Univerzitet u Beogradu - Elektrotehnički fakultet

Multiprocesorki sistemi (13S114MUPS, 13E114MUPS)



Domaći zadatak 1 – OpenMP

Izveštaj o urađenom domaćem zadatku

|  |  |
| --- | --- |
| Predmetni asistent: | Studenti: |
| doc. dr Marko Mišić | Lazar Cvetković 2016/0127  Strahinja Stefanović 2016/0130 |

Beograd, mart 2020.

Sadržaj

[Sadržaj 2](#_Toc36550726)

[1. Problem 1 – Julia Set (ručna raspodela) 3](#_Toc36550727)

[1.1. Tekst problema 3](#_Toc36550728)

[1.2. Delovi koje treba paralelizovati 3](#_Toc36550729)

[1.2.1. Diskusija 3](#_Toc36550730)

[1.2.2. Način paralelizacije 3](#_Toc36550731)

[1.3. Rezultati 3](#_Toc36550732)

[1.3.1. Logovi izvršavanja 3](#_Toc36550733)

[1.3.2. Grafici ubrzanja 9](#_Toc36550734)

[1.3.3. Diskusija dobijenih rezultata 9](#_Toc36550735)

[2. Problem 2 – Julia Set (worksharing) 10](#_Toc36550736)

[2.1. Tekst problema 10](#_Toc36550737)

[2.2. Delovi koje treba paralelizovati 10](#_Toc36550738)

[2.2.1. Diskusija 10](#_Toc36550739)

[2.2.2. Način paralelizacije 10](#_Toc36550740)

[2.3. Rezultati 10](#_Toc36550741)

[2.3.1. Logovi izvršavanja 10](#_Toc36550742)

[2.3.2. Grafici ubrzanja 16](#_Toc36550743)

[2.3.3. Diskusija dobijenih rezultata 16](#_Toc36550744)

[3. Problem 3 – Julia Set (tasks) 17](#_Toc36550745)

[3.1. Tekst problema 17](#_Toc36550746)

[3.2. Delovi koje treba paralelizovati 17](#_Toc36550747)

[3.2.1. Diskusija 17](#_Toc36550748)

[3.2.2. Način paralelizacije 17](#_Toc36550749)

[3.3. Rezultati 17](#_Toc36550750)

[3.3.1. Logovi izvršavanja 17](#_Toc36550751)

[3.3.2. Grafici ubrzanja 23](#_Toc36550752)

[3.3.3. Diskusija dobijenih rezultata 23](#_Toc36550753)

[4. Problem 4 – Izoštravanje slike 24](#_Toc36550754)

[4.1. Tekst problema 24](#_Toc36550755)

[4.2. Delovi koje treba paralelizovati 24](#_Toc36550756)

[4.2.1. Diskusija 24](#_Toc36550757)

[4.2.2. Način paralelizacije 24](#_Toc36550758)

[4.3. Rezultati 24](#_Toc36550759)

[4.3.1. Logovi izvršavanja 24](#_Toc36550760)

[4.3.2. Grafici ubrzanja 28](#_Toc36550761)

[4.3.3. Diskusija dobijenih rezultata 28](#_Toc36550762)

[5. Problem 5 – MRI Gridding 29](#_Toc36550763)

[5.1. Tekst problema 29](#_Toc36550764)

[5.2. Delovi koje treba paralelizovati 29](#_Toc36550765)

[5.2.1. Diskusija 29](#_Toc36550766)

[5.2.2. Način paralelizacije 29](#_Toc36550767)

[5.3. Rezultati 29](#_Toc36550768)

[5.3.1. Logovi izvršavanja 29](#_Toc36550769)

[5.3.2. Grafici ubrzanja 32](#_Toc36550770)

[5.3.3. Diskusija dobijenih rezultata 32](#_Toc36550771)

1. Problem 1 – Julia Set (ručna raspodela)
   1. Tekst problema

Paralelizovati program koji formira sliku tačaka koje pripadaju Julia skupu tačaka (https://en.wikipedia.org/wiki/Julia\_set). Neka se posmatra skup tačaka *(x, y)* u na pravougaonom domenu *x, y* ∈ [-1,5, 1.5] i neka važi *z = x+yi*. Julia skup je skup tačaka za koji iteracija *z = z2 + c* ne divergira za određene zadate početne uslove. U zadatom programu početni uslov odgovara *c=- 0.8+0.156i*. Ukoliko u bilo kom trenutku važi *1000 < |z|,* smatra se da tačka *z* ne pripada Julia skupu. Program formira sliku u *Targa* (*.tga*) formatu koja se može otvoriti u nekom od namenskih pregledača slika. Program se nalazi u datoteci **julia.c** u arhivi koja je priložena uz ovaj dokument, dok se primeri izlaznih datoteka nalaze u direktorijumu **output**. Prilikom paralelizacije nije dozvoljeno koristiti direktive za podelu posla (*worksharing* direktive), već je iteracije petlje koja se paralelizuje potrebno raspodeliti ručno. Obratiti pažnju na ispravno deklarisanje svih promenljivih prilikom paralelizacije. Program testirati sa parametrima koji su dati u datoteci **run**. [1, N]

