Sveučilište u Zagrebu
Fakultet organizacije i informatike Varaždin

Osnove distribuiranih sustava

Seminarski rad iz kolegija Teorija Informacijskih sustava

Dobrica Pavlinušić, student prve godine poslijediplomskog studija
informacijskih sustava, smjer multimedija

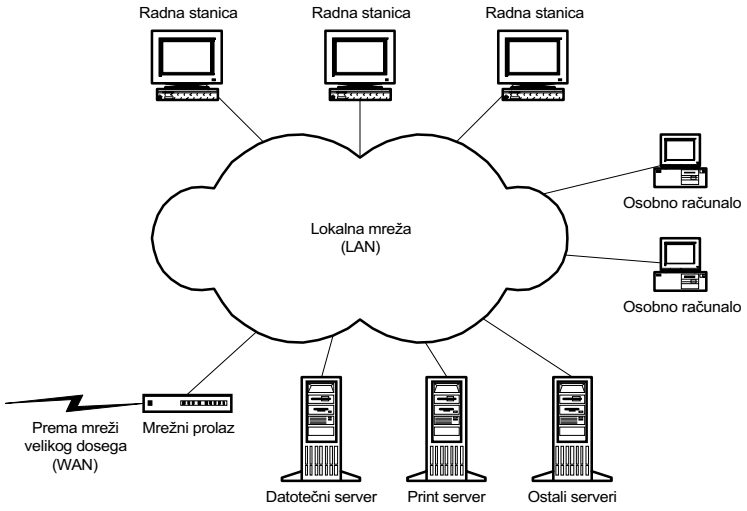
U Varaždinu, 8. siječnja 1998.

Sadržaj:

1. Uvod

Distribuirani sustavi se sastoje od skupa samostalnih računala povezanih računarskim mrežama i opremljenih programskom podrškom za distribuirane sustave. Programska podrška omogućava računalima da koordiniraju svoje aktivnosti i da dijele sustavne resurse - hardware, software i podatke. Korisnici distribuiranog sustava bi trebali primjećivati samo jedan integrirani sustav iako on može biti implementiran pomoću mnogo različitih računala na različitim lokacijama.

Razvoj distribuiranih sustava je slijedio pojavu brzih lokalnih mreža početkom sedamdesetih godina. Pojavom jakih osobnih računala, radnih stanica i servera došlo je do napuštanja centraliziranih i višekorisničkih računala koja su većinom bila smještena u računskim centrima, što je pridonijelo i razvoju distribuirane programske podrške, kao i distribuiranih aplikacija da bi se resursi novih računala efikasnije iskoristili.



Slika 1: Jednostavan distribuirani sustav

2. Osnovne karakteristike

Postoji šest osnovnih karakteristika distribuiranih sustava. To su: dijeljenje resursa (resource sharing), otvorenost (openess), istovremenost[1] (*concurrency*), skalabilnost (*scalability*), otpornost na pogreške (*fault tolerance*) te transparentnost (*transparency*). Niti jedna od tih karakteristika nije posljedica distribuiranih sustava. Distribuirani sustavi moraju biti pažljivo planirani i izvedeni na taj način da osiguraju svaku od tih karakteristika.

2.1 Dijeljenje resursa

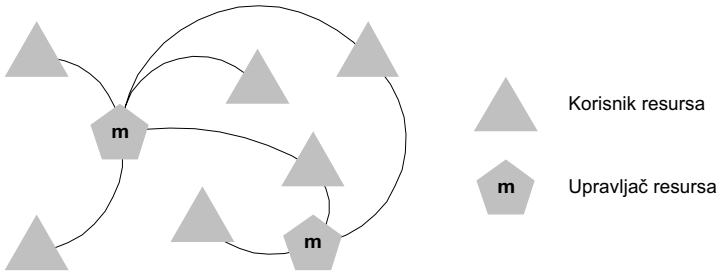
Iako je pojam resursa apstraktan, on najbolje opisuje što je sve moguće dijeliti u distribuiranim sustavima. Prednosti dijeljena resursa kao što su baza podataka, programi, dokumentacija i ostale informacije su se prvi puta pokazale pojavom višekorisničkih sustava[2] te sustava sa podjelom vremena[3] početkom 1960-ih, te pojavom višekorisničkih UNIX operacijskih sustava 1970-ih godina.

- Oprema kao što su štampači, diskovi velikog kapaciteta i ostali uređaji mogu biti dijeljeni zbog lakoće upotrebe i pristupačnosti.
- Dijeljenje podataka je jedan od osnovnih zahtjeva u mnogim računarskim primjenama.
  - Timovi za razvoj programske podrške mogu pristupati dosada razvijenim dijelovima programa i dijeliti iste alate za razvoj i na taj način koristiti samo jednu kopiju prevodilaca, biblioteka i editora. Na taj način nova verzija alata postaje odmah dostupna svim korisnicima.
  - Mnoge komercijalne aplikacije omogućavaju korisnicima korištenje objekata u jednoj zajedničkoj bazi podataka
  - Najperspektivnije područje primjene distribuiranih sustava je podrška grupama korisnika u timskom i kooperativnom radu[4] koje uvelike ovisi o dijeljenu podataka između programa na različitim radnim stanicama.

