Assigment 3: Job Dispatcher

Nume si prenume: Afilipoae George-Marian
Grupa: UPT, AC, CTI-RO, 3C.1.1

Compilare si rulare:

mpicc -o p clust.c main.c

mpirun -np x(numar procese) ./p => cu pana la numarul maxim de procese/2 ale laptopului

mpirun --oversubscribe -np x(numar procese) ./p => cu oricate procese

Acest proiect este un sistem distribuit de procesare a unor comenzi pentru calcularea de valori matematice (primi, divizori primi, anagrame) folosind tehnici de paralelizare, implementat cu ajutorul bibliotecii MPI (Message Passing Interface) pentru calculul distribuit.

Descriere generala a proiectului:

Acest proiect include doua componente principale: un master si mai multi workeri care efectueaza calculul efectiv pentru fiecare comanda primita. Sistemul citeste un fisier de comenzi, le proceseaza si returneaza rezultatele intr-un fisier corespunzator fiecarui client. Comenzile procesate includ calculul numerelor prime, numarul de divizori primi ai unui numar si generarea de anagrame pentru un cuvant dat.

1.**Master**:

- Initiaza MPI si gestioneaza logica principala a aplicatiei.
- Citeste comenzi dintr-un fisier si le trimite workerilor pentru procesare, prin intermediul **`MPI_Send`**.
- Dupa procesarea comenzilor, master-ul colecteaza rezultatele de la workeri folosind **`MPI_Recv`** si le scrie intr-un fisier de iesire corespunzator fiecarui client.
- Masoara timpii de executie pentru procesarea in mod serial si paralel si calculeaza speedup-ul.
- Inregistreaza loguri pentru fiecare comanda trimisa si rezultatul primit.

2.Workeri:

- Receptioneaza comenzi de la master folosind **`MPI_Recv`**.
- In functie de tipul comenzii, efectueaza calculele corespunzatoare:
  - **PRIMES**: Calculeaza numarul de numere prime pana la un numar dat.

        - **PRIMEDIVISORS**: Calculeaza numarul de divizori primi ai unui numar dat.
        - **ANAGRAMS**: Genereaza toate anagramele posibile ale unui cuvant dat.
    - Dupa ce termina procesarea, worker-ul trimite rezultatul inapoi la master cu **`MPI_Send`**.
    - Ciclu se repeta pentru fiecare comanda primita.

Detalii tehnice:

- **MPI** este utilizat pentru comunicarea intre procesele distribuite. Aceasta tehnologie permite unui numar de procese sa comunice si sa coopereze intre ele pentru a efectua sarcini distribuite.
    - **`MPI_Send`**: Este utilizat pentru trimiterea comenzilor de la master catre workeri.
    - **`MPI_Recv`**: Este utilizat pentru receptia comenzilor si a rezultatelor intre procese.
    - **`MPI_Wtime`**: Este folosit pentru masurarea timpilor de executie si pentru calculul speedup-ului.

Structura fisierelor de input si output:

- Fisierul de comenzi contine o lista de comenzi de procesat, fiecare comanda fiind de forma: **`CLI <client_id> <command_type> <parameter>`**.
- Fisierele de output pentru fiecare client sunt denumite **`<client_id>_serial.txt`** sau **`<client_id>_parallel.txt`**, in functie de modul de executie.
- Logurile sunt stocate intr-un fisier separat pentru a urmari trimiterea si receptia comenzilor.

Algoritmi utilizati:

- **Calculul numerelor prime**: Algoritmul verifica fiecare numar intre 2 si `n`, testand daca este divizibil cu orice numar intre 2 si radical din `n`.
- **Calculul divizorilor primi**: Algoritmul verifica fiecare divizor al unui numar si determina daca acesta este prim.
- **Generarea de anagrame**: Algoritmul genereaza toate permutarile posibile ale unui cuvant dat.

Performanta si scalabilitate:

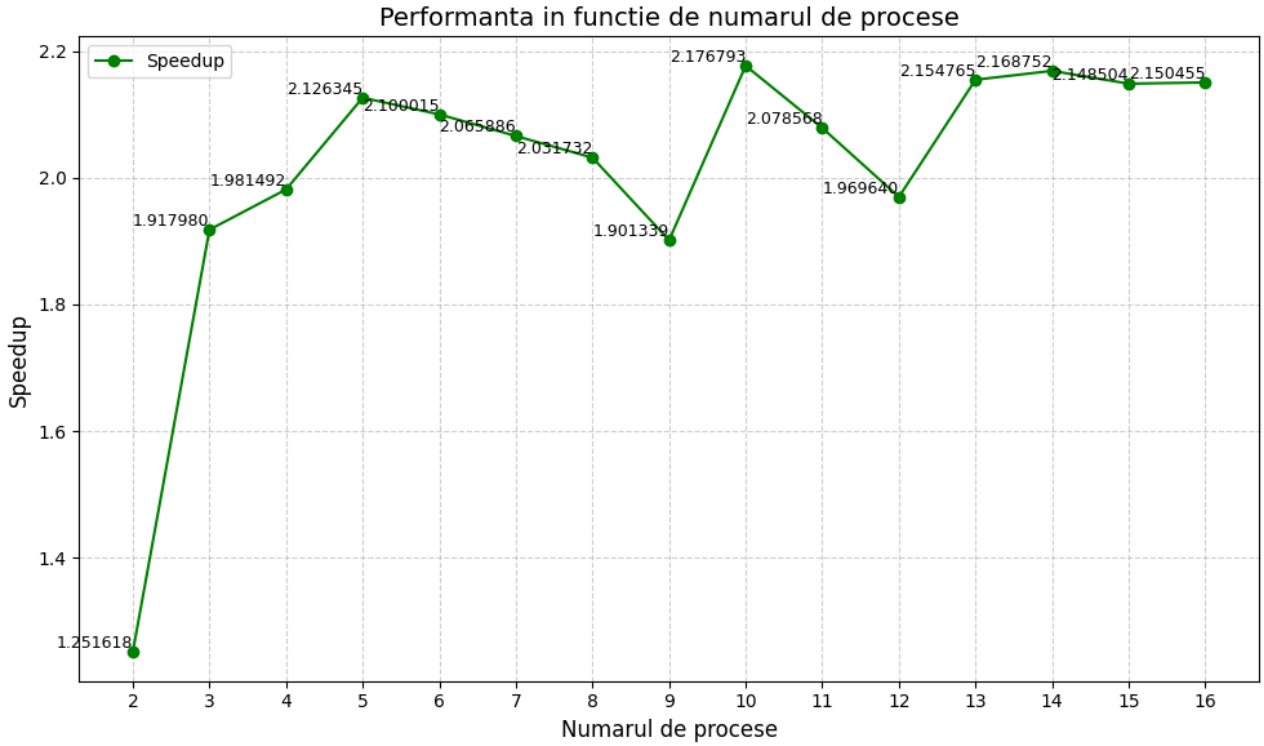
In cadrul acestui proiect, scalabilitatea este evaluata prin calcularea speedup-ului: raportul dintre timpul de executie in mod serial si timpul de executie in paralel.

Speedup-ul este calculat si stocat intr-un fisier **`speedup.txt`**, iar rezultatele sunt afisate pentru diferite numere de procese.

Concluzie:

Acest proiect demonstreaza utilizarea MPI pentru paralelizarea calculului pe mai multe procese, imbunatatind performanta in comparatie cu executia seriala.

Pentru fisierul **commands2.txt:**



Pentru fisierul **commands.txt:**