UNIVERZITET U BEOGRADU - ELEKTROTEHNIČKI FAKULTET

MULTIPROCESORKI SISTEMI (13S114MUPS, 13E114MUPS)

DOMAĆI ZADATAK 1 – OPENMP

Izveštaj o urađenom domaćem zadatku

Predmetni saradnici:

doc. dr Marko Mišić

dipl. ing. Pavle Divović

Studenti:

Matija Dodović 2018/0072

Miloš Milošević 2018/0445

Beograd, april 2022.

SADRŽAJ

1. PROBLEM 1 - SIMPLEX

1.1. Tekst problema

Paralelizovati program koji računa integral funkcije `F` na osnovu unutrašnjosti *simplex*-a (https://en.wikipedia.org/wiki/Simplex) u 20 dimenzija korišćenjem Monte Carlo metode. U izvornom kodu data je matrica eksponenata jednačine i ivica *simplex*-a. Ulazni parametar programa je broj iteracija aproksimacije. 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`.

1.2. Delovi koje treba paralelizovati

1.2.1. Diskusija

While petlja koja prolazi kroz sve iteracije (lon(n) iteracija) invocira izračunavanje u kome se računa ne može da se paralelizuje zato što je rezultat jedne iteracije potreban za sledeću iteraciju. *simplex_sample* funkcija vrši izračunavanja koristeći neki od algoritama za generisanje radnom brojeva, koji u osnovi ima *seed*. Vrednost *seed*-a se menja za svako sledeće izračunavanje u zavisnosti od prethodne vrednosti, tako da ništa vezano za random izračunavanje ne može da se paralelizuje. Ono što može da se paralelizuje jeste dvostruka for petlja, u funkciji *simplex_unit_to_general*, koja na određeni način kopira vrednosti (uz dodatna izračunavanja) iz jedne matrice u drugu. Sve ostalo u *simplex_sample* funkciji, u proizvoljnoj dubini poziva funckija, što može da se paralelizuje što može da se paralelizuje daje lošije rezultate usled malog i jednostavnog posla koju bi svaka nit izvršavala, dok su režijski troškovi konstantni. Rezultati *simplex_sample* funckije se koriste za izračunavanje krajnjeg rezultata za svaku iteraciju. Dvostruka for petlja koja to radi se može paralelizovati.

Usled specifičnosti problema uočava se da krajnji rezultat programa, koji se u ovoj konstelaciji izračunava u svakoj iteraciji, i u svakom trenutku za određeni broj iteracija predstavlja konačno rešenje, može se izdvojiti. Izračunavanje koje je u osnovi algoritam za generisanje random brojeva, ostaje u petlji koja će sad da radi duplo manje iteracija. Nakon te petlje se pokreće finalni algoritam za random brojeve i pokreće se dvostruka petlja za generisanje rezultata. Ta petlja, prema ranijoj diskusiji može da se paralelizuje. Ova šandarmacija sa kodom u osnovnom smislu donosi poboljšanje, dok paralelizacije unose dodatna poboljšanja.

1.2.2. Način paralelizacije

Osnovni zadatak, u kome nije dozvoljeno da se koristre *worksharing* direktive, pristupa se osnovnoj metodi paralelizacije (na oba mesta paralelizacije) tako da se unutrašnjost spoljašnje for petlje proglasi kao *unit-of-work* i da se ukupan broj *unit-of-work*-ova podeli po nitima, tako da svaka dobije podjednak broj *unit-of-work*-ova.

1.3. Rezultati

Predstavljeni su rezultati za 4 slučaja izvršavanja koda. Legenda se nalazi u nastavku:

basic_sequential	Nepromenjen sekvencijalni kod
basic_omp	Nepromenjen kod, paralelizovan ručnom raspodelom posla
modified_sequential	Modifikovan sekvencijalni kod
modifiec_omp	Modifikovan kod, paralelizovan ručnom raspodelom posla

1.3.1. Logovi izvršavanja

U nastavku se nalaze rezultati za 4 slučaja izvršavanja koda, za različit broj iteracija i različit broj niti.

