Elektronski fakultet **3. 9. 2021.**
Katedra za računarstvo

PARALELNI SISTEMI

1. U programskom jeziku C++ napisati:
a) Klasu CudaComplex za predstavljanje kompleksnih brojeva. Klasa treba da sadrži realni *(r)* i imaginarni *(i)* deo kompleksnog broja koji su tipa float, kao i funkcije i/ili operatore za množenje kompleksnih brojeva.
b) Funkciju ***int julia(int x, int y)*** koja korišćenjem prethodno napisane klase implementira pseudokod koji je dat u nastavku.

```
function julia( x,  y )
   scale = 1.5;
   retval = 255;
   jx = scale * (DIM/2 - x)/(DIM/2)
   jy = scale * (DIM/2 - y)/(DIM/2)

   CudaComplex a(jx, jy)

   for (i=0; i<200; i++)
      a = a * a
      magnitude2 = a.r * a.r + a.i * a.i

      if (magnitude2 > 1000)
         retval = 0;
      end if
   end for
   return retval;
end function
```

c) Funkciju calculate(unsigned char** A) koja popunjava kvadratnu matricu A veličine N na sledeći način: A [i,j] = julia (j,i);
d) Šta je potrebno izmeniti/dodati u klasi CudaComplex i funkciji *julia* da bi se kod mogao izvršavati na GPU?
e) Korišćenjem CUDA tehnologije, napisati kernel funkciju koja odgovara funkciji *calculate*. Obratiti pažnju na efikasnost paralelizacije.
f) Za prethodno napisan kernel, napisati i host kod kojim se omogućava njegovo pozivanje. Pretpostaviti veličinu bloka od 256 niti i broj blokova ne veći od 256.

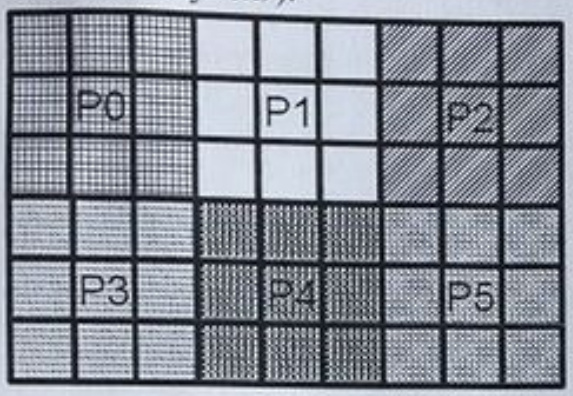
2. a) Napisati MPI program koji realizuje množenje matrice A_{nxn} i matrice B_{nxn}, čime se dobija rezultujuća matrica C_{nxn}. Matrice A i B se inicijalizuju u master procesu. Broj procesa je p i uređeni su kao matrica qxq (q^2=p). Matrica A je podeljena u blokove od po n/q vrsta, a matrica B podeljena u blokove od po n/q kolona. Master proces distribuira odgovarajuće blokove matrica A i odgovarajuće blokove matrice B svim procesima. Nakon distribuiranja, procesi u i-toj $(0 \le i \le q-1)$ vrsti matrice procesa sadrže *i, i+q, i+2q, ..., i+n -q* vrstu matrice A. Slično, nakon distribuiranja, procesi u j-toj $(0 \le j \le q-1)$ koloni matrice procesa sadrže *j, j+q, j+2q, ..., j+n -q* kolonu matrice B. Nakon toga, svaki proces obavlja odgovarajuća izračunavanja i učestvuje u generisanju rezultata koji se prikazuje u master procesu. Predvideti da se slanje podataka blokova matrica svakom procesu za svaku matricu obavlja odjednom. Slanje podataka blokova matrice A i B i Generisanje rezultata implementirati korišćenjem grupnih operacija i funkcija za kreiranje novih komunikatora.
b) Distribuciju podataka blokova matrice A i B po procesima implementirati korišćenjem Point-to-Point operacija i bez funkcija za kreiranje novih komunikatora.

3. Napisati MPI program koji vrši paralelni upis i čitanje binarne datoteke, prema sledećim zahtevima:

1. Svaki proces upisuje po 9 proizvoljnih celih brojeva u datoteku *file1.dat*. Upis se vrši upotrebom pojedinačnih pokazivača, dok redosled podataka u fajlu ide od podataka poslednjeg do podataka prvog procesa.

2. Ponovo otvoriti datoteku. Svaki proces vrši čitanje upravo upisanih podataka upotrebom funkcija sa eksplicitnim pomerajem. Proveriti ispravnost pročitanih podataka.

3. Upravo pročitane podatke upisati u novu datoteku, tako da podaci svakog procesa čine deo matrice dimenzija MxN, gde vrednosti M i N unosi korisnik. Upis izvršiti na način prikazan na slici (za slučaj matrice dimenzija 6x9):



U poslednjem zahtevu posebno obratiti pažnju na efikasnost paralelizacije upisa.

4. Napisati OpenMP kod koji sadrži sledeću petlju:

```
j = N + 1;
x = 0;

for (i = 0; i < N; i++)
{
   x = x + a[i];
   b[i] = b[i] + b[j++];
}
```

i proučiti da li je moguće izvršiti njenu paralelizaciju. Ako nije, transformisati petlju tako da paralelizacija bude moguća. Nakon petlje treba prikazati vrednosti za promenljivu x, generisanu u okviru petlje. Testiranjem sekvencijalnog i paralelnog rešenja za proizvoljno N i proizvoljan broj niti, pokazati korektnost paralelizovanog koda.