Izraz **upravljač resursa**[5] opisuje programski modul koji upravlja određenim resursom. Slijedeća slika prikazuje distribuirani sustav sastavljen od upravljača resursa i korisnika resursa. Korisnici resursa komuniciraju sa upravljačima resursa da bi dobili mogućnost korištenja dijeljenog resursa.

---

[1] Originalna engleska riječ je *concurrency* koja se po Random House Webster rječniku objašnjava kao istovremeno izvršavanje ili kooperacija.
[2] Engleski izraz je *multi-user* i opisuje sustav koji omogućuje da više korisnika istovremeno koristi istu računarsku opremu. Problemi koji se pri tome pojavljuju je praćene vlasnika svakoga procesa
[3] Engleski naziv *time sharing* opisuje da se više programa može istovremeno izvršavati. Kako postoji samo jedan procesor koji izvršava program, svaki od programa dobiva svoj djelić vremena po unaprijed određenom algoritmu (Round robin, FIFO i slično).
[4] Engleski naziv za grupni rad je *groupware*, a također se koristi i skraćenica CSCW tj. *Computer supported cooperative working*.
[5] Od engleskog naziva *resource manager*.

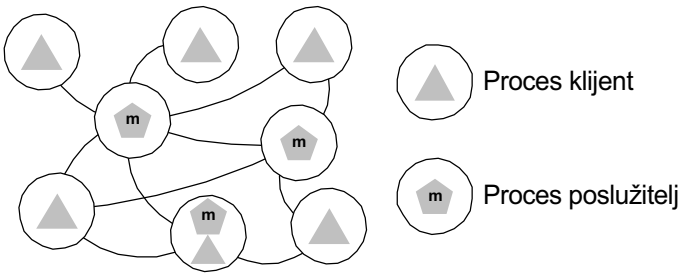


Slika 2: Upravljači i korisnici resursa

Iz takvog razmatranja razvijaju se dva osnovna modela: klijent-poslužitelj model i model zasnovan na objektima.

Klijent-poslužitelj model

To je najviše primjenjivan model distribuiranih sustava. Sastoji se od procesa poslužitelja koji su upravljači resursa određenog tipa i od procesa klijenta koji za izvršavanje svog zadatka zahtijevaju pristup dijeljenim resursima. Sami poslužitelji mogu trebati neke od dijeljenih resursa, pa tako mogu istovremeno biti klijenti nekim drugim poslužiteljima.



Slika 3: Procesi klijenti i poslužitelji

Proces se u ovom kontekstu shvaća jednako kao u operacijskim sustavima: program u izvođenju. Pojednostavljeni pogled na klijent-poslužitelj model može biti centralizirani poslužitelj resursa. Međutim, centralizirano posluživanje je neželjeno u distribuiranim sustavima. Zbog toga se pravi razlika između poslužitelja (*servers*) i usluga koje oni pružaju (*services*). Usluga je apstraktni entitet koji se može pružati preko nekoliko poslužitelja na različitim računalima koja surađuju preko mreže.
Klijent-poslužitelj model je uspješno primijenjen kod dijeljenja različitih resursa: elektronske pošte, mrežnih novosti, kod dijeljenja datoteka, sinkronizacija satova računala u mreži[6], dijeljenja prostora na diskovima i drugim. 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

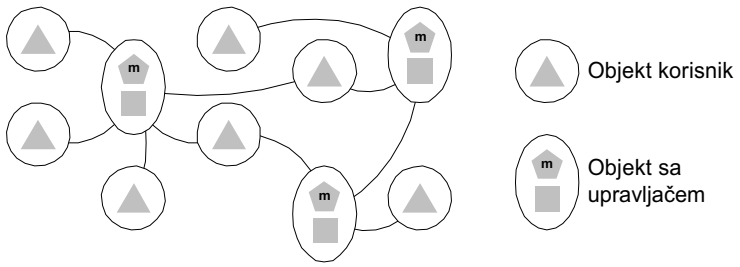
[6] Sinkronizacija satova na mreži važna je u slučajevima kada više klijenata pristupa zajedničkim resursima. U slučaju da jedan od klijenata promjeni podatke na zajedničkom resursu, mora postojati ...nastavak...

mrežno sučelje se smatraju najmanjim skupom resursa koji moraju ostati lokalni[7]. Ti se ključni resursi mogu dijeliti samo među procesima koji se izvode na istom računalu.

Klijent-poslužitelj model ne zadovoljava sve uvjete - neke primjene zahtijevaju čvršću povezanost između klijenta nego što je to moguće u klijent-poslužitelj modelu, međutim, on je pogodan za veliki broj primjena i kao baza za opće distribuirane sustave.

Model zasnovan na objektima

Ovaj model je sličan tradicionalnom objektnom modelu koji se koristi kod programiranja. On svaki izvršni dio programa promatra kao objekt koji posjeduje sustav za razmjenu poruka pomoću kojeg se pristupa do mogućnosti objekta. U objektnom distribuiranom modelu, svaki dijeljeni resurs promatra se kao objekt. Objekti su jednoznačno određeni i mogu mijenjati položaj na mreži bez mijenjanja identiteta. Kada program pokaže potrebu za resursom, on šalje poruku koja sadrži zahtjev za objektom u zajednički red poruka koji dijele svi objekti. Poruka se prosljeđuje odgovarajućoj proceduri ili procesu koji izvodi zahtjevanu operaciju i šalje odgovor ukoliko je to potrebno.



