**Laborator 3 - PPD**

De Miros Razvan-Andrei

Grupa 234, semigrupa 2

05.12.2022

**Analiza Cerintelor**

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Numerele initiale sunt reprezentate de cifre cuprinse intre 0-9, pe prima pozitie fiind o cifra diferita de 0.

Pe prima linie din fiecare fisier se scrie lungimea numarului, iar in continuare numarul propriu-zis.

**Proiectare:**

**Varianta 0:**

Text

Description automatically generated

* Citirea numerelor se face integral in variabile de tip “string”, nemaifacand conversia la tipul “int” al cifrelor
* Adunarile se fac in felul urmator:
* Text

  Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

* Codul se personalizeaza pe cazurile de testare avute. Varianta de sus se foloseste in cazul de testare 3 cand n1\_size < n2\_size.
* Altfel (cand n1\_size == n2\_size):
* Text

  Description automatically generated
* Se observa ca daca avem cazul in care ultima suma produce carry (adunarea primelor 2 cele mai semnificative cifre din numere), se scrie in fisier prima data carry si apoi array-ul rezultat care reprezinta suma.

**Varianta 1**

Text

Description automatically generated

* Citirea se face in chunkuri din fisiere;
* Text

  Description automatically generated
* Citirea o facem de la finalul fisierului inspre inceput, mutand cursoarele in mod adecvat;
* Text

  Description automatically generated
* In cazul de testare 3, citim intreg numarul 1 si acelasi chunk de lungime egala din numarul 2 (25000 caractere).
* Text

  Description automatically generated
* Fiecarui proces care face lucru de calcul , I se transmite din procesul 0, dimensiunile numerelor, dimensiunea chunkurilor ce urmeaza sa fie transmise, si chunkurile propriu-zise.

A picture containing text

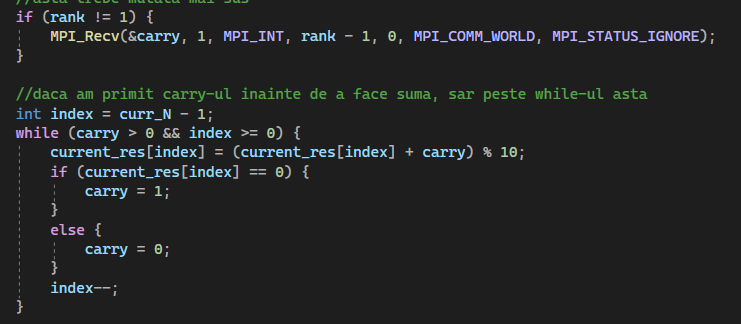
Description automatically generated

Text

Description automatically generated

- mai intai se face suma numerelor primite,

- apoi se asteapta carry



Se reiau calculele cu carry de la finalul numarului actualizand acolo unde este necesar.

A screenshot of a computer

Description automatically generated with medium confidence

* La final procesul trimite carry-ul local urmatorului process, si dupa rezultatul sumei chunkurilor procesului 0 (root).
* In procesul root (0):
* Text

  Description automatically generated
* Se asteapta carry de la procesul p – 1:
* Se scrie in fisier in cazul in care avem (demo v4)
* A screenshot of a computer

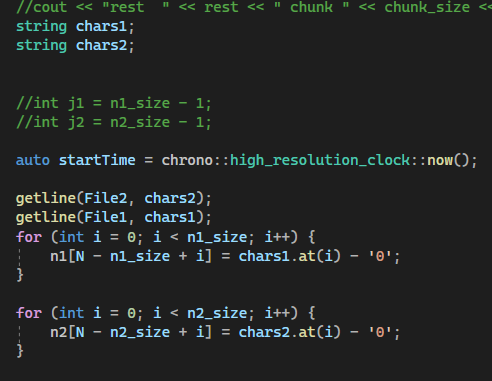
  Description automatically generated with medium confidence
* Se primesc rezultatele si se scriu in fisier pe masura ce se primesc.

**Varianta 2**

Text

Description automatically generated

* Se citesc dimensiunile si incrementam N astfel incat se poate imparti egal la p.
* N / p va reprezenta dimensiunea chunkului transmis fiecarui proces cu functia Mpi\_Scatter;
* Daca N se actualizeaza, numerele se completeaza cu zerouri la inceput



* Se citesc numerele integral si se convertesc la tipul int in array-uri globale.
* 
* Se transmite fiecarui process dimensiunea chunkului ce urmeaza sa fie primit.
* Text

