Laborator 12

Tutorial llnl
MPI The complete Reference

Exerciții

1. (**nonBlocking.c**) Implementați un program MPI ce are două procese. Procesul cu rank 0 trimite procesului cu rank 1 un vector folosind funcții non-blocante. Afișați vectorul după primire.
2. Scrieți în readme ce se întâmplă dacă imediat dupa send, în procesul cu rank 0 este modificat vectorul.
3. Scrieți în readme ce se întâmplă dacă modificare făcută în vector este la o poziție foarte mare 100.000+.
4. (**sendIsBlocking.c**) Pornind de la programul dat arătați că MPI_Send poate fi blocant. Notați cum ați încercat și care este rezultatul în readme.
5. (**lider.c**) Implementați alegerea liderului.
   - Se folosește un algoritm de tip heart-beat.
   - Toți liderii pot fi inițiatori.
   - Știm că rețeaua are mai puțin de 100 de noduri.
6. (**epidemic.c**) Implementați algoritmul epidemic.
   - ATENȚIE: Un nod poate comunica doar cu vecinii. MPI vă va permite să trimiteți mesaje de la orice nod la oricare altul, dar noi vrem să simulăm o rețea reală în care se poate comunica doar cu vecinii. Astfel, nu aveți voie să faceți send sau recv, decât de la un nod care este în lista de vecini.
   - Se începe de la programul anterior, avem nevoie de un lider ales. Faceți o copie codului anterior.
   - Fiecare proces are o valoare. Pentru toți valoarea este 0 cu excepția liderului, acesta are valoarea 1.
   - Valoarea este transmisă și la fiecare pas se modifică ca fiind media între valoarea primită și cea locală.
   - Se vor folosi exclusiv funcții de comunicare MPI_Sendrecv.
   - Presupunem că nu știm mărimea rețelei, scopul este să o aflăm.

Exercițiile de la 1 la 6 sunt **obligatorii**. Conceptele explorate sunt esențiale pentru obținerea notei **minime** de promovare.

Vă recomandăm, pentru a crește șansele de a obține o notă cât mai mare să explorați și următoarele exerciții:

7. Implementați un program în MPI care să facă calcule în timp ce sunt transmise date.