Slika 4: Objekti i upravljači

Slika pokazuje privlačnu jednostavnost takvog modela. On na ujednačen način omogućava pristup **svim** djeljivim resursima od strane objekata korisnika. Kao i kod klijent-poslužitelj modela, objekti mogu biti istovremeno i upravljači i korisnici. Dok je kod klijent-poslužitelj modela način imenovanja ovisio o poslužitelju koji je pružao uslugu[8], kod objektnog modela je imenovanje svih resursa uvijek ostvareno na isti način.

Kod primjene objektnog modela pojavljuju se određeni problemi. On zahtijeva da se upravljači objekata nalaze na istom mjestu kao i objekti kojima oni upravljaju zbog toga što oni posjeduju znanje o trenutnom stanja upravljanog objekta. To je jednostavno kod sustava kod kojih se objekti ne mogu pomicati i taj pristup je primijenjen u većini današnjih objektnih distribuiranih sustava[9]. U

---

način da se svi klijenti o tome obavijeste. Često se koristi rješenje u kojem svaki resurs ima i vremensku oznaku (*time stamp*) zadnje promjene.

[7] U novije vrijeme NC (*network computer*) inicijativa se zapravo zasniva na distribuiranim sustavima. Primijetiti ćete da su upravo ti resursi prisutni u svakom mrežnom računalu.

[8] Primjer toga je SQL baza podataka oblikovana prema klijent-server modelu. Klijent mora znati točno ime servera kojem pristupa da bi mogao poslati svoj upit prema SQL bazi.

[9] Primjeri takvih sustava su Argus, Amoeba i Mach.

eksperimentalnoj fazi nalaze se primjene koje omogućavaju nezavisno razmještanje objekata od njihovih upravljača[10].

2.2 Otvorenost

Otvorenost je karakteristika operacijskih sustava koja definira mogućnost proširivanja na različite načine. Sustav može biti otvoren na sklopovskoj razini - npr. za dodavanje dodatnih vanjskih uređaja, memorije ili komunikacijskih uređaja, ili na programskoj razini - za dodavanje novih funkcija operacijskom sustavu, novih komunikacijskih protokola ili servisa za dijeljenje resursa. Osnovna pretpostavka za otvorenost su standardi koji definiraju sustav na taj način da se može proširivati neovisno o dobavljaču opreme ili programa. Otvorenost distribuiranih sustava se većinom odnosi na mogućnost dodavanja servisa za dijeljenje resursa bez uznemirivanja ili udvajanja postojećih resursa.

Otvorenost se dakle, može svesti na:

- Karakteristiku otvorenih sustava da svoja sučelja publiciraju[11].
- Otvoreni distribuirani sustavi se baziraju na pružanju ujednačenih mehanizama komunikacije među procesima[12] i publiciranju sučelja da bi se omogućio pristup do dijeljenih resursa.
- Otvoreni distribuirani sustavi se mogu graditi od sklopovlja i programske podrške različitih proizvođača. Međutim, da bi se korisnici zaštitili od neusklađenosti, svi dijelovi se moraju brižljivo testirati s obzirom na publicirana sučelja.

2.3 Istovremenost

Kada na istom računalu postoji nekoliko procesa, oni se odvijaju istovremeno. Ako je računalo opremljeno samo sa jednom centralnom upravljačkom jedinicom, onda se to postiže naizmjeničnim izvršavanjem procesa. Ukoliko računalo ima *N* centralnih upravljačkih jedinica, do *N* procesa se može izvršavati paralelno što također rezultira *N*-terostrukim poboljšanjem.

U distribuiranom sustavu se nalazi više računala, svako sa jednom ili više centralnih upravljačkih jedinica. Ako se radi o *M* centralnih upravljačkih jedinica, onda se istovremeno može izvršavati *M* procesa. Istovremenost se postiže u dva slučaja (kao što je to prikazano na slici):

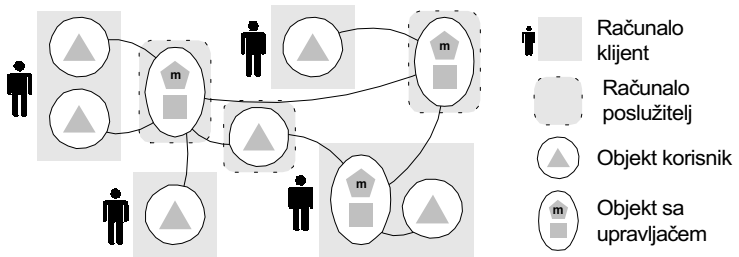
1. Kada više korisnika istovremeno koriste program.
2. Kada više procesa poslužitelja istovremeno poslužuju zahtjeve sa više procesa korisnika.

---

[10] Takvi eksperimentalni sustavi su Arjuna, Clouds i Emerald.

[11] Specificiranjem i dokumentiranjem glavnih načina pristupa sustavu omogućava se standardizacija sučelja. Postupak je sličan formalnoj standardizaciji procedura (koju provode npr. ISO ili ANSI), ali je manje formalan i moguće ga je obaviti u kraćem vremenu.