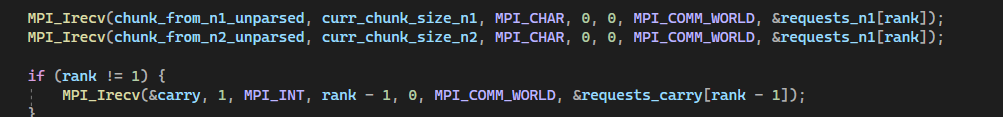
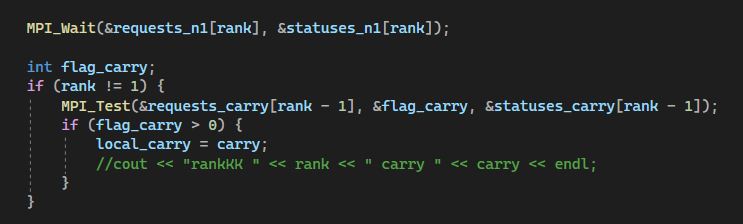
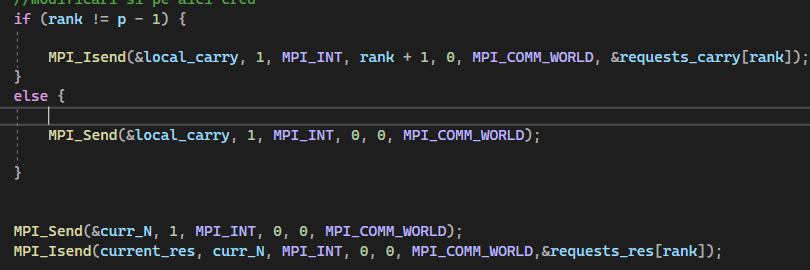
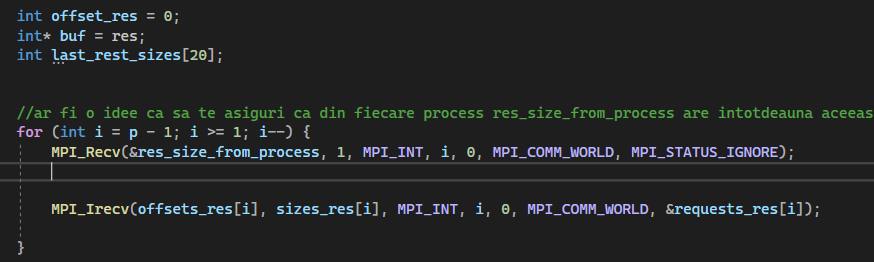
  Description automatically generated
* Se transmite cu scatter cate un chunk din array-ul global la fiecare proces.
* Sumele si transmiterea carry-ului se face asemanator ca in celelalte variante



* La final in procesul 0, se face gather in array-ul global “res”.

**Varianta 3**

* Codul este asemanator variantei 1 cu urmtoarele modificari:
* La transmiterea chunkurilor:
* Graphical user interface

  Description automatically generated
* Ne folosim de offseturi si de array-uri globale pentru a nu astepta dupa fiecare citire si transmitere de chunkuri, folosim varianta non-blocking
* 
* Primim non-blocking in restul proceselor
* 
* Verificam daca am primit intre timp carry-ul.
* Daca este primit, nu vom mai folosi 2 carry-uri (cel local si cel primit) si deci la final dupa ce sumele se vor termina vom transmite direct local\_carry.
* Daca nu este primit pana in acel punct, executia sumelor este asemanatoare cu cea din varianta 1.
* 
* La final se transmit non-blocking carry-ul si rezultatul.
* 
* La primire ne folosim tot de offseturi, pentru a nu bloca si a trebui sa asteptam primirea de la fiecare proces.

Varianta 0)

|  |  |  |  |
| --- | --- | --- | --- |
| Caz testare | | Nr procese | Timp executie(ms) |
| 1. N1 = n2 = 18 |  | secvential | 0.46233 |
| 1. N1 = n2 = 1000 | | secvential | 0.80699 |
|  |
|  |
| 1. n1 = 100   N2 = 100000 | | secvential | 4.73102 |

Varianta 1)

|  |  |  |
| --- | --- | --- |
| Caz testare | Nr procese | Timp executie(ms) |
| 1. N1 = n2 = 18 | 4 |  |
| 2.50777 |
|  | 8 | 3.41774 |
|  | 16 | 4.37847 |
| 1. N1 = n2 = 1000 |  |  |
| 4 | 4.37847 |
| 8 | 5.68964 |
| 16 | 4.68964 |
| 1. n1 = 100   N2 = 100000 | 4 | 36.79951 |
| 8 | 37.6099 |
| 16 | 40.6099 |

Varianta 2)

|  |  |  |
| --- | --- | --- |
| Caz testare | Nr procese | Timp executie(ms) |
| 1. N1 = n2 = 18 | 4 |  |
| 1.35067 |
|  | 8 | 1.93162 |
|  | 16 | 3.16132 |
| 1. N1 = n2 = 1000 | 4 |  |
| 1.50844 |
| 8 | 2.27382 |
| 16 | 3.57758 |
| 1. n1 = 100   N2 = 100000 | 4 | 25.9375 |
| 8 | 26.92501 |
| 16 | 29.35798 |

Varinata 3)

|  |  |  |
| --- | --- | --- |
| Caz testare | Nr procese | Timp executie(ms) |
| 1. N1 = n2 = 18 | 4 | 2.54457 |
|  | 8 | 3.07425 |
|  | 16 | 4.35384 |
| 1. N1 = n2 = 1000 | 4 |  |
| 3.00189 |
| 8 | 3.6484 |
| 16 | 4.96536 |
| 1. n1 = 100   N2 = 100000 | 4 | 35.75574 |
| 8 | 32.8287 |
| 16 | 34.59195 |

**Comparatie v1 vs v3**

Varianta 1)

*Varianta 3)*

*Observatii!:*

* *Am observant ca in varianta 3, prin transmiterea asincrona avem eficienta mai buna. (mica dar, observabila).*
* *In timpul de executie am inclus si citerea dar si scrierea in fisier, prin urmare e posibil ca varianta 3 sa fie mult mai eficienta decat in rezultatele date.*
* *Varianta secventiala ramane cea mai buna.*