# Arhitectura sistemelor de calcul paralel - examen

15 iunie 2023

Durata examenului este de două ore. Tratați următoarele subiecte:

1. **(4p)** Scrieți un program C MPI care să poată fi rulat cu patru procese și să asigure următoarea funcționalitate:
   (a) Toate procesele își stochează rangul în variabila *my_rank* și declară un șir *double data[10];* pe care îl inițializează cu valori generate aleator în intervalul (0,1) (puteți folosi funcțiile GSL pe care le cunoașteți, în particular *gsl_rng_uniform()*, cu inițializarea corespunzătoare a mediului RNG din GSL; a se vedea cap. 17 din manualul *gsl-ref.pdf*);
   (b) Procesul 0 colectează toate datele/șirurile și le stochează într-un fișier text *data.out* sub forma:
   *rank: data[0] data[1] ... data[9] [NEWLINE]*
   unde *rank* este rangul procesului care a generat șirul respectiv (fiecare linie din fișier conține datele provenind de la un proces, ordinea liniilor fiind ordinea rangurilor).
   (c) Funcții MPI sugerate: *MPI_Init(), MPI_Comm_size(), MPI_Comm_rank(), MPI_Send(), MPI_Recv(), MPI_Finalize()*.

2. **(5p)** Algoritmul CQUAD este o procedură dublu-adaptativă de integrare numerică a funcțiilor, inclusiv a celor cu singularități integrabile în domeniul de integrare. Domeniul de integrare este divizat în subintervale și la fiecare iterație subintervalul care dă cea mai mare eroare estimată este subdivizat în două părți egale. În felul acesta eroarea este redusă rapid, pe măsură ce subintervalele se concentrează în jurul punctelor dificile din domeniu (aici singularitatea, aproximațiile succesive asigurând convergența rapidă a integralei). Programul secvențial *int_CQUAD.c*, atașat, calculează numeric integrala:

$$\int_0^1 \frac{\ln(x)}{\sqrt{x}} dx, \tag{1}$$

   care prezintă o singularitate logaritmică în 0. Vi se cere să-l paralelizați, astfel încât să asigure următoarea funcționalitate:
   (a) Vor fi lansate în execuție *nproc* procese, procesul cu rangul 0 (procesul master) definind valorile $a$ și $b$ și distribuind inițial o partiție uniformă a intervalului de integrare $(a, b)$ (*nproc* subintervale egale, fiecare de lungime $\frac{b-a}{nproc}$);
   (b) O soluție posibilă ar fi ca fiecare proces să definească un șir *double llims[2];* în care să recepționeze limitele de integrare inferioară și superioară de la procesul cu rang 0 (evident, acesta își va defini local limitele de integrare);
   (c) Partiția inițială va fi: procesul cu rangul 0 tratează primul subinterval, procesul cu rangul 1 pe al doilea, etc.;
   (d) Toate procesele vor defini suplimentar variabilele *double local_result, local_error;*, care vor stoca, respectiv, rezultatul și eroarea estimată pentru calculul local;
   (e) Urmează partea secvențială, în care va fi aplicat algoritmul de integrare exact la fel ca în programul secvențial, însă valorile calculate vor fi stocate în *local_result* și *local_error*;
   (f) Valorile locale vor fi acumulate de procesul cu rangul 0 în variabilele *result* și *error* (o soluție elegantă constă în utilizarea funcției *MPI_Reduce()*);
   (g) Procesul cu rangul 0 scrie într-un fișier text rezultatul final și estimarea marginii superioare a erorii;
   (h) Funcții MPI sugerate: *MPI_Init(), MPI_Comm_size(), MPI_Comm_rank(), MPI_Send(), MPI_Recv(), MPI_Reduce(), MPI_Finalize()*

Se acordă 1p din oficiu. Funcțiile MPI indicate mai sus sunt doar sugestii de lucru. Orice soluție corectă, care asigură funcționalitatea cerută, va fi luată în considerare. Întrucât sunt utilizate funcții incluse în biblioteca GSL, compilarea se face cu:

mpicc -o nume_program fisier_sursa.c -lgslcblas -lgsl -lm
Dacă sistemul dumneavoastră de operare este Linux Ubuntu, compilati cu:
mpicc -o nume_program fisier_sursa.c -lgsl -lgslcblas -lm

**Succes!**