Paralelni sistemi

Projektovanje paralelnih algoritama

- Ne postoji jedinstveni recept za projektovanje
  - zahteva određeni stepen kreativnosti
- Ian Foster je 1995 predložio opštu metodologiju projektovanja paralelnih algoritama koja se sastoji od 4 koraka:
  - Dekompozicija
  - komunikacija
  - aglomeracija
  - preslikavanje

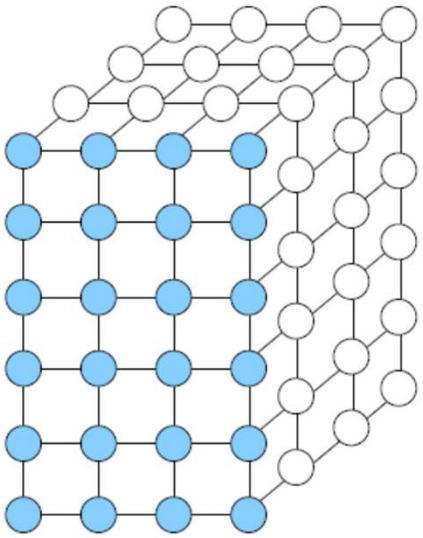
Dekompozicija

- Cilj dekompozicije je otkrivanje paralelizma
  - ovo je jedini korak u kome se detektuje paralelizam
    - svi ostali koraci ga redukuju
  - Postoje dva potencijalna izvora paralelizma:
    - podaci (domenska dekompozicija) i (data parallelism)
    - izračunavanje (funkcionalna dekompozicija) (task parallelism)
  - to su dva komplementarana metoda za ekstrakciju paralelizma.

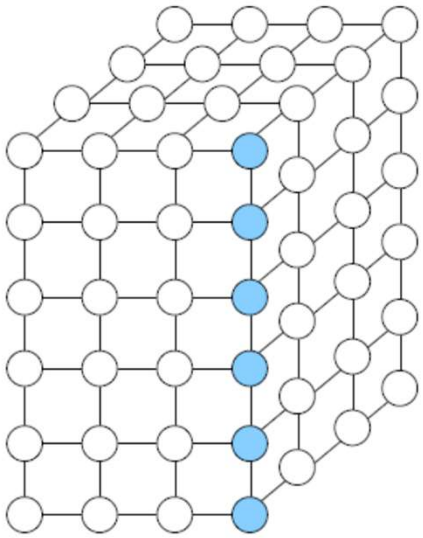
Domenska dekompozicija

- Cilj je rastaviti podatke na mnogo manjih delovana kojima se može primeniti paralelna obrada (izračunavanje)
  - Ova paralelna izračunavanja ćemo zvati primitivni zadaci (taskovi)
- Kod domenske dekompozicije, najbolje je identifikovati najveće i najčešće korišćenje podatke, izvršiti njihovu dekompoziciju na mnogo malih, po mogućnosti identičnih delova, i dodeliti svaki taj deo primitivnom zadatku.
  - Npr. ako imamo 3D model nekog objekta predstavljen kao skup 3D tačaka na površini tog objekta, i želimo da rotiramo taj objekat u prostoru, tada (teorijski) možemo primeniti istu operaciju na svakoj tački paralelno i svaku tačku dodeliti primitivnom zadatku.

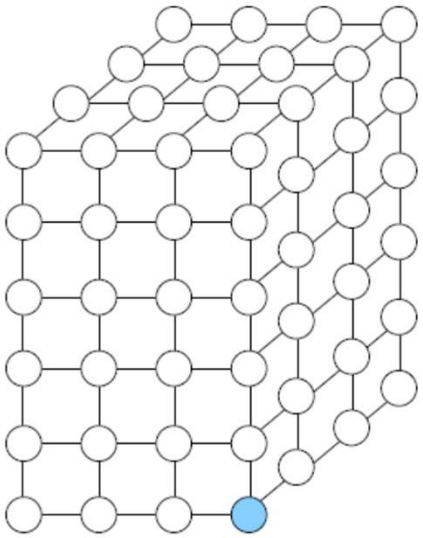
- Neka je, npr. zadata trodimenzionalna matrica dimenzija m×n×k, gde su m broj vrsta i n broj kolona u jednoj ravni, a k broj m×n ravni
- cilj dekompozicije je da podeli ovu matricu.
  - gruba (krupna) dekompozicija bi bila podeliti ovu matricu na k ravni i svaku ravan dodeliti na obradu jednom zadatku
  - finija dekompozicija bi bila podela na n×k kolona i dodela svake kolone jednom zadatku
  - najfinija dekompozicija bi svaki element matrice dodelila jednom zadatku, kreirajući tako n×m×k zadataka
    - ova dekompozicija obezbeđuje najveći paralelizam i bila bi najbolji izbor



