UNIVERZITET U BEOGRADU - ELEKTROTEHNIČKI FAKULTET

MULTIPROCESORKI SISTEMI (13S114MUPS, 13E114MUPS)


# DOMAĆI ZADATAK 2 – MPI

Izveštaj o urađenom domaćem zadatku

Predmetni asistent:

doc. dr Marko Mišić

Studenti:

Ema Pajić 2016/0017

Nikola Aleksić 2016/0022

Beograd, april 2020.

# SADRŽAJ

# 1. PROBLEM 1 – JULIA SET

## 1.1. 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. Proces sa rangom 0 treba da učita ulazne podatke, raspodeli posao ostalim procesima, na kraju prikupi dobijene rezultate i ravnopravno učestvuje u obradi. Za razmenu podataka, koristiti rutine za kolektivnu komunikaciju. Program testirati sa parametrima koji su dati u datoteci run. [1, N]

## 1.2. Delovi koje treba paralelizovati

### *1.2.1. Diskusija*

U funkciji julia_set uočene su 2 ugnježdene for petlje koje je moguće paralelizovati. Postoji i for u funkciji julia, ali njega ne vredi paralelizovati jer bi granularnost bila prevelika.

### *1.2.2. Način paralelizacije*

Kako u toku rada ovog zadatka treba koristiti kolektivnu komunikaciju. Korišćena je operacija gather. Paralelizujemo određivanjem promenljivih chunk, start i end koje će svakoj niti na osnovu njenog ranka dodeliti neki deo for petlje. Pošto treba paziti na balansiranost opterećenja niti, a tačke ne konvergiraju istom brzinom (neke i divergiraju), ako nitima dodelimo samo geometrijski bliske tačke može da se desi da je neka nit mnogo opterećenija od neke druge. Iz tog razloga pravimo po 30 razdvojenih blokova za svaku nit (koji idu kružno) i tome služe promenljive offset i cyclicChunk. Pokušali smo da napravimo da sve radi pomoću jednog gathera, ali nismo uspeli.

## 1.3. Rezultati

Vremena sekvencijalnog i paralelnog izvrsavanja u tabeli su srednje vrednosti 3 pokretanja programa sa zadatim parametrima i navedeni su u formatu (vreme sekvencijalnog izvršavanja) / (vreme paralelnog izvršavanja) = ubrzanje.

| broj niti / ulazi (h, w, cnt) | 1 | 2 | 4 |
|---|---|---|---|
| 500 500 200 | 0.038/0.43 = 0.88 | 0.038/0.023 = 1.652 | 0.038/0.014 = 2.714 |
| 500 500 500 | 0.048/0.054 = 0.88 | 0,047/0.029 = 1.620 | 0.047/0.015 = 3.133 |
| 500 500 1000 | 0.048/0.053 = 0.906 | 0.048/0.030 = 1.600 | 0.048/0.016 = 3.000 |
| 1000 1000 200 | 0.151/0.155 = 0.97 | 0.153/0.085 = 1.800 | 0.151/0.046 = 3.282 |
| 1000 1000 500 | 0.192/0.189 = 0.984 | 0.189/0.103 = 1.835 | 0.187/0.058 = 3.224 |
| 1000 1000 1000 | 0.193/0.20 = 0.965 | 0.193/0.105 = 1.838 | 0.195/0.060 = 3.217 |
| 2000 1000 200 | 0.303/0.307 = 0.987 | 0.303/0.168 = 1.804 | 0.303/0.085 = 3.564 |
| 2000 1000 500 | 0.377/0.380 = 0.992 | 0.376/0.204 = 1.843 | 0.374/0.112 = 3.339 |
| 2000 1000 1000 | 0.387/0.391 = 0.990 | 0.387/0.212 = 1.850 | 0.387/0.115 = 3.365 |
| **Srednje ubrzanje** | **0.950** | **1.760** | **3.204** |

### *1.3.1. Logovi izvršavanja*

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

```
JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.044
Sequential time: 0.038
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_500_500_200.tga'

JULIA Set
```

```
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.050
Sequential time: 0.047
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_500_500_500.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.051
Sequential time: 0.049
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_500_500_1000.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.158
Sequential time: 0.151
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_1000_1000_200.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.191
Sequential time: 0.186
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_1000_1000_500.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.200
Sequential time: 0.193
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_1000_1000_1000.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.308
Sequential time: 0.302
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_2000_1000_200.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.381
Sequential time: 0.383
Test PASSED
```

```
TGA_WRITE:
  Graphics data saved as 'output/dz_julia_2000_1000_500.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.393
Sequential time: 0.387
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_2000_1000_1000.tga'
```