Broj iteracija: 50000

```
OMP_NUM_THREADS=1
[basic_sequential] Elapsed (s): 0.877177
[basic_omp] Elapsed (s): 0.872136
[modified_sequential] Elapsed (s): 0.471000
[modifiec_omp] Elapsed (s): 0.476205
Test PASSED
speedup [basic_omp]: 1.00578006
speedup [modified_sequential]: 1.862371
speedup [modifiec_omp]: 1.842015
```

```
OMP_NUM_THREADS=2
[basic_sequential] Elapsed (s): 0.877177
[basic_omp] Elapsed (s): 0.478376
[modified_sequential] Elapsed (s): 0.471000
[modifiec_omp] Elapsed (s): 0.262717
Test PASSED
speedup [basic_omp]: 1.833656
speedup [modified_sequential]: 1.862371
speedup [modifiec_omp]: 4.338867
```

```
OMP_NUM_THREADS=4
[basic_sequential] Elapsed (s): 0.877177
[basic_omp] Elapsed (s): 0.271638
[modified_sequential] Elapsed (s): 0.471000
[modifiec_omp] Elapsed (s): 0.160525
Test PASSED
speedup [basic_omp]: 3.2292131
speedup [modified_sequential]: 1.862371
speedup [modifiec_omp]: 5.4644261
```

```
OMP_NUM_THREADS=8
[basic_sequential] Elapsed (s): 0.877177
[basic_omp] Elapsed (s): 0.218500
[modified_sequential] Elapsed (s): 0.471000
[modifiec_omp] Elapsed (s): 0.129345
Test PASSED
speedup [basic_omp]: 4.01454
speedup [modified_sequential]: 1.862371
speedup [modifiec_omp]: 6.781684
```

Grupa listing 1. Poređenje rezultata za 50000 iteracija

Broj iteracija: 100000

```
OMP_NUM_THREADS=1
[basic_sequential] Elapsed (s): 1.842235
[basic_omp] Elapsed (s): 1.849445
[modified_sequential] Elapsed (s): 1.005575
[modifiec_omp] Elapsed (s): 1.005181
Test PASSED
speedup [basic_omp]: 0.996102
speedup [modified_sequential]: 1.83202148
speedup [modifiec_omp]: 1.832739
```

```
OMP_NUM_THREADS=2
[basic_sequential] Elapsed (s): 1.842235
[basic_omp] Elapsed (s): 0.960102
[modified_sequential] Elapsed (s): 1.005575
[modifiec_omp] Elapsed (s): 0.549721
Test PASSED
speedup [basic_omp]: 1.91879092
speedup [modified_sequential]: 1.83202148
speedup [modifiec_omp]: 3.351518
```

```
OMP_NUM_THREADS=4
[basic_sequential] Elapsed (s): 1.842235
[basic_omp] Elapsed (s): 0.561320
[modified_sequential] Elapsed (s): 1.005575
[modifiec_omp] Elapsed (s): 0.343836
Test PASSED
speedup [basic_omp]: 3.28196929
speedup [modified_sequential]: 1.83202148
speedup [modifiec_omp]: 5.35788864
```

```
OMP_NUM_THREADS=8
[basic_sequential] Elapsed (s): 1.842235
[basic_omp] Elapsed (s): 0.422844
[modified_sequential] Elapsed (s): 1.005575
[modifiec_omp] Elapsed (s): 0.268574
Test PASSED
speedup [basic_omp]: 4.35677224
speedup [modified_sequential]: 1.83202148
speedup [modifiec_omp]: 6.85931996
```

Grupa listing 2. Poređenje rezultata za 100000 iteracija

Broj iteracija: 1000000

```
OMP_NUM_THREADS=1
[basic_sequential] Elapsed (s): 15.740635
[basic_omp] Elapsed (s): 15.896292
[modified_sequential] Elapsed (s): 8.639757
[modifiec_omp] Elapsed (s): 8.475198
Test PASSED
speedup [basic_omp]: 0.99028
speedup [modified_sequential]: 1.821884
speedup [modifiec_omp]: 1.85726
```

```
OMP_NUM_THREADS=2
[basic_sequential] Elapsed (s): 15.740635
[basic_omp] Elapsed (s): 8.397400
[modified_sequential] Elapsed (s): 8.639757
[modifiec_omp] Elapsed (s): 4.638564
Test PASSED
speedup [basic_omp]: 1.874465
speedup [modified_sequential]: 1.754856
speedup [modifiec_omp]: 3.393428
```

