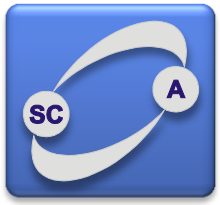


UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA
KATEDRA ZA AUTOMATIKU I UPRAVLJANJE SISTEMIMA

Konkurentno i distribuirano programiranje

Distribuirani sistemi

Distribuirano programiranje

Potreba

- Danas razvoj softvera mora uključiti tehnike za implementaciju paralelnog programiranja (PP) i distribuiranog programiranja (DP)
  - Aplikacije treba da se izvršavaju mreži: intranet-u ili Internetu
  - Tipično se traže dobre performanse (brzo izvršavanje)
- Softver mora biti dizajniran da uposli više procesora i/ili više povezanih računara

- Neki primeri zahteva su:
  - Višestruki i jednovremeni *download*-ovi podataka sa Interneta
  - Sofver dizajniran za emitovanje video zapisa mora renderovati grafiku i procesirati zvuk jednovremeno i tečno bez trzavica
  - Web server treba jednovremo da opslužuje veliki broj korisnika
- Da bi ispunila zahteve korisnika savremena softverska rešenja moraju biti složenija i “pametnija” od ranijih.

Izazovi

- Pisanje paralelnih i distribuiranih programa ima 3 izazova:
  - Identifikovanje prirodnog paralelizma u problemu koji se rešava
  - Podela softvera prema zadacima koji se mogu jednovremeno izvršavati
  - Koordinacija tih zadataka da bi izvršavali osnovni zadatak
    - Efikasno!
- Izazove prate poteškoće
  - Deljenje resursa*: Data race*, *deadlock* i njihovo otkrivanje
  - Delimični otkazi i lokalizovanje grešaka
  - Merenje vremena i sinhronizacija zadataka
  - Komunikacione greške i “ćutanje” sagovornika
    Razlike u protokolima
  - Nepostojanje globalnog stanja i centralizovane kontrole resursa
  - ...

Konkurentnost

- Za događaje kažemo da su konkurentni kada se pojave u istom vremenskom intervalu (VI)
- Zadaci koji se izvršavaju u istom VI su **konkurentni zadaci**
  - Podrazumeva se da zadaci postoje u istom VI
- Tehnike konkurentnosti se koriste da bi program
  - jednovremeno opsluživao više korisnika
  - uradio više tokom istog perioda
    - Ako se zadaci izvršavaju jedan iza drugog postoje čekanja gde se gubi vreme
  - bio jednostavniji.
- Dobrim dizajnom program se može podeliti na zadatke koji se mogu izvršavati jednovremeno/distribuirano

- Primer: dijeta i vežbanje imaju smisla kada se zajedno sprovode
  - jednovremeno? U istom času: jedemo i vežbamo? U istom periodu – mesecu: prve 2 nedelje jedemo, a naredne vežbamo?

