**CS230 SKRIPTA – BY OTO 5460**

**L01**

1. **Sta je centralizovani racunarski sistem?**

Centralizovani racunarski sistem je arhitektura u kojoj je sav softver, podaci i kontrola koncentrisana u jednom centralnom racunaru ili serveru. Svi korisnici pristupaju resursima preko tog centralnog cvora.

1. **Koje su prednosti decentralizovanih racunarskih sistema u odnosu na centralizovane?**

Decentralizovani racunarski sistemi nude bolju skalabilnost i otpornost na otkaze u poredjenju sa centralizovanim sistemima. Svaki cvor u decentralizovanom sistemu moze funkcionisati nezavisno, sto doprinosi vecoj pouzdanosti i smanjenju rizika od kvara.

1. **Sta je distribuirani racunarski sistem i koja je njegova osnovna karakteristika?**

Distribuirani racunarski sistem je arhitektura u kojoj se procesi i resursi racunarskog sistema nalaze na vise fizicki razdvojenih racunara koji su povezani mrezom. Osnovna karakteristika distribuiranih sistema je transparentnost, tj. sposobnost da se korisnicima i aplikacijama sakrije da se procesi i resursi nalaze na razlicitim racunarima.

1. **Kako se postize transparentnost u distribuiranim sistemima?**

Transparentnost u distribuiranim sistemima postize se kroz upotrebu middleware-a, sloja softverske infrastrukture koji pruza standardizovan interfejs aplikacijama. Middleware omogucava aplikacijama da pristupaju procesima i resursima distribuiranog sistema na isti nacin bez obzira na njihovu fizicku lokaciju.

1. **Koje su cetiri vrste transparentnosti u distribuiranim sistemima?**

Cetiri vrste transparentnosti u distribuiranim sistemima su: transparentnost pristupa, lokacije, relokacije i migracije. Transparentnost pristupa se odnosi na sakrivanje razlika u nacinima pristupa resursima. Transparentnost lokacije se odnosi na sakrivanje fizicke lokacije resursa. Transparentnost relokacije se odnosi na pomeranje objekata bez uticaja na korisnike. Transparentnost migracije podrzava mobilnost procesa i resursa bez prekida komunikacije.

1. **Kako se ostvaruje transparentnost pristupa u distribuiranim sistemima?**

Transparentnost pristupa ostvaruje se kroz standardizaciju interfejsa za pristup resursima distribuiranog sistema. To znaci da aplikacije pristupaju resursima na isti nacin bez obzira na njihovu fizicku lokaciju ili tehnicke specifikacije.

1. **Koja je uloga midlvera u distribuiranim sistemima?**

Midlver je sloj softverske infrastrukture koji se nalazi izmedju transportnog i aplikativnog sloja distribuiranog sistema. Njegova uloga je da pruzi standardizovan interfejs aplikacijama i omoguci transparentan pristup procesima i resursima distribuiranog sistema.

1. **Sta je transparentnost replikacije i zasto je vazna u distribuiranim sistemima?**

Transparentnost replikacije se odnosi na sakrivanje cinjenice da postoje vise kopija resursa ili da vise procesa radi u zakljucanom rezimu, omogucavajuci da jedan preuzme kontrolu ako drugi ne uspe. Ovo je vazno jer omogucava vecu pouzdanost i dostupnost sistema kroz redundanciju, a korisnici ne moraju biti svesni ove slozenosti.

1. **Sta je transparentnost konkurencije i kako se postize u distribuiranim sistemima?**

Transparentnost konkurencije se odnosi na to da korisnici ne primete da drugi koriste iste resurse na kojima oni rade istovremeno. To se postize kroz mehanizme zakljucavanja koji omogucavaju da se resursi koriste na kooperativan nacin, bez konflikata ili primetnog ometanja korisnika.

1. **Kakva je uloga maskiranja kvarova u distribuiranim sistemima i koje su poteskoce u ostvarivanju ovog cilja?**

Maskiranje kvarova je kljucna za odrzavanje pouzdanosti distribuiranih sistema, omogucavajuci da korisnici ili aplikacije ne primete kada deo sistema ne radi kako treba, i da se sistem automatski oporavi od otkaza. Glavna poteskoca je razlikovanje izmedju mrtvih i sporo reagujucih procesa, kao i nemogucnost preciznog utvrdjivanja uzroka kvara u slozenim distribuiranim okruzenjima.

1. **Sta znaci otvorenost distribuiranih sistema i zasto je vazna?**

Otvorenost distribuiranih sistema se odnosi na sposobnost sistema da pruzi komponente koje se lako mogu koristiti ili integrisati u druge sisteme, kao i da podrzava komunikaciju sa komponentama van samog sistema. Ovo je vazno jer omogucava fleksibilnost, interoperabilnost i mogucnost prilagodjavanja sistema promenljivim zahtevima i tehnoloskim trendovima.

1. **Sta podrazumeva pouzdanost distribuiranih sistema i kako se moze meriti?**

Pouzdanost distribuiranih sistema se odnosi na sposobnost sistema da kontinuirano pruza usluge u skladu sa ocekivanjima, cak i u prisustvu kvarova ili otkaza. Pouzdanost se moze meriti kroz velicine kao sto su srednje vreme do otkaza (MTTF), srednje vreme do oporavka (MTTR) i srednje vreme izmedju otkaza (MTBF), sto omogucava procenu performansi i otpornosti sistema na greske.

1. **Koje su kljucne tehnike skaliranja distribuiranih sistema i kako se razlikuju?**

Kljucne tehnike skaliranja distribuiranih sistema ukljucuju vertikalno skaliranje (povecanje kapaciteta postojecih resursa) i horizontalno skaliranje (dodavanje novih resursa radi prosirenja kapaciteta sistema). Vertikalno skaliranje se odnosi na nadogradnju postojecih komponenti, dok horizontalno skaliranje podrazumeva dodavanje novih cvorova u sistem radi distribucije opterecenja i povecanja performansi.

------------------------------------------------------------------------------------------------------------------------------------------

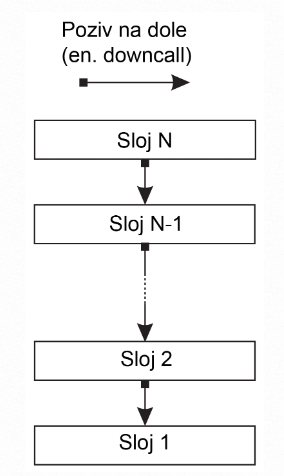
------------------------------------------------------------------------------------------------------------------------------------------

**L02**

**SLOJEVITA ARHITEKTURA**

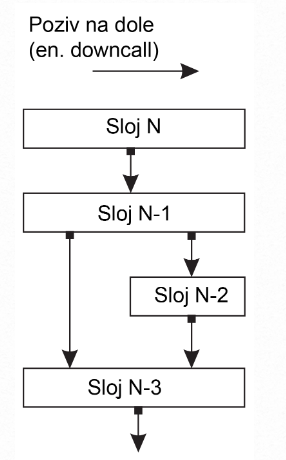
1. **Kako je organizovana jednosmerna arhitektura bez preskakanja sloja i u kojem kontekstu je cesto uporedjena?**

Jednosmerna arhitektura bez preskakanja sloja organizovana je tako da komponente komuniciraju iskljucivo ka sledecem nizem sloju, slicno kao u mreznoj komunikaciji. Ova organizacija omogucava jednostavnu strukturu gde se komponente oslanjaju na komunikaciju samo sa slojem ispod sebe, cesto uporedjujuci se sa standardnom organizacijom u mreznoj komunikaciji.



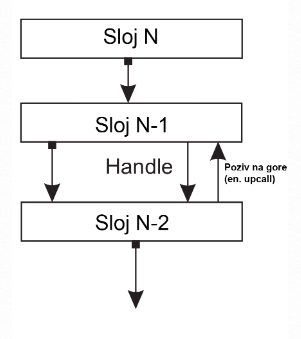
1. **Kako se koristi jednosmerna arhitektura sa preskakanjem sloja i koja je uloga slojeva u takvoj organizaciji?**

Jednosmerna arhitektura sa preskakanjem sloja koristi se kada aplikacija koristi specijalizovane biblioteke koje su implementirane koriscenjem drugih biblioteka. Slojevi su organizovani tako da aplikacija koristi biblioteke na nizim slojevima, ali moze direktno komunicirati sa specijalizovanim bibliotekama bez prolaska kroz sve slojeve. Na primer, aplikacija A se implementira na sloju N−1, specijalizovana matematicka biblioteka na sloju N−2, dok je zajednicka OS biblioteka na sloju N−3.



1. **Kako se definise dvosmerna arhitektura i u kojim situacijama se moze primeniti?**

Dvosmerna arhitektura se javlja kada nizi sloj inicira zahtev prema visem sloju. Ovo se cesto desava kada operativni sistem ili neki drugi nizi sloj signalizira dogadjaj, nakon cega visi sloj izvrsava odgovarajucu operaciju koju je prethodno preneo referencu. Na primer, kada operativni sistem signalizira pojavu dogadjaja, moze pozvati korisnicki definisanu operaciju koja je prethodno preneta referencom, obicno nazvanom ruckom (handle).

****

**MONOLITNA ARHITEKTURA**

1. **Kako se moze definisati monolitna arhitektura i kako funkcionise (primer)?**

Monolitna arhitektura je tradicionalni pristup razvoju aplikacija u kojem su svi delovi aplikacije integrisani u jednu celinu. U ovom modelu, aplikacija se sastoji od jednog softverskog modula koji obuhvata sve funkcionalnosti, poput korisnickog interfejsa, poslovne logike i pristupa podacima.

Na primeru veb sajta za rezervaciju hotela, monolitna arhitektura podrazumeva da se svi delovi aplikacije, poput pretrage hotela, procesa rezervacije, interakcije sa bazom podataka i eksternim servisima, nalaze u jednom softverskom modulu. Svaki zahtev korisnika prolazi kroz sve ove delove aplikacije unutar istog sistema.

1. **Kako se prednosti monolitne arhitekture manifestuju u procesu razvoja aplikacija?**

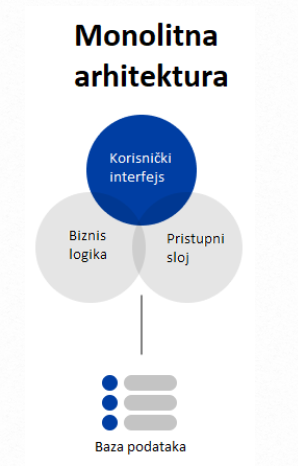
Prednosti monolitne arhitekture ukljucuju jednostavnost razvoja, organizaciju koda na jasan i strukturiran nacin, jednostavno skladistenje podataka u jednu bazu, lako rasporedjivanje aplikacije i relativnu skalabilnost, sto omogucava jednostavno odrzavanje aplikacije od strane jednog tima.

1. **Koje su glavne izazove i nedostatke monolitne arhitekture u procesu razvoja i odrzavanja aplikacija?**

Glavni izazovi monolitne arhitekture ukljucuju poteskoce pri izvodjenju velikih promena, nestabilnost aplikacije pri promenama u velikom obimu, kolateralnu stetu koja moze biti generisana od manjih promena, nekontrolisani rast aplikacije i organizaciju koda, kao i poteskoce u odrzavanju i skaliranju kako aplikacija postaje sve kompleksnija.

1. **Koja je glavna mana monolitne arhitekture?**

Monolitna arhitektura se suocava s izazovom rastuce slozenosti i skaliranjem aplikacije jer sve funkcije i komponente aplikacije deluju unutar istog koda. To znaci da kako aplikacija postaje veca i kompleksnija, teze je odrzavati cist i organizovan kod. Skaliranje monolitne aplikacije takodje postaje izazovno jer je tesko dodavati nove resurse ili instance aplikacije bez povecanja kompleksnosti i rizika od gresaka.