[12] IPC - *interprocess communication*



Slika 5: Istovremenost kod distribuiranih sustava

Istovremenost i paralelnost se, dakle, pojavljuju prirodno kod distribuiranih operacijskih sustava kao posljedica smještanja procesa poslužitelja na različita računala. Takva podjela procesa rezultira paralelnošću izvršavanja na različitim računalima. Istovremeni pristup i mijenjanje dijeljenih resursa mora biti usklađeno.

2.4 Skalabilnost

Distribuirani sustavi moraju djelovati učinkovito bez obzira na veličinu. Također mora postojati mogućnost proširivanja sukladno sa rastom potreba. Najmanji praktičan distribuirani sustav sastoji se od dvije radne stanice i datotečnog poslužitelja. Distribuirani sustavi sagrađeni oko lokalne mreže mogu imati više stotina radnih stanica i više poslužitelja različitih namjena. Više lokalnih mreža može biti međusobno povezano tako da više tisuća računala zajedno čine jedan distribuiran sustav koji omogućava dijeljenje resursa među njima[13].

Potreba za skalabilnošću nije samo problem opreme ili mrežnih karakteristika. Taj problem prožima sve dijelove distribuiranih sustava. Za razliku od centraliziranih sustava gdje su resursi kao što su memorija, upravljačke jedinice i ulazno-izlazni kanali ograničeni i ne mogu se dodavati proizvoljno, distribuirani sustavi omogućavaju dodavanje takvih resursa bez fiksnih ograničenja. Međutim, ukoliko sustav nije bio planiran sa skalabilnošću u vidu, mogu se pojaviti ograničenja.

2.5 Otpornost na pogreške

Računarski sustavi ponekad zataje. Kada se pogreška pojavi u sklopovlju ili programskoj podršci, mogu se dobiti pogrešne vrijednosti ili nepotpuni rezultati.

Dizajn sustava otpornih na greške temelji se na dva osnovna načela, po jedno za svaki tip grešaka:

- *udvajanje sklopovlja:* korištenjem više istih komponenti sklopovlja,
- *otklanjanje programskih grešaka:* oblikovanje programa na način da se sami mogu oporaviti od grešaka.

Da bi se stvorio sustav otporan na sklopovske greške, često se koriste dva međusobno povezana sustava za korištenje samo jedne aplikacije, gdje jedan služi kao rezerva ukoliko prvi sustav otkaže.

[13] Internet, kao najveća svjetska mreža, sastoji se od mnogo povezanih lokalnih mreža. Njezini servisi su odličan primjer distribuiranog sustava.

Kod distribuiranih sustava redundantnost može biti na mnogo finijem stupnju - pojedini servisi koji su kritični za funkcioniranje sustava mogu biti udvojeni. Udvojeno sklopovlje se može koristiti za nekritične poslove kada oba sustava rade bez grešaka da bi se povećale performanse. Poslužitelji mogu biti dizajnirani na taj način da otkrivaju greške kod obrade u udvojenom sustavu. U tom slučaju oba poslužitelja obrađuju iste podatke, te nakon svake obrade međusobno kontroliraju rezultate. Nakon greške, klijenti se preusmjeruju na drugi sustav. Zbog takvih karakteristika, otpornost na neke pogreške sklopovlja se može osigurati distribuiranim sustavima po relativno maloj cijeni.

Otklanjanje programskih pogreški uključuje vraćanje na prethodno stanje u slučaju pojave pogreške[14]. Raspoloživost sklopovlja je također na visokoj razini, jer se u slučaju kvara radne stanice rad može nastaviti na bilo kojoj drugoj, dok se kod ispada sklopovlja poslužitelja servisi mogu pokrenuti na drugom poslužitelju. Mreže na kojima se temelje distribuirani sustavi najčešće nisu otporne na pogreške. Ispad mreže onemogućava daljnje korištenje distribuiranog sustava do popravka ispada. Mnogo se truda ulaže u oblikovanje mreža otpornih na ispade.

2.6 Transparentnost

Transparentnost se definira kao skrivanje pojedinih komponenti distribuiranog sustava od krajnjeg korisnika i programera aplikacija na način da se sustav shvaća kao cjelina, a ne kao skup nezavisnih dijelova.

Dijeljenje sustava na dijelove je osnovna karakteristika distribuiranih sustava. Posljedice toga uključuju potrebu za komunikacijom među dijelovima i potrebu za upravljanjem i integracijom dijelova. Podjela omogućava stvarni paralelizam u izvođenju programa, prikrivanje ispada opreme i oporavljanje od pogrešaka bez uznemirivanja cijelog sustava, izolaciju i kontrolu komunikacijskih kanala kao mjeru zaštite sadržaja kod prijenosa, te rast ili smanjivanje sustava dodavanjem ili oduzimanjem komponenti.

ANSA Reference Manual i *International Standards Organization's Reference Model for Open Distributed Processing (RM-ODP)* određuju osam tipova transparentnosti. To su i osnovni motivi za izgradnju distribuiranih sustava.

