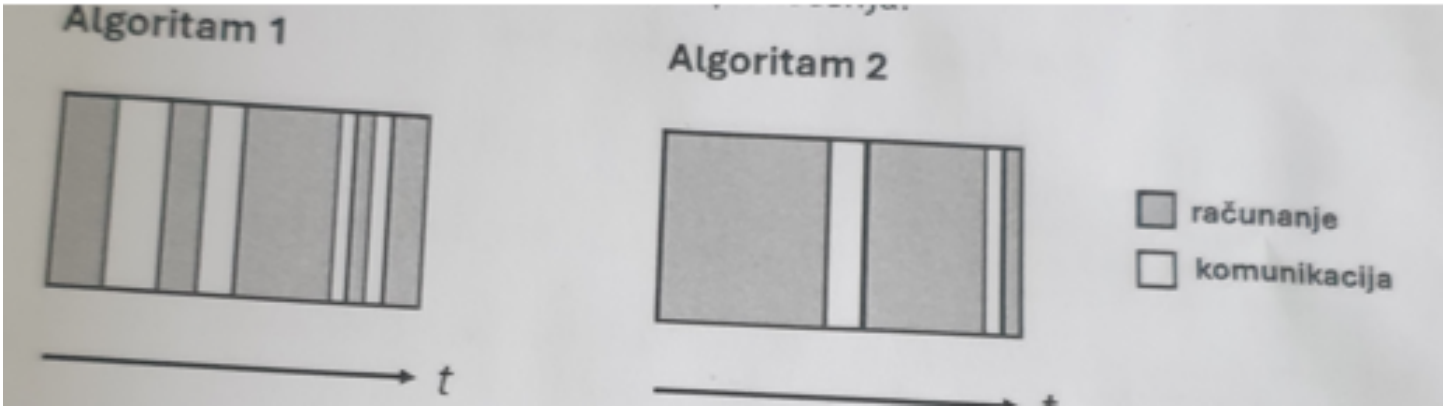
Paralelno programiranje - Međuispit 2024

1.(5) Napišite algoritam za EREW PRAM računalo koji za zadani niz A[] koji sadrži N prirodnih brojeva pronalazi najveći neparni broj. Pretpostavite da niz sadrži barem jedan neparni broj. Na raspolaganju su scan i reduce funkcije za proizvoljne operacije (ne i sortiranje). Netrivijalne operacije (npr. ona koje uključuju grananja) potrebno je definirati algoritamski. Ocijenite složenost algoritma.

2.(5) Napišite algoritam za CRCW PRAM kojí će odrediti broj čestica čija se putanja preklapa u sljedećem vremenskom intervalu delta t. Čestice su predstavljene točkama u 2D ravnini. U ravnini je ukupno ? (ne vidi se) čestica, a za svaku je poznat trenutni položaj zapisan u strukturi Polozaj i vektor brzine zapisan u strukturi Vektor. Niz P[] sadrži N položaja, a polje V[] sadrži N vektora. Za određivanje hoće li se dogoditi preklapanje u putanji (sudar) između dvije čestice u intervalu delta t, na raspolaganju je funkcija bool sudar(Polozaj pl, Polozaj p2, Vektor v1, Vektor v2). Na raspolaganju su scan i reduce funkcije za proizvoljne operacije (ne i sortiranje). Netrivijalne operacije (npr. one koje uključuju grananja) potrebno je definirati algoritamski.

3.(3) Usporedite zrnatost dvaju algoritama čiji je tijek izvođenja prikazan na donjoj slici te ukratko objasnite utjecaj zrnatosti na ujednačavanje opterećenja.