**MIKROSERVISNA ARHITEKTURA**

1. **Koje su kljucne karakteristike mikroservisne arhitekture u pogledu komunikacije i organizacije servisa?**

Mikroservisi su najcesce procesi koji komuniciraju putem mreze, koristeci protokole nevezane za odredjeni tip tehnologije. Takodje, organizovani su oko funkcionalnosti i mogu se posebno isporucivati, sto ih cini lako zamenjivim.

1. **Kako se aplikacije zasnovane na mikroservisnoj arhitekturi razlikuju od tradicionalnih monolitnih aplikacija?**

Aplikacije zasnovane na mikroservisnoj arhitekturi prirodno se namecu modularnom strukturom i podrzavaju kontinuiranu isporuku softvera. Za razliku od monolitnih aplikacija, izmena jednog dela aplikacije zahteva ponovno bildovanje i isporuku samo jednog ili malog broja servisa. Takodje, mikroservisne aplikacije se pridrzavaju principa sitno granuliranog interfejsa, poslovno orijentisanog razvoja, poliglotnog programiranja i decentralizovane kontinuirane isporuke, sto ih cini agilnijim i skalabilnijim.

1. **Kako razdvajanje duznosti u mikroservisnoj arhitekturi doprinosi brzini projekta i kontroli evolucije aplikacije?**

Razdvajanje duznosti u mikroservisnoj arhitekturi omogucava posebnim timovima da nezavisno razvijaju svaki mikroservis, sto ubrzava proces projekta. Svaki servis se fokusira na jednu ulogu, pridrzavajuci se principa jedne odgovornosti. Ovo rezultira labavo povezanim servisima koji omogucavaju lokalizovan uticaj promena unutar konkretnog servisa, dok ostatak aplikacije ostaje stabilan i netaknut.

1. **Koji su glavni izazovi i nedostaci mikroservisne arhitekture?**

Glavne mane mikroservisne arhitekture obuhvataju nelogicno razdvajanje, vise mreznih interakcija, problem skladistenja i deljenja podataka, kao i slozenost testiranja. Nelogicno razdvajanje moze dovesti do kompleksnosti i tezine odrzavanja, dok vise mreznih interakcija moze povecati kompleksnost i smanjiti performanse aplikacije. Skladistenje i deljenje podataka mogu biti izazovni zbog potrebe za odrzavanjem nezavisnosti izmedju mikroservisa, a testiranje aplikacija moze biti slozeno zbog velikog broja komponenti. Upravljanje ovim izazovima zahteva pazljivo planiranje, implementaciju pouzdanih mehanizama komunikacije, sinhronizacije podataka i testiranja, kako bi se osigurala stabilnost i efikasnost mikroservisne arhitekture.

------------------------------------------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------------------------------------------------------------

**L03**

**RESTFUL ARHITEKTURA (REPRESENTATIONAL STATE TRANSFER)**

1. **Koje su kljucne karakteristike RESTful arhitekture i zasto je postala popularna?**

Kljucne karakteristike RESTful arhitekture obuhvataju jedinstvenu semu imenovanja resursa, isti interfejs za sve servise, potpunu opisnost poruka i izvrsenje bez stanja. Postala je popularna zbog svoje jednostavnosti i sirokog prihvatanja za veb integraciju servisa.

1. **Koji su prednosti i nedostaci RESTful arhitekture u poredjenju sa drugim pristupima poput SOAP-a?**

Prednosti RESTful arhitekture ukljucuju jednostavnost i fleksibilnost, dok nedostaci mogu obuhvatiti komplikacije kod distribuiranih transakcija i kompleksnih komunikacionih sema. SOAP, s druge strane, definise komunikacioni protokol i propisuje format poruka, sto moze rezultirati vecim brojem parametara po operaciji.

1. **Kako se razlikuju RESTful i SOAP interfejsi, kao i koje su implikacije tih razlika na funkcionalnost aplikacija poput Amazon S3?**

RESTful interfejsi obicno nude samo osnovne operacije poput GET, PUT, POST i DELETE, dok SOAP interfejsi mogu imati vise operacija i ogranicenje u broju parametara po operaciji. Ove razlike mogu uticati na kompleksnost i efikasnost komunikacije izmedju aplikacija. Na primer, u slucaju Amazon S3, REST interfejs se oslanja na URI-je za opis resursa, dok SOAP interfejs ima ogranicenje u broju parametara po operaciji.

**PRETPLATNICKE ARHITEKTURE**

1. **Kakve su osnovne karakteristike koordinacije u pretplatnickim arhitekturama?**

Pretplatnicke arhitekture karakterise koordinacija procesa putem komunikacije i saradnje, gde se procesi posmatraju kao autonomno operativne jedinice. Koordinacija moze biti direktna, putem postanskog sanduceta, zasnovana na dogadjajima ili sa deljenim prostorom za podatke.

1. **Kako se razlikuju sistemi sa deljenim prostorom za podatke od sistemima zasnovanih na dogadjajima?**

Sistemi sa deljenim prostorom za podatke omogucavaju procesima komunikaciju putem struktuiranih podataka, tzv. torki, koje se smestaju u zajednicki prostor. S druge strane, sistemi zasnovani na dogadjajima omogucavaju procesima objavljivanje obavestenja o dogadjajima na koje su zainteresovani, a pretplatnici se pretplacuju na odredjene dogadjaje.

1. **Koje su prednosti i izazovi pretplatnickih arhitektura?**

Pretplatnicke arhitekture nude fleksibilnost i skalabilnost u distribuiranim sistemima, omogucavajuci efikasnu komunikaciju izmedju procesa. Medjutim, izazovi se javljaju u implementaciji efikasnog uparivanja pretplata sa obavestenjima, sto moze postati usko grlo, narocito kada su u pitanju sigurnost i privatnost podataka.

1. **Sta predstavlja magistrala dogadjaja (event-bus)?**

Magistrala dogadjaja (event bus) je mehanizam koji omogucava komunikaciju izmedju razlicitih delova softverskog sistema putem razmene dogadjaja. To je kanal preko kojeg se dogadjaji mogu emitovati (publikovati) od strane jednog dela sistema i primati (pretplatiti se) od strane drugih delova sistema koji su zainteresovani za te dogadjaje.

**ORGANIZAIJCA MIDLVERA**

1. **Kakvu ulogu ima midlver u distribuiranim sistemima i kako se organizuje?**

Midlver predstavlja sloj softvera iznad operativnih sistema racunara u distribuiranim sistemima, omogucavajuci efikasno deljenje i raspodelu resursa preko mreze. Organizuje se tako da sakriva razlike u hardveru i operativnim sistemima od aplikacija i pruza sredstva za komunikaciju izmedju komponenti razlicitih distribuiranih aplikacija.

1. **Koje usluge pruza midlver u distribuiranim sistemima i kako se razlikuju od usluga operativnih sistema?**

Midlver pruza usluge za medjuaplikacionu komunikaciju, zastitu, maskiranje i oporavak od gresaka, i druge. Glavna razlika u odnosu na usluge operativnih sistema je to sto su usluge midlvera prilagodjene mreznom okruzenju. Vecina tih usluga je korisna za mnoge aplikacije, cime midlver funkcionise kao kontejner zajednickih komponenti i funkcija koje aplikacije vise ne moraju implementirati odvojeno.

1. **Zasto je midlver vazan za razvoj distribuiranih aplikacija?**

Midlver je vazan jer omogucava efikasno deljenje resursa i komunikaciju izmedju komponenti distribuiranih aplikacija, cime olaksava razvoj slozenih sistema. Takodje, midlver pruza standardizovane usluge kao sto su medjuaplikaciona komunikacija i zastita, cime smanjuje potrebu za ponovnim implementiranjem tih funkcionalnosti u svakoj aplikaciji pojedinacno.

1. **Koja je svrha omotaca (wrappera) u organizaciji midlvera i kako resavaju problem nekompatibilnih interfejsa?**

Omotaci, ili adapteri, su posebne komponente koje nude interfejs prihvatljiv za klijentsku aplikaciju, transformisuci funkcije komponente u one koje su pogodne za koriscenje u aplikaciji. Oni resavaju problem nekompatibilnih interfejsa omogucavajuci aplikacijama da koriste funkcionalnosti postojecih komponenti.

**OMOTACI (WRAPPERI) I BROKERI**

1. **Sta je brokerski sistem poruka i kako funkcionise?**

Brokerski sistem poruka je tip midlvera koji se sastoji od logicki centralizovane komponente, brokera, koji upravlja komunikacijom izmedju razlicitih aplikacija. Aplikacije salju zahteve brokeru koji ih obradjuje i prosledjuje odgovarajucim aplikacijama, kombinuje i transformise odgovore, i vraca rezultate pocetnoj aplikaciji.

1. **Kako brokerski sistem poruka smanjuje potrebu za razvojem omotaca?**

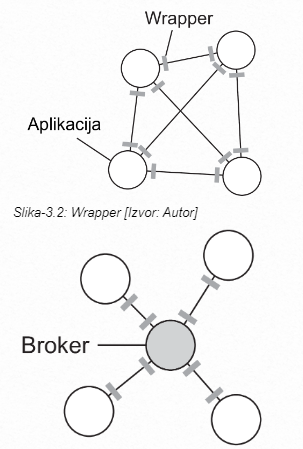
Brokerski sistem poruka smanjuje potrebu za razvojem omotaca tako sto omogucava da svaka aplikacija komunicira samo sa brokerom, umesto direktno sa drugim aplikacijama. Ovo smanjuje broj potrebnih omotaca sa O(N^2) na O(N), sto cini sistem skalabilnijim i efikasnijim.

1. **Kako omotaci doprinose prosirivosti sistema i otvorenosti?**

Omotaci omogucavaju prosirivost sistema tako sto omogucavaju integraciju postojecih komponenti sa novim aplikacijama, transformisuci funkcionalnosti i interfejse komponenti u interfejse prihvatljive za koriscenje u novim kontekstima. Time se postize otvorenost sistema za nove zahteve i promene.

1. **Zasto razvoj omotaca po potrebi moze postati neefikasan i nepraktican?**

Razvoj omotaca po potrebi moze postati neefikasan i nepraktican jer sa porastom broja aplikacija, broj potrebnih omotaca raste kvadratno u odnosu na broj aplikacija. To znaci da bi sa N aplikacija, bilo potrebno razviti O(N^2) omotaca, sto nije skalabilno.



**INTERCEPTORI**

1. **Kako interceptor doprinosi prilagodjavanju midlvera specificnim potrebama aplikacije?**

Interceptor prekida uobicajeni tok kontrole i omogucava izvrsenje aplikacijski specificnog koda, sto omogucava prilagodjavanje midlvera specificnim potrebama aplikacije. To je kljucno za postizanje otvorenosti i fleksibilnosti sistema.

**VISESLOJNE KLIJENT SERVER ARHITEKTURE**

1. **Sta predstavlja osnovnu klijent-server arhitekturu?**

Osnovna klijent-server arhitektura je oblik distribuirane arhitekture u kojoj se procesi u sistemu dele na dve glavne grupe: servere, koji implementiraju odredjenu uslugu, i klijente, koji zahtevaju tu uslugu slanjem zahteva serveru i cekanjem odgovora.

1. **Kako se ostvaruje komunikacija izmedju klijenta i servera u osnovnoj klijent-server arhitekturi?**

Komunikacija izmedju klijenta i servera moze biti ostvarena jednostavnim protokolom bez veze, gde klijent salje zahtev serveru i ceka odgovor, bez odrzavanja stalne veze.

