**Send si Recv:**

 **Procesul 0 (Master)** este responsabil pentru citirea segmentelor de date din fișiere, trimiterea acestora către procesele de lucru și colectarea rezultatelor.

 **Procesele Worker** primesc segmente de date, efectuează adunarea pentru secțiunile proprii și propagă „carry”-ul către procesele următoare.

**Abordarea adunării**

1. **Citirea datelor din fișiere de la final spre început**:
   * Fiecare număr este stocat într-un fișier, iar citirea începe de la sfârșit pentru a obține corect cifrele semnificative. Astfel, fiecare proces primește un segment (subsecțiune) din număr, necesar pentru adunare.
2. **Împărțirea datelor**:
   * Procesul 0 împarte datele din fișiere în funcție de numărul de procese și trimite fiecărui proces Worker un segment de dimensiune aproximativ egală, împreună cu indicii start și end care specifică partea de procesat.
   * Segmentarea este realizată astfel încât procesul 0 distribuie SIZE / (numprocs - 1) elemente fiecărui Worker și ajustează pentru eventualele elemente rămase (restul).
3. **Adunarea vectorilor și propagarea „carry”-ului**:
   * Fiecare proces adună cifrele corespunzătoare din cele două segmente primite și gestionează „carry”-ul (transportul) rezultat.
   * Dacă procesul nu este primul, primește „carry”-ul de la procesul anterior. Dacă transportul calculat este mai mare decât 9 (din adunare), acesta este trimis la procesul următor.
   * Adunarea este realizată folosind o funcție specializată care returnează atât suma pentru segmentul respectiv, cât și „carry”-ul rezultat.
4. **Colectarea rezultatelor și calculul final**:
   * Procesul 0 primește rezultatele de la fiecare proces Worker și reface vectorul final al sumei.
   * În cazul în care mai există „carry” rezidual după ce au fost combinate toate segmentele, acesta este adăugat la finalul rezultatului pentru a completa calculul corect al sumei.
5. **Salvarea rezultatului și măsurarea timpului de execuție**:
   * După finalizarea calculului, procesul 0 salvează vectorul rezultat într-un fișier și afișează timpul total de execuție, măsurat cu ajutorul funcțiilor de cronometrare (std::chrono).

**Scatter si Ghater**

* **Procesul 0** joacă rolul de Master și gestionează citirea inițială a datelor, distribuirea acestora și prelucrarea rezultatului final.
* **Procesele de lucru (Workers)** primesc un segment din numere, calculează suma parțială a segmentelor și propagă „carry”-ul (transportul) rezultat din calcule.

**Pașii programului**

1. **Citirea inițială a datelor și pregătirea pentru distribuire**:
   * **Procesul 0** citește cele două numere din fișiere și le stochează în vectori (vector1 și vector2).
   * Dimensiunea maximă dintre cele două numere este extinsă pentru a fi divizibilă cu numărul de procese (MPI\_Comm\_size), astfel încât fiecare proces să primească un segment de dimensiune aproape egală. Vectorii sunt completați cu 0 la sfârșit, dacă este necesar.
2. **Broadcast-ul dimensiunii vectorilor**:
   * După ce procesul 0 a stabilit dimensiunea finală a vectorilor, acesta trimite (MPI\_Bcast) informația despre dimensiune tuturor proceselor, asigurând sincronizarea între ele.
3. **Împărțirea segmentelor de date între procese**:
   * Fiecare proces primește un segment din cei doi vectori (corespunzător unei secțiuni din numerele de adunat) prin funcția MPI\_Scatter, distribuind vectorii inițiali în sub-vectori (chunks) pentru procesele Worker.
   * Fiecare proces calculează astfel un segment din suma celor două numere, păstrând „carry”-ul local pentru segmentul său.
4. **Calcularea sumei pentru fiecare segment și propagarea „carry”-ului**:
   * **Fiecare proces** realizează adunarea segmentului primit prin funcția addLargeNumbers, care returnează atât suma segmentului, cât și „carry”-ul rezultat.
   * Începând de la procesul cu rangul 1, fiecare proces primește „carry”-ul de la procesul anterior folosind MPI\_Recv. Acesta este utilizat pentru recalcularea ultimei cifre a segmentului său, asigurându-se că transportul este luat în considerare.
   * Fiecare proces trimite apoi „carry”-ul rezultat următorului proces folosind MPI\_Send.
5. **Colectarea rezultatelor și tratarea „carry”-ului final**:
   * Procesul 0 adună segmentele prelucrate folosind MPI\_Gather pentru a construi vectorul complet al sumei.
   * **Ultimul proces** trimite „carry”-ul final către procesul 0, care adaugă acest „carry” la rezultatul final, dacă este necesar, asigurând corectitudinea sumei.
6. **Afișarea rezultatului și măsurarea timpului de execuție**:
   * Procesul 0 afisează și salvează vectorul rezultat într-un fișier și raportează timpul total de execuție (folosind std::chrono pentru măsurare precisă).