4. (4) Rasporedite operacije koje izvode četiri procesora na APRAM računalu u što manji broj asinkronih odsječaka odvojenih sinkronizacijskim ogradama. Varijable A, B, C i D se nalaze u globalnoj memoriji a znak „ * “ označava lokalne operacije. Pretpostavite da prvi sa izvođenjem počinje procesor 1. Redoslijed operacija koje treba izvesti pojedini procesor treba ostati očuvan.

procesor1 | procesor2 | procesor3 | procesor4

---

citaj A | citaj B | citaj C | pisi D

citaj B | * | citaj A | *

* | * | * | pisi D

pisi C | pisi B | pisi A |

5.(4) U sljedećim izrazima definirana je binarna asocijativna operacija "o"

(x,i) o (y,j) = (z,k), gdje je z=max(x,y)

{i, ako x>y

k={j, ako y>x

{min(i,j), ako x=y

Pretpostavite da je neutralni element ove operacije uređeni par (0,0). Provedite prescan algoritam za zadanu operaciju na nizu uređenih parova [(2,0) (4,1) (3,2) (1,3) (7,4) (4,5) (7,6) (5,7)] uz proizvoljan broj procesora. Prikažite izvedbu algoritma u obliku stabla i tablično u obliku niza memorijskih lokacija.

6.(3) PRAM program za n = 1024 procesora izvodi se na APRAM računalu na p procesora, čiji je vrijednost sinkronizacije jednak B = 10. Opišite simbolički kako se jedna EREW PRAM instrukcija prilagođena izvedbi na p APRAM procesora. Navedite broj korištenih APRAM procesora (p), ako je poznato da je na APRAM računalu potrebno 74 koraka za jednu EREW PRAM instrukciju, a trajanje pristupa globalnoj memorijskoj lokaciji je 4 koraka. (rj: 64)

7.

a)(1) Ukupan broj operacija množenja u provedbi postupka *_prescan niza n elemenata na PRAM računalu uz p procesora, gdje je p <(n/2), iznosi:_____

b)(1) Na računalu EREW PRAM, optimalan broj koraka paralelnog algoritma prescan na nizu s 45 elemenata iznosi:_____

c)(1) Operacija kojom iz niza [1 -1 2 -3 4 -5 5 -3 4 -7] dobivamo niz [1 1 2 3 4 5 5 5 5 7] je:_____

d)(1) Nakon uspješne provedbe operacije MPI_Reduce, rezultat operacije nalazi se na koliko procesa:_____

e)(1) Povratak iz blokirajuće MPI funkcije znači:_____

f)(1) Ukupan broj poruka koji se razmijeni u provedbi komunikacijske strukture binarnog stabla (npr. algoritam reduciranja) dubine d iznosi:_____

g)(1) Amdahlov zakon povezuje sljedeće veličine:_____

8.(4) Zadan je dio ispravno inicijaliziranog MPI programa koji se izvodi u world_size procesa. Hoće li prilikom izvođenja programa doći do potpunog zastoja? Prikažite razmjenu poruka u programu (u obliku tablice prikazane na dnu stranice, gdje svaki stupac sadrži tijek naredbi jednog od procesa, a strelice označavaju poruke) ako pretpostavimo da je program pokrenut u 3 procesa.

```
int data=0;
int dest, tag;
bool poslano=false;

while(1) (
    if (!poslano) {
        dest = (my_id+1)% world_size; tag= 1;
        data = pripremi_podatak();
        MPI Send(&data, 1, MPI_INT, dest, tag, MPI_COMM_WORLD);
        poslano = true;
    }
    MPI Recv(&data, 1, MPI_INT, ANY_SOURCE, ANY_TAG, MPI_COMM WORLD, &status);
    if (status.MPI_TAG == 1){
        data = racunaj_nesto(data);
        dest = status.MPI_SOURCE; tag = 2;
        MPI Send(&data, 1, MPI_INT, dest, tag, MPI_COMM_WORLD);
    }
    else if(status.MPI_TAG == 2){
        pohraniRezultat(data);
        poslano = false;
        sleep(rand()%10);
    }
```

---

proces 0 | proces 1 | proces 2

====================

| |