Haja Florin-Gabriel

Programare Paralelă și Distribuită – Laborator 3

Obiectiv:

* Intelegerea/aprofundarea sablonului “producator-consumator”
* Intelegerea/aprofundarea sincronizarii conditionale
* Intelegerea/aprofundarea excluderii mutuale (granularitatea sectiunilor scritice)

Se considera n polinoame reprezentate prin lista de monoame (reprezentare: lista inlantuita ordonata dupa exponentii monoamele).

Se cere adunarea polinoamelor folosind o implementare multithreading (p threaduri).

Polinoamele se citesc din fisiere – cate un fisier pentru fiecare polinom:

- un fisier contine informatii de tip (coeficient, exponent) pentru fiecare monom al

unui polinom,

(conditie: fisierele nu contine monoame cu coeficient egal cu 0)

Metoda:

1. Se creeaza o lista inlantuita - L corespunzatoare unui polinom nul.
2. Primul thread citeste cate un monom si il adauga intr-o structura de date de tip coada.

(conditie – pentru structura de tip coada NU se admite folosirea unei structuri de date pentru care partea de sincronizare este deja implementata!!!)

1. Celelalte threaduri preiau cate un monom din coada si il aduna la polinomul reprezentat in lista L.
   * Se continua operatiile 1.,.2., 3. pana cand toate monoamele, din toate fisierele, sunt adunate la lista L.
2. Rezultatul obtinut in lista L se scrie intr-un fisier rezultat

(conditie: fisierul nu contine monoame cu coefficient egal cu 0)

**Sincronizare la nivel de lista!!!**

Limbaj: la alegere intre Java si C++

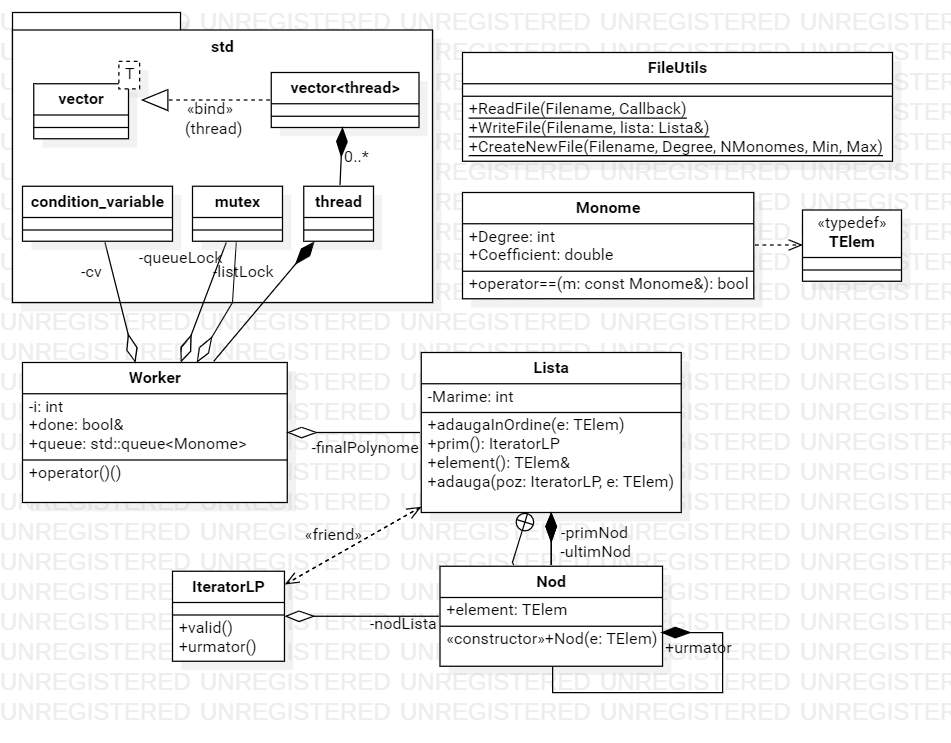
Analiza timpului de executie pentru urmatoarele cazuri:

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

Fisierele input se creeaza prin generare de numere aleatoare!

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.

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. polynomes | Max degree | No. monomes |
| 1 | C++ | 0.002 | 1 | 10 | 1000 | 50 |
| 2 | C++ | 0.0026 | 4 | 10 | 1000 | 50 |
| 3 | C++ | 0.003 | 6 | 10 | 1000 | 50 |
| 4 | C++ | 0.0024 | 8 | 10 | 1000 | 50 |
| 5 | C++ | 0.001 | 1 | 5 | 10000 | 100 |
| 6 | C++ | 0.002 | 4 | 5 | 10000 | 100 |
| 7 | C++ | 0.0024 | 6 | 5 | 10000 | 100 |
| 8 | C++ | 0.002 | 8 | 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