Pretpostavljam da će u tom području svaki put se dogoditi RH; vidjeti će da je vrijeme valjanosti prošlo, pitati će poslužitelja za novim podatkom, poslužitelj će odgovoriti da se nije promjenio i cache vraća isti podatak.

## **27.**

Ako jedan host sadrži više usluga (Multiple services per host), onda to nije skalabilno rješenje, jer bi za skaliranje trebalo dodati cijeli novi host sa svim tim uslugama.

U slučaju da imamo jednu uslugu po hostu (Single service per host), npr jedan host ima bazu drugi aplikacijsku logiku, onda kada bi trebalo skalirati npr samo bazu podataka, mogli bi dodati samo host sa bazom, što znači da je skaliranje puno jednostavnije.

Sad, ako je cijela krupnoznata, znači da sve usluge idu na jedan host, što znači slabije skaliranje.

Ako je aplikacija sitnoznata, moguće je podijeliti usluge po različitim hostovima što olakšava skalabilnost.

1. Ispravno oblikovanim REST sučeljem (engl. *REST API*) smatra se ono koje je klijentima izloženo na način
- A) GET /deleteUser/1234
  - B) GET /deleteUser?id=1234
  - C) DELETE /deleteUser/1234
  - D) DELETE /users/1234
  - E) POST /users/1234/delete
2. Cjevovodni režim rada HTTP protokola (engl. *HTTP pipelining*) omogućava
- A) slanje velikih HTTP poruka u više mrežnih paketa
  - B) istodobno posluživanje više različitih korisnika
  - C) prijenos zvuka i videa HTTP protokolom
  - D) korištenje HTTP protokola za prijenos poruka drugih protokola
  - E) upućivanje HTTP zahtjeva prema poslužitelju bez čekanja HTTP odgovora na prethodno upućeni zahtjev
3. Nakon početne uspostave veze, protokol *WebSocket* omogućava komunikaciju klijenta i poslužitelja koja je
- A) jednosmjerna od klijenta prema poslužitelju
  - B) jednosmjerna od poslužitelja prema klijentu
  - C) dvosmjerna sinkrona
  - D) dvosmjerna asinkrona
  - E) nije omogućena izravna komunikacija klijenta i poslužitelja, već isključivo putem posrednika
4. Nakon početne uspostave veze, tehnologija *Server-Sent Events* omogućava komunikaciju klijenta i poslužitelja koja je
- A) jednosmjerna od klijenta prema poslužitelju
  - B) jednosmjerna od poslužitelja prema klijentu
  - C) dvosmjerna sinkrona
  - D) dvosmjerna asinkrona
  - E) nije omogućena izravna komunikacija klijenta i poslužitelja, već isključivo putem posrednika
5. Protokolom *WebSocket* moguće je prenositi
- A) isključivo tekstualne poruke
  - B) isključivo binarne poruke
  - C) tekstualne i binarne poruke
  - D) binarne poruke izravno, a tekstualne samo ako su prethodno kodirane u Base64 format
  - E) tekstualne poruke izravno, a binarne samo ako su prethodno kodirane u Base64 format
6. Tehnologijom *Server-Sent Events* moguće je prenositi
- A) isključivo tekstualne poruke
  - B) isključivo binarne poruke
  - C) tekstualne i binarne poruke
  - D) binarne poruke izravno, a tekstualne samo ako su prethodno kodirane u Base64 format
  - E) tekstualne poruke izravno, a binarne samo ako su prethodno kodirane u Base64 format

6. Točan je E (označeno crveno)

7. U sustavu s pravilnim periodičkim promjenama gdje je period promjene dugačak (primjerice, svakih nekoliko sati), a količina korisnih podataka značajno veća od zaglavlja HTTP protokola, promjene s poslužitelja na klijente preporučljivo je izvesti
- ☒ A) tehnikom prozivanja (*polling*)
  - ☐ B) tehnikom blokirajućeg prozivanja (*long polling*)
  - ☐ C) protokolom *WebSocket*
  - ☐ D) tehnologijom *Server-Sent Events*
  - ☐ E) izbor dojavne tehnike nema utjecaja na radna svojstva sustava
8. Ujednačavanje XML-a neizostavan je korak kada se provodi
- ☐ A) bilježenje korištenja
  - ☐ B) autorizacija
  - ☐ C) autentikacija
  - ☒ D) digitalno potpisivanje
  - ☐ E) kriptiranje
9. Glavna zadaća RPC sustava je
- ☒ A) normalizacija semantike poziva funkcija u istim i različitim adresnim prostorima
  - ☐ B) vrednovanje semantike poziva funkcija u različitim adresnim prostorima
  - ☐ C) normalizacija adresnog prostora poziva funkcija u udaljenom sustavu
  - ☐ D) opisivanje semantike slanja parametara procedure koristeći IDL
  - ☐ E) omogućavanje asinkronog poziva udaljene procedure
10. U sustavima zasnovanim na WS-\* tehnološkom stogu, WSDL služi kao
- ☐ A) HTTP
  - ☐ B) RPC
  - ☒ C) IDL
  - ☐ D) SOAP
  - ☐ E) EDI
11. Promjena trećeg broja u zapisu verzije sustava koji koristi semantičko verzioniranje označava
- ☐ A) promjene nekompatibilne s prethodnim verzijama
  - ☐ B) promjene kompatibilne s prethodnim verzijama
  - ☒ C) ispravke manjih grešaka postojeće funkcionalnosti
  - ☐ D) promjene samo u sučelju sustava
  - ☐ E) promjene samo na poslužiteljskom dijelu sustava
12. Poredati vrste testova, u kontekstu mikro-usluga, od sporijih prema bržima: 1. E2E (*end-to-end*) testovi, 2. testovi usluga (*service tests*), 3. jedinični testovi (*unit tests*)
- ☒ A) 1, 2, 3
  - ☐ B) 3, 2, 1
  - ☐ C) 1, 3, 2
  - ☐ D) 3, 1, 2
  - ☐ E) 2, 3, 1
13. Oblikovni obrazac *Tolerant Reader* koristi se za
- ☐ A) pojednostavljenje sučelja složenog sustava
  - ☐ B) presretanja poruka
  - ☐ C) čitanje poruka promjenjivog sadržaja
  - ☒ D) čitanje poruka promjenjive strukture
  - ☐ E) lakše ispitivanje (testiranje) sustava
14. Poredati serijalizacijske formate prema veličini (prosječne) poruke, od manje prema većoj. Napomena: Protobuf i Thrift s odnose na kanonske inačice binarnih formata ugrađenih u istoimene tehnologije.
- ☐ A) Protobuf, Thrift, JSON, XML, SOAP
  - ☒ B) Protobuf, JSON, XML, SOAP, Thrift
  - ☐ C) JSON, Protobuf, Thrift, XML, SOAP
  - ☐ D) JSON, XML, SOAP, Protobuf, Thrift
  - ☐ E) SOAP, Protobuf, Thrift, JSON, XML
15. Osnovni skup tehnologija *Web Services* tehnološkog stoga (WS-\*) čine
- ☐ A) SOAP, WSDL, UDDI
  - ☐ B) XML, JSON, REST
  - ☐ C) RPC, TCP, HTTP
  - ☒ D) SOAP, XML Schema, XPath
  - ☐ E) WSDL, XML-RPC, AJAX

7. Točan A.

8. Ne radimo.

15. Točan A.



SRETNO SVIMA :)

A ne samo njima