KLIJENT-POSLUŽITELJ SUSTAV

SADRŽAJ:

1.UVOD

Za razvoj raspodijeljenih sustava važan je razvoj sve jeftinijih računala koja imaju sve veće mogućnosti te razvoj računalnih mreža.

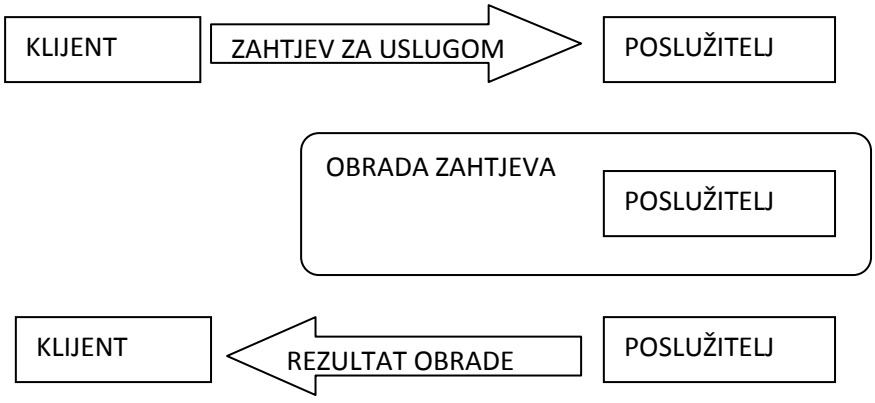
Do osamdesetih godina prošlog stoljeća vrijedilo je Grosch-ovo pravilo da je "moć (broj instrukcija u jedinici vremena) procesora proporcionalna kvadratu njegove cijene". Ili: ako biste izdvojili duplo više novca dobili biste 4 puta bolje performanse.Razvojem mikroprocesora Grosch-ovo pravilo više ne vrijedi. Ulaganjem duplo više novca često se može se dobiti samo isti procesor koji radi na nešto većoj frekvenciji.Rezultat ovakvog razvoja je da se ne isplati ulagati u veliko računalo sa snažnim procesorom već se više isplati imati veći broj računala sa procesorom srednje snage.Svako novo računalo u mreži dati će duplo bolje performanse.Raspored računala, komunikacija i odnos među računalima u mreži dovodi do stvaranja 2 modela: *klijent-poslužiteljskog* i *partnerskog modela*.Kod partnerskog modela svaki čvor je ravnoparan dok kod klijent-poslužiteljskog modela postoji jasna razlika između čvorova koji pohranjuju i nude podatke (poslužitelji) te čvorova koji podatke potražuju, obrađuju ili stvaraju (klijenti).U ovom radu obrađujemo model klijent-poslužitelj.

2. KLIJENT-POSLUŽITELJ ARHITEKTURA

Sustavom klijent-poslužitelj smatra se svaki sustav koji se može raščlaniti na dva dijela: prvi dio (*klijent*) koji traži obavljanje nekog zadatka i drugi dio *(poslužitelj*) koji traženi zadatak obavlja.Model *klijent/poslužitelj* danas je jedan od najprimjenjivanijih modela.Svakodnevno ga susrećemo kod dijeljenja različitih resursa:elektronske pošte, mrežnih novosti, kod dijeljenja datoteka, dijeljenja prostora na diskovima, pristupa internetu, baza podataka. Međutim, nije moguće sve resurse dijeliti na taj način. Neki resursi moraju ostati lokalni za svako računalo: radna memorija (RAM), centralna upravljačka jedinca (CPU) i mrežno sučelje se smatraju najmanjim skupom resursa koji moraju ostati lokalni.Ti se ključni resursi mogu dijeliti samo među procesima koji se izvode na istom računalu.

2.1 Komunikacija između klijenta i poslužitelja

Komunikaciju klijenta i poslužitelja možemo podijeliti na fazu zahtjeva u kojoj klijent poslužitelju šalje zahtjev za obavljanjem usluge, fazu obrade u kojoj se zahtjev obrađuje i fazu slanja rezultata obrade. Klijent odnosno poslužitelj mogu biti računala ali i neki drugi elementi, primjerice programske komponente.Okruženje klijent-poslužitelj je heterogeno, tj. sklopovska oprema i operacijski sustavi klijenta i poslužitelja ne moraju biti jednaki.