1. **Sta su idempotentni zahtevi u kontekstu klijent-server komunikacije?**

Idempotentni zahtevi su oni zahtevi cije ponavljanje nece promeniti rezultat ili stanje sistema. Ovo je vazno za tretiranje izgubljenih ili dupliciranih poruka u klijent-server komunikaciji.

1. **Koja je prednost pouzdanog protokola orijentisanog ka uspostavljanju veze u odnosu na protokole bez veze u klijent-server arhitekturi?**

Pouzdan protokol orijentisan ka uspostavljanju veze omogucava bolje upravljanje izgubljenim ili ostecenim porukama, sto je kljucno za pouzdan rad u nepouzdanim mreznim okruzenjima.

1. **Koje su dve glavne organizacije u viseslojnim arhitekturama?**

Dve glavne organizacije u viseslojnim arhitekturama su podela na klijentsku i serversku masinu i podela na slojeve korisnickog interfejsa, procesiranja i podataka.

**PEER TO PEER ARHITEKTURA**

1. **Koja je osnovna karakteristika peer-to-peer (P2P) sistema?**

Osnovna karakteristika P2P sistema je da su svi cvorovi jednaki i da svaki cvor istovremeno deluje kao klijent i server.

1. **Koje su dve vrste preklapajucih mreza u P2P sistemima?**

Dve vrste preklapajucih mreza u P2P sistemima su strukturisane i nestruktuirane mreze.

------------------------------------------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------------------------------------------------------------

**L04**

1. **Koja je uloga niti u distribuiranim sistemima?**

Niti igraju kljucnu ulogu u distribuiranim sistemima jer omogucavaju paralelno izvrsavanje zadatka. To znaci da vise niti moze istovremeno izvrsavati razlicite delove programa ili zadatke, sto doprinosi efikasnosti i skalabilnosti sistema.

1. **Sta je virtuelni procesor i kako se koristi u operativnom sistemu?**

Virtuelni procesor je apstrakcija fizickog procesora koja omogucava operativnom sistemu da pokrece vise virtuelnih procesa istovremeno na jednom fizickom procesoru. On se koristi za efikasno upravljanje resursima i omogucava multitasking i paralelno izvrsavanje vise procesa.

1. **Kako funkcionise tabela procesa u operativnom sistemu i koji su kljucni elementi koje sadrzi?**

Tabela procesa je struktura podataka koja se koristi u operativnom sistemu za pracenje svih pokrenutih procesa. Kljucni elementi koji se obicno nalaze u tabeli procesa ukljucuju identifikator procesa (PID), stanje procesa, prioritete, informacije o resursima koje proces koristi (poput memorije i fajlova), informacije o pristupu resursima, informacije o procesorskom vremenu i informacije o roditeljskom procesu. Ova tabela omogucava operativnom sistemu efikasno upravljanje procesima, njihovo pracenje i pruzanje resursa koje procesi zahtevaju za izvrsavanje.

1. **Sta obuhvata kontekst procesa i zasto je vazan u operativnom sistemu?**

Kontekst procesa obuhvata sve informacije o trenutnom stanju procesa koje su neophodne za njegovo privremeno zaustavljanje i kasnije nastavljanje izvrsavanja. To ukljucuje vrednosti registara CPU-a, stanje memorije, otvorene datoteke i ostale resurse koje proces koristi. Kontekst procesa je vazan jer omogucava operativnom sistemu da efikasno upravlja vise procesima tako sto im omogucava da se privremeno zaustave i ponovo pokrenu, cime se postize multitasking i paralelno izvrsavanje.

1. **Sta je interprocesna komunikacija?**

Interprocesna komunikacija (IPC) je proces razmene podataka ili informacija izmedju razlicitih procesa koji se izvrsavaju na istom racunarskom sistemu ili cak na razlicitim racunarima u mrezi. Ova komunikacija omogucava procesima da saradjuju, razmenjuju informacije ili sinhronizuju svoje aktivnosti radi postizanja zajednickog cilja.

1. **Koje su glavne metode interprocesne komunikacije i kako one omogucavaju komunikaciju izmedju razlicitih procesa?**

Glavne metode interprocesne komunikacije (IPC) obuhvataju mehanizme kao sto su razmena poruka (message passing) i deljenje memorije (shared memory).

Razmena poruka omogucava procesima da komuniciraju slanjem poruka jedan drugom. Poruke mogu biti asinhrono ili sinhrono dostavljene, a mehanizmi za to ukljucuju redove poruka, cevovode (pipes), signale i sockets.

Deljenje memorije omogucava procesima da pristupaju zajednickom prostoru u memoriji, omogucavajuci im da dele podatke. Ovo se obicno postize kroz deljene memorijske segmente ili mapiranje fajlova u memoriju.

Obe metode omogucavaju komunikaciju izmedju razlicitih procesa tako sto omogucavaju slanje podataka ili sinhronizaciju izmedju njih, cime se olaksava saradnja i deljenje resursa. Medjutim, svaka metoda ima svoje prednosti i mane, a izbor metode zavisi od specificnih zahteva aplikacije i ogranicenja operativnog sistema.

1. **Koji su glavni nedostaci many-to-one threading modela u implementaciji niti na nivou korisnickog prostora?**

Glavni nedostaci many-to-one threading modela ukljucuju cinjenicu da vise niti mapira na jedan entitet koji moze biti rasporedjen, sto znaci da blokirajuci sistemski pozivi mogu blokirati ceo proces i sve druge niti u tom procesu. Ovo ogranicenje onemogucava efikasno koriscenje visenitnih aplikacija, posebno u situacijama kada nekoliko niti istovremeno radi na razlicitim zadacima.

1. **Koje su prednosti one-to-one threading modela u odnosu na many-to-one threading model kada je rec o implementaciji niti?**

Prednosti one-to-one threading modela ukljucuju to da svaka nit predstavlja jedinicu koja moze biti rasporedjena, sto omogucava da blokiranje jedne niti ne utice na ostale niti. Ovaj model takodje omogucava efikasnu upotrebu viseprocesorskih sistema, jer svaka nit moze biti dodeljena razlicitom procesoru ili jezgru radi paralelnog izvrsavanja. Osim toga, ovaj model olaksava sinhronizaciju i upravljanje nitima, jer operacije sa nitima, poput kreiranja, brisanja i sinhronizacije, mogu biti obavljene direktno od strane jezgra operativnog sistema.

**VISENITNI KLIJENTI**

1. **Sta je vreme ukupnog puta (round-trip time) i zasto je vazno u kontekstu distribuiranih sistema, posebno veb pretrazivaca?**

Vreme ukupnog puta (round-trip time) u kontekstu distribuiranih sistema predstavlja vreme potrebno da se poruka ili zahtev posalje sa izvora do odredista, a zatim da se odgovor vrati nazad na izvor. Ovo vreme ukljucuje vreme potrebno za slanje podataka preko mreze, obradu na odredisnom uredjaju i povratnu putanju podataka do izvora.

1. **Sta je strategija rotacije (round-robin) i kako se koristi u kontekstu distribuiranih sistema?**

Strategija rotacije (round-robin) je jedan od nacina balansiranja opterecenja u distribuiranim sistemima. Ova strategija funkcionise tako sto se zahtevi ili konekcije ravnomerno rasporedjuju medju dostupnim serverima ili resursima, jedan za drugim, u krug. Na primer, kada klijent salje zahtev za veb stranicu, strategija rotacije ce proslediti taj zahtev prvom dostupnom serveru, a zatim sledeci zahtev proslediti drugom serveru, i tako dalje, u krug.

U kontekstu distribuiranih sistema, strategija rotacije omogucava efikasno iskoriscenje resursa jer se opterecenje ravnomerno rasporedjuje medju serverima. Ovo pomaze u izbegavanju preopterecenja pojedinacnih servera i osigurava bolju dostupnost usluge za klijente. Sa visenitnim klijentom, ova strategija omogucava uspostavljanje paralelnih konekcija sa razlicitim replikama servera, sto doprinosi brzem prenosu podataka i smanjenju vremena potrebnog za prikazivanje veb stranica.

1. **Kako visenitni klijent funkcionise u kontekstu veb pretrazivaca i koje su prednosti ovog pristupa?**

Visenitni klijent u kontekstu veb pretrazivaca funkcionise tako sto omogucava istovremeno obavljanje vise zadataka, kao sto su preuzimanje HTML datoteke, slika, ikona, itd., sa veb servera. Kada korisnik otvori odredjenu veb stranicu, glavna HTML datoteka se prvo preuzima, a zatim prikazuje korisniku. Istovremeno, posebne niti se aktiviraju da preuzmu ostale delove stranice, kao sto su slike, ikone i drugi elementi.

Ovaj pristup omogucava sto brze prikazivanje delova veb stranice korisniku, dok se ostali delovi jos uvek preuzimaju. Dok korisnik cita tekst i interaguje sa stranicom, pretrazivac nastavlja sa preuzimanjem preostalih delova stranice. Kako se delovi preuzimaju, prikazuju se korisniku u realnom vremenu, sto doprinosi boljem korisnickom iskustvu.

Prednosti visenitnog klijenta u veb pretrazivacu ukljucuju efikasnije iskoriscenje vremena tokom preuzimanja veb stranica, brze prikazivanje sadrzaja korisnicima, kao i mogucnost otvaranja vise konekcija istovremeno sto moze poboljsati performanse, posebno u slucaju kada je server opterecen ili spor.

**VISENITNI SERVERI**

1. **Kako se dispecer, radne niti (working threads) i endpointovi koriste u arhitekturi visenitnog servera, posebno u kontekstu obrade zahteva u distribuiranim sistemima?**

Dispecer je komponenta visenitnog servera koja prima dolazne zahteve od klijenata i prosledjuje ih odgovarajucim radnim nitima na obradu. Radne niti (working threads) su niti koje izvrsavaju zadatke koje im je dodelio dispecer, kao sto su operacije nad datotekama ili bilo koja druga obrada zahteva. Endpointovi predstavljaju tacke na kojima klijenti mogu pristupiti serveru i slati zahteve. Ovi zahtevi se prosledjuju serveru putem endpointa, gde ih dispecer distribuira radnim nitima na obradu. Ova arhitektura omogucava efikasnu obradu vise zahteva istovremeno u visenitnom okruzenju

1. **Kako se visenitni server realizuje kao masina konacnog stanja (finite state machine)?**

Visenitni server realizuje se kao masina konacnog stanja tako sto svaka nit reaguje na dolazne zahteve kao na dogadjaje. Kada zahtev stigne, nit pregleda trenutno stanje i u skladu sa tim reaguje na zahtev. Ukoliko je moguce, zahtev se obradjuje iz kes memorije. U suprotnom, nit moze zakazati asinhronu operaciju, kao sto je operacija citanja sa diska, i nastaviti sa procesiranjem drugih zahteva. Kada se operacija zavrsi, nit se obavestava i nastavlja sa zavrsetkom obrade zahteva. Ovaj pristup omogucava efikasno upravljanje resursima i smanjenje vremena cekanja, cime se postizu bolje performanse servera.

------------------------------------------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------------------------------------------------------------

**L05**

**VIRTUELIZACIJA**

1. **Sta podrazumeva virtuelizacija racunarskih resursa?**

Virtuelizacija podrazumeva apstrakciju racunarskih resursa, sto ukljucuje razdvajanje resursa racunara u vise zasebnih radnih okruzenja kroz tehnike poput hardverskog ili softverskog particionisanja, timesharinga, masinske simulacije, emulacije i drugih.

1. **Koja je svrha virtuelizacije u IT infrastrukturi?**

Svrha virtuelizacije u IT infrastrukturi je efikasnije koriscenje racunarskih resursa putem omogucavanja pokretanja vise virtuelnih masina na jednoj fizickoj masini, sto omogucava deljenje resursa na nekoliko razdvojenih okruzenja i istovremeno pokretanje razlicitih operativnih sistema i aplikacija.