Transparentnost pristupa

omogućava da se lokalni i udaljeni resursi predstavljaju na jednak način[15]

Transparentnost pozicije

omogućava da informacije o objektima ne ovise o fizičkom smještaju

[14] Engleski izraz je *roll back*, i označava vraćanje podataka u stanje prije transakcije kod koje je došlo do programske pogreške.

[15] Primjer transparentnosti pristupa su računala na Internetu kojima možemo pristupati korištenjem *telnet* protokola.

Transparentnost istovremenosti

pruža mogućnost da više procesa istovremeno pristupa podacima bez problema koji se pri tome mogu pojaviti

Transparentnost udvajanja

služi da se poveća pouzdanost i kvaliteta pristupa udvajanjem objekata

Transparentnost na pogreške

omogućava izvršavanje korisničkih zadataka bez obzira na ispade opreme

Transparentnost na promjene lokacije sadržaja

bez utjecaja na korisnike

Transparentnost performansi

omogućava prilagođivanje sustava za veća opterećenja bez uznemirivanja korisnika[16]

Transparentnost na rast

omogućuje rast bez promjena strukture sustava ili algoritama

Dvije najvažnije transparentnosti su transparentnost na pristup i transparentnost pozicije. Njihova prisutnost ili nedostatak najjače utječu na uporabu distribuiranih resursa. Jednim imenom se nazivaju mrežna transparentnost[17].

[16] Jedan od načina je preseljenje procesa na druge procesore ako se pokaže potreba za većim resursima nego što ga trenutni procesor može pružiti.
[17] *Network transparency = access + location transparency*

3. Oblikovanje distribuiranih sustava

Osnovni ciljevi oblikovanja distribuiranih sustava su slijedeći:

- performanse
- pouzdanost
- skalabilnost
- ujednačenost
- sigurnost

Iako oblikovanje distribuiranih sustava zahtijeva razmatranje i ostalih problema nevezanih sa distribucijom, mi ćemo se ograničiti na slijedeće:

- Imenovanje - imena bi trebala biti nezavisna od lokacije
- Komunikacije - potrebno je optimalno koristiti komunikacijske resurse
- Struktura programske podrške - mora osigurati otvorenost, što se postiže definiranjem sučelja
- Raspoređivanje opterećenja - postizanje dobrih svojstava sustava da bi se postigli najbolji rezultati bez obzira na opterećenje sustava
- Ujednačenost sustava - problem ujednačenosti je jedan od najvećih problema distribuiranih sustava.

3.1 Imenovanje

Proces koji zahtjeva resurs sa kojim ne upravlja mora znati njegovo ime ili identifikator. Ime označava naziv koji ima značenje za korisnika ili sustav, dok identifikator označava naziv koji ima smisla **samo** za sustav.

Kažemo da je ime *određeno (resolved)* ili pretvoreno u mrežnu adresu koja je pogodna za pokretanje akcije nad resursom ili objektom na koji se ime odnosi. U distribuiranim sustavima određeno ime (*resolved name*) je najčešće **komunikacijski identifikator** zajedno sa ostalim atributima koji su korisni za uspostavljanje veze[18].

Imenovanje uključuje nekoliko bitnih odluka:

- Određivanje prikladnog prostora imena (*name space*) za svaki tip resursa. Prostor imena može biti beskonačan ili konačan, strukturirani ili jednostavan. Resursi istog tipa moraju imati različita imena, bez obzira na njihovu lokaciju. Kod objektnog sustava, svi objekti dijele isti prostor imena.
- Iz imena se mora moći odrediti komunikacijski identifikator.

Imena se često određuju u zavisnosti od konteksta. Zbog toga je potrebno navesti kako ime, tako i kontekst za to ime. Treba izbjegavati uključivanje mrežne adrese ili druge lokacijske oznake u ime

[18] Kod veza na Internetu komunikacijski identifikator se sastoji od IP adrese i porta (npr. `161.53.120.3:25`). Kod Mach distribuiranog sustava, komunikacijski identifikator je samo oznaka port-a, dok portovi mogu mijenjati svoju lokaciju.

resursa. To direktno narušava transparentnost pozicije spomenutu prije. Određivanje odgovarajućeg sustava imena je veoma važno za distribuirane sustave.

3.2 Komunikacije

Distribuirani sustavi su sastavljeni od dijelova koji su fizički i logički odvojeni i koji moraju komunicirati da bi mogli međusobno djelovati. Pretpostaviti ćemo da su komponente koje upravljaju ili trebaju resurse realizirane kao procesi. To je točna pretpostavka za klijent-poslužitelj model, dok kod objektnog modela praktična realizacija također može počivati na tom principu.

Komunikacija između dva procesa uključuje slanje i primanje što ima za posljedicu:

a) *prijenos podataka* iz procesa koji šalje, procesu koji prima i
b) kod nekih komunikacija *sinkronizaciju* slanja sa primanjem, tako da je proces koji šalje ili prima privremeno zaustavljen dok ne izvrši do kraja akciju koja je inicirala zaustavljanje (npr. pisanje na disk).

Kod slučaja (a) oba procesa dijele isti komunikacijski kanal, dok je ponašanje opisano u slučaju (b) karakteristično za svako komuniciranje.

