Laborator 03

Tutorial llnl
MPI The complete Reference

Exerciții

Pentru fiecare exercițiu se va scrie în fișierul REPORT.txt rezultatul rulărilor și răspunsul la întrebări.

Toate soluțiile din acest laborator necesită o implementare scalabilă (cu cât creștem numărul de procese cu atât va executa mai rapid programul). Pentru a obține o implementare scalabilă trebuie:

- **Să inițializați/afișați doar de pe procesul cu rank 0.**
- **Să funcționeze cu un singur proces.**
- **Să minimizați comunicația (transmiteți doar strictul necesar de date).**
- **Să transmiteți mai multe date o dată, în loc de multe transmisii mici.**
- **Să folosiți N destul de mare încât timpul de execuție cu un proces să depășească 1 secundă. Timpii mici sunt irelevanți.**
- **Să fie oprită afișare când măsurați (argument program printLevel să fie 0).**
- **Să comparați doar 1, 2, 4 procese.**
  - **De la 4 în sus e dificil ca un program să scaleze.**
  - **Dacă depășiți numărul de core-uri complete (non-hyper threading) timpul de execuție va crește.**

Timpul de execuție se va măsura din linia de comandă cu utilitarul time.

```
time mpirun -n 2 ./myprogram N printLevel
```

Pentru a putea face măsurători fără a recompila, programele vor primi ca parametri:

- **N – numărul de elemente (mărimea reală va fi N*N pentru matrice)**
- **printLevel – 0 = fără printare, 1 = printare full, 2 = printare parțială**

1. (**1_vectorPower.c**) Pornind de la codul dat (care calculează o putere a lui 42 folosind un calcul modulo) implementați programul distribuit cu MPI care să scaleze cu numărul de procese.
2. (**2_4_6_8_timpi.csv**) Demonstrați că implementarea este scalabilă măsurând timpii de execuție. Completați coloana din fișierul excel.
3. (**3_rankSort.c**) Implementați soluția distribuită MPI pentru sortarea cu algoritmul Rank Sort. Rank Sort este prezentat la începutul cursului din link. Sunt suficiente explicațiile despre varianta secvențială dar dacă vreți să vă inspirați și din partea de paralel puteți ignora partea cu bariere și conflicte de memorie.
4. (**2_4_6_8_timpi.csv**) Demonstrați că implementarea este scalabilă măsurând timpii de execuție. Completați coloana din fișierul excel.

5. (**5_monteCarloPi.c**) Implementați soluția distribuită MPI pentru algoritmul Monte Carlo pentru estimare valorii numărului **π (PI)**. Pentru a înțelege algoritmul secvențial puteți viziona video-urile de aici și aici.
6. (**2_4_6_8_timpi.csv**) Demonstrați că implementarea este scalabilă măsurând timpii de execuție. Completați coloana din fișierul excel.

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. (**7_matrixMultiply.c**) Implementați soluția distribuită MPI pentru înmulțirea a două matrici.
8. (**2_4_6_8_timpi.csv**) Demonstrați că implementarea este scalabilă măsurând timpii de execuție. Completați coloana din fișierul excel.