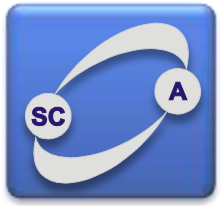


UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA
KATEDRA ZA AUTOMATIKU I UPRAVLJANJE SISTEMIMA

Procesi

Distribuirani sistemi

Distribuirano programiranje

Procesi u operativnom sistemu

- Iz ugla operativnog sistema (OS): **Proces je program koji se izvršava**
- Zadatak OS je da rukuje i raspoređuje procese na izvršenje (priključi na procesor)
  - Svaki proces ima utisak da je procesor samo njegov
    - OS pomaže u transparenciji: Procesi dele CPU i druge hw resurse
  - OS brzo smenjuje procese u izvršavanju tako da se stiče utisak njihovog jednovremenog rada – *multitasking*

- Više procesa može koristiti isti resurs
  - OS nudi mehanizme za pristup deljenim resursima
  - Ovim aspektima se bavi konkurentno programiranje

Procesi i niti

- Proces
  - Izvršava program
  - Stanje procesa (Kontekst procesa) je veliko
    - CPU kontekst (CPU registri)
    - memorijske mape - Modifikovanje registara u MMU
    - informacije za upravljanje procesima – npr. blokiranje na *mutex*-u
    - otvorene datoteke, brojačke (statističke) informacije, privilegije, ...
- Nit (*thread*)
  - Izvršava deo zadatka procesa – takođe se priključuje na procesor
  - Kontekst niti je manji od konteksta procesa
    - OS ga brže sačuva i restaurira

Niti u nedistribuiranom sistemu

- Proces sa jednom niti je često blokiran
  - Npr. čekanje na unos sa tastature, čekanje na odgovor servera, ...

- Višestruke niti (*multithreading*)
  - koriste snagu multiprocesorskog sistema – bolje performanse
    - jednovremeno se izvršavaju i na jednom procesoru
  - Komunikacija između niti je preko deljene memorije
    - Potrebna je sinhronizacija pristupa toj memoriji
  - Pogodne su za velike aplikacije

- Više niti/procesa olakšavaju softversko inženjerstvo
  - Izdvajanje celina
  - Kodiranje zahteva dodatan intelektualan napor

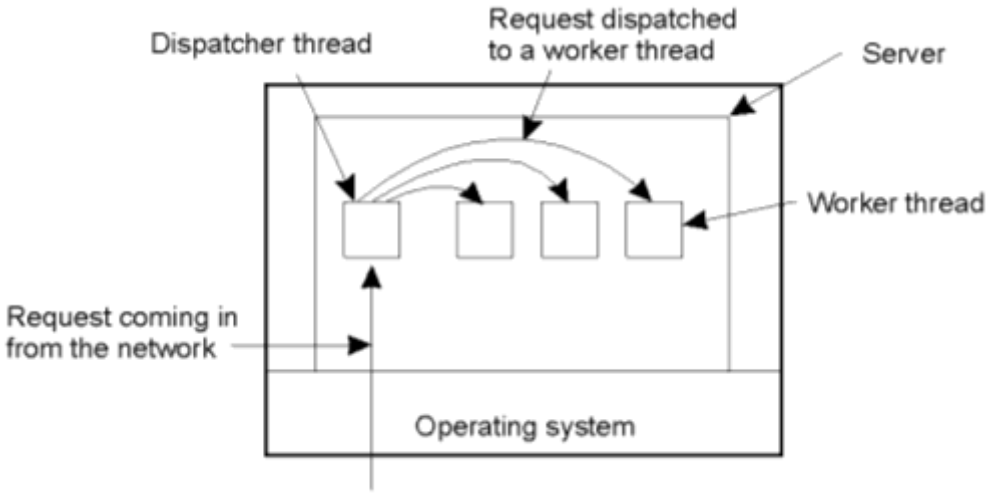
Procesi/niti u distribuiranom sistemu

- Efikasna organizacija klijent-server veze zahteva više programskih niti
  - Jednovremeno izvršavanje komunikacije i lokalne obrade podataka

- Klijenti sa više niti
  - Višestruke komunkacije sa blokirajućim pozivima
  - Izdvojena obrada podataka koji pristižu
  - Jednovremeno pribavljanje podataka od više servera
  - Primer: Web browser
- Serveri sa više niti
  - Jednovremeno opsluživanje više klijenata
  - (ostale već navedene prednosti)