1. **Kako se virtuelizacija realizuje na fizickom racunaru?**

Virtuelizacija se realizuje na fizickom racunaru kroz izradu virtuelnih masina (VM), softverskih implementacija racunara koje izvrsavaju programe na isti nacin kao prava masina. Kroz ovaj proces, fizicki resursi racunara se deluju na nacin da podrze rad vise izolovanih radnih okruzenja, svako sa svojim operativnim sistemom i aplikacijama.

1. **Sta je Virtual Machine Monitor (VMM)?**

Virtual Machine Monitor (VMM) je komponenta softvera koja omogucava kreiranje i upravljanje virtuelnim masinama na fizickom racunaru. VMM je odgovoran za izolaciju virtuelnih masina jednih od drugih, kao i od fizickog hardvera, omogucavajuci im da deluju kao nezavisni entiteti sa sopstvenim operativnim sistemima i aplikacijama.

1. **Kako se definisu pojmovi host (domacin) i guest (gost) u kontekstu virtuelizacije?**

U kontekstu virtuelizacije, host (domacin) je osnovni racunar na kojem se izvrsava virtualizacija, dok je guest (gost) virtuelna masina koja se pokrece unutar tog osnovnog racunara.

1. **Koja je osnovna razlika izmedju emulacije i virtuelizacije?**

Osnovna razlika izmedju emulacije i virtuelizacije lezi u nacinu na koji se pristupa hardverskim resursima. Emulacija podrazumeva apstrakciju hardvera i arhitekture virtuelne masine, dok se kod virtuelizacije jednostavno izoluje virtuelna masina unutar postojeceg hardvera.

1. **Sta podrazumeva emulacija u kontekstu virtuelnih masina?**

Emulacija u kontekstu virtuelnih masina znaci da se emulira hardverska struktura, na primer x86 arhitektura, i koristi dodatni sloj za prevodjenje i preusmeravanje na nivou gostujuce virtuelne masine.

1. **Kako virtuelizacija funkcionise u odnosu na host racunar i gostujucu virtuelnu masinu?**

Kod virtuelizacije, host racunar propusta izvrsavanje gostujuce virtuelne masine na svom postojecem hardveru. Gostujuca virtuelna masina funkcionise izolovano unutar hosta, bez potrebe za dodatnim slojevima prevodjenja i preusmeravanja, sto omogucava bolje performanse i brzi rad u poredjenju sa emulacijom.

**X86 VIRTUELIZACIJA**

1. **Kako se x86 virtuelizacija realizuje?**

X86 virtuelizacija postize se dodavanjem virtuelnog sloja izmedju hardvera i operativnog sistema.

1. **Koje su prednosti x86 virtuelizacije?**

X86 virtuelizacija omogucava konkurentan rad vise instanci operativnog sistema na istom racunaru, dinamicku raspodelu resursa poput CPU-a, skladista, memorije i I/O uredjaja, te unapredjenje konsolidacije servera, agilnosti baza podataka i kontinuiteta poslovanja.

1. **Koji su osnovni karakteristike privilegovanog i neprivilegovanog pristupa u kontekstu racunarskih sistema?**

U racunarskim sistemima, privilegovani pristup omogucava softveru koriscenje svih hardverskih instrukcija, dok neprivilegovani pristup ogranicava softver na izvrsavanje samo neprivilegovanih instrukcija. Privilegovani rezim se koristi za izvrsavanje operacija koje zahtevaju visok nivo privilegija, poput upravljanja hardverom, dok se neprivilegovani rezim koristi za izvrsavanje korisnickih programa i operacija koje ne zahtevaju visok nivo privilegija.

**VIRTUELNE MASINE**

1. **Koja je uloga hipervizora u virtuelizaciji?**

Hipervizor je specijalizovani softver koji omogucava kreiranje i upravljanje virtuelnim masinama na fizickom racunaru. On emulira hardverske resurse i omogucava svakoj virtuelnoj masini da radi kao zaseban operativni sistem, nezavisno od drugih virtuelnih masina i fizickog hardvera. Na taj nacin, hipervizor omogucava deljenje hardverskih resursa izmedju vise virtuelnih masina, sto povecava iskoriscenje hardvera i olaksava upravljanje racunarskim resursima.

1. **Koje su osnovne razlike izmedju hipervizora tipa 1 i hipervizora tipa 2?**

Osnovna razlika izmedju hipervizora tipa 1 i hipervizora tipa 2 je u tome gde se svaki od njih izvrsava i kako pristupa hardveru. Hipervizor tipa 1, poznat kao "bare metal" hipervizor, radi direktno na hardveru, dok hipervizor tipa 2 radi kao aplikacija unutar postojeceg operativnog sistema. To znaci da hipervizor tipa 1 ne zahteva host operativni sistem, dok hipervizor tipa 2 zahteva host operativni sistem na koji se instalira.

1. **Kako radi hipervizor tipa 1 u odnosu na hipervizor tipa 2 u pogledu performansi i fleksibilnosti?**

Hipervizor tipa 1 pruza bolje performanse i vecu fleksibilnost u poredjenju sa hipervizorom tipa 2. To je zbog toga sto hipervizor tipa 1 radi direktno na hardveru, sto smanjuje overhead i omogucava efikasnije koriscenje resursa. Sa druge strane, hipervizor tipa 2 mora raditi kroz host operativni sistem, sto moze dovesti do veceg overheada i ogranicenja u performansama.

1. **Sta je paravirtuelizacija i kako se razlikuje od tradicionalnih metoda virtuelizacije?**

Paravirtuelizacija je proces postizanja virtuelizacije pri cemu se trajno uklanjaju osetljive instrukcije, umesto da se emuliraju. Umesto toga, hipervizoru se upucuju pozivi za hipervizor, sto omogucava efikasnije izvrsavanje i bolje performanse. Paravirtuelizacija se razlikuje od tradicionalnih metoda virtuelizacije koje pokusavaju da emuliraju kompletnu hardversku arhitekturu, sto moze dovesti do veceg overheada i gubitka performansi.

------------------------------------------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------------------------------------------------------------

**L06**

**UMREZENI KORISNICKI INTERFEJS**

1. **Koje su karakteristike i prednosti umrezenih korisnickih interfejsa?**

Umrezeni korisnicki interfejs omogucava da se sve obradi i skladisti na serveru, sto cini klijentsku masinu terminalom bez potrebe za lokalnim skladistenjem. Ovakav pristup rezultira aplikaciono neutralnim resenjem, sto olaksava upravljanje sistemom i omogucava laksu podrsku razlicitim uredjajima. Ova tehnologija postaje sve popularnija sa sve vecom dostupnoscu interneta i povecanim koriscenjem mobilnih uredjaja.

**X WINDOW SYSTEM (X11)**

1. **Sta je X Window System (X11) i kakvu ulogu ima u kontroli korisnickog interfejsa?**

X Window System, ili X11, je protokol i softverski sistem koji omogucava kontrolu korisnickog interfejsa, ukljucujuci monitor, tastaturu i mis. On se koristi za bit-mape terminala, podrzavajuci tradicionalne i moderne uredjaje poput pametnih telefona i tableta.

1. **Kako X jezgro funkcionise i koje zadatke obavlja u X Window System-u?**

X jezgro obavlja upravljacke zadatke specificne za terminal, pruzajuci interfejs za kontrolu ekrana i hvatanje dogadjaja tastature i misa.

1. **Koja je uloga Xlib biblioteke u X Window System-u i kako aplikacije komuniciraju sa X jezgrom?**

Xlib je biblioteka koja omogucava aplikacijama da komuniciraju sa X jezgrom putem X protokola, slanjem zahteva za kreiranje prozora, postavljanje boja i drugih operacija.

**ANATOMIJA SERVERA**

1. **Kako se organizuju serveri u slucaju iterativnog servera?**

U slucaju iterativnog servera, sam server obradjuje zahtev i, ako je potrebno, vraca odgovor klijentu koji je poslao zahtev.

1. **Sta karakterise konkurentni server?**

Konkurentni server ne obradjuje zahtev sam, vec ga prosledjuje zasebnoj niti ili drugom procesu, nakon cega odmah ceka sledeci dolazni zahtev.

1. **Koja je alternativna implementacija konkurentnog servera?**

Alternativna implementacija konkurentnog servera je kreiranje novog procesa za svaki novi dolazni zahtev, sto se cesto koristi u mnogim Unix sistemima.

1. **Sta je krajnja tacka (endpoint) u kontekstu kontaktiranja servera?**

Krajnja tacka (endpoint) je mesto na koje klijenti salju zahteve, takodje nazvano port, na masini na kojoj server radi. Svaki server slusa odredjenu krajnju tacku.

1. **Kako klijenti obicno saznaju krajnju tacku servisa?**

Jedan pristup je globalno dodeljivanje krajnjih tacaka za dobro poznate servise, poput FTP-a koji koristi TCP port 21 ili HTTP-a koji koristi TCP port 80. Ove krajnje tacke su dodeljene od strane Internet Assigned Numbers Authority (IANA), i klijentu je potrebno samo da pronadje mreznu adresu masine na kojoj server radi. Alternativno, servisi koji ne zahtevaju unapred dodeljene krajnje tacke mogu koristiti dinamicki dodeljene krajnje tacke.

1. **Kako se resava problem pronalazenja krajnje tacke za servise koji koriste dinamicki dodeljene krajnje tacke?**

Jedno resenje je imati poseban proces nazvan daemon koji radi na svakoj masini na kojoj su serveri. Demon prati trenutnu krajnju tacku svakog servisa koji implementira lokalno postavljeni server. Klijent ce prvo kontaktirati daemona, zatraziti krajnju tacku, a zatim kontaktirati specificni server.

1. **Koje su dve osnovne metode za prekid servera prilikom prekida komunikacije sa klijentom?**

Dve osnovne metode za prekid servera prilikom prekida komunikacije sa klijentom su: izlazak korisnika iz klijentske aplikacije, sto automatski prekida vezu sa serverom, ili slanje "out-of-band" (OOB) podataka koji signaliziraju serveru da je potrebno prekinuti vezu.

1. **Sta su "out-of-band" (OOB) podaci i kako se koriste za prekid servera?**

"Out-of-band" (OOB) podaci su podaci koji se obradjuju od strane servera pre bilo kojih drugih podataka od tog klijenta. Koriste se za prekid servera tako sto se serveru salje posebna krajnja tacka na koju klijent salje OOB podatke ili se OOB podaci salju preko iste veze preko koje klijent salje originalni zahtev, na primer, putem TCP-a.

1. **Kako server moze osluskivati OOB podatke za prekid veze sa klijentom?**

Server moze osluskivati OOB podatke tako sto omoguci posebnu krajnju tacku na koju klijent salje OOB podatke i istovremeno osluskuje (sa nizim prioritetom) tacku kroz koju prolaze normalni podaci. Alternativno, OOB podaci se mogu slati preko iste veze preko koje klijent salje originalni zahtev, a server reaguje na primljene OOB podatke prekidom veze.

1. **Koja je bitna razlika izmedju opsteg objektnog servera i klasicnijih servera?**

Bitna razlika je sto objektni server sam po sebi ne pruza odredjenu uslugu, vec usluge implementiraju objekti koji se nalaze na serveru. Server pruza samo nacin za pozivanje lokalnih objekata na osnovu zahteva sa udaljenih klijenata.

1. **Kako objekti funkcionisu unutar objektnog servera?**

Objekti u objektnom serveru se sastoje od dva dela: podataka koji predstavljaju njihovo stanje i koda za izvrsavanje njihovih metoda. Implementacija metoda i razdvajanje podataka zavisi od konkretnog objektnog servera.