**Listing 1. Poređenje sekvencijalnog izvršavanja i izvršavanja sa 1 niti Julia**

```
JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.023
Sequential time: 0.038
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_500_500_200.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.029
Sequential time: 0.047
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_500_500_500.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.029
Sequential time: 0.048
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_500_500_1000.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.082
Sequential time: 0.159
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_1000_1000_200.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.102
Sequential time: 0.189
Test PASSED

TGA_WRITE:
```

```
  Graphics data saved as 'output/dz_julia_1000_1000_500.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.105
Sequential time: 0.193
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_1000_1000_1000.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.170
Sequential time: 0.311
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_2000_1000_200.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.205
Sequential time: 0.374
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_2000_1000_500.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.213
Sequential time: 0.387
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_2000_1000_1000.tga'
```

**Listing 2. Poređenje sekvencijalnog izvršavanja i izvršavanja sa 2 niti Julia**

```
JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.010
Sequential time: 0.038
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_500_500_200.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.015
Sequential time: 0.047
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_500_500_500.tga'
```

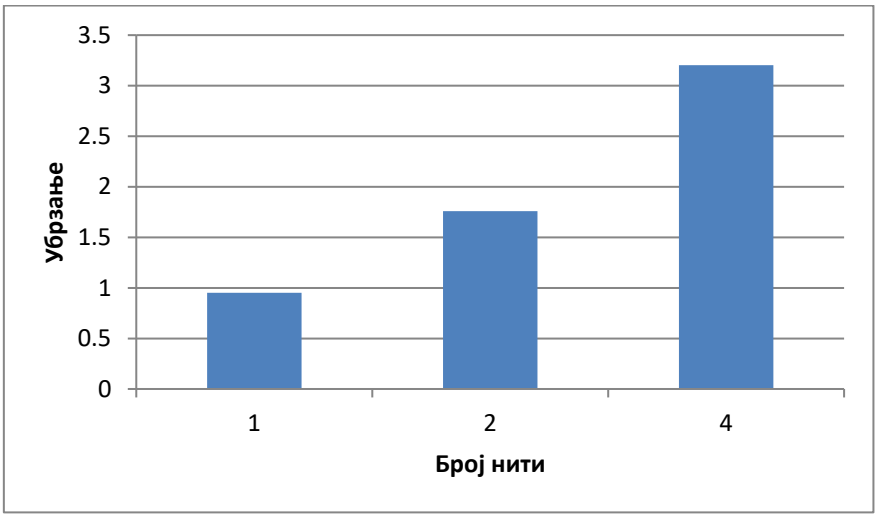
```
JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.016
Sequential time: 0.048
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_500_500_1000.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.048
Sequential time: 0.151
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_1000_1000_200.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.063
Sequential time: 0.186
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_1000_1000_500.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.060
Sequential time: 0.193
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_1000_1000_1000.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.087
Sequential time: 0.305
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_2000_1000_200.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.102
Sequential time: 0.374
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_2000_1000_500.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.109
Sequential time: 0.387
```

```
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_2000_1000_1000.tga'
```

**Listing 3. Poređenje sekvencijalnog izvršavanja i izvršavanja sa 4 niti Julia**

### *1.3.2. Grafici ubrzanja*

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




**Slika 1. Grafik zavisnosti ubrzanja algoritma bez worksharing direktiva od broja niti**

### *1.3.3. Diskusija dobijenih rezultata*

Paralelizacija je ubrzala izvršavanje za 2 i 4 niti, a pri radu sa 1 niti nije znacajno usporila. Problem je embarrassingly parallel i svaka nit dobija da računa pripadnost Julia setu za deo tačaka, jedino je potrebno paziti na opterećenje po nitima. Pošto tačke ne konvergiraju istom brzinom (a neke i divergiraju), ako nitima dodelimo samo geometrijski bliske tačke može da se desi da je neka nit mnogo opterećenija od neke druge. Iz tog razloga pravimo po 30 razdvojenih blokova za svaku nit (koji idu kružno). Najveće ubrzanje se dostiglo sa 8 niti gde primećujemo da je za veće dimenzije problema ubrzanje značajno u odnosu na izvršavanje sa 4 niti. Što se tiče manjih dimenzija problema (500x500), povećavanje broja niti sa 4 na 8 ne doprinosi daljem ubrzanju.