Slika 1.Kominikacija klijent/poslužitelj

Primjer komunikacije: pretraživač (IE, Mozzila...) je *klijent* program pokrenut na korisnikovu računalu s kojim može pristupiti informacijama pohranjenim na nekom mrežnom

poslužitelju.Korisnik pristupa bankarskim uslugama sa svog računala koristeći pretraživač kao *klijent* da bi poslao zahtjev mrežnom *poslužitelju* banke. Taj mrežni *poslužitelj* proslijeđuje zahtjev svojoj vlastitoj bazi podataka koja je *klijentski* program i šalje zahtjev *poslužitelju* baze podataka neke druge banke da bi od nje preuzeo podatke o računu. Nakon što se podatci o računu pronađu, proslijeđuju se *klijentskoj* bazi podataka odnosno bazi podataka banke koja je poslala zahtjev, ona podatke vraća mrežnom *poslužitelju* banke. Podatci se prikazuje u pretraživaču odakle je zahtjev i poslan.

2.1.1 Funkcije klijenta

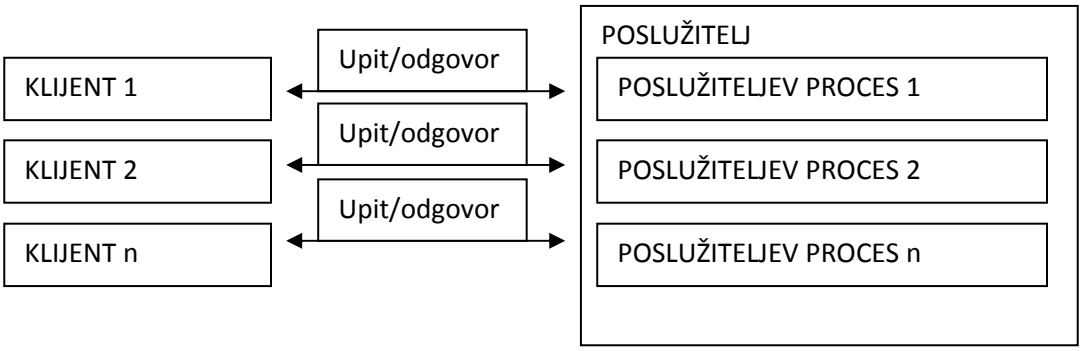
- osiguravanje korisničkog sučelja
- provjera sintaksne ispravnosti unesenih podataka
- translacija korisnikovog upita u oblik standardiziran protokolima prijenosa podataka
- slanje upita poslužitelju
- primanje poslužiteljevog odgovora
- translacija odgovora u oblik čitljiv korisniku
- prikaz rezultata korisniku

2.1.2 Funkcije poslužitelja

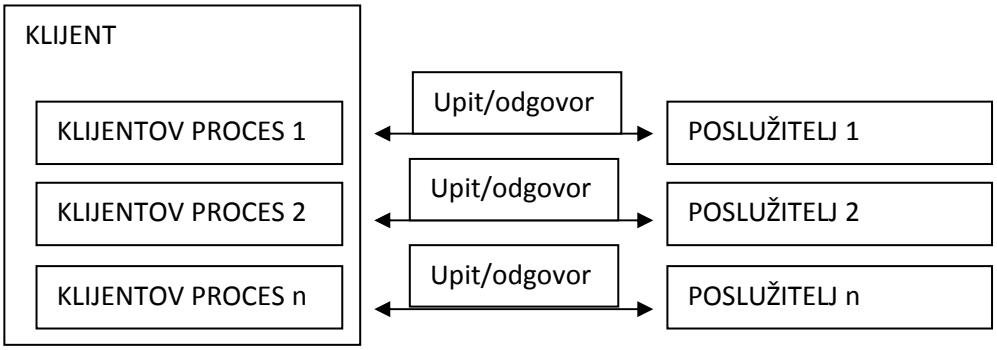
- prihvat klijentovog zahtjeva
- obrada zahtjeva
- slanje rezultata klijentu