------------------------------------------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------------------------------------------------------------

**L07**

**OSI REFERENTNI MODEL**

1. **Sta predstavlja OSI referentni model?**

OSI referentni model je dizajniran da omoguci otvorenim sistemima da komuniciraju preko mreze. To je standardni model koji identifikuje razlicite nivoe ukljucene u komunikaciju, daje im standardna imena i ukazuje koji nivo treba da obavi odredjeni posao.

1. **Koje su dve vrste komunikacionih usluga prema OSI modelu?**

Prema OSI modelu, postoje dve vrste komunikacionih usluga - usluga orijentisana ka povezivanju (connection-oriented service) i usluga bez povezivanja (connectionless service).

1. **Koji je primer usluge orijentisane ka povezivanju prema OSI modelu?**

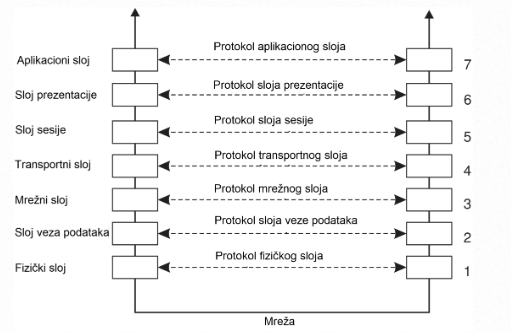
Primer usluge orijentisane ka povezivanju prema OSI modelu je telefon, gde pre razmene podataka posiljalac i primalac eksplicitno uspostavljaju vezu pre nego sto razmene podatke.

1. **Koji je primer usluge bez povezivanja prema OSI modelu?**

Primer usluge bez povezivanja prema OSI modelu je stavljanje pisma u postansko sanduce, gde posiljalac jednostavno salje prvu poruku kada je spreman, bez potrebe za prethodnim podesavanjima.

1. **Koji su sedam slojeva OSI referentnog modela?**

Sedam slojeva OSI referentnog modela su: Fizicki sloj, Sloj veze podataka, Mrezni sloj, Transportni sloj, Sloj sesije, Sloj prezentacije i Aplikacioni sloj.



1. **Koje komunikacione usluge pruza fizicki sloj prema OSI modelu?**

Fizicki sloj se bavi standardizacijom nacina kako su dva racunara povezana i kako se 0 i 1 reprezentuju.

1. **Koje su osnovne funkcije sloja veze podataka prema OSI modelu?**

Sloj veze podataka pruza sredstva za detektovanje i mozda ispravljanje gresaka u prenosu, kao i protokole za odrzavanje sinhronizacije izmedju posiljaoca i primaoca.

1. **Sta radi mrezni sloj prema OSI modelu?**

Mrezni sloj sadrzi protokole za rutiranje poruke kroz racunarsku mrezu, kao i protokole za upravljanje zagusenjem.

1. **Koje usluge pruza transportni sloj prema OSI modelu?**

Transportni sloj pruza protokole koji direktno podrzavaju aplikacije, kao sto su oni koji uspostavljaju pouzdanu komunikaciju ili podrzavaju realno vreme tokom prenosa podataka.

1. **Koje komunikacione usluge pruza sloj sesije prema OSI modelu?**

Sloj sesije pruza podrsku za sesije izmedju aplikacija, ukljucujuci uspostavljanje, odrzavanje i prekid sesija.

1. **Koja je osnovna uloga sloja prezentacije prema OSI modelu?**

Sloj prezentacije propisuje kako se podaci predstavljaju na nacin koji je nezavisan od racunara na kojima se izvrsavaju aplikacije koje komuniciraju, ukljucujuci enkripciju, kompresiju i formatiranje podataka.

1. **Sta obuhvata aplikacioni sloj prema OSI modelu?**

Aplikacioni sloj obuhvata protokole za e-postu, pristup vebu, prenos datoteka i druge aplikacije koje korisnici direktno koriste za komunikaciju preko racunarske mreze.

1. **Kako proces P komunicira sa udaljenim procesom Q prema OSI modelu?**

Proces P formira poruku i prosledjuje je sloju aplikacije kroz interfejs sloja aplikacije. Poruka se zatim prosledjuje kroz razlicite slojeve modela prema sloju fizickog prenosa.

1. **Kako svaki sloj obradjuje poruku tokom njenog prenosa prema OSI modelu?**

Svaki sloj dodaje ili ispituje svoje vlastito zaglavlje na poruku koja prolazi kroz njega, pri cemu se zaglavlje koristi za informacije o protokolu sloja i procesu komunikacije.

**MIDLWARE PROTOKOLI**

1. **Kako se midlver definise u kontekstu OSI referentnog modela, i koje karakteristike ima?**

Midlver je aplikacija koja logicki zivi, uglavnom, u OSI sloju aplikacije, ali koja sadrzi mnogo protokola opste namene koji zasluzuju svoje sopstvene slojeve, nezavisno od drugih, specificnijih aplikacija.

**TIPOVI KOMUNIKACIJE (POSTOJANA & NEPOSTOJANA) (SINHRONA & ASINHRONA)**

1. **Kako se komunikacija razlikuje izmedju postojane (persistant) i nepostojane (transient) komunikacije u kontekstu midlvera u racunarstvu klijent-server?**

Postojana komunikacija podrazumeva da se poruka koja je poslata za prenos cuva od strane komunikacionog middleware-a sve dok se ne dostavi primaocu, cime se omogucava da aplikacija za slanje ne mora nastaviti sa izvrsavanjem nakon slanja poruke.

Suprotno tome, kod nepostojane komunikacije, poruka se cuva u komunikacionom sistemu samo dok aplikacije za slanje i primanje izvrsavaju, i ako komunikacioni sistem ne moze da dostavi poruku zbog prekida u prenosu ili neaktivnosti primaoca, poruka ce biti odbacena.

1. **Kako se razlikuju asinhrona i sinhrona komunikacija u kontekstu racunarstva, i koje su karakteristicne osobine svake?**

Asinhrona komunikacija je karakterizovana time da posiljalac nastavlja odmah nakon slanja svoje poruke za prenos, dok je poruka privremeno smestena od strane middleware-a.

Sa druge strane, sinhrona komunikacija zahteva da posiljalac bude blokiran dok se njegov zahtev ne zavrsi ili ne dobije potvrdu.

**RPC (REMOTE PROCEDURE CALL)**

1. **Sta je udaljeni poziv procedura (RPC) i kako funkcionise u distribuiranim sistemima?**

Udaljeni poziv procedura (RPC) je koncept koji omogucava programima da pozivaju procedure koje se nalaze na drugim masinama u distribuiranim sistemima. Kada proces na masini A pozove proceduru na masini B, pozivajuci proces na A je suspendovan, a izvrsavanje pozvane procedure se odvija na B. Informacije se mogu prenositi od pozivaoca do pozvane procedure u parametrima, a rezultati se mogu vratiti izvrsitelju poziva. RPC omogucava programerima da izbegnu eksplicitno rukovanje komunikacijom izmedju procesa, cime se postize transparentnost pristupa u distribuiranim sistemima.

1. **Koja je uloga server stub-a u udaljenom pozivu procedura (RPC) i kako funkcionise na strani servera?**

Server stub je komponenta u sistemu udaljenog poziva procedura (RPC) koja se nalazi na strani servera. Njegova uloga je da prima poruke koje stizu preko mreze, raspakuje parametre iz tih poruka i poziva odgovarajuce server procedure kako bi izvrsio zahtev klijenta. Server stub transformise zahteve koji dolaze preko mreze u lokalne pozive procedura na serveru, omogucavajuci serveru da izvrsi zahtevane operacije kao da su direktno pozvane od strane klijenta. Nakon zavrsetka izvrsenja, server stub pakuje rezultat u poruku i salje ga nazad klijentu preko mreze.

1. **Sta je marshaling parametara i zasto je bitno u udaljenom pozivu procedura?**

Marshaling parametara se odnosi na proces upakivanja parametara funkcije ili procedure u format koji je pogodan za prenos preko mreze. Ovaj proces je vazan u udaljenom pozivu procedura jer omogucava klijentskom stub-u da pravilno upakuje parametre u poruku koja ce biti poslata serveru.

------------------------------------------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------------------------------------------------------------

**L08**

**KOMANDA MAKE – SINHRONIZACIJA SATOVA U DISTRIBUIRANOM SISTEMU**

1. **Sta je osnovni cilj komande make?**

Osnovni cilj make-a je automatski odrediti koje delove velikog programa treba ponovo kompajlirati i izdati neophodne komande za njihovo ponovno kompajliranje.

1. **Kako make odredjuje koje delove programa treba ponovo kompajlirati?**

Make odredjuje koje delove programa treba ponovo kompajlirati tako sto pregleda vremenske oznake (timestamps) kada su poslednji put modifikovani svi izvorni i objektni fajlovi i uporedjuje ih. Ako je izvorni fajl noviji od odgovarajuceg objektnog fajla, potrebno je ponovno kompajliranje.

1. **U kojim operativnim sistemima je komanda make posebno popularna?**

Komanda make je posebno popularna u Unix, Linux i drugim operativnim sistemima slicnim Unix-u.

1. **Da li bi bilo moguce sinhronizovati sve satove u distribuiranom sistemu?**

Iako je teoretski moguce postici odredjeni stepen sinhronizacije, praksa pokazuje da zbog razlicitih faktora kao sto su kasnjenja u mrezi i razlike u brzini rada satova na razlicitim racunarima, apsolutna sinhronizacija svih satova u distribuiranom sistemu je veoma tesko ostvariva.

1. **Sta moze biti posledica nesinhronizovanih satova u distribuiranom sistemu?**

Nesinhronizovani satovi mogu dovesti do toga da build sistem kao sto je make ne prepozna da je izvorni fajl izmenjen, jer bi mu mogao biti dodeljen stariji vremenski pecat (timestamp) nego odgovarajucem objekt fajlu. Kao rezultat toga, make nece pozvati kompilator, sto moze dovesti do mesavine starih i novih objektnih fajlova u izvrsnom binarnom programu. Ovo moze uzrokovati pad programa, sto ce programeru otezati identifikaciju problema u kodu.

**FIZICKI CASOVNICI**

1. **Sta je racunski tajmer i kako funkcionise?**

Racunski tajmer je precizno izradjen kvarcni kristal koji oscilira na dobro definisanoj frekvenciji kada je pod tenzijom. Uz kristal idu dva registra: brojac i registar za cuvanje. Svaka oscilacija smanjuje brojac za jedan, a kada brojac dodje do nule, generise se prekid i brojac se ponovo ucitava iz registra za cuvanje. Na ovaj nacin, tajmer moze generisati prekide na zeljenoj frekvenciji.

1. **Kako se odrzava tacnost softverskog sata na racunaru?**

Kada se sistem prvi put pokrene, korisnik unosi trenutnu vremensku zonu ili vreme, koje se zatim konvertuje u broj taktova posle nekog poznatog pocetnog datuma i sacuva u memoriji. Vecina racunara ima baterijom podrzanu CMOS RAM memoriju koja cuva datum i vreme, tako da se ne moraju unositi prilikom svakog sledeceg pokretanja. Procedura za obradu prekida dodaje jedan u vreme sacuvano u memoriji prilikom svakog taktovanja sata, cime se softverski sat odrzava azurnim.

1. **Sta uzrokuje odstupanje sata (clock skew) u distribuiranim sistemima?**

Odstupanje sata u distribuiranim sistemima uzrokuje to sto svi n kristali rade sa malo razlicitim brzinama, sto dovodi do toga da softverski satovi postepeno izlaze iz sinhronizacije i daju razlicite vrednosti kada se ocitavaju.