* 1. Delovi koje treba paralelizovati
     1. Diskusija

Moguće je paralelizovati samo glavnu funkciju koja pokreće izračunavanje svih tačaka skupa **julia\_set**, odnosno dvostruku for petlju unutar nje. Funkciju **julia** koja izračunava tačnu vrednost tačke nije moguće paralelizovati zato što postoji zavisnost između susednih iteracija petlje koju ona izvršava.

* + 1. Način paralelizacije

Funkcija **julia\_set** je paralelizovana sa #pragma omp parallel navođenjem koji podaci su privatni, a koji deljeni, a onda je unutar nje izvršena ručna raspodela posla između niti, tako da svaka nit dobije chunkSize deo posla.

* 1. Rezultati
     1. Logovi izvršavanja

Ovde su dati logovi izvršavanja za definisane test primere i različit broj niti.

Input values: 500 500 200

Number of threads: 1

Sequential execution time: 0.017251

Parallel execution time: 0.017059

Test PASSED

Input values: 500 500 500

Number of threads: 1

Sequential execution time: 0.021523

Parallel execution time: 0.021443

Test PASSED

Input values: 500 500 1000

Number of threads: 1

Sequential execution time: 0.022286

Parallel execution time: 0.022222

Test PASSED

Input values: 1000 1000 200

Number of threads: 1

Sequential execution time: 0.068673

Parallel execution time: 0.067723

Test PASSED

Input values: 1000 1000 500

Number of threads: 1

Sequential execution time: 0.085169

Parallel execution time: 0.084322

Test PASSED

Input values: 1000 1000 1000

Number of threads: 1

Sequential execution time: 0.088170

Parallel execution time: 0.088109

Test PASSED

Input values: 2000 1000 200

Number of threads: 1

Sequential execution time: 0.137578

Parallel execution time: 0.137771

Test PASSED

Input values: 2000 1000 500

Number of threads: 1

Sequential execution time: 0.172194

Parallel execution time: 0.170904

Test PASSED

Input values: 2000 1000 1000

Number of threads: 1

Sequential execution time: 0.176485

Parallel execution time: 0.177054

Test PASSED

Input values: 500 500 200

Number of threads: 2

Sequential execution time: 0.017250

Parallel execution time: 0.008783

Test PASSED

Input values: 500 500 500

Number of threads: 2

Sequential execution time: 0.021525

Parallel execution time: 0.010932

Test PASSED

Input values: 500 500 1000

Number of threads: 2

Sequential execution time: 0.022275

Parallel execution time: 0.011266

Test PASSED

Input values: 1000 1000 200

Number of threads: 2

Sequential execution time: 0.068740

Parallel execution time: 0.034488

Test PASSED

Input values: 1000 1000 500

Number of threads: 2

Sequential execution time: 0.085631

Parallel execution time: 0.043132

Test PASSED

Input values: 1000 1000 1000

Number of threads: 2

Sequential execution time: 0.088418

Parallel execution time: 0.044643

Test PASSED

Input values: 2000 1000 200

Number of threads: 2

Sequential execution time: 0.136695

Parallel execution time: 0.068806

Test PASSED

Input values: 2000 1000 500

Number of threads: 2

Sequential execution time: 0.172234

Parallel execution time: 0.086180

Test PASSED

Input values: 2000 1000 1000

Number of threads: 2

Sequential execution time: 0.177156

Parallel execution time: 0.088764

Test PASSED

Input values: 500 500 200

Number of threads: 4

Sequential execution time: 0.017288

Parallel execution time: 0.008797

Test PASSED

Input values: 500 500 500

Number of threads: 4

Sequential execution time: 0.021452

Parallel execution time: 0.010221

Test PASSED

Input values: 500 500 1000

Number of threads: 4

