Tema 3

Cerinta:

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.

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.

Functii:

Proiectul are 4 functii:

* generateNumber(digitsCount: Int, filename: String)
  + genereaza un numar cu digitsCount cifre si il scrie in fisier
* run0() – rezolva problema secvential
* run1(int argc, char\*\* argv) - rezolva problema folosind Send si Recv
* run2(int argc, char\*\* argv) - rezolva problema folosinf Scatter si Gather
* main(int argc, char\*\* argv)

Explicatii:

Run1: Funcția implementează adunarea paralelă de numere mari: Master-ul (rank 0**)** citește numerele din fișiere, le împarte în bucăți egale și le trimite worker-ilor; Workerii calculează suma cifrelor cu transport (carry) și trimit rezultatele parțiale la master; Master-ul combină rezultatele și ajustează carry-ul final, verificând apoi corectitudinea cu un fișier de referință și afișează durata execuției.

Run2: Funcția implementează adunarea paralelă a două numere mari folosind MPI. Procesul 0 citește numerele, le împarte în bucăți egale și le distribuie către procese cu MPI\_Scatter. Fiecare proces calculează suma cifrelor primite, gestionează transportul (carry) și trimite rezultatul parțial înapoi cu MPI\_Gather. Procesul 0 reunește rezultatele și verifică corectitudinea comparând cu un fișier de referință. La final, afișează timpul de execuție.

Teste

P = 4

|  |  |  |
| --- | --- | --- |
| **Varianta** | **Caz** | **Timp** |
| **Varianta 1** | Numar 1 = “123456789123456789” = Numar2 | 3.7502 |
| N\_1=1000 si N\_2=1000 (random digits | 6.0362 |
| N1 = 100, N2 = 100000 | 58.0776 |
|  | N\_1=N\_2=16 ; Numar1=”9999 4444 4444 9999” Numar1=”9999 5555 5555 9999 |  |
| **Varianta 2** | Numar 1 = “123456789123456789” = Numar2 | 3.4088 |
| N\_1=1000 si N\_2=1000 (random digits | 5.2034 |
| N\_1=100 si N\_2=100000 (random digits | 61.3928 |
|  | N\_1=N\_2=16 ; Numar1=”9999 4444 4444 9999” Numar1=”9999 5555 5555 9999 | 4.0277 |

P = 8

|  |  |  |
| --- | --- | --- |
| **Varianta** | **Caz** | **Timp** |
| **Varianta 1** | Numar 1 = “123456789123456789” = Numar2 | 6.4442 |
| N1 = 1000, N2 = 1000 | 11.448 |
| N\_1=100 si N\_2=100000 (random digits | 71.670 |
| **Varianta 2** | Numar1 = Numar2 = 123456789123456789 | 9.138 |
| N1 = 1000, N2 = 1000 | 11.702 |
| N\_1=100 si N\_2=100000 (random digits) | 73.682 |

P = 16

|  |  |  |
| --- | --- | --- |
| **Varianta** | **Caz** | **Timp** |
| **Varianta 1** | Numar 1 = “123456789123456789” = Numar2 | 17.328 |
| N\_1=1000 si N\_2=1000 (random digits | 19.269 |
| N\_1=100 si N\_2=100000 (random digits | 77.9932 |
| **Varianta 2** | Numar 1 = “123456789123456789” = Numar2 | 17.3652 |
| N\_1=1000 si N\_2=1000 (random digits | 18.5147 |
| N\_1=100 si N\_2=100000 (random digits) | 91.440 |

Timpul de execuție crește odată cu numărul de procese, iar varianta 1 este ușor mai rapidă decât varianta 2, însă diferența de performanță nu este semnificativă.