# 2. PROBLEM 2 – JULIA SET

## 2.1. Tekst problema

Prethodni program paralelizovati korišćenjem manager - worker modela. Proces gospodar (master) treba da učita neophodne podatke, generiše poslove, deli posao ostalim procesima i ispiše na kraju dobijeni rezultat. U svakom koraku obrade, proces gospodar šalje procesu radniku na obradu jednu jedinicu posla čiji veličinu treba pažljivo odabrati. Proces radnik prima podatke, vrši obradu, vraća rezultat, signalizira gospodaru kada je spreman da primi sledeći posao i ponavlja opisani postupak dok ne dobije signal da prekine sa radom. Veličinu jedne jedinice posla prilagoditi karakteristikama programa. Ukoliko je moguće, koristiti rutine za neblokirajuću komunikaciju za razmenu poruka. Program se nalazi u datoteci julia.c u arhivi koja je priložena uz ovaj dokument, dok se primeri izlaznih datoteka nalaze u direktorijumu output. Program testirati sa parametrima koji su dati u datoteci run. [1, N]

## 2.2. Delovi koje treba paralelizovati

### *2.2.1. Diskusija*

U funkciji julia_set uočene su 2 ugnježdene for petlje koje je moguće paralelizovati. Postoji i for u funkciji julia, ali njega ne vredi paralelizovati jer bi granularnost bila prevelika.

### *2.2.2. Način paralelizacije*

U tekstu zadatka rečeno je da se paralelizacija izvrši korišćenjem manager-worker modela. Proces sa rankom 0 je manager i on učitava podatke i šalje ih workerima. Jedan task smo uzeli da nam je jedan red. Na početku pošalje po task svakom slobodnom workeru, i ako ih ima više nego što je potrebno pošalje poruku delu koji je višak da se ugasi. Workeri primaju podatke, ako su dobili poruku da se ugase, gase se, u suprotnom rade potreban posao nad podacima koje su primili i nakon što završe, rezultate šalju manageru. Pri primanju poruke da je neki worker završio sa obradom, manager prima rezultat od njega i šalje mu podatke za dalju obradu, ukoliko ih ima. Na kraju manager čuva rezultat u sliku. Sva komunikacija se vrši porukama između samo managera i odgovarajućeg workera. Svaka send operacija je neblokirjuća, jer nam za njih nije bitno kada će tačno stići, međutim recv operacije jesu blokirajuće jer recv radimo u trenutku kada je neophodno da se obrade podaci.

## 2.3. Rezultati

Vremena sekvencijalnog i paralelnog izvrsavanja u tabeli su srednje vrednosti 3 pokretanja programa sa zadatim parametrima i navedeni su u formatu (vreme sekvencijalnog izvršavanja) / (vreme paralelnog izvršavanja) = ubrzanje.

| broj niti / ulazi (h, w, cnt) | 2 | 4 |
|---|---|---|
| 500 500 200 | 0.038/0.041 = 0.927 | 0.038/0.015 = 2.533 |
| 500 500 500 | 0.047/0.051 = 0.922 | 0.047/0.018 = 2.611 |
| 500 500 1000 | 0.048/0.052 = 0.923 | 0.048/0.020 = 2.400 |
| 1000 1000 200 | 0.151/0.156 = 0.968 | 0.151/0.053 = 2.849 |
| 1000 1000 500 | 0.186/0.191 = 0.974 | 0.186/0.066 = 2.818 |
| 1000 1000 1000 | 0.193/0.196 = 0.985 | 0.193/0.071 = 2.718 |
| 2000 1000 200 | 0.302/0.309 = 0.977 | 0.303/0.109 = 2.779 |
| 2000 1000 500 | 0.376/0.379 = 0.992 | 0.374/0.134 = 2.791 |
| 2000 1000 1000 | 0.391/0.393 = 0.995 | 0.388/0.136 = 2.853 |
| **Srednje ubrzanje** | **0.963** | **2.710** |