Sequential execution time: 0.022364

Parallel execution time: 0.010796

Test PASSED

Input values: 1000 1000 200

Number of threads: 4

Sequential execution time: 0.068671

Parallel execution time: 0.031837

Test PASSED

Input values: 1000 1000 500

Number of threads: 4

Sequential execution time: 0.085738

Parallel execution time: 0.040660

Test PASSED

Input values: 1000 1000 1000

Number of threads: 4

Sequential execution time: 0.088548

Parallel execution time: 0.042070

Test PASSED

Input values: 2000 1000 200

Number of threads: 4

Sequential execution time: 0.136851

Parallel execution time: 0.063490

Test PASSED

Input values: 2000 1000 500

Number of threads: 4

Sequential execution time: 0.171495

Parallel execution time: 0.080923

Test PASSED

Input values: 2000 1000 1000

Number of threads: 4

Sequential execution time: 0.177753

Parallel execution time: 0.083506

Test PASSED

Input values: 500 500 200

Number of threads: 8

Sequential execution time: 0.017256

Parallel execution time: 0.006628

Test PASSED

Input values: 500 500 500

Number of threads: 8

Sequential execution time: 0.021547

Parallel execution time: 0.008395

Test PASSED

Input values: 500 500 1000

Number of threads: 8

Sequential execution time: 0.022135

Parallel execution time: 0.008776

Test PASSED

Input values: 1000 1000 200

Number of threads: 8

Sequential execution time: 0.068531

Parallel execution time: 0.024677

Test PASSED

Input values: 1000 1000 500

Number of threads: 8

Sequential execution time: 0.085784

Parallel execution time: 0.031320

Test PASSED

Input values: 1000 1000 1000

Number of threads: 8

Sequential execution time: 0.088931

Parallel execution time: 0.032082

Test PASSED

Input values: 2000 1000 200

Number of threads: 8

Sequential execution time: 0.137369

Parallel execution time: 0.048342

Test PASSED

Input values: 2000 1000 500

Number of threads: 8

Sequential execution time: 0.171658

Parallel execution time: 0.061728

Test PASSED

Input values: 2000 1000 1000

Number of threads: 8

Sequential execution time: 0.177747

Parallel execution time: 0.064156

Test PASSED

Listing 1. Izlaz programa

* + 1. Grafici ubrzanja

U okviru ove sekcije su dati grafici ubrzanja u odnosu na sekvencijalnu implementaciju.

Slika 1. Grafik zavisnosti ubrzanja

* + 1. Diskusija dobijenih rezultata

Na osnovu dobijenih merenja može se zaključiti da je paralelizovana verzija brža, a primetno je i još veće povećanje brzine izvršavanja sa povećanjem broja niti.

1. Problem 2 – Julia Set (worksharing)
   1. Tekst problema

Prethodni program paralelizovati korišćenjem direktiva za podelu posla (*worksharing* direktive). Obratiti pažnju na raspodelu opterećenja po nitima i testirati program za različite načine raspoređivanja posla. Program testirati sa parametrima koji su dati u datoteci **run**. [1, N]

* 1. Delovi koje treba paralelizovati
     1. Diskusija

Moguće je paralelizovati samo glavnu funkciju koja pokreće izračunavanje svih tačaka skupa **julia\_set**, odnosno dvostruku for petlju unutar nje. Funkciju **julia** koja izračunava tačnu vrednost tačke nije moguće paralelizovati zato što postoji zavisnost između susednih iteracija petlje koju ona izvršava.

* + 1. Način paralelizacije

Korišćena je OpenMP **for** direktiva koja vrši paralelizaciju petlji uz direktivu **schedule**(static,1) koja uvodi cikličnu blokovsku raspodelu iteracija.

* 1. Rezultati
     1. Logovi izvršavanja

Ovde su dati logovi izvršavanja za definisane test primere i različit broj niti.