1D Decomposition



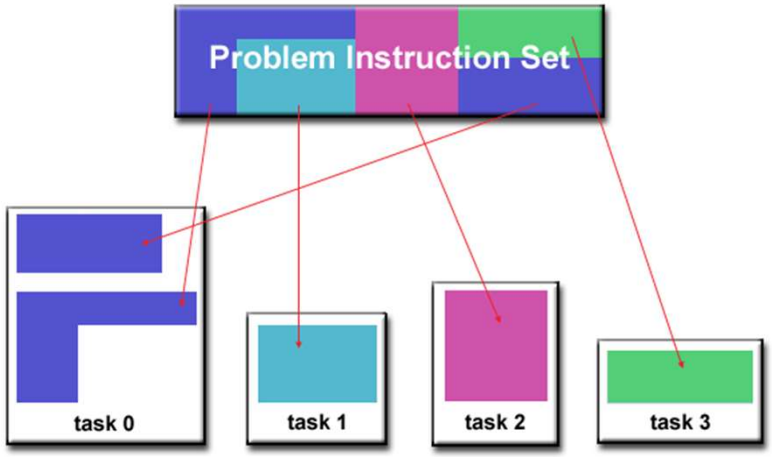
2D Decomposition

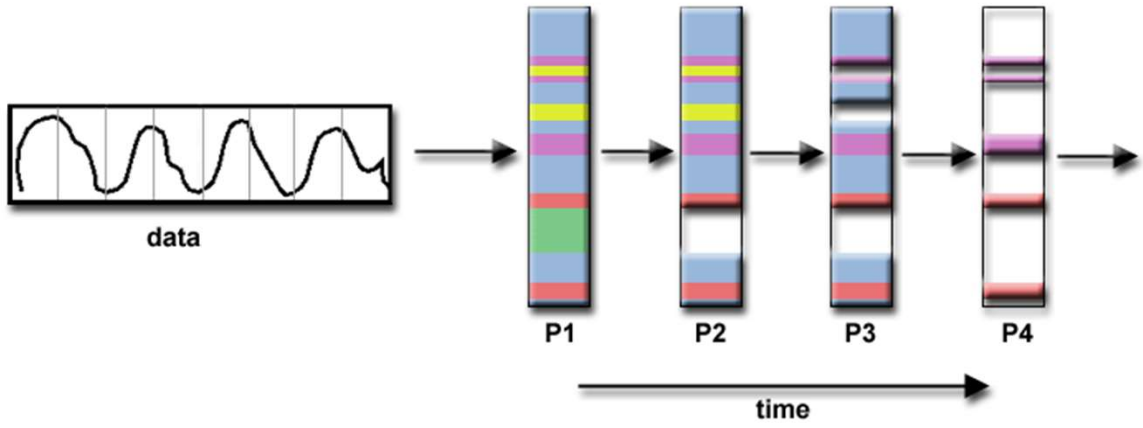


3D Decomposition

Funkcionalna dekompozicija

- Ponekad problem koji treba da se reši ne poseduje visoki stepen paralelizma na nivou podataka, već poseduje tvz. funkcionalni paralelizam
  - funkcionalni paralelizam postoji kada postoje različite operacije koje se mogu jednovremeno obavljati, obično nad različitim skupom podataka.
  - ove funkcije se dodeljuju različitim primitivnim zadacima (taskovima) na izvršenje i identifikuju se podaci nad kojima se funkcije obavljaju
  - u ovom slučaju akcenat je na izračunavanju koje treba da se obavi.





