ALGORITMI PARALELI SI DISTRIBUITI:
Tema #3 Prelucrarea de imagini folosind rețele MPI

Termen de predare: 11 Ianuarie 2017, ora 23:55

Responsabili Tema: *Elena Apostol, Vlad Drăgoi, Dragoș Comăneci*

Cuprins

1. Cerințele temei

O imagine este reprezentarea unei poze si consta de obicei într-o matrice de numere (m x n). Fiecare element al unei imagini poarta numele de pixel si are o valoare care exprima intensitatea luminii sau culoarea (RGB).

Pentru simplificarea algoritmilor, in cadrul temei, se vor utiliza imagini in tonuri de gri (*gray-scale*) pentru care fiecare pixel va avea o valoare in intervalul *0 - 255*. Imaginile vor fi in formatul PGM (vezi Secțiunea 4)

Exista o multitudine de algoritmi de procesare de imagini si fiecare poate beneficia de pe urma paralelizării. Mai mult, o aceeași operație este aplicata de multe ori unui flux de imagini. Operațiile de procesare de imagini variază de la algoritmi pe pixeli individuali (cum ar fi ajustarea contrastului), la operații locale pe grupuri de pixeli (uniformizare, reducerea zgomotului) pana la operații globale pe toți pixelii (codare, decodare).

Aceasta tema își propune sa implementeze o aplicație distribuita de procesare a unui set de imagini folosind procese modelate într-o topologie de tip graf.

Tema consta din 3 parți: aflarea arborelui de acoperire, procesarea fiecărei imagini in parte si statistica referitoare la numărul de linii prelucrate la nivel de proces. Detaliile de implementare pentru fiecare din aceste parți le găsiți in Secțiunea 2.

2. Implementare

2.1 Stabilirea arborelui de acoperire

Se considera ca procesele MPI comunica folosind o topologia predefinita. Topologia este stabila, nu vor apărea modificări ale sale in timpul rulării programului.
La pornire fiecare proces MPI va citi dintr-un fișier lista sa de adiacenta. In acest mod fiecare nod cunoaște doar nodurile cu care este conectat direct si poate comunica doar cu acestea.

In prima etapa a aplicației distribuite fiecare nod al grafului va rula un *algoritm unda* pentru stabilirea arborelui de acoperire. La sfârșitul acestei etape toate nodurile vor determina nodul părinte si nodurile copii. Un nod își elimina vecinii care nu fac parte din arborele de acoperire.

Nodul 0 va fi considerat rădăcina arborelui de acoperire.

Se considera un singur fișier cu topologia (dat ca argument aplicației) din care fiecare nod va citi linia cu lista de adiacenta care ii aparține.

2.2 Prelucrarea distribuita de imagini folosind filtre

După stabilirea arborelui de acoperire, programul va trebui sa aplice diferite filtre pe imaginile de intrare, citite ca matrici de dimensiune (m x n). Pentru o imagine se poate alege unul dintre următoarele filtre: {"smooth", "blur", "sharpen", "mean_removal"}. Modul de implementare a fiecărui filtru este descris in **Secțiunea 3**.

Modelul de prelucrare

Prelucrare unei imagini va fi implementata folosind paradigma Heartbeat. Idea de rezolvare se bazează pe împărțirea imaginii in P fâșii sau blocuri de pixeli. Fiecărui proces (numit Worker in cadrul paradigmei) i se va asigna un bloc. Pe lângă blocul de procesat un proces va primi si cele doua linii de graniță (sus si jos). Aceste linii de graniță sunt necesare deoarece filtrele vor acționa la nivel de pixel (punct din matrice), iar *pentru a calcula noua valoare a unui pixel e necesara si cunoașterea valorilor pixelilor vecini.*

In cadrul acestei teme doar nodurile frunza din arborele de acoperire vor fi Workeri. Orice alt nod din arborele de acoperire va împarți in mod egal blocul primit de la părinte la fiecare nod copil.

Observație: Daca s-ar ajunge la cazul *(puțin probabil)* in care blocul nodului curent ar conține mai puține linii decât numărul de noduri copii, doar primele noduri copii vor primi cate o linie din bloc (împreună cu granițele aferente).