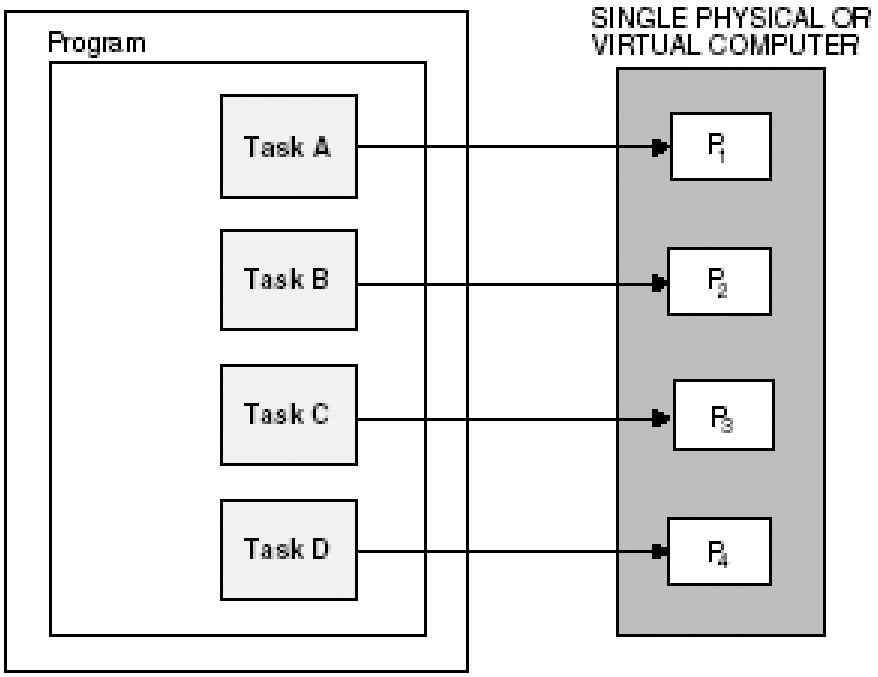
PP, DP i konkurentnost

- PP i DP su dva osnovna pristupa za postizanje konkurentnosti
  - Radi se o dve paradigme koje se nekad prepliću

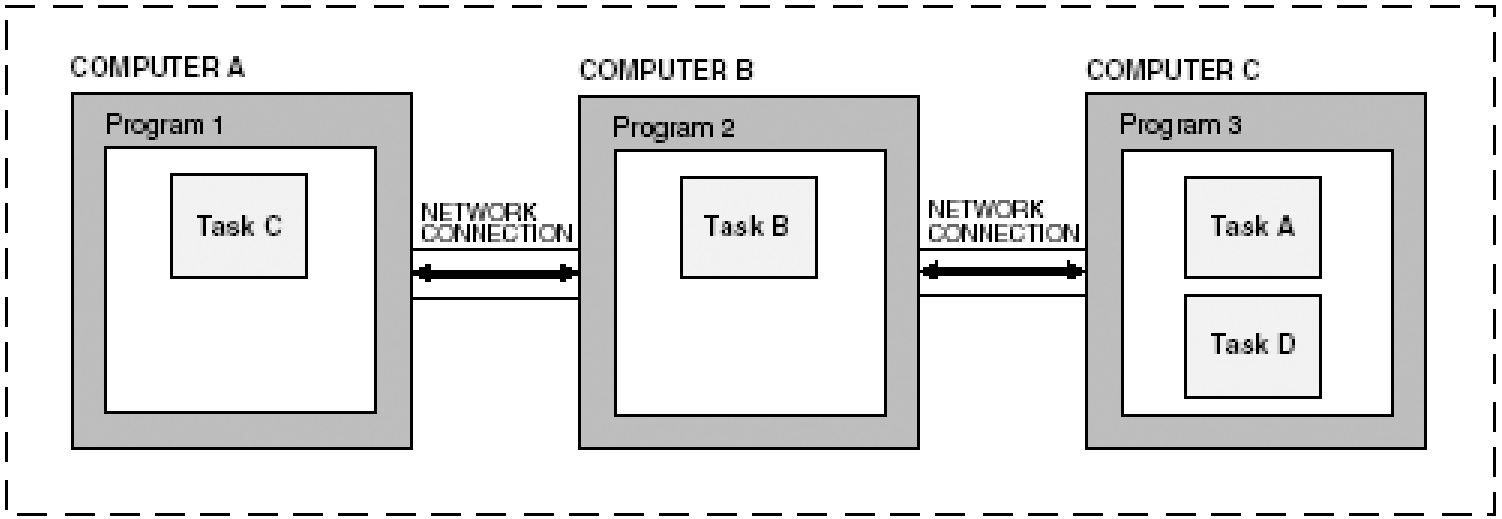
- U PP program koristi 2 ili više procesora unutar istog fizičkog ili virtuelnog računara
- U DP program koristi isti princi PP, ali procesi mogu (a ne moraju) biti u različitim računarima

- Često programi imaju i paralelno i distribuirano izvršavanja

Paralelni program



Distribuiran program



Prednosti PP

- Dobro dizajnirani paralelni program se izvršava brže od njemu odgovarajućeg sekvencijalno organizovanog programa
  - BRŽE == BOLJE
- Neki problemi se prirodno rešavaju kolekcijom paraleleno izvršavanih zadataka
  - Primena u inženjerstvu, nauci, računarskoj inteligenciji, ...
  - Često diktirano specijalizovanim hardverom

Prednosti DP

- Generalno DP omogućava softveru da se iskoriste resurse locirane u računarskoj mreži

