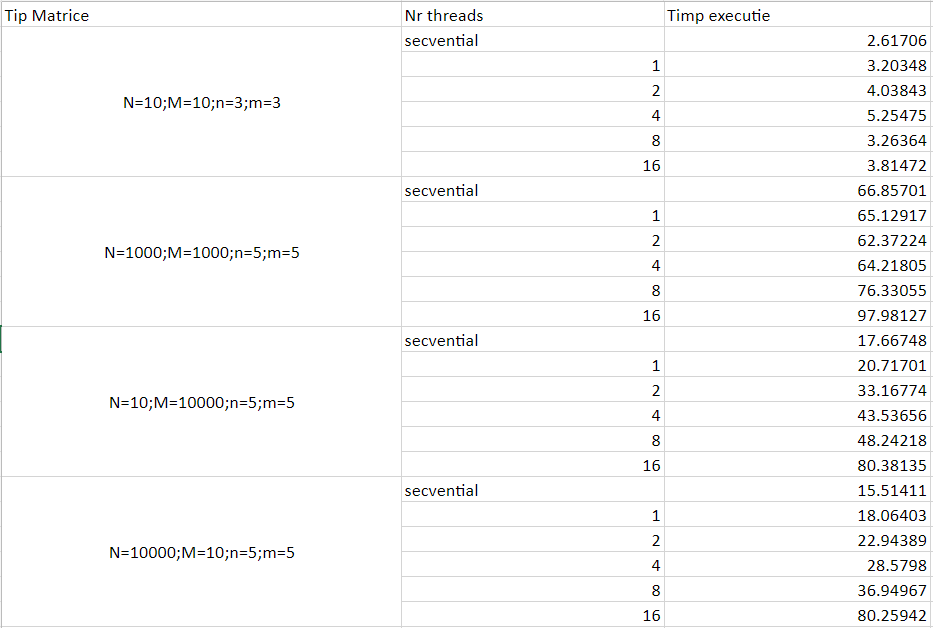
Documentatie PPD Lab 2

* Info:
  + OS: Windows 10
  + Processor: 11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz 2.30 GHz
  + Cores: 8
  + Logical Processors: 16
  + RAM: 16.0 GB (15.7 GB usable)
* Idee:
  + Pentru fisierele de intrare formatul este astfel: Daca dorim numarul 12345, fisierul are pe prima linia 5 (nr de cifre), iar pe linia 2 cifrele in ordine inversa (54321)
  + Vom creea o structura/clasa big\_number ce are numarul de cifre si cifrele in sine salvate (cifrele sunt reprezentate in ordine inversa)
  + Clasa va avea o metoda de adunare, care primeste alt numar, si va aduna atat cat poate la sine, iar daca da overflow va semnala asta prin return
  + Putem considera un big\_number (123456789) ca putand fi reprezentat echivalent printr-o fragmentare acestuia in big\_numbers mai mici (123, 45 si 6789 spre exemplu)
  + FIe P numarul de procese de tip worker, ce realizeaza adunari
  + Fragmentam astfel numarele in P partitii, o partitie avand N/P sau N/P+1 cifre, unde N=max(cifrele primului numar, cifrele celui de al doilea numar) (in cazul lui V2, lui N i se va aduna cel mai mic numar astfel incat el sa devina multiplu de P si sa se poata imparti egal)
  + Fiecare proces aduna la numarul cu mai multe cifre numarul cu numar mai mic de cifre (daca au numar egal de cifre, se alege oricare), impreuna cu eventualul carry de la procesul anterior
  + Daca apare carry, il trimite la procesul urmator – ultimul proces il va trimite la procesul master
  + Procesul master primeste rezultatele si reconstruieste raspunsul, apoi, daca are carry de la ultimul proces, trateaza numarul ca si cand ar incepe cu un extra 1
* Before word:
  + Pentru varianta 3, am realizat 2 subvariante
  + V3\_1 este cea mai eficienta implementare, facand trimiteri de date asincrone doar unde chiar se observa imbunatatiri (toate send-urile si receive-urile de carry se fac asincron, inafara de cel din master)
  + V3\_2 realizeaza aceleasi trimiteri asincrone ca V3\_1 + tot ceea ce trimite/primeste master-ul se face asincron, cu exceptia primirii ultimului carry
* Specificatii:
  + Specifcatiile sunt puse pe codul sursa, la fiecare metoda, impreuna cu detaliile extra de implementare
* In C++:
  + Diagrama de clase:



* + Rezultate:



* + LEGENDA: P – nr de procese
  + Pentru N1\_Size=N2\_Size=18:
    - Varianta secventiala este cea mai eficienta – de asteptat pentru valori atat de mici
    - Cu cresterea numarului de procese, toate variantele devin mai slabe, garantat datorita timpului de trimitere/primire a datelor
    - Overall: Secvential>MPI\_V1(P=2)>MPI\_V3\_1(P=2) >MPI\_V2(P=4)>MPI\_V3\_2(P=2)
    - Cu cat logica e mai simplista, rezolvarea e mai eficienta
  + Pentru N1\_Size=N2\_Size=1000:
    - Varianta secventiala este cea mai eficienta – numarul de cifre nu este indeajuns de mare sa se observe o diferenta
    - Cu cresterea numarului de procese, toate variantele (inafara de MPI\_V3\_2) - in continaure, nu-i indeajuns de mare numarul de cifre probabil si se observa timpul de trimitere/primire
    - MPI\_V3\_2 pare sa fie mai fluctuant ca timp, posibil datorita timpilor de asteptare – sunt atat de multe request-uri dupa care se asteapta (pentru fiecare array si carry de trimis/primit) incat fluctuatiile de la ei se resimt mai tare in parent
    - Overall: Secvential> MPI\_V3\_1(P=2) >MPI\_V2(P=2)>MPI\_V1(P=4)>MPI\_V3\_2(P=8)
    - Cu cat logica e mai simplista, rezolvarea e mai eficienta
  + Pentru cazurile N1\_Size=100 si N2\_Size=100000:
    - Varianta MPI\_V1 cu P=2 este cea mai eficienta
    - Cu cresterea numarului de procese, toate variantele (inafara de MPI\_V3\_2) - am si rerulat, pare ca pentru 2 si 4 procese sta foarte mult MPI\_V3\_2, probabil din cauza send-urilor mari si a wait-urilor?
    - MPI\_V3\_2 pare sa fie mai fluctuant ca timp, posibil datorita timpilor de asteptare – sunt atat de multe request-uri dupa care se asteapta (pentru fiecare array si carry de trimis/primit) incat fluctuatiile de la ei se resimt mai tare in parent
    - Overall: MPI\_V1(P=2)>MPI\_V2(P=2)> MPI\_V3\_1(P=2) > MPI\_V3\_2(P=8) >Secvential
    - Cu cat logica e mai simplista, rezolvarea e mai eficienta
  + Concluzii:
    - Intre toate variantele sunt diferente foarte mici de timp intre variantele cele mai bune
    - In principiu, cresterea numarului de procese scade eficienta...