### *2.3.1. Logovi izvršavanja*

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

```
JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.043
Sequential time: 0.038
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_500_500_200.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
```

```
Parallel time: 0.051
Sequential time: 0.047
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_500_500_500.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.056
Sequential time: 0.050
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_500_500_1000.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.167
Sequential time: 0.165
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_1000_1000_200.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.214
Sequential time: 0.188
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_1000_1000_500.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.213
Sequential time: 0.195
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_1000_1000_1000.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.335
Sequential time: 0.315
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_2000_1000_200.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.378
Sequential time: 0.375
Test PASSED

TGA_WRITE:
```

```
  Graphics data saved as 'output/dz_julia_2000_1000_500.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.422
Sequential time: 0.408
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_2000_1000_1000.tga'
```

**Listing 4. Poređenje sekvencijalnog izvršavanja i izvršavanja sa 2 niti Julia2**

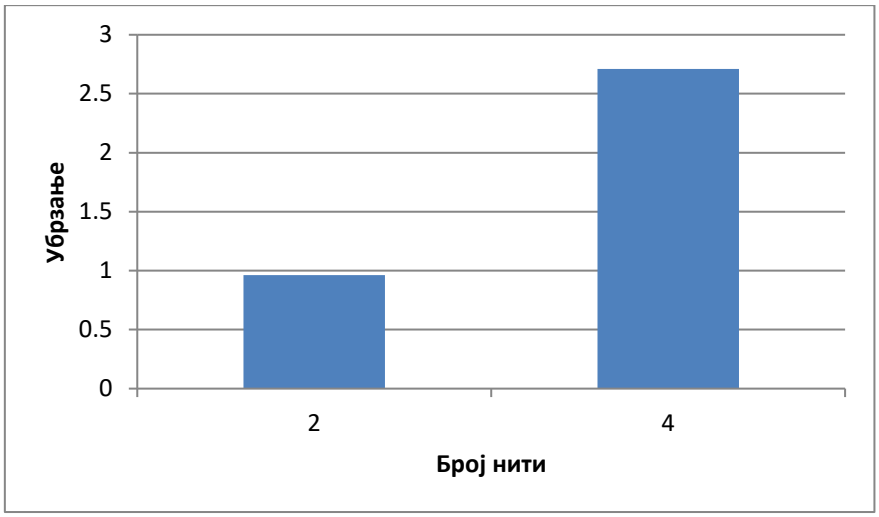
```
JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.016
Sequential time: 0.038
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_500_500_200.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.018
Sequential time: 0.047
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_500_500_500.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.019
Sequential time: 0.048
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_500_500_1000.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.053
Sequential time: 0.151
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_1000_1000_200.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.072
Sequential time: 0.187
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_1000_1000_500.tga'

JULIA Set
```

```
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.073
Sequential time: 0.193
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_1000_1000_1000.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.110
Sequential time: 0.305
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_2000_1000_200.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.130
Sequential time: 0.376
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_2000_1000_500.tga'

JULIA Set
  Plot a version of the Julia set for Z(k+1)=Z(k)^2-0.8+0.156i
Parallel time: 0.134
Sequential time: 0.387
Test PASSED

TGA_WRITE:
  Graphics data saved as 'output/dz_julia_2000_1000_1000.tga'
```

**Listing 5. Poređenje sekvencijalnog izvršavanja i izvršavanja sa 4 niti Julia2**

### *2.3.2. Grafici ubrzanja*

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




**Slika 2. Grafik zavisnosti ubrzanja algoritma pomocu for worksharing direktive od broja niti**

### *2.3.3. Diskusija dobijenih rezultata*

Ovaj zadatak je isti kao prethodni, samo što je trebalo da se paralelizacija izvrši manager-worker modelom. Iz tog razloga, pokretanje sa jednim procesom nema smisla (jer manager ne treba da se uključi u rad poslova već samo raspoređuje) i svodi se na sekvencijalnu implementaciju. Da se ne bi javljala greška napravili smo da se ukoliko postoji samo 1 proces izvrši samo sekvencijalna implementacija. Paralelizacija je ubrzala izvršavanje za 4 procesa, a pri radu sa 2 procesa (1 manager i 1 worker) malo usporila. Pri radu sa 2 procesa jasno je da će doći do usporenja jer idalje samo 1 proces radi potreban posao a dodatno vreme se troši na nepotrebnu komunikaciju. Rezultati sa 4 procesa pokazuju da je došlo do ubzanja.