U slučaju da su *klijent* i *poslužiteljski* procesi smješteni u dva ili više nezavisnih i umreženih računala, *poslužiteljski* proces može osigurati usluge za više od jednog *klijenta*. Pored toga,

klijent može zahtijevati usluge i od više *poslužitelja* iz okruženja bez obzira na njihove lokacije ili fizičke karakteristike računala na kojima se nalaze poslužiteljski procesi. Mreža služi povezivanju *poslužitelja* i *klijenata* zajedno osiguravajući medij kroz koji klijenti i poslužitelj komuniciraju.



Slika 2.Više klijenata ima zahtjev na istom poslužitelju



Slika 3.Jedan klijent šalje zahtjeve na više različitih poslužitelja

Klijent i poslužitelj zahtjevaju velik broj ispunjenih konvencija prije nego što mogu komunicirati. Taj komplet sadrži protokol koji mora biti implementiran u oba kraja spajanja, na klijentu i poslužitelju. Primjeri spajanja su TELNET protokol koji se koristi kod povezivanja na udaljeno računalo, mrežni datotečni prijenosnik FTP i najviše korišteni hipertekstualni prijenosni protokol, HTTP.

Klijent sadrži sklopovske i programske komponente i poželjno je da one posjeduju sljedeće karakteristike:
-Operativni sustav koji je sposoban da podrži „*multitasking*“
-Komunikacijske sposobnosti

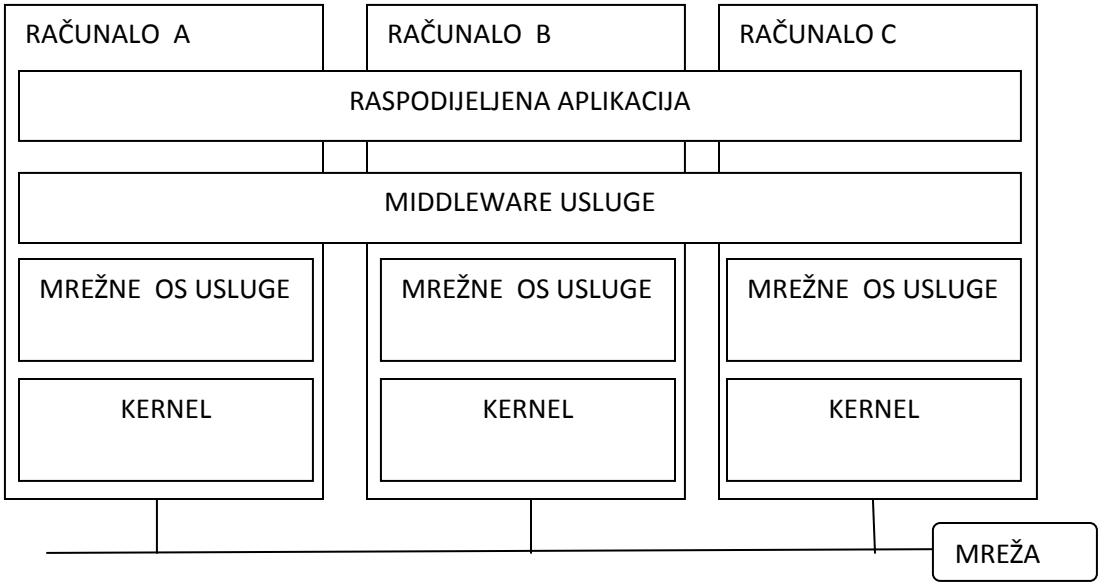
Klijent aplikacija se spaja s operativnim sustavom radi korištenja “*multitaskinga*” i grafičkog korisničkog sučelja koje on osigurava. Ona se još spaja i s mrežnom programskom komponentom komunikacijskog posrednika radi pristupa poslužiteljima.

Sklopovska komponenta komunikacijskog posrednika (mrežna kartica i mrežni kabal) fizički prenosi zahtijeve i odgovore na zahtijeve između klijenta i poslužitelja.Dok se zahtijev izvršava na poslužitelju, klijent može izvršavati druge poslove.