Un bloc ajuns la un nod frunza va fi prelucrat folosind filtru aferent.

La terminarea prelucrării locale, nodul frunza va trimite blocul rezultat (prin aplicarea filtrului) la părinte.

Un nod intermediar in arborele de acoperire va concatena bucățile de matrice primite de la toți copii, si trimite mai departe către părinte.

Nodul rădăcina (*rank 0*) va cumula bucățile primite si le va scrie intr-un fișier de ieșire (rezultând imaginea prelucrata).

Acești pași se vor repeta pentru fiecare imagine citita de nodul rădăcini. Rădăcina va trimite blocurile din următoarea imagine, doar după terminarea procesării imaginii anterioare.

După scrierea in fișier a ultimei imagini, rădăcina va trimite mai jos in arbore mesaje cu tag-ul de terminare.

2.3 Statistici pentru numărul de linii procesate

In ultima etapa, la recepționarea mesajului cu tag de terminare (*de la părinte*), o frunza va răspunde cu numărul de linii procesate in total. După trimiterea mesajului de tip statistica, procesul își termina execuția.

Observație: La acest număr se vor aduna liniile de la toate blocurile procesare. Nu contează ca dimensiunea unei linii poate fi diferita de la o imagine la alta.

Aceste statistici (de la toate nodurile) vor ajunge la rădăcina. Acesta le va scrie in fișierul de ieșire, dat ca parametru.

Observație: Rădăcina va scrie numărul de linii prelucrate pentru toate nodurile, in ordinea crescătoare a rank-urilor. Pentru rădăcina si nodurile intermediare se va scrie „0” (linii procesare).

3. Filtre

Un filtru de convoluție este in esența o matrice de forma
0 0 0
0 1 0
0 0 0 / 1 + 0
aplicata punctului curent si punctelor din jurul lui. Fiecare punct capătă astfel o pondere, iar suma valorilor ponderate este împărțita la un factor si câteodată se aduna si un deplasament.
Matricea de mai sus se numește filtru identitate deoarece lasă neschimbata imaginea originala.
De obicei factorul este suma tuturor ponderilor, astfel încât valoarea finala sa fie in intervalul 0..255. Sunt cazuri in care suma ponderilor este 0 (de exemplu filtrele *emboss*). Atunci factorul va fi 1 iar deplasamentul va fi 127 (o valoare obișnuita), pentru a lumina imaginea finala.

Filtrele de convoluție sunt de dimensiuni variate, 3x3 este o dimensiune normala dar exista de exemplu si filtre 7x7. Unele din filtre sunt simetrice, aplicate uniform punctelor din jur, altele sunt nesimetrice, cum ar fi filtrele de detectare a marginilor.
Cu cat un filtru este de dimensiune mai mare cu atât va rămâne o margine mare care nu poate fi prelucrata. Pentru o matrice 3x3 precum cea din exemplu va rămâne cate un pixel neprelucrat pe fiecare margine.
In cadrul acestei teme se vor procesa toți pixelii, chiar si cei de pe marginile imaginii. Pentru acest lucru veți borda cu 0 matricea (m x n) cu imaginea.

A. Uniformizarea (*smooth*)
Aceasta operație presupune crearea unui pixel care are ca valoare media tuturor punctelor din jur, inclusiv a propriei valori. Valoarea ponderilor va fi deci 1 pentru toate punctele iar factorul va fi 9. Putem scrie deci matricea ca
1 1 1
1 1 1
1 1 1 / 9 + 0

B. Uniformizarea triunghiulara (*blur*)
Filtrul triunghiular localizează schimbările de culoare dintr-o imagine si creează culori intermediare pentru a netezi marginile. Filtrul este:
1 2 1
2 4 2
1 2 1 / 16 + 0
Acest filtru da un efect circular deoarece pixelii care sunt mai departe de margine au pondere mai mica. Efectul obținut este similar cu o fotografie pentru care distanta focala nu a fost bine aleasa.