Fosterova lista za procenu dekompozicije

- Dekompozicija treba da zadovolji (što je moguće više) sledeće kriterijume:
  - Broj zadataka (taskova) koji proizilazi iz dekompozicije treba da bude bar za red veličine veći od broja procesora u sistemu.
    - u protivnom postojaće malo fleksibilnosti u sledećim stadijumima projektovanja
  - redundantna izračunavanja i redundantne podatke treba svseti na minimum
    - u protivnom se može desiti da rezultujući algoritam ne može da se primeni na probleme velikog obima.
  - Primitivni zadaci (taskovi) treba da su približno istog obima
    - u protivnom će biti treško dodeliti svakom procesoru približno jednaku količinu posla, što će dovesti do degradiranja performansi.
  - Broj taskova treba da je rastuća funkcija obima problema.
    - idealno bi bilo da povećanje obima problema dovede do povećanja broja taskova, a ne veličine taska.
  - Dobro je identifikovati nekoliko alternativnih dekompozicija
    - na taj način se maksimizira fleksibilnost sledećih koraka u procesu projektovanja paralelnog algoritma.
  - Istražiti mogućnosti i domenske i funkcionalne dekompozicije.

Komunikacija

- Kada je izračunavanje podeljeno na nekoliko zadataka (taskova) koji se mogu izvršavati na različitim procesorima, neki od podataka potrebnih tasku mogu biti u lokalnoj memoriji procesora, ali neki mogu biti u memoriji drugog procesora.
  - Zbog toga se javlja potreba za razmenom podataka, tj. za komunikacijom
    - Ona je posledica paralelizacije. To je overhead koji se plaća paralelizaciji
    - potrebno je prvo identifikovati gde se javlja potreba za komunikacijom.
  - Lokanla komunikacija
    - zadatak dobija podatke od malog broja drugih zadataka
  - globalna komunikacija
    - podaci se dobijaju od velikog broja zadataka

Fosterova lista za procenu komunikacje

⁕ Kriterijumi za procenu

- Da li svi zadaci imaju približno jednak obim komunikacije.
  - Nebalansiranost u komunikaciji sugeriše da se paralelni algoritam neće dobro skalirati sa porastom obima problema.
- Svaki zadatak treba da komunicira sa malim brojem suseda.
  - Ako zadatak treba da komunicira sa velikim brojem drugih zadataka to će kreirati veliki overhead kod izvršenja paralelnog programa.
- Komunikacija između zadataka treba da se obavlja konkurentno, tj. više zadataka može međusobno da komunicira u isto vreme
  - u protivnom paralelni algoritam će biti neefikasan i slabo skalabilan

Aglomeracija

- Aglomeracija podrazumeva kombinovanje grupa od dva iliviše zadataka u veće zadatke, kako bi se smanjio broj zadataka i potreba za komunikacijom.
  - alomeracijom se sa sitno-zrnastog (fine grained) paralelizma prelazi na srednje-zrnati (coarse grained) paralelizam.
    - Njegova svrha je poboljšanje performansi i pojednostavljenje programiranja.
  - Aglomeracija je problem optimizacije
    - Veoma često ciljevi su u suprotnosti, pa je potrebno napraviti kompromise.
  - Jedan od načina za poboljšanje performansi je smanjenje komunikacija. K
    - ada se dva zadatka koji međusobno razmenjuju podatke kombinuju u jedan zadatak, nema potrebe za komunikacijom. To se zove povećanje lokalizacije.

Preslikavanje (mapping)

- Podrazumeva dodelu zadataka procesorima
- Cilj: minimizirati ukupno vreme izvršenja programa
- Smernice:
  - zadaci koji mogu da se izvršavaju konkurentno se dodeljuju (preslikavaju) različitim procesorima.
  - Zadaci koji zahtevaju čestu međusobnu komunikaciju se dodeljuju istom procesoru.
  - poželjno je da svi procesori dobiju istu količinu posla
    - neravnomerna raspodela poslova dovodi do lošijih peformansi.
    - procesor koji obavlja najveći posao će definisati vreme rada celog sistema.
  - Na žalost da bi se postiglo izbalansirano opterećenje procesora nekada se moraju primeniti komplikovani algoritmi

Projektovanje paralelnih algoritama