Raspodjela zadataka između *klijenta* i *poslužitelja* čini bitan element pri kreiranju svake aplikacije koja se izvršava u okruženju *klijent-poslužitelj*.
U nekim prilikama *klijenti* su nedjelotvorni uređaji za zahtjevne aplikacije s velikim performansama.U tom slučaju izračunavanje je napravljeno na visokoučinkovitim *poslužiteljima* (eng.*high-performance server*).Danas je takav pristup manje korišten, ali još uvijek ima područje djelovanja kao što je izračunavanje virtualnih stvarnosti u filmskim scenama

2.1.3 Middleware

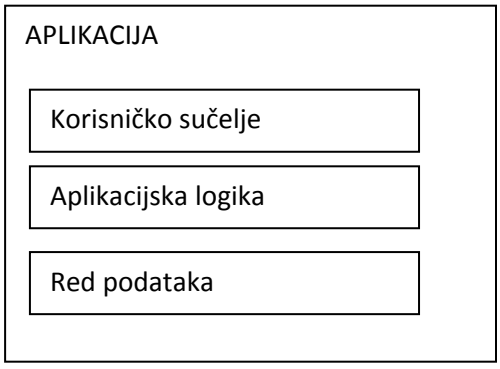
Da bi smo osigurali pristup poslužiteljima, čiji se procesi ne izvršavaju na istom stroju kao i klijent, obično se koristi „*middleware*“. *Middleware* služi kao umrežavajuća komponenta između klijent-poslužitelj sustava, mora se izvoditi i na klijentu i na poslužitelju. On daje sve što je potrebno da biste dobili zahtjev od klijenta do poslužitelja i da poslužitelj da odgovor natrag do klijenta.*Middleware* često olakšava komunikaciju između različitih tipova računalnih sustava. Ova komunikacija omogućuje prijelaznu platformu (eng.*cross-platform*) između klijent i poslužitelj, i omogućuje različitim vrstama klijenata da pristup istim podacima. Na taj način se izbjegavaju problemi koji nastaju zbog različitosti računala.



Slika 4.Middleware

3. 2-REDNA I 3-REDNA ARHITEKTURA KLIJENT-POSLUŽITELJ

Može se reći da poslužitelj gotovo uvijek ima ulogu čuvanja podataka a klijent osigurava korisničko sučelje.Aplikacijska logika, koja određuje što bi se s podatcima trebalo događati, može biti raspodijeljena između klijenta i poslužitelja.



Slika 5.Osnovni elementi aplikacije

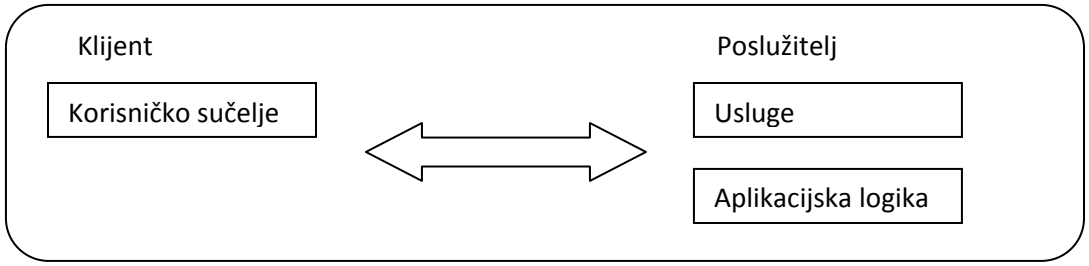
U početku razvoja modela *klijent-poslužitelj*, aplikacijska logika čvrsto se vezivala ili za klijenta ili za poslužitelja. Tako dolazimo do pojmova "debelog" odnosno "tankog" klijenta i "debelog"

odnosno "tankog" poslužitelja. Oblikovatelji *klijent-poslužitelj* aplikacija su se na početku oblikovanja morali odlučiti za jednu od sljedeće dvije opcije:

