Calcul paralel folosind MPI

Responsabili de temă: Mihai Carabaş, Larisa Grigore

Data publicării: 30.11.2013

Data ultimei modificări a enunţului: 30.11.2013

Termenul de predare: 14.12.2013, ora 23:55

Cerință

Să se scrie un program care să calculeze mulțimile Mandelbrot [1] și Julia [2] pentru o funcție polinomială complexă de forma *f(z)=z2+c* şi să le afişeze sub formă de imagini grayscale. Programul va fi scris in C/C++ și va fi paralelizat utilizând MPI.

Definiții

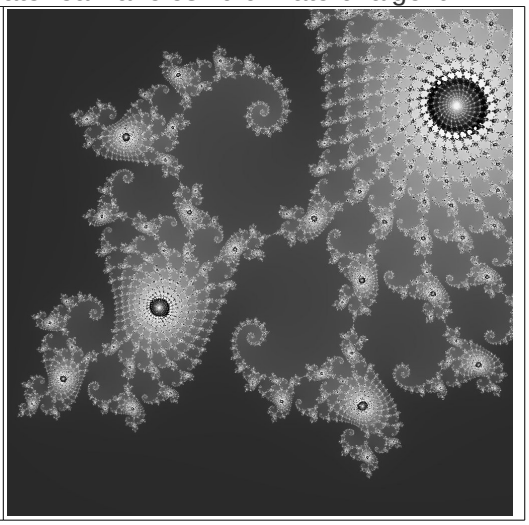
1. Mulțimea Mandelbrot

Fie familia de polinoame complexe *Pc:* $\mathbb{C}\rightarrow\mathbb{C}$, definite de *Pc(z)=z2+c* , cu c un numar complex. Definim *multimea Mandelbrot* ca fiind multimea punctelor c pentru care secventa *0,Pc(0), Pc(Pc(0)),...* nu tinde către infinit.

$M=\{ c \mid \exists s\in\mathbb{R} \text{ a.i. } \forall n\in\mathbb{N}, |Pcn(0)|<s \}$

Generarea și reprezentarea mulțimii Mandelbrot se poate realiza folosind următorul algoritm:

```
foreach c in the complex plane do
    z ← 0 + 0i
    step ← 0
    while |z| < 2 and step < MAX_STEPS do
        z ← z*z + c
        step ← step + 1
    color ← step mod NUM_COLORS
    plot(c.x,c.y,color)
```



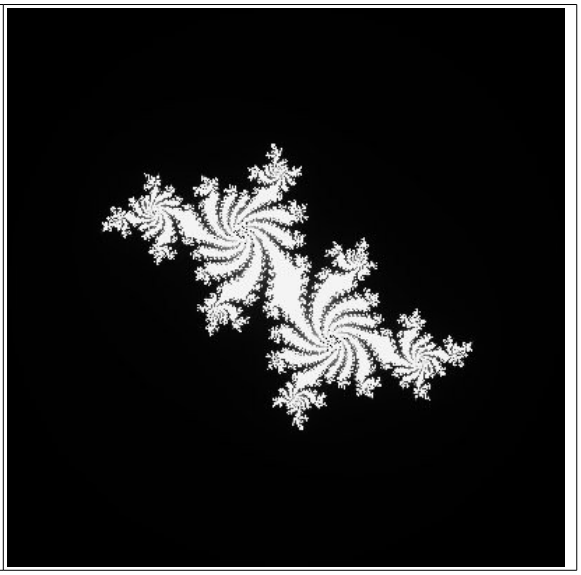
2. Mulțimi Julia

Fie *f(z):* $\mathbb{C}\rightarrow\mathbb{C}$, *f(z)=P(z)/Q(z)* o funcție rațională complexă. *Mulțimea Julia plină Jf* a funcției este mulțimea punctelor din planul complex care au o orbită mărginită în raport cu *f*.

$Jf = \{ z\in\mathbb{C} \mid \exists s\in\mathbb{R} \text{ a.i. } \forall n\in\mathbb{N}, |f\ n(z)|<s \}$

Generarea și reprezentarea mulțimii Julia pline pentru o functie *f(z)=z2+c* se pot realiza folosind următorul algoritm:

```
foreach z in the complex plane do
    step ← 0
    while |z| < 2 and step < MAX_STEPS do
        z ← z*z + c
        step ← step + 1
    color ← step mod NUM_COLORS
    plot(z.x,z.y,color)
```



Paralelizare

Cei doi algoritmi se vor paraleliza folosind MPI, procesele lucrând pe submulțimi de date aproximativ egale. Astfel, un proces master va citi fișierul de intrare, după care va trimite x_min, x_max, y_min, y_max, rezoluția și numărul de iterații celorlaltor procese (se consideră deci că **doar master-ul** are acces la fișier și la aceste informații). Mai departe, fiecare proces în parte (**inclusiv master-ul**) își va calcula intervalul din matricea finală de pixeli pe care lucrează și va aplica algoritmul corespunzător asupra acestuia. La final, fiecare proces va trimite master-ului datele calculate, iar master-ul va crea matricea imaginii finale și o va scrie în fișierul PGM de ieșire.

