

*Algoritmi paraleli avansați*

Extragerea elementelor distincte dintr-un vector

Student: Mărginea Maria

Profesor: s.l. dr. ing. Prodan Remus

SIC, AN I

2022/2023

Cuprins

[I. Tema proiectului 3](#_Toc139842024)

[II. Noțiuni teoretice 3](#_Toc139842025)

[III. Implementarea problemei 3](#_Toc139842026)

[Procesarea serială 4](#_Toc139842027)

[Procesarea paralelă 5](#_Toc139842028)

[IV. Date experimentale 6](#_Toc139842029)

[V. Serial vs Paralel 7](#_Toc139842030)

[VI. Concluzii 8](#_Toc139842031)

[VII. Bibliografie 8](#_Toc139842032)

# Tema proiectului

Tema proiectului reprezintă extragerea elementelor distincte dintr-un vector. Proiectul are ca date de intrare un vector de n numere și returnează un alt vector cu elementele distincte din vectorul de intrare. Am studiat diferențele dintre cele două tipuri de procesări: serială și paralelă și am analizat performanțele și avantajele acestor tipuri.

Am ales această temă deoarece am vrut să observ performanțele și avantajele procesării paralele în comparație cu procesarea serială.

# Noțiuni teoretice

Pentru realizarea proiectului am ales limbajul de programare C. Pentru procesarea serială am folosit următoarele biblioteci:

* #include <stdio.h>
* #include <sys/time.h> . Am folosit această bibliotecă pentru a calcula timpul de execuție.

Pentru procesarea paralelă am folosit următoarele biblioteci:

* #include <stdio.h>
* #include <mpi.h> . MPI(Message Passing Interface) este o bibliotecă ce este utilizată pentru procesarea paralelă a informației. MPI este utilizat cu precădere în cazul în care se dorește comunicarea și implicit procesarea informației pe mai multe PC-uri. Aceasta oferă funcții specifice pentru inițializare, închidere și comunicare între procese cum ar fi : MPI\_Init, MPI\_Finalize, MPI\_Send, MPI\_Recv , MPI\_Reduce, MPI\_Gather, MPI\_Gatherv, etc.
* #include <sys/time.h> . Am folosit această bibliotecă pentru a calcula timpul de execuție.

Am utilizat utilitarul make pentru automatizarea procesului de compilare si executate.

# Implementarea problemei

Proiectul conține fișierul serial.c unde am realizat rezolvarea problemei printr-o metodă clasică, serială. In fișierul paralel.c am rezolvat problema prin procesare paralelă. Fișierul test.c este un script care îmi generează vectorii aleatori pe care îi voi folosi în proiect.

Fișierul make conține acțiuni pentru compilare, curățare și executare proiect:

* Make clean – șterge fișierul paralel (fișierul executabil)
* Make all – compilează proiectul și generează executabilul pentru procesarea paralelă.

## Procesarea serială

Algoritmul pentru afișarea elementelor distincte dintr-un vector primește, ca date de intrare, un vector de diferite dimensiuni. Pe baza acestui vector se realizează verificarea elementelor, daca nu este ”vizitat”, este afișat, adică afișează doar prima apariție a unui număr. Este calculat timpul de execuție a acestui algoritm și apoi este afișat.

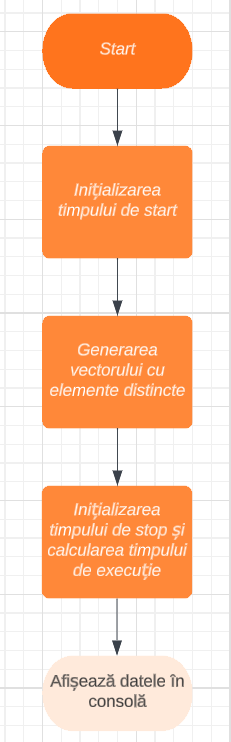


Figura I. Fluxul de date pentru procesarea serială

## Procesarea paralelă

Setul de date pentru test este identic cu cel de la modelul serial. La începutul programului se inițializează numărul de procese pentru care se dorește procesarea datelor în mod paralel, se calculează numărul de elemente pentru fiecare proces și se alocă memorie pentru vectorul final.

Se apelează funcția care verifică care elemente sunt distincte din setul de date de intrare. Această funcție primește ca argumente: setul de date de intrare, dimensiunea acestuia, rank (procesul: 0 este Master și ceilalți sunt Workeri) și numărul de procese. Pentru fiecare proces în parte se determină elementele distincte. Rezultatul obținut după apelul funcției este prelucrat în așa fel încât să se extragă elementele distincte din rezultatele fiecărui proces.

Este calculat timpul de execuție a acestui algoritm și apoi este afișat.