1. **Zasto je sinhronizacija satova vazna u distribuiranim sistemima, cak i ako nije potrebno tacno stvarno vreme?**

Sinhronizacija satova je vazna jer omogucava cvorovima u distribuiranom sistemu da se sloze oko trenutnog vremena ili relativnog vremena dogadjaja. Na primer, za pokretanje make-a je vazno da se cvorovi sloze da je input.o zastareo u odnosu na novu verziju input.c. Ovo omogucava pracenje dogadjaja medjusobno, sto je kljucno za ispravno funkcionisanje sistema.

**LOGICKI CASOVNICI**

1. **Sta su logicki casovnici (logical clocks) i zasto su vazni u distribuiranim sistemima?**

Logicki casovnici su koncept koji se koristi u distribuiranim sistemima da bi se pratio redosled dogadjaja izmedju cvorova. Vazni su jer omogucavaju da se dogadjaji (kao sto je stvaranje nove verzije fajla) prate i sinhronizuju bez potrebe za tacnim stvarnim vremenom, omogucavajuci doslednost i koordinaciju u sistemu.

1. **Sta predstavlja relacija "desi se pre" (happens-before) i kako se primenjuje?**

Relacija "desi se pre" (happens-before) koju je definisao Lamport znaci da se svi procesi slazu da se dogadjaj a desava pre dogadjaja b. Relacija "desi se pre" se direktno primenjuje u dva slucaja:

* Ako su a i b dogadjaji u istom procesu, i a se desava pre b, tada je relacija a → b istinita.
* Ako je a dogadjaj slanja poruke od jednog procesa, a b je dogadjaj prijema poruke od drugog procesa, tada je a → b takodje istinito. Poruka ne moze biti primljena pre nego sto bude poslata, jer je potrebno konacno, nenulto (pozitivno) vreme da stigne.

1. **Kako funkcionise softverski brojac u kontekstu distribuiranog sistema i kako se vrednosti brojaca razlikuju izmedju procesa?**

U distribuiranom sistemu, svaki proces se izvrsava na razlicitim masinama, svaka sa svojim sopstvenim casovnikom implementiranim kao softverski brojac. Ovaj brojac se povecava za odredjenu vrednost svakih T vremenskih jedinica. Medjutim, vrednost kojom se povecava brojac razlikuje se po procesu. Na primer, casovnik u procesu P1 povecava se za 6 jedinica, u procesu P2 za 8 jedinica, a u procesu P3 za 10 jedinica. Ove razlike uzrokuju da se softverski satovi postepeno izlaze iz sinhronizacije, sto moze dovesti do razlicitih vrednosti vremena kada se ocitavaju, i potencijalno do problema u koordinaciji dogadjaja medju procesima.

1. **Kako funkcionise Lamportov algoritam za sinhronizaciju satova u distribuiranim sistemima?**

Lamportov algoritam funkcionise tako sto koristi relaciju "desi se pre" (happens-before) da sinhronizuje satove medju procesima u distribuiranom sistemu. Kada proces posalje poruku, ona nosi vreme slanja prema satu posiljaoca. Kada poruka stigne kod primaoca, ako sat primaoca pokazuje vrednost pre vremena kada je poruka poslata, primalac pomera svoj sat napred tako da bude za jedan veci od vremena slanja. Na ovaj nacin se osigurava da redosled dogadjaja bude ocuvan. Na primer, ako je poruka m3 poslata u 60, mora stici u 61 ili kasnije, pa primalac pomera svoj sat na 61 ili vise. Slicno, poruka m4 koja je poslata kasnije stize u 70, cime se izbegavaju nemoguce situacije sa negativnim vremenima putovanja poruke.

**MEDJUSOBNO ISKLJUCENJE**

1. **Sta je medjusobno iskljucenje i zasto je kljucno za operativne sisteme i distribuirane sisteme?**

Medjusobno iskljucenje je osnovni koncept koji osigurava da konkurentni procesi ili niti ne mesaju kada pristupaju deljenim resursima, kao sto su promenljive, datoteke ili hardverski uredjaji. U operativnim sistemima, medjusobno iskljucenje je neophodno kako bi se sprecili konflikti koji mogu dovesti do ostecenja podataka ili neusaglasenosti. Mehanizmi medjusobnog iskljucivanja, kao sto su zakljucavanja, semafori ili muteksi, namecu pravilo prema kojem samo jedan proces moze pristupiti deljenom resursu u jednom trenutku, dok ostali moraju cekati da se resurs oslobodi. Ovo promovise redovno izvrsavanje i odrzava integritet sistema, sto je kljucno za stabilnost i pouzdanost operativnih sistema. U distribuiranim sistemima, medjusobno iskljucenje je takodje vazno kako bi se sprecilo ostecenje ili nekonzistentnost deljenih resursa, sto je kljucno za konkurentno i pouzdano izvrsavanje procesa u takvim okruzenjima.

1. **Sta su locks (zakljucavanja) i zasto su vazna za operativne sisteme i distribuirane sisteme?**

Locks (zakljucavanja) su mehanizmi koji omogucavaju kontrolu pristupa deljenim resursima tako sto omogucavaju samo jednom procesu ili niti da pristupi resursu u odredjenom trenutku. Oni su vazni za operativne sisteme jer sprecavaju konflikte i neusaglasenost prilikom pristupa deljenim resursima, sto doprinosi stabilnosti i pouzdanosti sistema. U distribuiranim sistemima, locks se koriste na slican nacin kako bi se osiguralo da samo jedan proces moze pristupiti deljenim resursima u isto vreme, cime se sprecava konflikt i ostecenje podataka.

1. **Kako se koriste semafori (semafores) u upravljanju medjusobnim iskljucenjem u visenitnim i viseprocesnim okruzenjima?**

Semafori (semaphores) su mehanizmi medjusobnog iskljucenja koji omogucavaju kontrolu pristupa deljenim resursima kroz koncept brojaca. Oni se koriste u visenitnim i viseprocesnim okruzenjima tako sto procesima omogucavaju da cekaju na slobodan resurs i da ga zatim oslobode kada ga zavrse koristiti. Ovaj mehanizam pomaze u sprecavanju sukoba i trkacih uslova, cime se osigurava da resursi budu pravilno korisceni i da se integritet podataka odrzava.

1. **Sta su mutexi (muteksi) i kako se koriste za obezbedjivanje medjusobnog iskljucenja u operativnim sistemima?**

Mutexi (muteksi) su mehanizmi medjusobnog iskljucenja koji omogucavaju samo jednom procesu ili niti da pristupi deljenom resursu u jednom trenutku. Oni se koriste u operativnim sistemima za sprecavanje trkacih uslova i ostecenja podataka prilikom pristupa deljenim resursima. Mutexi obezbedjuju siguran pristup resursima tako sto procesima omogucavaju da ih zakljucaju prilikom koriscenja i otkljucaju kada zavrse, cime se osigurava redosled i integritet podataka.

1. **Sta su race conditions (uslovi trke) i zasto su problematicni u visenitnim i viseprocesnim okruzenjima?**

Race conditions (uslovi trke) su problematicne situacije koje se javljaju u visenitnim i viseprocesnim okruzenjima kada vise procesa ili niti pokusava pristupiti deljenom resursu istovremeno, sto moze rezultirati neodredjenim ili nekonzistentnim stanjem resursa. Race conditions su problematicni jer mogu dovesti do ostecenja podataka ili neusaglasenosti, te stoga zahtevaju koriscenje mehanizama medjusobnog iskljucenja kao sto su locks, semafori ili mutexi radi sprecavanja ovakvih situacija.

1. **Kako funkcionise centralizovan algoritam za medjusobno iskljucenje u distribuiranim sistemima, i koje su karakteristike ovog pristupa?**

Centralizovan algoritam za medjusobno iskljucenje u distribuiranim sistemima funkcionise tako sto se jedan proces bira kao koordinator. Kada neki proces zeli pristupiti deljenom resursu, salje zahtevnu poruku koordinatoru, navodeci koji resurs zeli da pristupi. Koordinator zatim proverava da li je taj resurs trenutno dostupan. Ako resurs nije zauzet, koordinator odobrava zahtev i salje odgovor procesu traziocu, omogucavajuci mu pristup resursu. U suprotnom, ako resurs vec koristi drugi proces, koordinator odbija zahtev ili ga blokira dok resurs ne postane dostupan. Ovaj pristup centralizovanog medjusobnog iskljucenja je jednostavan za implementaciju i odrzavanje, ali moze biti nepouzdan u slucaju kvara ili preopterecenja koordinatora.

1. **Kako centralizovani algoritam za medjusobno iskljucenje u distribuiranim sistemima obavlja oslobadjanje resursa i kako procesi nastavljaju sa radom nakon oslobadjanja?**

Kada proces zavrsi sa koriscenjem deljenog resursa, salje poruku koordinatoru kako bi oslobodio svoj ekskluzivni pristup resursu. Koordinator, po prijemu ove poruke, uzima prvu stavku sa liste odlozenih zahteva za pristup resursu. Zatim, salje poruku odobrenja tom procesu, omogucavajuci mu pristup resursu. Ako je proces bio blokiran, tj. ako je ovo prva poruka koju prima nakon blokade, proces se odblokira i moze pristupiti resursu. Ako je vec poslata eksplicitna poruka koja odbija dozvolu, proces ce morati periodicno da proverava dolazni saobracaj ili ce kasnije biti blokiran. Kada proces primi odobrenje od koordinatora, moze nastaviti sa radom koristeci deljeni resurs.

1. **Kako distribuirani algoritam koristi Lamportove logicke casovnike i kako funkcionise njegovo resenje za medjusobno iskljucenje u distribuiranim sistemima?**

Distribuirani algoritam koristi Lamportove logicke casovnike kako bi omogucio rasporedjivanje svih dogadjaja u sistemu i jednoznacno odredio redosled dogadjaja. Kada proces zeli pristupiti deljenom resursu, salje poruku koja sadrzi ime resursa, broj svog procesa i trenutno (logicko) vreme svim drugim procesima. Kada proces primi zahtevnu poruku od drugog procesa, akcija koju preduzima zavisi od njegovog trenutnog stanja u vezi sa resursom. Ako proces ne pristupa resursu i ne zeli mu pristupiti, salje poruku OK posiljaocu. Ako vec ima pristup resursu, ne odgovara na zahtev vec stavlja zahtev u red. Ako zeli pristupiti resursu ali ga jos uvek nema, uporedjuje timestamp dolazne poruke sa timestampom poruke koju je sam poslao. Ako dolazna poruka ima nizi timestamp, proces salje nazad poruku OK. Ako njegova sopstvena poruka ima nizi timestamp, proces stavlja dolazni zahtev u red i ne salje nista.

1. **Kako funkcionise algoritam prstena za postizanje medjusobnog iskljucivanja u distribuiranim sistemima?**

Algoritam prstena koristi logicki prsten u kojem su procesi rasporedjeni tako da svaki proces zna ko je sledeci u redu. Kada se prsten inicijalizuje, procesu P0 se daje zeton (token). Token cirkulise oko prstena i prenosi se od jednog procesa do sledeceg u sekvenci. Kada proces dobije token, proverava da li treba da pristupi deljenom resursu. Ako treba, koristi resurs, obavlja potrebne zadatke, i zatim prosledjuje token dalje. Proces ne moze odmah ponovo koristiti resurs sa istim tokenom. Ako proces ne treba resurs, jednostavno prosledjuje token sledecem procesu. Token cirkulise kontinuirano, osiguravajuci da samo jedan proces moze pristupiti resursu u bilo kom trenutku, cime se garantuje medjusobno iskljucivanje. Redosled u kojem token cirkulise sprecava da bilo koji proces bude zanemaren.