# 3. PROBLEM 3 – IZOŠTRAVANJE PGM SLIKA

## 3.1. 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. Ukoliko je moguće, koristiti rutine za neblokirajuću komunikaciju za razmenu poruka. Program testirati sa parametrima koji su dati u datoteci run. [1, N]

## 3.2. Delovi koje treba paralelizovati

### *3.2.1. Diskusija*

Paralelizacija je moguća u dosharpen funkciji. Glavni zadatak za paralelizaciju je računanje matrice konvolucije.

### *3.2.2. Način paralelizacije*

Da bi se računanje matrice konvolucije izvršilo paralelno svaki od procesa pravi svoju matricu konvolucije i računa promenljive start, chunk i end na osnovu svog ranka. Da bi svi mogli da imaju podatke iz fuzzyPadded koji su potrebni za računanje matrice konvolucije, broadcastuje se fuzzy[0] gde je pre toga učitana slika iz ulaznog fajla u procesu sa rankom 0. Nakon paralelnog računanja konvolucije, radimo redukciju convolution[0] i promenljive pixcount. Nakon toga proces sa rankom 0 čuva izoštrenu sliku. Za redukciju je korišćena opcija MPI_IN_PLACE da bi se rezultat sacuvao u istom nizu iz kog se šalje.

## 3.3. Rezultati

Vremena sekvencijalnog i paralelnog izvrsavanja u tabeli su srednje vrednosti 3 pokretanja programa sa zadatim parametrima i navedeni su u formatu (vreme sekvencijalnog izvršavanja) / (vreme paralelnog izvršavanja) = ubrzanje.

| broj niti / slika i dimenzija | 1 | 2 | 4 |
|---|---|---|---|
| balloons_noisy 640x480 | 3.15/3.16 = 1.00 | 3.18/1.64 = 1.94 | 3.17/0.92 = 3.44 |
| bone_scint 1129x1865 | 21.48/21.41 = 1.00 | 21.35/10.88 = 1.96 | 21.51/5.72 = 3.45 |
| fuzzy 564x770 | 4.45/4.45 = 1.00 | 4.42/2.28 = 1.94 | 4.42/1.28 = 3.45 |
| lena512 512x512 | 2.70/2.68 = 1.01 | 2.67/1.39 = 1.92 | 2.82/0.80 = 3.54 |
| man 1024x1024 | 10.68/10.73 = 0.99 | 10.74/5.52 = 1.95 | 10.76/3.04 = 3.54 |
| Rainier_blur 1920x1080 | 21.11/21.19 = 0.99 | 21.11/10.76 = 1.96 | 21.19/5.71 = 3.71 |
| **Srednje ubrzanje** | **1.000** | **1.945** | **3.520** |

### *3.3.1. Logovi izvršavanja*

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

```
Input file is: data/balloons_noisy.pgm
Image size is 640 x 480
Parallel time: 3.14
Sequential time: 3.13
Test PASSED

Input file is: data/bone_scint.pgm
Image size is 1129 x 1865
Parallel time: 21.55
Sequential time: 21.55
Test PASSED

Input file is: data/fuzzy.pgm
Image size is 564 x 770
Parallel time: 4.53
Sequential time: 4.55
```

```
Test PASSED

Input file is: data/lena512.pgm
Image size is 512 x 512
Parallel time: 2.68
Sequential time: 2.77
Test PASSED

Input file is: data/man.pgm
Image size is 1024 x 1024
Parallel time: 10.66
Sequential time: 10.87
Test PASSED

Input file is: data/Rainier_blur.pgm
Image size is 1920 x 1080
Parallel time: 21.70
Sequential time: 21.45
Test PASSED
```