C. *Sharpen*
Este aproximativ inversul filtrului *blur*, scopul lui este detectarea diferențelor intre pixeli si accentuarea acestor diferențe. Formula sa este:
 0 -2   0
-2 11 -2
0   -2   0 / 3 + 0

Cu cat mai mare este diferența intre punctele cu ponderi negative si punctul care este modificat cu atâta este mai mare schimbarea valorii punctului central. Gradul de accentuare este stabilit de ponderea din centru.

D. Eliminarea mediilor (*mean removal*)
Ca si *sharpen*, este tot un filtru de accentuare. Dar *sharpen* funcționează doar pe orizontala si pe verticala pe când acest filtru acționează si pe diagonale. Tot la fel, valoarea centrala poate fi modificata pentru a modifica gradul de accentuare.
-1 -1 -1
-1 9 -1
-1 -1 -1 / 1 + 0

4. Formatul datelor de intrare/ieșire.

Programul va primi ca argumente in linia de comanda: fișierul ce conține topologia, fișierul de listare a imaginilor de procesat si fișierul de ieșire cu statistica pe linii.

Rularea aplicației:

mpirun -np N ./filtru topologie.in imagini.in statistica.out

Fișierul de topologie < *topologie.in* >

- Fiecare linie va avea următorul format *id_nod: vecin-1 vecin-2 ... vecin-i*

Exemplu:
0: 1 2
1: 0 3 4 5 6
2: 0 7 8
3: 1
4: 1 9 10
5: 1
6: 1
7: 2
8: 2 11
9: 4
10: 4
11: 8

Fișierul de listare a imaginilor < *imagini.in* >

Acesta are următorul format:

- pe linia I: numărul NF de imagini de intrare
- pe următoarele NF linii: filtru-i nume_imagine-i nume_imagine_prelucrata-i

Pentru filtru se poate pune: *smooth, blur, sharpen, mean_removal*

Exemplu:
2
blur image1.pgm image1-b.pgm
sharpen image2.pgm image2-s.pgm

Observație:

- Formatul unui fișier de tip imagine:
  - fișierul va avea formatul **PGM** a cărui descriere o găsiți aici si care poate fi vizualizat cu diverse viewere (gimp, xnview, etc.).

- pentru a converti o imagine dintr-un anume format (ex. JPG) in format PGM se poate folosii următoarea metoda: deschideți poza cu GIMP. Selectați **File →Export as...** Se va alege formatul PGM. La opțiunile de **„Data formatting”** selectați formatul **ASCII** (si NU **RAW**!). Imaginea rezultata va fi in tonuri de gri.

- Fișierul de ieșire cu imaginea prelucrata trebuie sa respecte același format ca si fișierul de intrare (PGM)

Exemplu: Poza aceasta in format JPG s-a convertit folosind GIMP. Rezultatul este aici.

Fișierul de ieșire < statistica.out >
Exemplu:
0: 0
1: 0
2: 0
3: 500
4: 500
5: 1000

5. Teste si punctare

Pentru a verifica funcționalitatea temei folosi aceste teste. *(testele urmează sa fie adăugate)*

Punctajul temei va fi distribuit astfel:

- 15p: Arborele de acoperire
- 50p: Implementarea corecta a filtrelor la nivelul proceselor frunza:
  - *"smooth"- 10p, "blur"-10p, "sharpen"-15p, "mean_removal"-15p*
- 20p: Afișarea corecta a imaginilor rezultate in urma operațiilor de filtrare
- 15p: Consistenta rezultate de statistica

6. Conținutul arhivei temei

„Fișierele și directoarele care contribuie la rezolvarea temei trebuie OBLIGATORIU împachetate într-o arhivă de tip '.zip', cu numele '**Grupa_NumePrenume_TemaX.zip**'”

Arhiva temei va conține codul sursa pentru tema împreună cu fișierele *makefile* si *Readme*.
Temele care nu conțin si aceste doua fișiere nu vor fi corectate!

Fișierul *makefile* va conține (obligatoriu) regulile „*build*” si „*clean*” (ștergere binar).
In urma compilării, binarul rezultat e obligatoriu sa se numească „filtru”.

7. Resurse (optional)

- Christian Graus: Convolution filters http://www.codeproject.com/cs/media/csharpfilters.asp