Input values: 500 500 200

Number of threads: 1

Sequential execution time: 0.017270

Parallel execution time: 0.017269

Test PASSED

Input values: 500 500 500

Number of threads: 1

Sequential execution time: 0.021640

Parallel execution time: 0.021550

Test PASSED

Input values: 500 500 1000

Number of threads: 1

Sequential execution time: 0.022361

Parallel execution time: 0.022359

Test PASSED

Input values: 1000 1000 200

Number of threads: 1

Sequential execution time: 0.068693

Parallel execution time: 0.068213

Test PASSED

Input values: 1000 1000 500

Number of threads: 1

Sequential execution time: 0.085807

Parallel execution time: 0.084797

Test PASSED

Input values: 1000 1000 1000

Number of threads: 1

Sequential execution time: 0.088607

Parallel execution time: 0.087926

Test PASSED

Input values: 2000 1000 200

Number of threads: 1

Sequential execution time: 0.137416

Parallel execution time: 0.137579

Test PASSED

Input values: 2000 1000 500

Number of threads: 1

Sequential execution time: 0.171184

Parallel execution time: 0.169200

Test PASSED

Input values: 2000 1000 1000

Number of threads: 1

Sequential execution time: 0.177115

Parallel execution time: 0.178276

Test PASSED

Input values: 500 500 200

Number of threads: 2

Sequential execution time: 0.017112

Parallel execution time: 0.008977

Test PASSED

Input values: 500 500 500

Number of threads: 2

Sequential execution time: 0.021384

Parallel execution time: 0.011066

Test PASSED

Input values: 500 500 1000

Number of threads: 2

Sequential execution time: 0.022206

Parallel execution time: 0.011497

Test PASSED

Input values: 1000 1000 200

Number of threads: 2

Sequential execution time: 0.068752

Parallel execution time: 0.035151

Test PASSED

Input values: 1000 1000 500

Number of threads: 2

Sequential execution time: 0.085953

Parallel execution time: 0.043635

Test PASSED

Input values: 1000 1000 1000

Number of threads: 2

Sequential execution time: 0.088555

Parallel execution time: 0.045384

Test PASSED

Input values: 2000 1000 200

Number of threads: 2

Sequential execution time: 0.137427

Parallel execution time: 0.069758

Test PASSED

Input values: 2000 1000 500

Number of threads: 2

Sequential execution time: 0.171686

Parallel execution time: 0.087674

Test PASSED

Input values: 2000 1000 1000

Number of threads: 2

Sequential execution time: 0.177823

Parallel execution time: 0.090221

Test PASSED

Input values: 500 500 200

Number of threads: 4

Sequential execution time: 0.017264

Parallel execution time: 0.004976

Test PASSED

Input values: 500 500 500

Number of threads: 4

Sequential execution time: 0.021499

Parallel execution time: 0.006136

Test PASSED

Input values: 500 500 1000

Number of threads: 4

Sequential execution time: 0.022238

Parallel execution time: 0.006174

Test PASSED

Input values: 1000 1000 200

Number of threads: 4

Sequential execution time: 0.068804

Parallel execution time: 0.019021

Test PASSED

Input values: 1000 1000 500

Number of threads: 4

Sequential execution time: 0.086037

Parallel execution time: 0.022208

Test PASSED

Input values: 1000 1000 1000

Number of threads: 4

Sequential execution time: 0.088621

Parallel execution time: 0.023573

Test PASSED

Input values: 2000 1000 200

Number of threads: 4

Sequential execution time: 0.137882

Parallel execution time: 0.038351

Test PASSED

Input values: 2000 1000 500

Number of threads: 4

Sequential execution time: 0.172453

Parallel execution time: 0.044747

Test PASSED

Input values: 2000 1000 1000

Number of threads: 4

Sequential execution time: 0.178053

Parallel execution time: 0.046701

Test PASSED

Input values: 500 500 200

Number of threads: 8

Sequential execution time: 0.017224

Parallel execution time: 0.002992

Test PASSED

Input values: 500 500 500

Number of threads: 8

Sequential execution time: 0.021380

Parallel execution time: 0.003581

Test PASSED

Input values: 500 500 1000

Number of threads: 8

Sequential execution time: 0.022186

Parallel execution time: 0.003644

Test PASSED

Input values: 1000 1000 200

Number of threads: 8

Sequential execution time: 0.068486

Parallel execution time: 0.010868

Test PASSED

Input values: 1000 1000 500

Number of threads: 8

Sequential execution time: 0.086100

Parallel execution time: 0.013100

Test PASSED

Input values: 1000 1000 1000

Number of threads: 8

Sequential execution time: 0.089045

Parallel execution time: 0.013537

Test PASSED

Input values: 2000 1000 200

Number of threads: 8

Sequential execution time: 0.137668

Parallel execution time: 0.021001

Test PASSED

Input values: 2000 1000 500

Number of threads: 8

Sequential execution time: 0.172592

Parallel execution time: 0.025566

Test PASSED

Input values: 2000 1000 1000

Number of threads: 8

Sequential execution time: 0.178679

Parallel execution time: 0.027804

Test PASSED

Listing 2. Izlaz programa

* + 1. Grafici ubrzanja

Slika 2. Grafik zavisnosti ubrzanja

* + 1. Diskusija dobijenih rezultata

Na osnovu dobijenih merenja može se zaključiti da je paralelizovana verzija brža, a primetno je i još veće povećanje brzine izvršavanja sa povećanjem broja niti.

Takođe, ovakav automatizovani način paralelizacije je i do nekoliko puta brži od ručne raspodele posla između niti. To poboljšanje se još više primećuje što je broj niti veći.