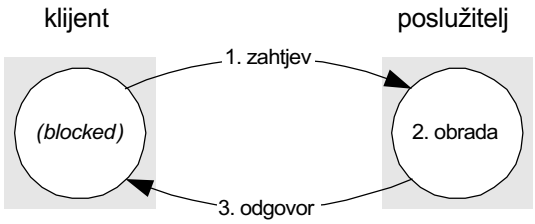
Osnovni način realizacije su šalji (*send*) i primi (*receive*) dijelovi koji zajedno čine akcije za **prijenos poruke** između dva procesa. Akcijom prijenosa poruke se određeni podaci (poruka) koju stvara proces koji šalje, prenose korištenjem određenog komunikacijskog mehanizma (kanala ili porta), do procesa koji podatke prima. Taj mehanizam može biti **sinkroni** (engleski izraz je *blocking*) što znači da pošiljalac čeka dok primalac ne primi poruku ili **asinkroni** (*non-blocking*) kod kojeg se poruka stavlja u niz poruka koje čekaju na primanje, dok proces koji šalje može nastaviti svoje izvršavanje.

Dva osnovna načina komuniciranja su **klijent-poslužitelj komuniciranje** između dva procesa i **grupno slanje** (*group multicast*) za komuniciranje između grupe procesa koji međusobno surađuju.

Klijent-poslužitelj komuniciranje

je orijentirano prema pružanju usluge. Razmjena podataka se sastoji od:

1. proces klijent šalje zahtjev procesu poslužitelju
2. obrada zahtjeva na poslužitelju
3. prijenos odgovora klijentu

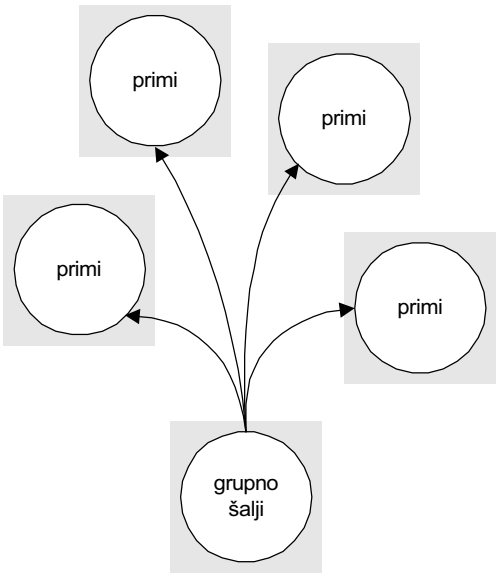


Slika 6: Klijent-poslužitelj komuniciranje

Takav način komuniciranja uključuje prijenos dvije poruke i sinkronizaciju klijenta i poslužitelja. Iako se ovaj način komunikacije može implementirati korištenjem osnovnih operacija šalji i primi, najčešće se upotrebljava iz viših programskih jezika preko **RPC**-a (*remote procedure calling*) sučelja koje skriva komunikaciju od korisnika.

Grupno slanje (*group multicast*)

Kod grupnog komuniciranja procesi razmjenjuju poruke na taj način da svaka poruka ide svim procesima u grupi, a ne samo jednom. Jednom šalji pozivu odgovara više primi poziva, po jedan na svakog člana grupe zbog toga jer dijele zajednički komunikacijski kanal. Takvo slanje naziva se **multicast**[19].



Slika 7: Grupno slanje

Grupno slanje ima slijedeće dobre strane:

- neovisno je o lokaciji objekta kojem je upućena poruka - poruka se šalje svim objektima, dok odgovara samo onaj kojem je namijenjena
- otpornost na pogreške - poruka može biti poslana istovremeno na više poslužitelja, tako da na nju odgovara jedan ili više od njih. Kvar jednoga od poslužitelja klijent ne primjećuje.
- istovremeno usklađivanje svih članova grupe - jedan poslužitelj može poslati vrijeme grupnim slanjem tako da se svi ostali poslužitelji i klijenti sinkroniziraju na to vrijeme.

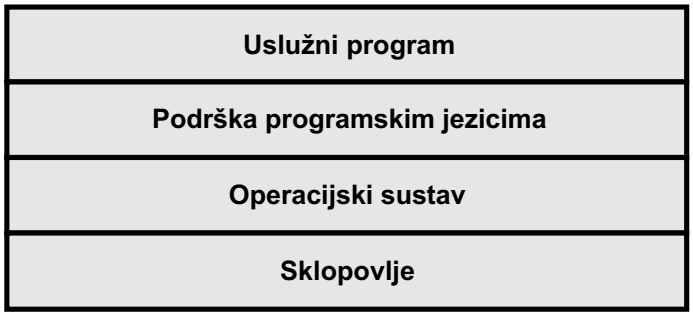
[19] *Multicast* se treba razlikovati od *broadcast*-a koji se često koristi u lokalnim mrežama. Broadcast je slanje svim stanicama spojenim na lokalnu mrežu, a ne samo određenoj grupi stanica, pa se takav način ne primjenjuje na većim mrežama.

3.3 Struktura programske podrške

Kod centraliziranih sustava glavna komponenta je operacijski sustav. On upravlja svim resursima i pruža usluge uslužnim programima i korisnicima koje uključuju:

- osnovno upravljanje resursima
  - dodjeljivanje i zaštita memorije
  - stvaranje i određivanje redoslijeda izvršavanja procesa
  - upravljanje vanjskim uređajima
- usluge programima i korisnicima
  - provjera korisnika i kontrola pristupa
  - upravljanje datotekama i pravima pristupa
  - usluge sata

Sve te servise obavlja *kernel* operacijskog sustava kod centraliziranog sustava.