Format date de intrare/ieșire

Pentru implementarea temei, trebuie să aveți un singur executabil care va primi 2 parametri: numele fișierului de intrare și numele fișierului de ieșire.

Fișierul de intrare va avea următorul format:

- prima linie: un întreg care definește tipul de mulțime care va fi generată (0 - pentu calculul mulțimii Mandelbrot; 1 - calculul mulțimii Julia)
- a doua linie: 4 numere reale (x_min, x_max, y_min, y_max) separate de spații, care definesc subspațiul din planul complex pentru care se realizează calculul. Intervalele pe care se va lucra sunt [x_min,x_max) si [y_min,y_max).
- a treia linie: un număr real care definește rezoluția (pasul) în cadrul subspațiului ales
- a patra linie: numărul maxim de iterații pentru generarea mulțimilor (MAX_STEPS)
- în cazul în care se realizeaza calculul mulțimii Julia (1 pe prima linie), pe cea de-a cincea linie se vor găsi 2 numere reale (și) separate de spațiu, care definesc parametrul complex al funcției

Exemple:

0 -2.5 1.0 -1.0 1.0 0.001 5000		Va genera mulțimea Mandelbrot între -2.5 și 1.0 pe axa OX, respectiv între -1.0 și 1.0 pe axa OY cu o rezoluție de 0.001. În cadrul algoritmului se va folosi un număr de maximum 5000 iterații.
1 -2.0 2.0 -2.0 2.0 0.001 5000		Va genera mulțimea Julia a functiei între -2.0 și 2.0 pe axa OX, respectiv între -2.0 și 2.0 pe axa OY cu o rezoluție de 0.001. În cadrul algoritmului se va folosi un număr de maximum 5000 iterații.

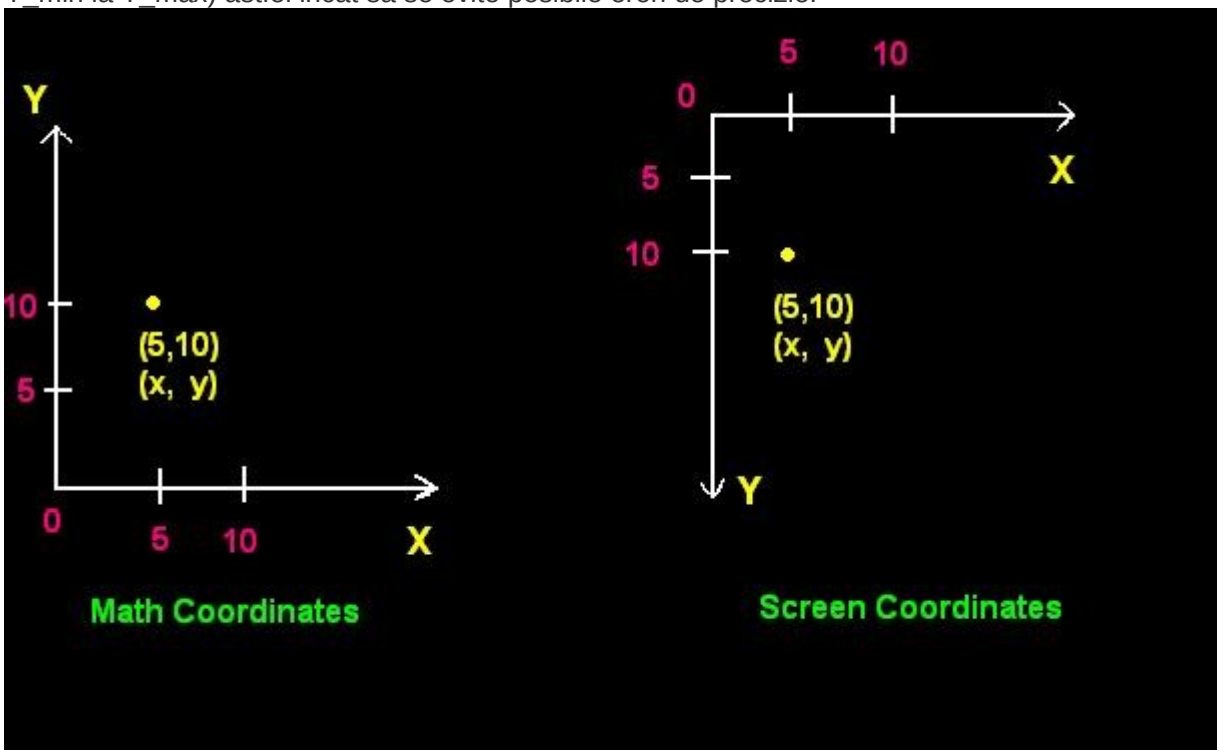
-0.6 0		

Rezultatele programului vor fi scrise în fișierul de ieșire în format PGM “plain” [3]. Imaginea rezultată va avea un numar de 256 de nuanțe de gri (valori intre 0 si 255). Dimensiunile imaginii finale se calculează pe baza x_min, x_max, y_min, y_max si rezoluție, conform formulelor:
Width=[(x_max-x_min)/resolution]
Height=[(y_max-y_min)/resolution]
În formulele anterioare *[x]* reprezintă partea întreagă a lui *x* (cel mai mare număr întreg mai mic sau egal cu *x*).