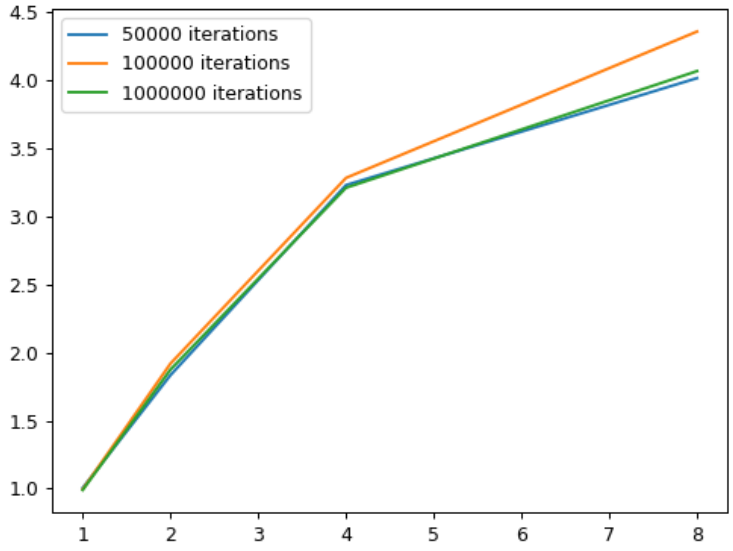
```
OMP_NUM_THREADS=4
[basic_sequential] Elapsed (s): 15.740635
[basic_omp] Elapsed (s): 4.904667
[modified_sequential] Elapsed (s): 8.639757
[modifiec_omp] Elapsed (s): 2.846133
Test PASSED
speedup [basic_omp]: 3.209318
speedup [modified_sequential]: 1.754856
speedup [modifiec_omp]: 5.530534
```

```
OMP_NUM_THREADS=8
[basic_sequential] Elapsed (s): 15.740635
[basic_omp] Elapsed (s): 3.871027
[modified_sequential] Elapsed (s): 8.639757
[modifiec_omp] Elapsed (s): 2.229195
Test PASSED
speedup [basic_omp]: 4.06627
speedup [modified_sequential]: 1.821884
speedup [modifiec_omp]: 7.06112969
```

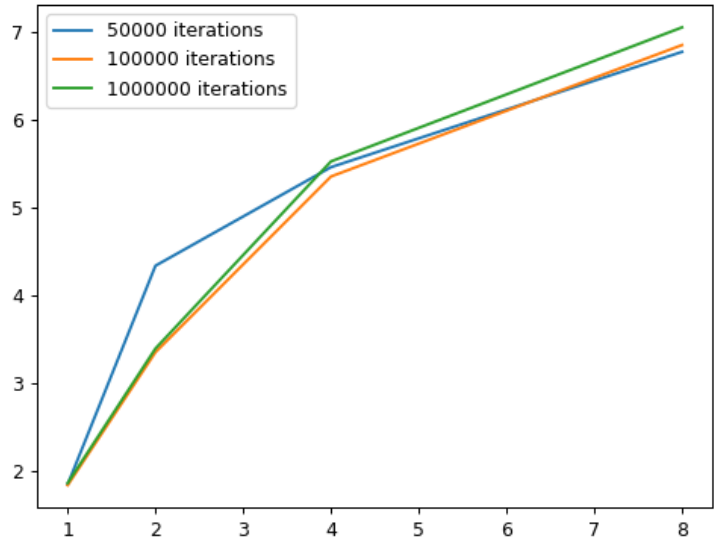
Grupa listing 3. Poređenje rezultata za 1000000 iteracija

1.3.2. Grafici ubrzanja

U nastavku su grafici ubrzanja osnovne implementacije algoritma sa paralelizovanim osnovnim algoritmom, kao i sa paralelizovanim modifikovanim kodom.



