Haja Florin-Gabriel

Programare Paralelă și Distribuită – Laborator 3

Aceeasi problema ca si la Laborator 4 dar cu urmatoarele modificari:

* Dintre cele p threaduri p1 sunt cititoare si p2 sunt de tip ‘worker’ (preiau din coada si adauga in lista rezultat – p=p1+p2

**Sincronizarea operatiile cu lista inlantuita se fac la nivel de nod nu pentru intraga lista!**

Limbaj: la alegere intre Java si C++

Analiza timpului de executie pentru urmatoarele cazuri:

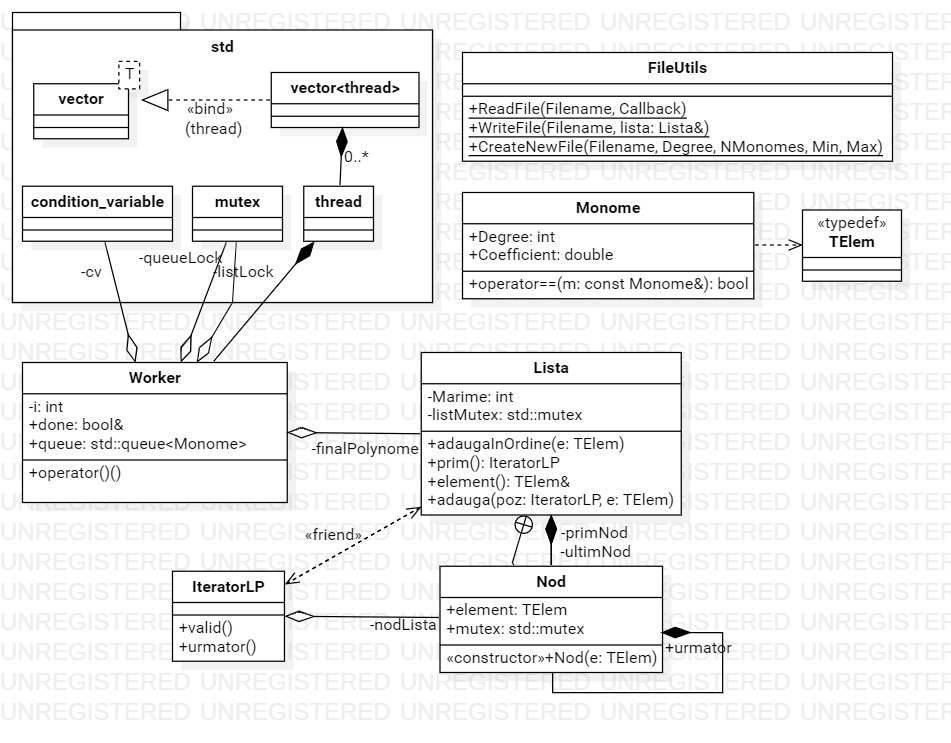
1. Rezolvare secventiala
2. 10 polinoame fiecare cu gradul maxim 1000 si cu maxim 50 monoame
   1. p = 4, 6, 8 ; si p1=2 , p1=3
   2. secvential ; si p1=2 , p1=3
3. 5 polinoame fiecare cu gradul maxim 10000 si cu maxim 100 monoame
   1. p = 4, 6, 8 ; si p1=2 , p1=3
   2. secvential ; si p1=2 , p1=3

Analiza comparativa a performantei –Lab 4) versus Lab 5)

Implementarea se bazează pe paradigma problemei producător-consumator. Principiul este de a crea p-1 obiecte de tip Worker, care așteaptă, prin intermediul variabilei condiționale cv, să fie introduse monoame în coada queue în timpul citirii celor n polinoame din fișiere.

Cazul sincronizării pe noduri l-am rezolvat adăugând câte un mutex în fiecare nod, blocându-le în momentul în care apelez operațiile din clasa Lista. La fiecare pas, blochez nodul curent și nodul anterior, pentru a asigura validitatea adresei următorului nod pentru funcțiile care folosesc iteratori care au ajuns pe poziții inferioare.

Diagrama de clase arată în felul următor:

Observăm următoarele rezultate pe o configurație cu procesor Intel Core i5-8250U, 8 GB RAM DDR4 la 2400 MHz și Windows 10 (build 19041.572):

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Run no. | Language | Execution time | No. of threads | No. of reading threads | No. polynomes | Max degree | No. monomes |
| 1 | C++ | 0.0018 | 1 | 1 | 10 | 1000 | 50 |
| 2 | C++ | 0.0016 | 1 | 1 | 10 | 1000 | 50 |
| 3 | C++ | 0.0028 | 4 | 2 | 10 | 1000 | 50 |
| 4 | C++ | 0.0038 | 4 | 3 | 10 | 1000 | 50 |
| 5 | C++ | 0.0032 | 6 | 2 | 10 | 1000 | 50 |
| 6 | C++ | 0.0026 | 6 | 3 | 10 | 1000 | 50 |
| 7 | C++ | 0.0026 | 8 | 2 | 10 | 1000 | 50 |
| 8 | C++ | 0.003 | 8 | 3 | 10 | 1000 | 50 |
| 9 | C++ | 0.0016 | 1 | 1 | 5 | 10000 | 100 |
| 10 | C++ | 0.001 | 1 | 1 | 5 | 10000 | 100 |
| 11 | C++ | 0.0014 | 4 | 2 | 5 | 10000 | 100 |
| 12 | C++ | 0.0022 | 4 | 3 | 5 | 10000 | 100 |
| 13 | C++ | 0.0018 | 6 | 2 | 5 | 10000 | 100 |
| 14 | C++ | 0.002 | 6 | 3 | 5 | 10000 | 100 |
| 15 | C++ | 0.004 | 8 | 2 | 5 | 10000 | 100 |
| 16 | C++ | 0.0032 | 8 | 3 | 5 | 10000 | 100 |

Graficul rulării programelor arată astfel:

Fig. 1 10 polinoame, cu grad maxim 1000, fiecare având 50 de monoame

Fig. 2 5 polinoame, cu grad maxim 10000, fiecare având 100 de monoame