O imagine de tip PGM “plain” are următorul format:

- prima linie: numărul magic specific formatului (“P2”)
- a doua linie: lățimea și înălțimea imaginii în pixeli (separate de spațiu)
- a treia linie: valoarea maximă de gri (în cazul de față va fi 255, adică NUM_COLORS - 1 din algoritmii prezentați în a doua secțiune)
- următoarele linii: valorile de gri ale pixelilor de pe fiecare linie din imagine, separate prin spații albe.

Pentru a deschide imagini PGM, puteți folosi diverse editoare de imagini (de exemplu, Gimp).
Atenție! Deoarece în cazul coordonatelor matematice, punctul (0,0) se află în partea de stânga-jos a axelor și axa OY este îndreptată în sus, iar pentru coordonatele ecran, punctul (0,0) se află în partea de stânga-sus și axa OY este îndreptată în jos, în momentul în care salvați datele în fișierul de ieșire, va trebui să scrieți coordonatele Y în ordine inversă, așa cum puteți vedea în imaginea de mai jos.
Recomandam totusi parcurgerea spatiului complex in ordine directa(de la X_min la X_max si de la Y_min la Y_max) astfel incat sa se evite posibile erori de precizie.



Testare

Temele se vor testa automat pe CLUSTER. Puteți găsi în secțiunea **Resurse Tema3** un set de teste publice (directorul ***in/*** din arhiva) și output-urile de referință (directorul ***out-ref/***) corespunzătoare. Pentru verificarea rezultatului vostru cu cele obținute de noi, puteţi folosi programul ***imgdiff*** care se află în arhivă (***bin/imgdiff*** - 64 biti și ***bin/imgdiff32*** - 32 biti). Deoarece se lucrează cu valori cu multe zecimale şi cu operaţii de radical şi ridicări la pătrat, pot apărea erori de

rotunjire în funcţie de compilatorul folosit. Din această cauză, există posibilitatea ca output-ul vostru să difere un pic de rezultatele noastre, însă ***imgdiff*** ia în considerare acest lucru când compară doua fişiere PGM (pe care le primește ca parametri în linia de comandă).

Notare

Tema se va trimite la adresa **http://vmchecker.cs.pub.ro** (login folosind ID-ul de cs.curs.pub.ro după care selectați APD din meniul de sus), într-o arhiva .zip care pe lângă fișierele sursă va trebui să conțină următoarele 2 fișiere:

- Makefile - cu directiva **build** care compilează tema voastră și generează un executabil numit **main**
- README în care să se descrie pe scurt implementarea temei

Atentie! Toate fișierele vor fi puse în rădăcina arhivei.

Atenție! NU contează numele arhivei.

Atenție! Testați tema pe cluster. Checker-ul ține cont și de timpii de rulare (tema trebuie să fie scalabilă - dacă se folosesc mai multe procese MPI timpul trebuie să fie mai mic).

Punctajul este divizat după cum urmează:
- **80p** - testarea automată pe VMCHECKER
- **20p** - claritatea codului și a explicațiilor.

Compilare și Rularea pe cluster

1. student@vm:~$ scp main.c mihai.carabas@fep.grid.pub.ro:
main.c 100% 179 0.2KB/s 00:00
2. student@vm:~$ ssh mihai.carabas@fep.grid.pub.ro
[mihai.carabas@fep-62-2 ~]$
3. *[mihai.carabas@fep-62-2 ~]$* ***module load libraries/openmpi-1.6-gcc-4.4.6***
4. [mihai.carabas@fep-62-2 ~]$ **mpicc main.c -o main**
5. [mihai.carabas@fep-62-2 ~]$ cat script.sh
#!/bin/bash
module load libraries/openmpi-1.6-gcc-4.4.6
mpirun ./main $1 $2
6. [mihai.carabas@fep-62-2 ~]$ **qsub -cwd -pe openmpi 4 -q ibm-quad.q script.sh fisier.in fisier.out**

- 4 - reprezinta numarul de procese MPI lansate
- script.sh - scriptul care se va rula pe cluster
- fisier.in - fisierul de intrare
- fisier.out - fisierul de iesire

7. [mihai.carabas@fep-62-2 ~]$ **qstat**
job-ID prior name user state submit/start at queue slots ja-task-ID

532370 0.00000 script.sh mihai.caraba qw 11/25/2013 18:49:19 4
8. Output-ul îl găsiți în fișierul script.sh.o532370 (este numele fișierului concatenat cu .o și ID-ul job-ului) după ce job-ul nu mai apare la comanda **qstat.**
[mihai.carabas@fep-62-2 ~]$ cat script.sh.o532370

Referințe

1. http://en.wikipedia.org/wiki/Mandelbrot_set
2. http://en.wikipedia.org/wiki/Julia_set
3. http://netpbm.sourceforge.net/doc/pgm.html