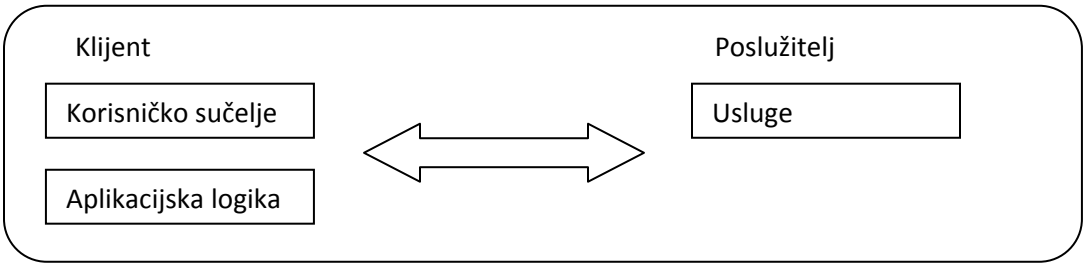
- "tanki" klijent - "debeli" poslužitelj
- "debeli" klijent - "tanki" poslužitelj

Ovakve arhitekture sustava *klijent-poslužitelj* nazivaju se *dvorednim arhitekturama sustava klijent-poslužitelj* (engl. 2-tier client-server arhitecture).

3.1. 2-redna arhitektura sustava klijent-poslužitelj



Slika 6.“Tanki“ klijent –“Debeli“ poslužitelj



Slika 7.“ Debeli“ klijent –“ Tanki“ poslužitelj

"Debeli" poslužiteljski sustavi, kao što su "*groupware*" sustavi i internet poslužitelji, povjeravaju više odgovornosti aplikacijskoj logici kod *poslužitelja*, dok "debeli" klijent sustavi, kao što su većina baznih sustava, predaju veću odgovornost *klijentu*.

Sustav *tankog klijenta* (engl.*Thin-client*) (a)

- Klijent je oblikovan da bude što je moguće više mali i jednostavniji.
- Većina posla obavlja se na poslužiteljskoj strani.
- Oblikovnu strukturu klijenta i izvršni kod jednostavno se preuzima preko računalne mreže.

Sustav *debelog klijenta* (engl. *Fat-client*) (b)

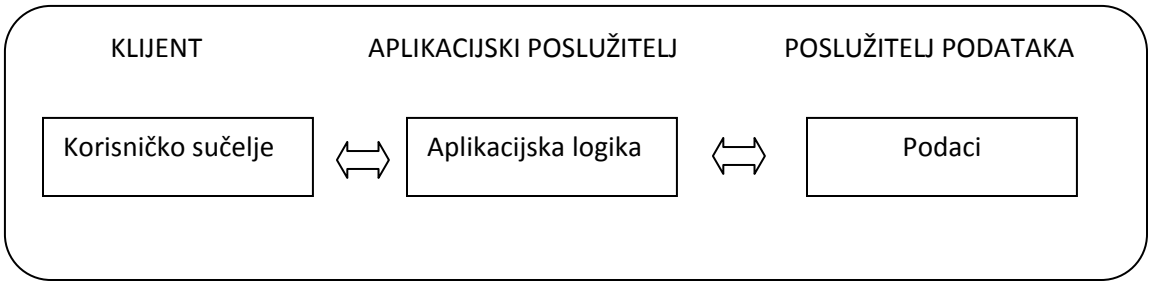
- Što je moguće više posla predaje se klijentima.
- Poslužitelj može rukovati s više klijenata.

2-redna arhitektura ima nedostatke bez obzira gdje se smjesti element aplikacijske logike. Kombinacija "tanki" klijent - "debeli" poslužitelj ima najveći nedostatak u preopterećivanju poslužitelja što rezultira smanjivanjem brzine izvršavanja aplikacija pri povećanju broja korisnika sustava. Kod kombinacije "debeli" klijent -"tanki" poslužitelj problemi nastaju kada se treba nešto promijeniti unutar aplikacijske logike jer je potrebno izvršiti promjene na svim klijent računalima što povećava cijenu održavanja sustava.