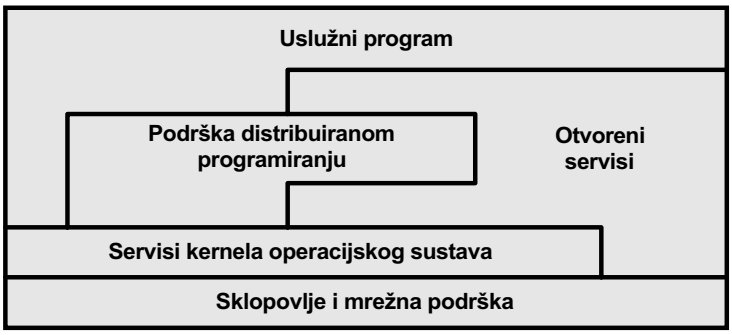


Slika 8: Klasični centralizirani sustav

Distribuirani sustavi pružaju te i druge usluge uslužnim programima na taj način da dodavanje novih servisa ne predstavlja problem. Da bi to bilo moguće, *kernel* se ograničava na pružanje samo osnovnih usluga upravljanja resursima:

- dodjeljivanje i zaštitu memorije
- stvaranje i određivanje redoslijeda izvršavanja procesa
- komunikaciju među procesima, i
- upravljanje vanjskim uređajima

Uvodi se novi tip servisa nazvan otvoreni servisi koji služi za pružanje usluga pristupa svim ostalim dijeljenim servisima i resursima. Svi servisi koji nemaju potrebu za pristupom *kernel*-ovim podacima ili sklopovlju implementirani su na istom nivou kao i uslužni programi.



Slika 9: Osnovni dijelovi distribuiranog sustava i njihov odnos

Svako od područja na gornjoj slici 9 predstavlja skup programskih ili sklopovskih komponenti koje se razlikuju po načinu korištenja i oblikovanju.

3.4 Raspoređivanje opterećenja

Kod klasičnih centraliziranih sustava, svi resursi centralne jedinice i memorije su na raspolaganju operacijskom sustavu u skladu sa trenutnim opterećenjem. Kod najjednostavnijih distribuiranih sustava resursi centralne jedinice i memorije su ograničeni najvećim raspoloživim resursom na jednoj od radnih stanica. Takav jednostavan model naziva se **radna stanica-poslužitelj** model. Smještanje tih resursa "blizu korisniku" (na njegovu radnu stanicu) ima mnoge prednosti naročito kod interaktivnih primjena i to je glavna prednost modela radna stanica-poslužitelj. Međutim, iz toga slijede i ograničenja: model ne koristi optimalno resurse niti omogućava korisniku sa velikim zahtjevima pristup do dodatnih resursa.

Zbog toga su se razvile dvije modifikacije tog modela. **Procesorski pool** je prva od njih, i uključuje dinamičko dodjeljivanje procesorskih resursa korisnicima. Druga varijanta je **upotreba neiskorištenih radnih stanica** koja koristi stanice koje se trenutno ne upotrebljavaju. Jedna od opcija su i **višeprocesorski sustavi sa dijeljenom memorijom** koji se mogu uspješno primijeniti u oba slučaja, naročito kod velikih opterećenja koja se onda mogu podijeliti na više procesora.

Procesorski pool

Kod procesorskog pool-a procesori se dodjeljuju procesima za cijelog njihovog izvršavanja, što rezultira dijeljenjem "procesora po procesu". Korisnik sa više procesa može iskoristiti više procesorske snage nego što jedna radna stanica može ponuditi[20].

Procesorki pool sastoji se od skupine jeftinih računala, od kojih se svako sastoji samo od procesora, memorije i mrežnog sklopovlja. Svaki procesor u pool-u ima svoj priključak na mrežu, kao i radne

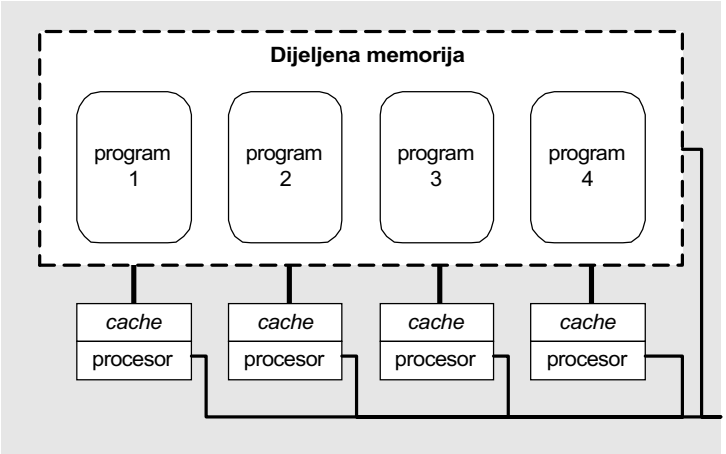
[20] Najčešći primjer je prevođenje C programa napisanog u više dijelova istovremeno. Kod njega se svaki dio programa prevodi istovremeno na svom procesoru. Kod procesorskog pool-a se program iz *n* dijelova može prevesti na *n* procesora uz *n*-terostruko skraćenje prevođenja uz uvjet da su dijelovi međusobno neovisni.

stanice te serveri. Procesori su smješteni na ploče, koje se montiraju u za to predviđene ormare (*rack*-ove). Procesori mogu biti različiti po arhitekturi da bi se omogućilo izvršavanje različite programske podrške.