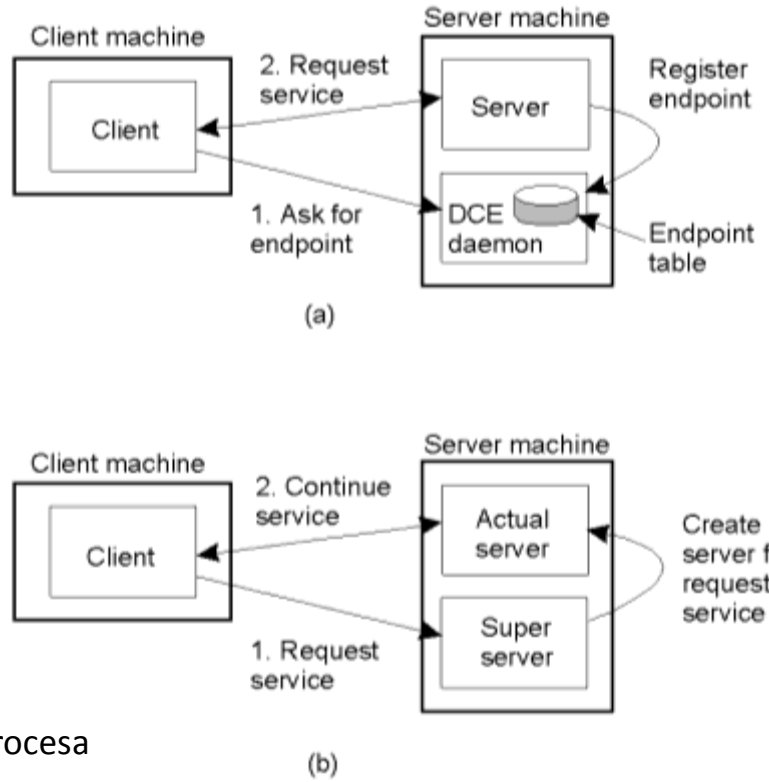
Dizajn servera

- Server je proces koji pruža neku uslugu klijentima (servis)
  - Čeka na zahtev klijenta
  - Procesira zahtev
  - Čeka na novi zahtev
- Organizacija servera
  - Iterativan server – sam obrađuje zahtev i vraća rezultat
  - Konkurentan server – zahtev prosleđuje drugoj niti/procesu i odmah je spreman da prihvati novi zahtev
    - *dispatcher/worker* model



Povezivanje klijenta na server

- Klijenti kontaktiraju server preko porta (*endpoint*-a)
  - Server “sluša” na određenom portu
    - više servera u istom računaru koristi razne portove
  - Kako klijent zna koji port da koristi?
    - Za neki servis se zna TCP port:
      FTP – 21; HTTP – 80; ...
    - Upotreba *daemon* procesa



a) Klijent-server povezivanje upotrebom *daemon* procesa
b) Klijent-server povezivanje upotrebom *Superserver*-a

Objektno-orijentisana arhitektura servera

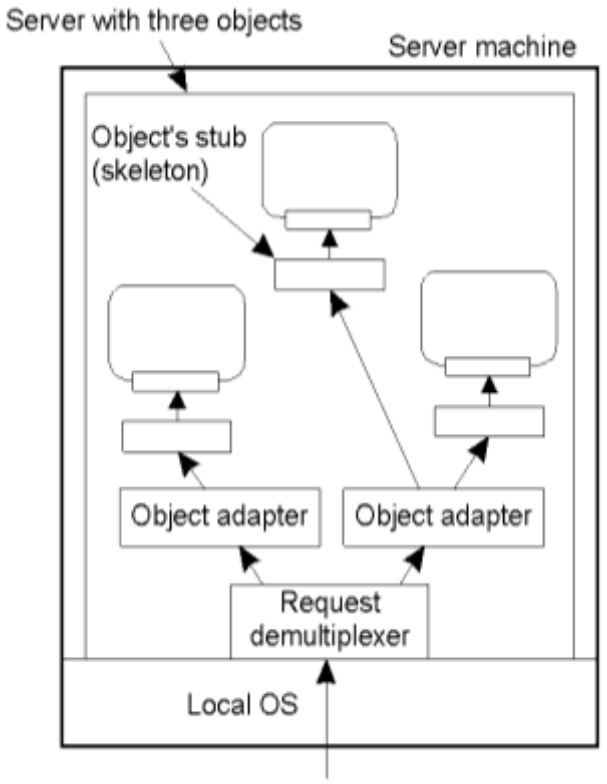
- Unutar servera postoje **lokalni objekti** koji pružaju usluge
  - Relativno je jednostavno širiti skup usluga dodavanjem novih objekata

- **Poseban objekat** prihvata pozive klijenata i prosleđuje ih (raspoređuje) lokalnim objektima
  - Politike aktivacije – pre poziva metode kod objekta se mora učitati
    - npr. objekat postoji samo tokom poziva klijenta
    - npr. objekat se smešta u izolovan adresni prostor
    - ...
  - Politika u odnosu na niti
    1. Jedna kontrolna nit za ceo server (najjednostavniji pristup)
    2. Za svaki objekat posebna nit
       - Server prosleđuje poziv niti objekta
       - Ako je nit zauzeta, novi poziv se privremeno stavlja u red
       - Serijalizacija poziva metoda objekta - Jednostavna implementacija
    3. Svaki poziv u zasebnoj niti
       - Metode implementiraju zaštitu od konkurentnog pristupa

Adapter objekta

- Jedan server može imati više politika aktivacija u isto vreme
  - Sprovode ih Adapteri objekata

- Adapter objekta
  - implementira određenu politiku aktivacije (može se konfigurisati)
  - je generička komponenta
    - Ima jedinstven interfejs koji ne zavisi od interfejsa objekta
  - može upravljati sa više objekata



Migracija koda

- Razlozi migracije koda (Premešta se kod, a ne podaci)
  - Prebacivanje sa opterećenog na neopterećen CPU (računar)
  - Često se mininizuju komunikacije (a ne opterećenje CPU)
    - Primer: obrada velike količine DB podataka se može preseliti na server
- Modeli migracije koda
  - *Weak mobility* – Nakon prenosa proces se pokreće od početka
    - Primer: *Java applet*
  - *Strong mobility* – Proces se privremeno zaustavi, prenese i nastavi da radi
- Problemi migracije
  - Pristup lokalnim resursima
    - Kod koji koristi lokalne resurse se nekada ne može migrirati
  - Heterogeni sistemi (drugačiji hardver)
    - Upotreba skript jezika i virtuelne mašine (npr. Java, C#)

Softverski agenti

- Softverski agenti su autonomne jedinice sposobne da obavljaju zadatak u saradnji sa drugim agentima (po mogućnosti udaljenim)
  - Agent je proces
  - Sposoban je da preuzme inicijativu
  - Npr. agenti koji dogovaraju sastanak

- Agentska tehnologija predstavlja osnovu za rad agenata
  - Podržane su migracije