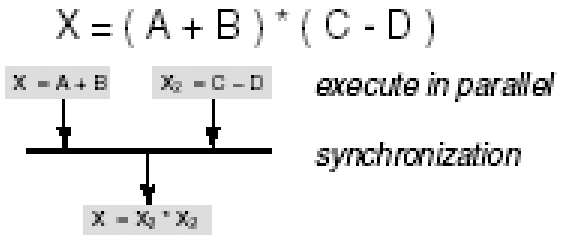
- Deljenje skupih resursa - program na jednom računaru pristupa hardveru drugog računara ili dobija uslugu softvera drugog računara
  - Resursi mogu biti veoma udaljeni (moraju biti povezani u mrežu!)
- Pristup ogromnoj količini informacija koja prevazilazi smeštajne kapacitete jednog računara – lociranoj u raznim čvorovima
- Veća procesorska (računarska) moć je na raspolaganju
- Udvajanje resursa
  - Redundantan hardver (i softver) može rešavati probleme ispada
  - Kopije podataka (replike) omogućavaju brži pristup

“Cene” paralelizma i distribucije

- PP i DP imaju svoju cenu
- Pre nego se program napiše mora se proći kroz dizajn rešenja koji obuhvata
  - Dekompoziciju – podela programa na delove
  - Komunikacije – neophodne za povezivanje delova
  - Sinhronizaciju – koordinacija rada delova

Nivoi konkurentnosti

- Konkurentnost se može pojaviti na više nivoa:
  - Nivo instrukcija
    - Delovi instrukcije se mogu paralelno izvršiti



  - Nivo funkcije/procedure
    - Funkcionalne celine se izvršavaju jednovremeno (tipično u programskim nitima)
  - Nivo objekta
    - Više objekata učestvuje u rešavanju zadatka. Svaki objekat se izvršava u zasebnom procesu ili niti.
    - Upotrebom CORBA-e objekat se može izvršavati u udaljenom računaru
  - Nivo aplikacije
    - Nekoliko aplikacija može sarađivati u rešavanju istog problema
    - Pojedinačne aplikacije su mogle biti razvijane nezavisno sa drugom namenom, ali su ponovo upotrebljene (*software reuse*) i sinhronizovane za jednu distribuiranu aplikaciju

Koordinacija zadataka

Zadaci moraju da komuniciraju i sinhronizuju svoj rad

- ***Data race*** – javlja se kada više zadataka menjaju deljeni podatak u isto vreme tako da konačna vrednost zavisi od redosleda pristupa
  - Rešenje: Sinhronizaciju pristupa zasnovati na poznatim pravilima
- **Beskonačno odlaganje** – javlja se kada zadatak čeka na događaj koji se ne desi, tako da on čeka, čeka, ...
- ***Deadlock*** – javlja se kada dva zadatka X i Y žele ekskluzivan pristup do dva resursa A i B, ali zahteve ispostavljaju u suprotnom redosledu. X nakon odobrenja pristupa A čeka na odobrenje za B, ali ga na dobija jer je Y već zauzeo taj resurs (B) i neće ga osloboditi jer čeka da se oslobodi A.
  - Veći broj deljenih resursa i zadataka komplikuje situaciju
- **Komunikacione poteškoće**
  - Komunikacija je nepouzdana; unosi promenljivo kašnjenje
  - Povezivanje heterogenih sistema nosi dodatne komplikacije

Hardverski i softverski otkazi

- U DS se nameće mnoštvo pitanja na koje treba imati spreman odgovor
  - Šta uraditi kada jedan procesor ili računar zakaže?
  - Da li tada program treba da prekine rad ili zadatak posveti drugom?
  - Šta kada komunikacioni link privremeno zakaže? Koliko čekati na odziv kod veoma opterećenog sistema (gustog mrežnog saobraćaja)?
  - Šta uraditi kada deo aplikacije zakaže?
  - Kako promena prava korisnika utiče na celu aplikaciju? Npr. deo aplikacije (koji je ranije radio bez greške) ne može da priđe podacima jer su promenjana prava pristupa.
  - ...

Preterivanje u paralelizmu

- Upošljavanje prevelikog broja procesora (procesa i programskih niti) može imati negativne posledice
  - Komunikacije među procesima u DS se usložnjavaju
  - Prekljućenja na jednom procesoru zahtevaju vreme
  - Složena sinhronizacija procesa
- Optimalan broj potrebnih procesa ili računara je često nepoznat
  - Pomažu: merenja performansi sistema i iskustvo