**Listing 6. Poređenje sekvencijalnog izvršavanja i izvršavanja sa 1 niti sharpen**

```
Input file is: data/balloons_noisy.pgm
Image size is 640 x 480
Parallel time: 1.64
Sequential time: 3.47
Test PASSED

Input file is: data/bone_scint.pgm
Image size is 1129 x 1865
Parallel time: 10.98
Sequential time: 21.36
Test PASSED

Input file is: data/fuzzy.pgm
Image size is 564 x 770
Parallel time: 2.32
Sequential time: 4.48
Test PASSED

Input file is: data/lena512.pgm
Image size is 512 x 512
Parallel time: 1.46
Sequential time: 2.72
Test PASSED

Input file is: data/man.pgm
Image size is 1024 x 1024
Parallel time: 5.47
Sequential time: 10.63
Test PASSED

Input file is: data/Rainier_blur.pgm
Image size is 1920 x 1080
Parallel time: 10.83
Sequential time: 20.99
Test PASSED
```

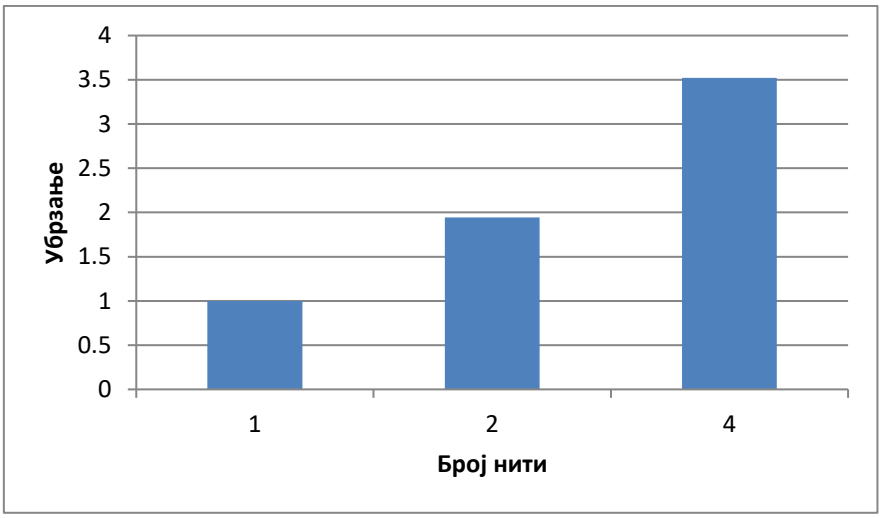
**Listing 7. Poređenje sekvencijalnog izvršavanja i izvršavanja sa 2 niti sharpen**

```
Input file is: data/balloons_noisy.pgm
Image size is 640 x 480
Parallel time: 0.91
Sequential time: 3.17
Test PASSED

Input file is: data/bone_scint.pgm
Image size is 1129 x 1865
Parallel time: 6.11
Sequential time: 21.54
Test PASSED

Input file is: data/fuzzy.pgm
Image size is 564 x 770
Parallel time: 1.30
Sequential time: 4.40
Test PASSED

Input file is: data/lena512.pgm
Image size is 512 x 512
Parallel time: 0.88
Sequential time: 2.75
Test PASSED

Input file is: data/man.pgm
Image size is 1024 x 1024
Parallel time: 3.13
Sequential time: 10.71
Test PASSED

Input file is: data/Rainier_blur.pgm
Image size is 1920 x 1080
Parallel time: 6.10
Sequential time: 21.29
Test PASSED
```

**Listing 8. Poređenje sekvencijalnog izvršavanja i izvršavanja sa 4 niti sharpen**

### *3.3.2. Grafici ubrzanja*

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




**Slika 3. Grafik zavisnosti ubrzanja algoritma pomocu taskova od broja niti**

### *3.3.3. Diskusija dobijenih rezultata*

Paralelizacija je ubrzala izvršavanje za 2 i 4 procesa, a pri radu sa 1 niti nije usporila. Najveće ubrzanje se dostiglo sa 4 niti - oko 3.5 puta, takođe primećujemo da ubrzanje nije puno zavisilo od dimenzije problema.

# 4. PROBLEM 4 – MRI-GRIDDING

## 4.1. 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, kao i na različite mogućnosti i nivoe na kojima se može obaviti paralelizacija koda. 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]

## 4.2. Delovi koje treba paralelizovati

### *4.2.1. Diskusija*

U funkciji gridding_Gold uočena je for petlja od 0 do n koju treba paralelizovati uzevši u obzir da je n veliki broj. Unutar ove for petlje, postoji još dosta forova koje je u principu moguće paralelizovati, ali se ta paralelizacija ne isplati. Prilikom rešavanja ovog zataka problem sa zadatim ACCURACY rešava se promenom sa float na double u strukturi cmplx.