------------------------------------------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------------------------------------------------------------

**L09**

**OSNOVE IMENSKIH PROSTORA**

**Imena, identifikatori, adrese**

1. **Sta je ime u distribuiranom sistemu?**

Ime u distribuiranom sistemu je niz bitova ili karaktera koji se koristi da se referise na entitet. Entitet moze biti resurs kao sto su racunari, stampaci, diskovi, datoteke, procesi, korisnici, postanski sanducici, Web stranice, graficki prozori, poruke, mrezne veze, i slicno.

1. **Sta je pristupna tacka i zasto je vazna?**

Pristupna tacka je posebna tacka putem koje mozemo pristupiti entitetu u distribuiranom sistemu. Vazna je jer omogucava izvrsavanje operacija nad entitetom. Ime pristupne tacke naziva se adresa.

1. **Sta je lokacijski nezavisno ime?**

Lokacijski nezavisno ime je ime entiteta koje je nezavisno od njegovih adresa. To ime omogucava fleksibilniju i laksu upotrebu jer ne zavisi od specificnih tacaka pristupa koje se mogu menjati.

1. **Zasto je korisno imati lokacijski nezavisno ime?**

Lokacijski nezavisno ime je korisno jer omogucava referenciranje na entitet bez obzira na promene u njegovim tackama pristupa. To znaci da reorganizacija sistema ili promena servera nece uzrokovati nevazece reference.

1. **Sta je pravi identifikator i koje su njegove osobine?**

Pravi identifikator je ime koje ima sledece osobine:

* Odnosi se na najvise jedan entitet.
* Svaki entitet se referise na najvise jednim identifikatorom.
* Uvek se odnosi na isti entitet (nikada se ne koristi ponovo).

1. **Kako identifikator olaksava jednoznacno referenciranje entiteta?**

Koriscenjem identifikatora, referenciranje na entitet postaje jednoznacno jer svaki identifikator referise na tacno jedan entitet. Na primer, procesi mogu lako proveriti da li referenciraju isti entitet poredjenjem identifikatora. Ovaj test nije moguc sa nejedinstvenim, neidentifikacionim imenima kao sto su uobicajena imena ljudi.

1. **koja je razlika izmedju adrese i identifikatora**

Adresa se obicno odnosi na specificno mesto ili lokaciju, dok se identifikator odnosi na jedinstveni naziv ili oznaku koja se koristi za identifikaciju entiteta. Adresa se moze odnositi na fizicku lokaciju, kao sto je IP adresa za mrezni uredjaj, ili fizicka adresa za postanske usluge. S druge strane, identifikator moze biti apstraktna oznaka, kao sto je korisnicko ime ili broj indeksa, koji se koristi za identifikaciju korisnika ili objekta. Dok adresa moze varirati ili se promeniti u vremenu, identifikator bi trebao ostati konstantan i jedinstven za odredjeni entitet.

1. **Sta predstavlja lokacijski nezavistan identifikator**

Lokacijski nezavisan identifikator predstavlja ime entiteta koje je nezavisno od njegovih adresa. Ovo ime omogucava lako i fleksibilno koriscenje entiteta, cak i kada se adresa promeni ili entitet prebaci na drugu tacku pristupa. Umesto koriscenja adresa kao referenci za entitet, koristi se jedinstveno ime koje nije povezano sa specificnom adresom, vec jasno identifikuje entitet bez obzira na njegovu fizicku lokaciju.

**Staticko Imenovanje**

1. **Sta su nestruktuirana ili ravna imena?**

Nestruktuirana ili ravna imena su identifikatori koji su jednostavno slucajni nizovi bitova. Oni ne sadrze nikakve informacije o tome kako pronaci tacku pristupa povezanog entiteta.

1. **Koja je kljucna osobina nestruktuiranih imena?**

Kljucna osobina nestruktuiranih imena je da ne sadrze nikakve informacije o tome kako pronaci tacku pristupa povezanog entiteta.

1. **Koja dva jednostavna resenja postoje za lociranje entiteta pomocu nestruktuiranih imena?**

Dva jednostavna resenja za lociranje entiteta pomocu nestruktuiranih imena su:

* Difuzno emitovanje (broadcasting).
* Pokazivaci za prosledjivanje (forwarding pointers).

1. **Kako funkcionise difuzno emitovanje u lokalnim mrezama?**

Difuzno emitovanje u lokalnim mrezama funkcionise tako sto se poruka sa identifikatorom entiteta emituje ka svakoj masini na mrezi. Svaka masina proverava da li ima taj entitet i samo masine koje mogu ponuditi tacku pristupa za entitet salju poruku odgovora sa adresom te tacke pristupa.

1. **Koji su nedostaci difuznog emitovanja u vecim mrezama?**

Nedostaci difuznog emitovanja u vecim mrezama ukljucuju trosenje propusnog opsega mreze na zahtevne poruke i prekomerno prekidanje host masina zahtevima na koje ne mogu odgovoriti.

1. **Sta je visestruko emitovanje i kako poboljsava difuzno emitovanje?**

Visestruko emitovanje (multicasting) je tehnika gde samo ogranicena grupa domacina prima zahtev, sto poboljsava difuzno emitovanje smanjenjem broja masina koje moraju obraditi svaki zahtev. Ethernet mreze podrzavaju visestruko emitovanje na nivou podataka direktno u hardveru.

1. **Kako funkcionisu pokazivaci za prosledjivanje?**

Pokazivaci za prosledjivanje funkcionisu tako da, kada se entitet premesti iz jedne lokacije (A) u drugu (B), ostavi iza sebe referencu na svoju novu lokaciju (B). Klijent moze pronaci trenutnu adresu entiteta prateci lanac pokazivaca prosledjivanja.

1. **Koja je svrha ARP protokola i kako funkcionise u racunarskim mrezama?**

ARP (Address Resolution Protocol) protokol koristi se za mapiranje IP adresa na fizicke MAC adrese u lokalnoj racunarskoj mrezi. Kada racunar zeli da komunicira sa drugim uredjajem u istoj mrezi, prvo koristi ARP za otkrivanje fizicke adrese tog uredjaja.

1. **Koje su prednosti pokazivaca za prosledjivanje?**

Prednost pokazivaca za prosledjivanje je njihova jednostavnost. Klijent moze pronaci trenutnu adresu entiteta prateci lanac pokazivaca, sto olaksava lociranje pokretnih entiteta.

1. **Koji su nedostaci pokazivaca za prosledjivanje?**

Nedostaci pokazivaca za prosledjivanje ukljucuju:

* Mogucnost da lanac postane predug i skupo za pracenje ako se entitet cesto krece.
* Potrebu da sve lokacije u lancu odrzavaju svoj deo pokazivaca, sto moze biti tesko za medjunarodne lokacije.
* Ranjivost na prekinute veze, jer gubitak bilo kog pokazivaca moze onemoguciti dostizanje entiteta.

1. **Sta je home lokacija u kontekstu mreza velikog obima?**

Home lokacija je mesto koje prati trenutnu lokaciju entiteta. Obicno se bira kao mesto gde je entitet kreiran i koristi se kao rezervni mehanizam za usluge lociranja bazirane na prosledjivackim pokazivacima.

1. **Kako se home lokacija koristi u Mobilni IP?**

U Mobilni IP, home lokacija je predstavljena home agentom koji se nalazi na lokalnoj mrezi odgovarajucoj mreznoj adresi sadrzanoj u IP adresi mobilnog host uredjaja. Sva komunikacija ka fiksnoj IP adresi mobilnog uredjaja prvobitno je usmerena ka home agentu.

1. **Sta se desava kada mobilni uredjaj predje na drugu mrezu u kontekstu Mobilni IP?**

Kada mobilni uredjaj predje na drugu mrezu, zahteva privremenu adresu (care-of address) koju koristi za komunikaciju. Ova privremena adresa se registruje kod home agenta. Kada home agent primi paket za mobilni uredjaj, proverava trenutnu lokaciju uredjaja i prosledjuje paket na privremenu adresu ako je uredjaj na drugoj mrezi.

**HIJERARHIJSKO IMENOVANJE**

1. **Kako je mreza organizovana u hijerarhijskoj semi imenovanja?**

Mreza je podeljena na kolekciju domena, sa jednim top-level domenom koji obuhvata celu mrezu. Svaki domen moze biti podeljen na vise manjih poddomena, pri cemu najnizi nivo domena, nazvan domen lista, obicno odgovara lokalnoj mrezi ili celiji u mobilnoj mrezi.

1. **Sta je direktorijumski cvor u hijerarhijskoj semi imenovanja?**

Direktorijumski cvor (dir(D)) je cvor koji prati entitete u odredjenom domenu. Postoji direktorijumski cvor za svaki domen, ukljucujuci top-level domen, ciji se direktorijumski cvor naziva korenski cvor.

1. **Koja je prednost hijerarhijske seme imenovanja?**

Prednost hijerarhijske seme imenovanja je u tome sto omogucava efikasno pracenje lokacije entiteta i smanjuje vreme potrebno za prenos poruka unutar manjih domena. Takodje, organizacija u domene i direktorijumske cvorove omogucava skalabilnost i jednostavnije upravljanje informacijama o lokaciji entiteta.

**DNS**

1. **Za sta se prvenstveno koristi Domain Name System (DNS)?**

DNS se prvenstveno koristi za pretragu IP adresa racunara i servera za elektronsku postu.

1. **Kako je organizovan imenski prostor DNS-a?**

Imenski prostor DNS-a hijerarhijski je organizovan kao stablo sa korenom. Oznaka je niz karaktera koji se ne razlikuju po velicini i sastoje se od alfanumerickih karaktera. Oznaka moze imati maksimalnu duzinu od 63 karaktera, a duzina potpunog imena putanje ogranicena je na 255 karaktera.

1. **Kako se predstavlja ime putanje u DNS-u?**

String reprezentacija imena putanje sastoji se od nabrajanja njenih oznaka, pocevsi od desne, i razdvajanja oznaka tackom ("."). Koren je predstavljen tackom.

1. **Kako se naziva svaki cvor u DNS imenskom prostoru?**

Svaki cvor u DNS imenskom prostoru naziva se imenom putanje, pri cemu oznaka vezana za dolaznu granu cvora takodje se koristi kao ime za taj cvor.

1. **Sta se podrazumeva pod domenom u DNS imenskom prostoru?**

Podstablo imenskog prostora DNS-a naziva se domenom, a ime putanje do njegovog korenskog cvora naziva se imenom domena.

1. **Sta je rekurzivni DNS upit?**

Rekurzivni DNS upit je proces u kojem klijent salje upit svom lokalnom DNS resolveru sa zahtevom da razresi naziv domena u IP adresu.

1. **Sta je iterativni DNS upit?**

Iterativni DNS upit je proces u kojem lokalni DNS resolver salje upite vise DNS serverima umesto da sam obradi sve korake procesa razresavanja.

**LDAP**

1. **Sta je LDAP direktorijumska usluga?**

LDAP (Lightweight Directory Access Protocol) direktorijumska usluga je distribuirana usluga za organizovanje i pristup informacijama o korisnicima, resursima i drugim objektima u mreznom okruzenju.

1. **Kakva je struktura LDAP direktorijumske usluge?**

LDAP direktorijumska usluga se sastoji od zapisa, koji se nazivaju direktorijumski unosima. Svaki unos sadrzi kolekciju parova (atribut, vrednost), gde svaki atribut ima odredjeni tip. Atributi mogu imati jednu ili vise vrednosti.