Izvedba sustava klijent-poslužitelj u 2-rednoj arhitekturi preporuča se kod izrade aplikacija kojima neće pristupati više od stotinjak korisnika, koje će koristiti samo jednu bazu podataka i koje će se izvršavati u okruženju brze i sigurne računalne mreže. Izrada zahtjevnijih aplikacija u ovoj arhitekturi ne preporuča se zbog već navedenih nedostataka ove arhitekture.

Nedostaci 2-redne arhitekture klijent-poslužitelj riješeni su odvajanjem elementa aplikacijske logike u zasebni red čime dolazimo do *troredne arhitekture sustava klijent poslužitelj* (engl. 3-tier client-server architecture).

3.2. 3-redna arhitektura sustava klijent-poslužitelj



Slika 8. 3-redna arhitektura klijent-poslužitelj

3-rednom arhitekturom klijent-poslužitelj olakšano je obavljanje izmjena unutar aplikacijske logike ali i unutar drugih elemenata sustava. Izmjene izvršene unutar aplikacijske logike ne moraju povlačiti potrebu za izmjenjivanjem korisničkog sučelja ili elementa podataka a vrijedi i obratno. Procesi postaju snažnije budući da oni mogu djelovati neovisno o klijentima i poslužiteljima.Osim toga, razdvajanje aplikacijske logike od podataka omogućuje da se podaci iz više izvora koriste u jednoj transakciji, bez kvara u *klijent-poslužitelj* modelu.

3-redne arhitekture *klijent-poslužitelj* je fleksibilnija od 2-redna arhitekture, jer razdvajanje aplikacijske logike iz klijenta i poslužitelja aplikacija daje aplikacijskoj logici novu razinu autonomije. Nadalje, prednost 3-redne arhitekture klijent-poslužitelj ogleda se i u mogućnosti repliciranja srednjeg reda aplikacije odnosno instalacije aplikacijske logike na više računala (aplikacijskih poslužitelja) čime se dodatno rasterećuje sustav i ubrzava rad aplikacije. Daljnjom parcijalizacijom dolazimo do n-rednih arhitektura.

Ovaj napredak u klijent-server arhitektura je u velikoj mjeri odgovoran za pojam raspodijeljenog prijenosa podataka.

4. ZAKLJUČAK

PREDNOSTI

-Upravljanje podacima je puno lakše jer su podaci na jednom mjestu. To dozvoljava brze pohrane podataka i efikasno upravljanje pogreškama. Ima više stupnjeva dozvola koji mogu omogućiti korisnicima da ne rade štetu podacima.

-Poslužiteljevo sklopovlje je dizajnirano da služi zahtjevima klijenata vrlo brzo. Svi podaci su procesuirani na poslužitelju i samo su rezultati vraćeni klijentu. Ovo omogućava smanjenje količine mrežnog prometa između poslužitelja i klijenta poboljšavajući mrežne performanse.

-Arhitektura „tankih" klijenata dopušta brzu zamjenu izbjeglih klijenata zbog toga što su podaci i aplikacije na poslužitelju.

NEDOSTACI

-*Klijent-poslužitelj* sustavi su jako skupi i trebaju mnogo održavanja.

-*Poslužitelj* sadrži samo jednu točku neuspjeha. Ako se neuspjesi na poslužitelju dogode moguće je da će sustav patiti u teškom zakašnjenju ili će se kompletno pokvariti te onemogućiti stotinama klijenata da rade sa svojim podacima i aplikacijama. Za to vrijeme dok je poslužitelj izvan funkcije kompaniji koja pruža uslugu nagomilavaju se troškovi.

5.LITERATURA

1. Klijent-poslužiteljski sustavi

URL: http://www.answers.com/topic/client-server-system (02/09/10).

2. Rosić, M.,Raspodijeljeni sustavi (prezentacija)

3. Korisnik-poslužiteljski (Klijent-server)

URL:www.efos.hr/informatika/so/.../Ivana_petrovic_klijent%20server.ppt (02/09/10).

4. Jan Maly, Robin: Comparison of Centralized(Client-Server) and Decentralized (Peer-to-Peer) Networking, Zurich 2003

URL: ftp://ftp.tik.ee.ethz.ch/pub/students/2002-2003-Wi/SA-2003-16.pdf (01/09/10).