1. Problem 3 – Julia Set (tasks)
   1. Tekst problema

Rešiti prethodni problem korišćenjem koncepta poslova (*tasks*). Obratiti pažnju na eventualnu potrebu za sinhronizacijom i testirati program za različite granularnosti poslova. Program testirati sa parametrima koji su dati u datoteci **run**. [1, N]

* 1. Delovi koje treba paralelizovati
     1. Diskusija

Moguće je paralelizovati samo glavnu funkciju koja pokreće izračunavanje svih tačaka skupa **julia\_set**, odnosno dvostruku for petlju unutar nje. Funkciju **julia** koja izračunava tačnu vrednost tačke nije moguće paralelizovati zato što postoji zavisnost između susednih iteracija petlje koju ona izvršava.

* + 1. Način paralelizacije

Funkcija julia\_set je paralelizovana sa #pragma omp parallel navođenjem koji podaci su privatni, a koji deljeni, a onda je unutar nje napravljena jedna nit koja vrši delegiranje poslova. Cela parallel direktiva se blokira čekajući da se prethodno napravljeni poslovi završe. Poslovi obavljaju unutrašnju for petlju.

* 1. Rezultati
     1. Logovi izvršavanja

Ovde su dati logovi izvršavanja za definisane test primere i različit broj niti.