### *4.2.2. Način paralelizacije*

Prvo master proces učita podatke i pošalje ostalim procesima. Ovo nismo merili u vremenu izvršavanja. Funkcija gridding_Gold se izvršava paralelno uz redukciju gridData i sampleDensity na kraju funkcije. Svaki proces izvršava deo pomenute for petlje i pre ulaska u for petlju svaki proces računa promenljive chunk, start i end. Svaki proces izvršava deo for petlje od start do end.

## 4.3. Rezultati

Vremena sekvencijalnog i paralelnog izvrsavanja u tabeli su srednje vrednosti 5 pokretanja programa sa zadatim parametrima i navedeni su u formatu (vreme sekvencijalnog izvršavanja) / (vreme paralelnog izvršavanja) = ubrzanje.

| broj niti / ulazi | 1 | 2 | 4 |
|---|---|---|---|
| small.uks | 1.95/1.98 = **0.98** | 1.96/1.24 = **1.58** | 1.97/0.98 = **2.01** |

### *4.3.1. 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
Sequential time: 1.92
Parallel time: 1.97
Test PASSED
```

**Listing 9. Poređenje sekvencijalnog izvršavanja i izvršavanja sa 1 niti mri-gridding**

```
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
Sequential time: 1.93
Parallel time: 1.25
Test PASSED
```

**Listing 10. Poređenje sekvencijalnog izvršavanja i izvršavanja sa 2 niti mri-gridding**

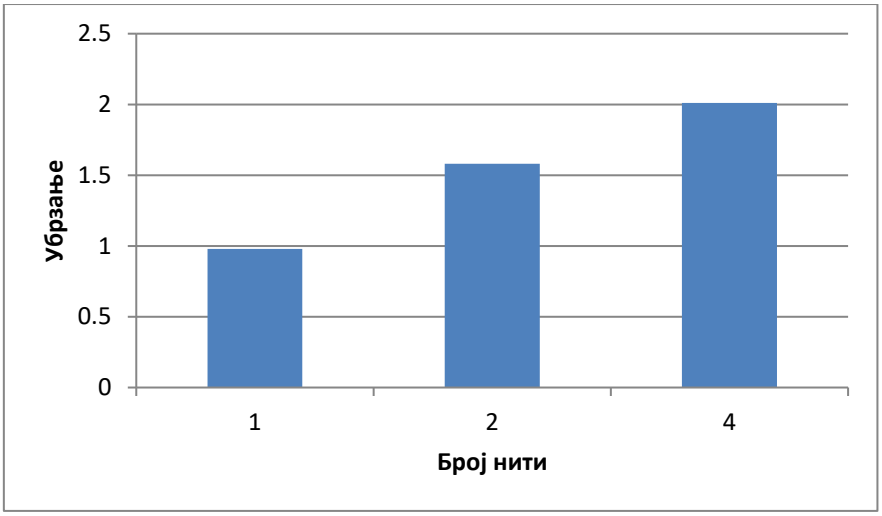
```
Reading parameters
  Number of samples = 2655910
  Grid Size = 256x256x256
  Input Matrix Size = 60x60x60
  Recon Matrix Size = 60x60x60
  Kernel Width = 5.000000
  KMax = 150.00 150.00 150.00
```

```
  Oversampling = 5.000000
  GPU Binsize = 32
  Use LUT = Yes
Reading input data from files
Generating Look-Up Table
Sequential time: 2.03
Parallel time: 0.98
Test PASSED
```

**Listing 11. Poređenje sekvencijalnog izvršavanja i izvršavanja sa 4 niti mri-gridding**

### *4.3.2. Grafici ubrzanja*

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




**Slika 4. Grafik zavisnosti ubrzanja naivnog algoritma od broja niti**

### *4.3.3. Diskusija dobijenih rezultata*

Paralelizacija je ubrzala izvršavanje za 2 i 4 procesa, a pri radu sa 1 procesom nije došlo do usporenja. Svaka tačka iz neuniformnog 3D prostora doprinosi susednim tačkama u regularnoj mreži u skladu sa Kaiser-Bessel funkcijom za određivanje rastojanja, pa svaki proces ima svoju kopiju koja se kasnije redukuje u proces sa rankom 0.