1. **Koja je svrha atributa imenovanja u LDAP-u?**

Atributi imenovanja u LDAP-u sluze za formiranje jedinstvenih imena za svaki zapis u direktorijumskoj usluzi. Ovi atributi se nazivaju relativno razlicita imena (RDN) i omogucavaju identifikaciju svakog zapisa na globalnom nivou. Hijerarhijski organizujuci se u stablo informacija direktorijuma (DIT), ovi atributi omogucavaju organizaciju podataka i efikasno upravljanje LDAP direktorijumskom strukturom.

------------------------------------------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------------------------------------------------------------

**L10**

**OSNOVE REPLIKACIJE PODATAKA**

1. **Koje su dve osnovne svrhe replikacije podataka?**

Dve osnovne svrhe replikacije podataka su povecanje pouzdanosti sistema i poboljsanje performansi. Repliciranje podataka omogucava sistemu da nastavi s radom nakon pada jedne replike, pruzajuci bolju zastitu od ostecenih podataka. Poboljsanje performansi postize se postavljanjem kopija podataka u blizini procesa koji ih koristi, cime se smanjuje vreme pristupa podacima.

1. **Koja je cena replikacije podataka?**

Cena replikacije podataka ukljucuje problem sa konzistentnoscu podataka, gde svaka izmena na jednoj kopiji zahteva modifikacije na svim kopijama kako bi se osigurala konzistentnost. Osim toga, replikacija podataka moze dovesti do povecane upotrebe mrezne propusnosti odrzavanjem svih kopija azurnim.

1. **Kako se replikacija koristi kao tehnika skaliranja?**

Odgovor: Replikacija se koristi kao tehnika skaliranja tako sto omogucava postavljanje kopija podataka blizu procesa koji ih koriste, cime se smanjuje vreme pristupa podacima i resavaju problemi sa skaliranjem u pogledu performansi.

1. **Kako se definise uska konzistentnost (tight consistency) u kontekstu replikacije podataka?**

Uska konzistentnost podrazumeva da su sve kopije podataka u sistemu uvek iste, sto znaci da ce operacija citanja uvek vratiti isti rezultat bez obzira na to na kojoj kopiji je izvrsena. Zahteva se da se azuriranja izvrsavaju na svim kopijama kao jedna atomicna operacija ili transakcija. Medjutim, odrzavanje uske konzistentnosti u distribuiranom okruzenju moze biti izazovno zbog sirokog raspona faktora, ukljucujuci brzinu izvrsavanja operacija i geografsku raspodelu replika.

**MODELI KONZISTENCIJE ZASNOVANE NA PODACIMA**

1. **Kako se klasifikuju operacije nad podacima u kontekstu replikacije podataka?**

Operacije nad podacima se klasifikuju kao operacije pisanja kada menjaju podatke, dok se u suprotnom klasifikuju kao operacije citanja. Ove operacije su bitne u kontekstu replikacije podataka, gde se lokalne kopije podataka cuvaju na vise masina. Operacije pisanja se propagiraju na druge kopije, dok se operacije citanja ocekuje da vrate vrednost koja pokazuje rezultate poslednje operacije pisanja nad tim podacima.

1. **Sta znaci sekvencijalna konzistentnost u kontekstu skladista podataka?**

Sekvencijalna konzistentnost podrazumeva da je rezultat svake operacije nad skladistem podataka isti kao da su sve operacije (citanje i pisanje) svih procesa izvrsene u nekom sekvencijalnom redosledu. Drugim recima, operacije svakog procesa se pojavljuju u nizu u redosledu koji je specificiran programom tog procesa, a svi procesi vide isto preplitanje operacija, iako se one mozda istovremeno izvrsavaju na razlicitim masinama.

1. **Sta podrazumeva model uslovne konzistencije u kontekstu skladista podataka?**

Model uslovne konzistencije zahteva da upisi koji su potencijalno uzrocno povezani budu vidjeni od svih procesa u istom redosledu, dok konkurentni upisi mogu biti vidjeni u razlicitom redosledu na razlicitim masinama. Ovaj model pravi razliku izmedju dogadjaja koji su uzrocno povezani i onih koji nisu, i zahteva da svi procesi prvo vide uzrocno povezane dogadjaje pre nego sto vide dogadjaje koji na njih ne uticu.

**GRUPISANJE OPERACIJA**

1. **Sta je kriticna sekcija i kako se koristi za upravljanje konkurentnoscu programa koji dele podatke?**

Kriticna sekcija predstavlja deo koda u programu koji sadrzi operacije citanja i pisanja nad deljenim podacima. Da bi se sprecili problemi konkurentnog pristupa, kriticna sekcija se zastiti mehanizmima sinhronizacije, poput ENTER\_CS i LEAVE\_CS operacija. Prilikom ulaska u kriticnu sekciju (ENTER\_CS), proces obezbedjuje da su svi podaci u lokalnom skladistu azurni i bezbedno izvrsava operacije citanja i pisanja. Nakon zavrsetka operacija, proces napusta kriticnu sekciju (LEAVE\_CS). Ovo osigurava da se niz operacija citanja i pisanja izvrsava kao atomski izvrsena jedinica, sprecavajuci konkurentne pristupe koji bi mogli narusiti konzistentnost podataka. Semantika ENTER\_CS i LEAVE\_CS operacija moze se definisati kroz koriscenje deljenih sinhronizacionih promenljivih ili katanaca (locks), koji obezbedjuju jedinstveni pristup deljenim podacima.

**MODELI KONZISTENCIJE ZASNOVANE NA KLIJENTU**

1. **Sta karakterise skladista podataka koja nude slab model konzistentnosti i koji su osnovni principi takvih skladista?**

Skladista podataka koja nude slab model konzistentnosti obicno karakterise odsustvo istovremenih azuriranja ili pretpostavka da se takva azuriranja mogu relativno lako resiti. Vecina operacija nad ovim skladistima podataka ukljucuje citanje podataka, dok su azuriranja manje ucestala ili se tretiraju kao retki slucajevi. Ovi sistemi obicno nude slabije forme konzistentnosti, poput konacne konzistentnosti. Principi takvih skladista ukljucuju uvodjenje posebnih modela konzistentnosti zasnovanih na klijentu, sto omogucava sakrivanje mnogih nekonzistentnosti na relativno jeftin nacin. Ovakvi modeli cesto pruzaju efikasan balans izmedju performansi i konzistentnosti u distribuiranim sistemima.

1. **Sta znaci konzistentnost monotonih citanja u kontekstu distribuiranih skladista podataka?**

Konzistentnost monotonih citanja u distribuiranim skladistima podataka znaci da ako proces jednom procita vrednost podatka x, svaka sledeca operacija citanja na x od strane tog procesa uvek ce vratiti istu tu vrednost ili noviju vrednost. Drugim recima, ovaj model konzistentnosti garantuje da proces nikada nece videti stariju verziju x nakon sto jednom vidi odredjenu vrednost x.

1. **Sta podrazumeva konzistentnost monotonog pisanja u skladistima podataka?**

Konzistentnost monotonog pisanja u skladistima podataka podrazumeva da se operacija pisanja od strane procesa na podatku x zavrsava pre bilo koje sledece operacije pisanja na x od istog procesa. Ovo znaci da se operacije pisanja propagiraju u tacnom redosledu ka svim kopijama skladista podataka, cime se osigurava ocuvanje redosleda azuriranja.

1. **Sta podrazumeva konzistentnost "citaj svoje upise" u kontekstu distribuiranih skladista podataka?**

Konzistentnost "citaj svoje upise" podrazumeva da ce efekat operacije pisanja od strane procesa na podatku x uvek biti vidjen od strane sledece operacije citanja na x od istog procesa. Drugim recima, operacija pisanja uvek je zavrsena pre sledece operacije citanja od strane istog procesa, bez obzira na lokaciju na kojoj se ta operacija citanja odvija. Ovo garantuje da korisnik uvek vidi svoje prethodne azuriranja i omogucava dosledno iskustvo koriscenja sistema, cak i u distribuiranim okruzenjima.

1. **Kako se definise konzistentnost "upisi koja prate citanja" u distribuiranim skladistima podataka?**

Konzistentnost "upisi koja prate citanja" znaci da ce operacija pisanja od strane procesa na podatku x nakon prethodne operacije citanja na x od istog procesa garantovano biti izvrsena na istoj ili novijoj vrednosti x koja je procitana. Drugim recima, svaka sledeca operacija pisanja od strane procesa na podatku x ce se izvrsiti na kopiji x koja je azurirana sa vrednoscu koja je najskorije procitana od strane tog procesa. Ovo obezbedjuje doslednost izmedju operacija citanja i pisanja, omogucavajuci procesima da prate i azuriraju najaktuelnije vrednosti podataka koje su procitali.

**UPRAVLJANJE REPLIKACIJOM**

1. **Sta su permanente replike i kako se koriste u distribuiranim sistemima?**

Permanente replike su osnovne kopije podataka u distribuiranom sistemu, cesto smestene na nekoliko servera na jednoj lokaciji ili na ogranicenom broju servera sirom interneta. One sluze kao pocetni skup replika i omogucavaju redundanciju i bolju dostupnost podataka.

1. **Koja je svrha replika iniciranih serverom i kako se koriste?**

Replike inicirane serverom su kopije podataka koje se privremeno postavljaju na razlicite lokacije radi poboljsanja performansi i brzeg odgovora na zahteve. Na primer, kada se na centralnom serveru pojavi neocekivan talas zahteva sa dalekih lokacija, server moze inicirati postavljanje replika na tim lokacijama radi efikasnijeg rukovanja sa zahtevima.

1. **Kako se distribuiraju veb sajtovi i koje su karakteristike ogledanja?**

Distribucija veb sajtova moze se vrsiti na nekoliko nacina, ukljucujuci ogledanje (mirroring). Ogledanje podrazumeva kopiranje veb sajta na ogranicen broj servera, nazvanih ogledala, koji su geografski rasporedjeni sirom interneta. Korisnici obicno biraju jedno od razlicitih ogledala ili se transparentno prosledjuju jednom od njih.

1. **Zasto je vazna replikacija u kontekstu veb hostinga?**

Replikacija je vazna u veb hostingu jer omogucava brze odgovore na zahteve korisnika i poboljsava dostupnost veb sajtova. Sa replikacijom, kopije veb sajtova se postavljaju na razlicite lokacije, sto omogucava distribuiranje opterecenja i smanjuje rizik od prekida usluge.

1. **Sta su klijentski kesevi i kako se koriste u distribuiranim sistemima?**

Klijentski kesevi su kopije podataka koje klijenti privremeno skladiste radi brzeg pristupa podacima. Klijenti ih koriste kako bi smanjili vreme potrebno za preuzimanje ili upisivanje podataka.

1. **Koje su karakteristike protokola zasnovanih na azuriranju sa strane servera?**

Protokoli zasnovani na azuriranju sa strane servera su oni u kojima se azuriranja propagiraju na druge replike cak i bez zahteva tih replika za azuriranjem. Obicno se koriste u situacijama kada je potrebna jaka konzistentnost.

1. **Kada se obicno koriste protokoli zasnovani na azuriranju sa strane servera?**

Oni se obicno koriste kada je potrebna jaka konzistentnost podataka, narocito u okruzenjima gde postoji visok odnos citanja prema azuriranju.

1. **Koje su prednosti i nedostaci push-based protokola u distribuiranim sistemima?**

Prednosti push-based protokola ukljucuju efikasno sirenje azuriranja i brzu konzistentnost podataka. Medjutim, nedostaci ukljucuju potencijalno veliku opterecenost mreze i servera

1. **Sta su protokoli zasnovani na povlacenju i kako funkcionisu?**

Protokoli zasnovani na povlacenju su oni u kojima se server ili klijent zahtevaju od drugih servera da im posalju sva azuriranja koja imaju u tom trenutku. Takodje se nazivaju pull-based protokolima i cesto se koriste kod klijentskih keseva.