|  |  |  |  |
| --- | --- | --- | --- |
| Tip Executie | Nr procese | Timp executie T1 | Timp executie T2 |
| scatter | 4 | 20.66109 | 2501.243 |
| 8 | 15.41321 | 2404.313 |
| 20 | 10.4698 | 2463.443 |
| 4 | 21.12743 | 2492.786 |
| 8 | 15.09624 | 2525.598 |
| 20 | 10.88843 | 2627.011 |
| send/recv | 5 | 1231.132 | 2476.246 |
| 9 | 1224.876 | 2466.9 |
| 21 | 1228.662 | 2473.888 |
| 5 | 1244.728 | 2510.385 |
| 9 | 1231.002 | 2483.689 |
| 21 | 1224.602 | 2469.507 |

*Comparatie in functie de p si de varianta*

Pentru prima varianta, folosind send si receive, diferentele intre timpii de executii sunt destul de mici, in cazul al doilea al rularii cu o noua matrice generate, timpii find imbunatati cu cat se creste numarul de procese, dar diferenta este una nesimnificativa avand in vedere media totala. In schimb, in cazul in care am folosit scatter si gather, timpul de calcul efectiv T1 a scazut considerabil, iar crescand numarul de procese, se poate vedea si o imbunatatire a timpilor pentru ambele cazuri. Cu toate acestea, un punct comun al ambelor variante este faptul ca operatiile de citire, respectiv salvare, a matricei sunt cele mai costisitoare din punct de vedere temporal.

*Comparatie cu ultimul laborator cazul Send/Recv*

Pentru varianta 1 a laboratorului, timpii efectivi de calculi scad considerabil fata de laboratorul trecut, fiind de 60 de ori mai rau decat cazul secvential pentru matricea de 1000x1000, fiind aproape de cazul cu 16 thread-uri pentru matricea de 10000x10000. De aici putem deduce ca in functie de dimensiunea datelor pe care le avem, trebuie sa alegem cu atentie metoda folosita.

*Comparatie cu ultimul laborator cazul Scatter/Gather*

Totusi, putem constanta o imbunatatire majora fata de laboratul trecut, unde in cazul 1000x1000, aveam un timp de aproximativ 21 de secunde abia la 8 thread-uri, aici aceasta performanta fiind obtinuta la 4 procese, echivalentul la 16 thread-uri, fiind 8 procese. De aici putem concluziona, ca crescand numarul de procese, putem creste si performanta mai mult decat am face-o creand thread-uri. Este de mentionat, ca timpul scos la procese, este de 3, 4, respectiv 6 ori mai bun fata de executia secventiala.