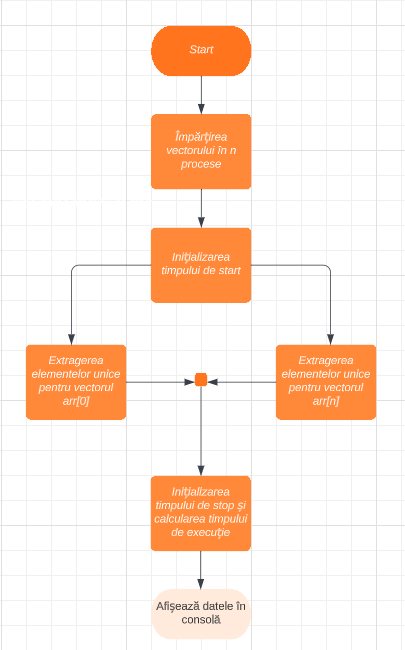


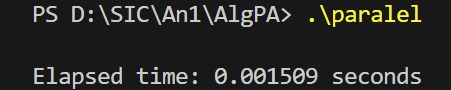
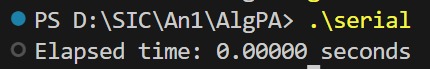
Figura II. Fluxul de date pentru procesarea paralelă

# Date experimentale

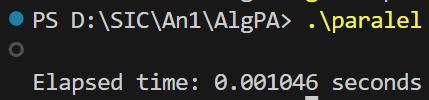
Pentru a testa proiectul, am făcut si o versiune pe Windows a proiectului cu procesare paralele.   
În continuare voi prezenta datele obținute local, pe laptop meu, iar la final voi prezenta și rezultatele obținute pe putty.

Am ales 5 dimensiuni diferite ale vectorului de intrare:

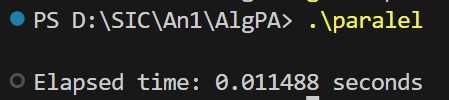
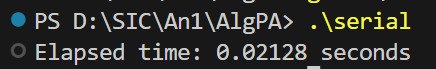
* N=10 – am obținut timpi mai buni pentru procesarea serială

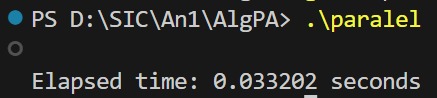
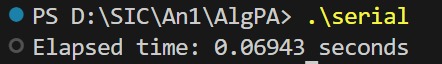
* N=500 – am obținut timpi mai buni pentru procesarea serială

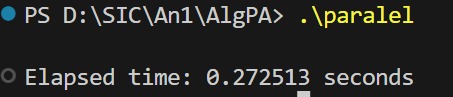
* N=5000 – am obținut timpi mai buni pentru procesarea paralelă

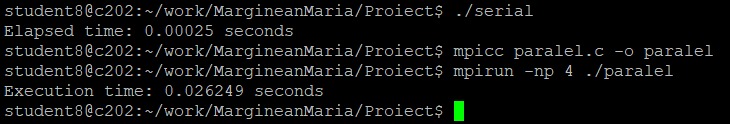
* N=10000 – am obținut timpi mai buni pentru procesarea paralelă

* N=50000 – am obținut timpi mai buni pentru procesarea paralelă

În continuare voi prezenta rezultatele obținute pe putty pentru n=500, procesarea paralelă fiind pe 4 procese.



# Serial vs Paralel

Pentru compararea eficienței dintre programul paralel și serial s-a utilizat un set de date de 5 dimensiuni multipli de 6 începând de la 10 la 50000 de elemente. Pentru procesarea paralelă s-au utilizat 4 procese.

În figura III se poate observa timpul necesar pentru procesarea fiecărui set de date pentru procesarea serială și paralelă. Se poate observa vizibil faptul că pentru seturile foarte mici de date procesarea serială este mai rapidă, în schimb pentru seturile mari de date procesarea paralelă cu 4 procese este de aproape 3 ori mai rapidă. Acest lucru este ușor de înțeles deoarece în cadrul procesării paralele pentru seturile mici de date timpul pentru transmiterea și primirea datelor între procese se ia în calcul, iar pentru seturile mici de date acești tip se pot observa vizibil.

În continuare se observă faptul că performanța pentru seturile mari de date este mult mai bună pentru procesarea paralelă a informației.

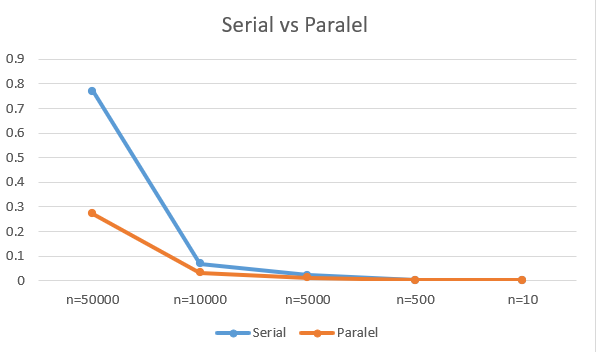


Figura III. Serial vs. Paralel

# Concluzii

În concluzie, se poate observa că pentru setul de date de intrare mari, procesarea paralelă este mult mai optimă și eficientă. Rezultatele obținute pentru a diferenția procesarea paralelă de cea serială sunt satisfăcătoare. Premisă faptului că procesarea paralelă este mai eficientă ca timp se confirmă. În cadrul experimentului, se mai pot lua în calcul pe viitor, și alte tipuri de analize precum: creșterea sau micșorarea numărului de procese paralele, schimbarea algoritmului, schimbarea limbajului de programare, realizarea de optimizări, etc.

Se poate observa faptul că pentru sisteme de operare diferite, timpul de execuție este diferit.

# Bibliografie

<http://dev.fiesc.usv.ro/remus/arhive/SPP/AP_Lab4_Introducere_MPI.pdf>

<http://dev.fiesc.usv.ro/remus/arhive/SPP/AP_Lab5_Programare_MPI.pdf>

[MPI - C++ Examples (sc.edu)](https://people.math.sc.edu/Burkardt/cpp_src/mpi/mpi.html)