Input values: 500 500 200

Number of threads: 1

Sequential execution time: 0.017220

Parallel execution time: 0.017280

Test PASSED

Input values: 500 500 500

Number of threads: 1

Sequential execution time: 0.021443

Parallel execution time: 0.021572

Test PASSED

Input values: 500 500 1000

Number of threads: 1

Sequential execution time: 0.022684

Parallel execution time: 0.022276

Test PASSED

Input values: 1000 1000 200

Number of threads: 1

Sequential execution time: 0.068353

Parallel execution time: 0.068004

Test PASSED

Input values: 1000 1000 500

Number of threads: 1

Sequential execution time: 0.085278

Parallel execution time: 0.084532

Test PASSED

Input values: 1000 1000 1000

Number of threads: 1

Sequential execution time: 0.088243

Parallel execution time: 0.087282

Test PASSED

Input values: 2000 1000 200

Number of threads: 1

Sequential execution time: 0.137413

Parallel execution time: 0.137620

Test PASSED

Input values: 2000 1000 500

Number of threads: 1

Sequential execution time: 0.171866

Parallel execution time: 0.170497

Test PASSED

Input values: 2000 1000 1000

Number of threads: 1

Sequential execution time: 0.176633

Parallel execution time: 0.177790

Test PASSED

Input values: 500 500 200

Number of threads: 2

Sequential execution time: 0.017278

Parallel execution time: 0.008932

Test PASSED

Input values: 500 500 500

Number of threads: 2

Sequential execution time: 0.021576

Parallel execution time: 0.011086

Test PASSED

Input values: 500 500 1000

Number of threads: 2

Sequential execution time: 0.022260

Parallel execution time: 0.011423

Test PASSED

Input values: 1000 1000 200

Number of threads: 2

Sequential execution time: 0.068546

Parallel execution time: 0.034785

Test PASSED

Input values: 1000 1000 500

Number of threads: 2

Sequential execution time: 0.085709

Parallel execution time: 0.043494

Test PASSED

Input values: 1000 1000 1000

Number of threads: 2

Sequential execution time: 0.088994

Parallel execution time: 0.044710

Test PASSED

Input values: 2000 1000 200

Number of threads: 2

Sequential execution time: 0.137721

Parallel execution time: 0.069701

Test PASSED

Input values: 2000 1000 500

Number of threads: 2

Sequential execution time: 0.172611

Parallel execution time: 0.086808

Test PASSED

Input values: 2000 1000 1000

Number of threads: 2

Sequential execution time: 0.177100

Parallel execution time: 0.089189

Test PASSED

Input values: 500 500 200

Number of threads: 4

Sequential execution time: 0.017345

Parallel execution time: 0.005039

Test PASSED

Input values: 500 500 500

Number of threads: 4

Sequential execution time: 0.022065

Parallel execution time: 0.006174

Test PASSED

Input values: 500 500 1000

Number of threads: 4

Sequential execution time: 0.022626

Parallel execution time: 0.006286

Test PASSED

Input values: 1000 1000 200

Number of threads: 4

Sequential execution time: 0.068589

Parallel execution time: 0.018111

Test PASSED

Input values: 1000 1000 500

Number of threads: 4

Sequential execution time: 0.085961

Parallel execution time: 0.022791

Test PASSED

Input values: 1000 1000 1000

Number of threads: 4

Sequential execution time: 0.089677

Parallel execution time: 0.023441

Test PASSED

Input values: 2000 1000 200

Number of threads: 4

Sequential execution time: 0.137393

Parallel execution time: 0.035900

Test PASSED

Input values: 2000 1000 500

Number of threads: 4

Sequential execution time: 0.172152

Parallel execution time: 0.046941

Test PASSED

Input values: 2000 1000 1000

Number of threads: 4

Sequential execution time: 0.178054

Parallel execution time: 0.048663

Test PASSED

Input values: 500 500 200

Number of threads: 8

Sequential execution time: 0.017195

Parallel execution time: 0.004247

Test PASSED

Input values: 500 500 500

Number of threads: 8

Sequential execution time: 0.021465

Parallel execution time: 0.004396

Test PASSED

Input values: 500 500 1000

Number of threads: 8

Sequential execution time: 0.022187

Parallel execution time: 0.004596

Test PASSED

Input values: 1000 1000 200

Number of threads: 8

Sequential execution time: 0.068411

Parallel execution time: 0.011484

Test PASSED

Input values: 1000 1000 500

Number of threads: 8

Sequential execution time: 0.086231

Parallel execution time: 0.013654

Test PASSED

Input values: 1000 1000 1000

Number of threads: 8

Sequential execution time: 0.088661

Parallel execution time: 0.014044

Test PASSED

Input values: 2000 1000 200

Number of threads: 8

Sequential execution time: 0.137968

Parallel execution time: 0.021451

Test PASSED

Input values: 2000 1000 500

Number of threads: 8

Sequential execution time: 0.172140

Parallel execution time: 0.026147

Test PASSED

Input values: 2000 1000 1000

Number of threads: 8

Sequential execution time: 0.177866

Parallel execution time: 0.026636

Test PASSED

Listing 3. Izlaz programa

* + 1. Grafici ubrzanja

U okviru ove sekcije su dati grafici ubrzanja u odnosu na sekvencijalnu implementaciju.

Slika 3. Grafik zavisnosti ubrzanja

* + 1. Diskusija dobijenih rezultata

Očigledno je da paralelizovana verzija sa 4 niti do određene složenosti ulaznih podataka daje najbolje rezultate, dok za ulazne podatke najveće veličine primat preuzima paralelizacija kojoj su na raspolaganju 8 niti. Sekvencijalna verzija i ona paralelna sa jednom niti su gotovo istih performansi. Dakle, zaključak je da do neke granice najbolje performanse su sa 4 niti, dok nakon nje to postaje sa 8 niti.

1. Problem 4 – Izoštravanje slike

U okviru ovog poglavlja je dat kratak izveštaj u vezi rešenja zadatog problema 4.

* 1. Tekst problema

Paralelizovati program koji izoštrava zadatu sliku u *Portable Graymap Format* (PGM) formatu. PGM format se može otvoriti u nekom od namenskih pregledača slika ili *online* na adresi http://paulcuth.me.uk/netpbm-viewer/. Program se nalazi u direktorijumu **sharpen** u arhivi koja je priložena uz ovaj dokument. Program se sastoji od više datoteka, od kojih su od interesa datoteke **sharpen**.**c**, **dosharpen**.**c** i **filter.c**. Obratiti pažnju na raspodelu opterećenja po nitima i testirati program za različite načine raspoređivanja posla. Program testirati sa parametrima koji su dati u datoteci **run**. [1, N]

* 1. Delovi koje treba paralelizovati
     1. Diskusija

Izvršena je paralelizacija četiri ugnježdene petlje koje izračunavaju konvoluciju, kao i sledeće dvostruke petlje koja taj dobijeni rezultat koristi za izoštravanje slike.

* + 1. Način paralelizacije

Korišćena je direktiva #pragma omp parallel for uz korišćenje odredbe **collapse(2)** kako bi se sažimanjem savršeno ugneždenih pravougaonih petlji postiglo još veće ubrzanje.

* 1. Rezultati
     1. Logovi izvršavanja

Ovde su dati logovi izvršavanja za definisane test primere i različit broj niti.

