n1Laborator L3
MPI

Scrieti un program bazat pe MPI care face suma a 2 numere mari.
'numar mare' = numar cu mai mult de 10 cifre

Consideratii generale:
Reprezentare unui numar = tablou de cifre (numere intregi fara semn - byte) in care cifra cea mai nesemnificativa este pe pozitia 0.
e.g. pentru numarul 123 cifra cea mai nesemnificata este 3, deci tabloul corespunzator este t[0]=3 t[1]=2 t[2]=1

Cele 2 numere mari se citesc din fisierele "Numar1.txt" (un numar cu N_1 cifre) si "Numar2.txt" (un numar cu N_2 cifre).
Fiecare din aceste fisiere contine la inceput un numar (N) care reprezinta numarul de cifre si apoi cifrele numarului respectiv.
Implementare > C++11.

Varianta 0 – implementare secventiala

Implementari MPI cu p procese:

Varianta 1 (comunicare standard)
IMPORTANT:
- Indiferent de implementarea MPI (Microsoft, OpenMPI, MPICH, ….) programele trebuie sa fie corecte (fara deadlock)! Acest lucru inseamna ca pentru comunicatia standard trebuie sa faceti proiectarea presupunand ca implementarea nu foloseste buffere – adica MPI_Send blocheaza procesul pana cand se termina receptia de catre procesul receptor. (vezi fisierul "MPI_Communication Modes.docs" din Files/Cursuri).
- Pentru a testa implementarea corecta – inlocuiti transmiterea standard cu cea sincrona MPI_Ssend.

– Se considera rezolvarea problemei prin executia urmatoarelor etape:

1) procesul 0
    a. seteaza o variabila *id_proces_curent*=1
    b. repeta urmatoarele actiuni pana cand se citesc toate cifrele numerelor
        i. citeste cate N/(p-1) cifre din cele 2 fisiere // Atentie -procesul 0 nu citeste toate cifrele la inceput
        ii. le trimite procesului "*id_proces_curent*"
        iii. incrementeaza "*id_proces_curent*"
2) procesele cu *id>0* fac suma cifrelor primite si calculeaza "report" (carry) corespunzator;
   Constrangere: procesele primesc mai intai cifrele pe care trebuie sa le adune si apoi primeste carry de la precedent
3) fiecare proces (cu exceptia ultimului) trimit "reportul" la procesul urmator care il foloseste pentru actualizarea rezultatului (procesul id=1 nu primeste carry - il considera egal 0)
4) rezultatul final se obtine in procesul 0, care scrie rezultatul in fisierul "Numar3.txt"
    a. procesul 0
        i. primeste cifrele sumei de la fiecare proces cu id>0
        ii. dupa ce primeste un segment de cifre de la un process il scrie in fisier (nu asteapta sa le primeasca pe toate)
    b. procesele cu id>1 trimit cifrelor sumei pe care le-a calculat catre procesul 0

Optimizare -!? Adunarea cifrelor dintr-un proces cu id>1 incepe inainte de a astepta carry.
Este posibil? Ce impune aceasta varianta?

Varianta 1.1 = Rezolvarea cu aceasta potentiala optimizare cu analiza comparativa a rezultatelor => 2 puncte suplimentar

Testare:

1) N_1=N_2=16 ; Numar1=”9999 4444 4444 9999” Numar1=”9999 5555 5555 9999” cu 5 procese
2) N_1=10000 si N_2=10000 (random digits) cu Numar de Procese: 5, 9, 17
3) N_1=100 si N_2=100000 (random digits) cu Numar de Procese: 5, 9, 17

(procesul 0 nu calculeaza sume de cifre – citeste, transmite, primeste si scrie date)

Varianta 2 (scatter / gather) si procesul 0 face
– considera rezolvarea problemei prin executia urmatoarelor etape:

1) procesul 0 citeste cele 2 numere si le stocheaza in 2 tablouri:
   -daca un numar are mai putine cifre se completeaza cu cifre nesemnificative
2) cifrele celor 2 numere se distribuire proceselor folosind MPI_Scatter (daca nu este valabila conditia p|N, unde N=max{N_1,N_2}, N_1 nr de cifre ale primului numar, N_2 nr de cifre ale celui de-al doilea, atunci se mareste N corespunzator si se completeaza cu 0-uri)
3) procesele fac suma cifrelor primite si calculeaza “report” (carry) pe care il trimit procesului urmator (cu exceptia ultimului proces care nu trimite carry)
4) rezultatul final se obtine in procesul 0 (se foloseste MPI_Gather)
5) procesul 0 scrie rezultatul in fisierul “Numar3.txt”

Testare:

1) N_1=N_2=16 ; Numar1=”9999 4444 4444 9999” Numar1=”9999 5555 5555 9999” cu 4 procese
2) N_1=1000 si N_2=1000 (random digits) cu Numar de Procese: 4, 8, 16
3) N_1=100 si N_2=100000 (random digits) cu Numar de Procese: 4, 8, 16

Varianta3– comunicare asincrona
Se considera rezolvarea problemei prin executia urmatoarelor etape:

1) procesul 0
- seteaza o variabila *id_proces_curent*=1
- repeta urmatoarele actiuni pana cand se citesc toate cifrele numerelor
   a. citeste cate N/p cifre din cele 2 fisiere // Atentie -procesul 0 nu citeste toate cifrele la inceput!
   b. le trimite procesului “*id_proces_curent*” folosind MPI_Isend
   c. incrementeaza “*id_proces_curent*”
2) un process cu id<>0 primeste setul de cifre de la procesul 0 si face adunarea intr-un vector rezultat si actualizeaza “reportul”(carry”) pe care il trimite la procesul urmator
   (atentie un proces cu id (id<>1, id<>0) primeste informatie de la procesul 0 si de la procesul (id-1) dar ordinea intre cele 2 nu este sigura … se cere sa se foloseasca **MPI_Irecv**)
3) rezultatul final se obtine in procesul 0 prin agregarea rezultatelor tuturor celorlalte procese; agregarea se va face folosind transmitere asincrona!!!
4) procesul 0 scrie rezultatul in fisierul “Numar3.txt”

Testare:

1) N_1=N_2=16 ; Numar1=”9999 4444 4444 9999” Numar1=”9999 5555 5555 9999” cu 5 procese
2) N_1=10000 si N_2=10000 (random digits) cu Numar de Procese: 5, 9, 17
3) N_1=100 si N_2=100000 (random digits) cu Numar de Procese: 5, 9, 17
   (procesul 0 nu calculeaza sume de cifre – citeste, transmite, primeste si scrie date)

Includeti in timpul de executie si citirea numerelor!

Nu uitati sa verificati corectitudinea prin compararea fisierelor rezultat obtinute la varianta secventiala cu rezultatele obtinute la variantele paralele pentru fiecare din cele 10 executii.