Upotreba neiskorištenih radnih stanica

Druga zanimljiva mogućnost je korištenje radnih stanica koje su neiskorištene ili malo iskorištene kao dodatnih izvora procesorskih resursa, slično procesorskom pool-u. Primjeri takvih sustava su korištenje crva (*worms*) koji putuju mrežom[21], traže neiskorištene stanice i tamo obavljaju svoje poslove, i premještanje procesa (*process migration*) koje omogućava da se procesi vrate svom "izvorštu" u slučaju da se stanica počne više iskorištavati.

Višeprocesorski sustavi sa dijeljenom memorijom



Slika 10: Višeprocerki sustav sa dijeljenom memorijom

Slika prikazuje jedan od načina rada paralelnih računala. Takav se pristup često koristi kod poslužitelja otvorenih distribuiranih sustava zbog toga što je veoma učinkovita u odnosu na svoju cijenu. Višeprocesorski sustav se sastoji od dva ili više nezavisnih procesora koji izvršavaju procese i međusobno dijele memoriju.

3.5 Ujednačenost sustava

Ujednačenost (*consistency*) je važan problem kod distribuiranih sustava zbog dijeljenja resursa i istovremenosti. Ovdje ćemo razmotriti samo najvažnije.

Ujednačenost kod obnavljanja sadržaja

Ovaj se problem javlja kada nekoliko procesa pristupa i mijenja podatke istovremeno. Mijenjanje podataka ne može biti trenutno, ali bi trebalo biti atomarna operacija, tako da svim ostalim procesima

[21] Ovdje se ne misli na poznati slučaj Internet Crv-a koji je prvi poznati računarski virus tog tipa, već na istraživanja provođena u *Xerox Palo Alto Research Lab (PARC)*.

izgleda kao da je trenutno. Problem se nije pojavio sa distribuiranim sustavima, već postoji svugdje gdje se dijele podaci.

Ujednačenost kod udvajanja

Kada se podaci sa jednog izvora udvoje na više različitih lokacija javlja se problem ukoliko se oni i promijene na jednoj od lokacija. Sve ostale lokacije moraju biti obaviještene o takvoj promjeni, da bi sve kopije podataka bile identične.

Ujednačenost pričuvne memorije (*cache*-a)

Problem ujednačenosti pričuvne memorije je specijalni slučaj problema kod udvajanja podataka i rješava se na slične načine[22].

Ujednačenost kod neotklonjivih grešaka

Kada kod klasičnog sustava dođe do greške koja prekida izvršavanje obrade, svi procesi se zaustavljaju. Međutim, kod distribuiranih sustava vjerojatno je da će kod ispada jedne od komponenti sustava ostali procesi biti u različitim točkama izvršavanja. Zbog toga je potrebno vratiti se na zadnje poznato i ispravno stanje (*roll back*) da bi se obrada mogla nastaviti.

Ujednačenost sata

Mnogi od algoritama distribuiranih sustava ovise o vremenskim oznakama (*time stamps*), tako da je ujednačenost satova kod distribuiranih sustava veoma bitna.

Ujednačenost korisničkog sučelja

Korisničko sučelje bi trebalo reagirati na komande korisnika dovoljno brzo da omogući praćenje promjena koje se dešavaju radom korisnika. Vrijeme kašnjenja se sastoji od:

- vremena potrebnog da se korisnički unos primi, obradi, da se izračunaju potrebne promjene na ekranu i
- vremena potrebnog da se promjene prenesu do korisničkog sustava prozora i da se obnovi slika na ekranu.

Po ergonomskim studijama, to vrijeme bi trebalo biti manje od 0.1 sekunde.

[22] Ovaj problem se najčešće dešava kada jedan od procesora želi promijeniti podatke u glavnoj memoriji, a neki od drugih procesora ima taj isti dio memorije u svojoj pričuvnoj memoriji.

4. Zaključak

Distribuirani sustavi danas postaju uobičajena pojava. Upotrebljavaju se u mnogim područjima, od višekorisničkih sustava opće namjene kao što je to UNIX, do mrežnih informacijskih servisa i multimedijskih aplikacija.

Oni omogućuju velike prednosti svojim korisnicima: dijeljenje resursa, otvorenost, istovremenost, skalabilnost, otpornost na pogreške i transparentnost.

5. Literatura

George Coulouris, Jean Dollimore, Tim Kindberg: Distributed systems, concepts and design; Queen Mary and Westfield College, University of London, Addison-Wesley Publishing Company, second edition, 1995.

Povijest ovoga dokumenta:

1. originalna verzija koja je predana kao seminarski rad završena je 8. siječnja 1998.
2. prva revizija koja ispravlja neke nejasnoće na koje mi je skrenula pažnju Sandra Medenjak napravljena je 4. lipnja 1999.