Input file: data\_dz1z4/balloons\_noisy.pgm

Number of threads: 1

Sequential execution time: 3.160817

Parallel execution time: 0.449893

Test PASSED

Input file: data\_dz1z4/bone\_scint.pgm

Number of threads: 1

Sequential execution time: 21.448036

Parallel execution time: 2.940236

Test PASSED

Input file: data\_dz1z4/fuzzy.pgm

Number of threads: 1

Sequential execution time: 4.436713

Parallel execution time: 0.615860

Test PASSED

Input file: data\_dz1z4/lena512.pgm

Number of threads: 1

Sequential execution time: 2.687834

Parallel execution time: 0.347967

Test PASSED

Input file: data\_dz1z4/man.pgm

Number of threads: 1

Sequential execution time: 10.718712

Parallel execution time: 1.508994

Test PASSED

Input file: data\_dz1z4/Rainier\_blur.pgm

Number of threads: 1

Sequential execution time: 21.040416

Parallel execution time: 2.830196

Test PASSED

Input file: data\_dz1z4/balloons\_noisy.pgm

Number of threads: 2

Sequential execution time: 3.205606

Parallel execution time: 0.232764

Test PASSED

Input file: data\_dz1z4/bone\_scint.pgm

Number of threads: 2

Sequential execution time: 21.326204

Parallel execution time: 2.072310

Test PASSED

Input file: data\_dz1z4/fuzzy.pgm

Number of threads: 2

Sequential execution time: 4.429806

Parallel execution time: 0.360594

Test PASSED

Input file: data\_dz1z4/lena512.pgm

Number of threads: 2

Sequential execution time: 2.685684

Parallel execution time: 0.197396

Test PASSED

Input file: data\_dz1z4/man.pgm

Number of threads: 2

Sequential execution time: 10.623472

Parallel execution time: 0.858018

Test PASSED

Input file: data\_dz1z4/Rainier\_blur.pgm

Number of threads: 2

Sequential execution time: 20.966091

Parallel execution time: 1.674089

Test PASSED

Input file: data\_dz1z4/balloons\_noisy.pgm

Number of threads: 4

Sequential execution time: 3.141195

Parallel execution time: 0.137025

Test PASSED

Input file: data\_dz1z4/bone\_scint.pgm

Number of threads: 4

Sequential execution time: 21.660731

Parallel execution time: 1.076808

Test PASSED

Input file: data\_dz1z4/fuzzy.pgm

Number of threads: 4

Sequential execution time: 4.413546

Parallel execution time: 0.200995

Test PASSED

Input file: data\_dz1z4/lena512.pgm

Number of threads: 4

Sequential execution time: 2.671352

Parallel execution time: 0.121599

Test PASSED

Input file: data\_dz1z4/man.pgm

Number of threads: 4

Sequential execution time: 10.646887

Parallel execution time: 0.522341

Test PASSED

Input file: data\_dz1z4/Rainier\_blur.pgm

Number of threads: 4

Sequential execution time: 21.022792

Parallel execution time: 1.080296

Test PASSED

Input file: data\_dz1z4/balloons\_noisy.pgm

Number of threads: 8

Sequential execution time: 3.160479

Parallel execution time: 0.141472

Test PASSED

Input file: data\_dz1z4/bone\_scint.pgm

Number of threads: 8

Sequential execution time: 21.295840

Parallel execution time: 1.115918

Test PASSED

Input file: data\_dz1z4/fuzzy.pgm

Number of threads: 8

Sequential execution time: 4.447348

Parallel execution time: 0.198538

Test PASSED

Input file: data\_dz1z4/lena512.pgm

Number of threads: 8

Sequential execution time: 2.667963

Parallel execution time: 0.121737

Test PASSED

Input file: data\_dz1z4/man.pgm

Number of threads: 8

Sequential execution time: 10.609205

Parallel execution time: 0.541231

Test PASSED

Input file: data\_dz1z4/Rainier\_blur.pgm

Number of threads: 8

Sequential execution time: 20.962118

Parallel execution time: 1.017262

Test PASSED

Listing 4. Ispis programa

* + 1. Grafici ubrzanja

U okviru ove sekcije su dati grafici ubrzanja u odnosu na sekvencijalnu implementaciju.

Slika 4. Grafik zavisnosti ubrzanja

* + 1. Diskusija dobijenih rezultata

Primetno je značajno ubrzanje, ali ono je u mnogome i posledica dodavanja funkcije **makeFilterMatrix** koja postupkom memoizacije samo jednom računa filtersku matricu, za razliku od sekvencijalnog izvršavanja koja u svakoj iteraciji to radi. Takođe, može se primetiti da ne postoji neki veliki dobit u performansama ukoliko se broj niti poveća sa 4 na 8.

1. Problem 5 – MRI Gridding

U okviru ovog poglavlja je dat kratak izveštaj u vezi rešenja zadatog problema 5.

* 1. Tekst problema

Paralelizovati program koji vrši mapiranje neuniformnih podataka u 3D prostoru na regularnu mrežu u 3D prostoru. Svaka tačka iz neuniformnog 3D prostora doprinosi susednim tačkama u regularnoj mreži u skladu sa *Kaiser-Bessel* funkcijom za određivanje rastojanja. Program se nalazi u direktorijumu **mri-gridding** u arhivi koja je priložena uz ovaj dokument. Program se sastoji od više datoteka, od kojih su od interesa datoteke **main**.**c** i **CPU\_kernels.c**. Analizirati dati kod i obratiti pažnju na način generisanja vrednosti tačaka u regularnoj mreži. Ukoliko je potrebno međusobno isključenje prilikom paralelizacije programa, koristiti dostupne OpenMP konstrukte. Obratiti pažnju na efikasnost međusobnog isključenja niti i po potrebi ga svesti na što je moguće manju meru uvođenjem pomoćnih struktura podataka. Ulazni test primeri se nalaze u direktorijumu **data**. Verifikaciju paralelizovanog rešenja vršiti nad nizovima **gridData** i **sampleDensity** iz glavnog programa. Način pokretanja programa se nalazi u datoteci **run**. [1, N]

* 1. Delovi koje treba paralelizovati
     1. Diskusija

Paralelizovana je funkcija **gridding\_Gold** koju ima i najviše smisla paralelizovati pošto su u njoj izvršava sav potreban račun.

* + 1. Način paralelizacije

Korišćena je direktiva #pragma omp parallel for uz odredbu **schedule**(dynamic, 5000) koja vrši raspodelu posla u pakete veličine 5000, a niti te pakete uzimaju u FIFO redosledu. Deljeni podaci koje je potrebno zaštiti sinhorinizacijonim primitivama su **gridData** i **sampleDensity** i tu svrhu korišćena je direktiva #pragma omp atomic koja daje najbolje performanse u poređenju sa bravama i kritičnim sekcijama.

* 1. Rezultati
     1. Logovi izvršavanja

Ovde su dati logovi izvršavanja za definisane test primere i različit broj niti.

Reading parameters

Number of samples = 2655910

Grid Size = 256x256x256

Input Matrix Size = 60x60x60

Recon Matrix Size = 60x60x60

Kernel Width = 5.000000

KMax = 150.00 150.00 150.00

Oversampling = 5.000000

GPU Binsize = 32

Use LUT = Yes

Reading input data from files

Generating Look-Up Table

Number of threds: 1

Sequential execution time: 1.559372

Parallel execution time: 4.078526

TEST PASSED - gridData

TEST PASSED - sampleDensity

Reading parameters

Number of samples = 2655910

Grid Size = 256x256x256

Input Matrix Size = 60x60x60

Recon Matrix Size = 60x60x60

Kernel Width = 5.000000

KMax = 150.00 150.00 150.00

Oversampling = 5.000000

GPU Binsize = 32

Use LUT = Yes

Reading input data from files

Generating Look-Up Table

Number of threds: 2

Sequential execution time: 1.559289

Parallel execution time: 2.154071

TEST PASSED - gridData

TEST PASSED - sampleDensity

Reading parameters

Number of samples = 2655910

Grid Size = 256x256x256

Input Matrix Size = 60x60x60

Recon Matrix Size = 60x60x60

Kernel Width = 5.000000

KMax = 150.00 150.00 150.00

Oversampling = 5.000000

GPU Binsize = 32

Use LUT = Yes

Reading input data from files

Generating Look-Up Table

Number of threds: 4

Sequential execution time: 1.563377

Parallel execution time: 1.372940

TEST PASSED - gridData

TEST PASSED - sampleDensity

Reading parameters

Number of samples = 2655910

Grid Size = 256x256x256

Input Matrix Size = 60x60x60

Recon Matrix Size = 60x60x60

Kernel Width = 5.000000

KMax = 150.00 150.00 150.00

Oversampling = 5.000000

GPU Binsize = 32

Use LUT = Yes

Reading input data from files

Generating Look-Up Table

Number of threds: 8

Sequential execution time: 1.559892

Parallel execution time: 1.057231

TEST PASSED - gridData

TEST PASSED - sampleDensity

Listing 5. Ispis programa

* + 1. Grafici ubrzanja

U okviru ove sekcije su dati grafici ubrzanja u odnosu na sekvencijalnu implementaciju.

Slika 5. Grafik zavisnosti ubrzanja

* + 1. Diskusija dobijenih rezultata

Može se primetiti da su režijski troškovi koje uvodi OpenMP u slučajevima kada se koristi jedna ili dve niti veći od dobiti koje uvodi paralelizam. Razlog tome je najverovatnije česta sinhronizacija na deljenim promenljivama.