Slika 1. Grafik zavisnosti ubrzanja osnovnog sekvencijalnog koda sa paralelizovanim osnovnim sekvencijalnim kodom



Slika 2. Grafik zavisnosti ubrzanja osnovnog sekvencijalnog koda sa paralelizovanim modifikovanim sekvencijalnim kodom

1.3.3. Diskusija dobijenih rezultata

Paralelizacija osnovnog sekvencijalnog donosi ubrzanje samo kada je broj niti veći od 1. Sa jednom niti su vremena dosta slična, sa veoma malim usporenjem u slučaju paralelizacije zbog

overheda u kreiranju bazena niti (sa samo jednom niti) i barijere na kraju paralelnog regiona. Povećanje broja niti poboljšava rezultate (smanjuje vremena izvršavanja). Ubrzanja koja se dobijaju modifikacijom koda u odnosu na neparalelizovan kod su primetna (reda veličine 2), dok su sa povećanjem broja niti sve veća ubrzanja usled paralelizacije modifikovanog koda (sa 2 puta za 2 niti do 4 puta za 8 niti).

2. PROBLEM 2 - SIMPLEX

2.1. Tekst problema

Problem 1 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`.

2.2. Delovi koje treba paralelizovati

2.2.1. Diskusija

Uočena mesta i motivacija za paralelizaciju tih mesta su ista kao i u prethodnom problemu.

2.2.2. Način paralelizacije

Paralelizacija se u ovom problemu izvodi koristeći *worksharing* direktivu for. Rasporedjivanje pojedinačnih *unit-of-work*-ova po nitima se kontroliše direktivom *schedule(kind, chunksize)*. Za parametar *kind* su varirane vrednosti *static*, *dynamic* i *guided*. A za chunksize vrednosti 1, 2, 5 (što reprezentuje male vrednosti *chunksize*), kao i vrednosti 10 i 30 (koje reprezentuju velike vrednosti *chunksize*).

2.3. Rezultati

Predstavljeni su rezultati za 2 slučaja izvršavanja koda (analogno se izvode zaključci i za ostala dva slučaja, na osnovu zaključaka iz problema 1, a ovde je fokus na odnosima performansi za različite načine rasporedjivanja poslova). Izabrano je da prikazani rezultati budu za modifikovanu verziju koda jer se malom izmenom u samom kodu postižu značajna ubrzanja, pa je ovde prikazan najbolji slučaj. Kako su najveća ubrzanja postignuta za 8 niti, ovde je u analizi korišćen taj broj niti, i ispitivan je uticaj raspredjivanja na performanse. Za logove izvršavanja su prikazana samo poredjenja za *static* rasporedjivanje, dok su analize za *dynamic* i *guided* ostavljene za deo oko grafika. Legenda se nalazi u nastavku:

modified_sequential	Modifikovan sekvencijalni kod
modifiec_for_omp	Modifikovan kod, paralelizovan *worksharing* direktivom *for*

2.3.1. Logovi izvršavanja

Prikazani su rezultati za raspored *schedule(static, 1)*:

Broj iteracija: 50000

```
OMP_NUM_THREADS=8
[modified_sequential] Elapsed (s): 0.508690
[modifiec_omp] Elapsed (s): 0.130439
Test PASSED
speedup [modifiec_omp]: 3.899831
```

Grupa listing 1. Poređenje rezultata za 50000 iteracija

Broj iteracija: 100000

```
OMP_NUM_THREADS=8
[modified_sequential] Elapsed (s): 1.005575
[modifiec_omp] Elapsed (s): 0.266943
Test PASSED
speedup [modifiec_omp]: 3.767003
```

Grupa listing 2. Poređenje rezultata za 100000 iteracija

Broj iteracija: 1000000

```
OMP_NUM_THREADS=8
[modified_sequential] Elapsed (s): 8.395269
[modifiec_omp] Elapsed (s): 2.270859
Test PASSED
speedup [modifiec_omp]: 3.696957
```

Grupa listing 3. Poređenje rezultata za 1000000 iteracija

Prikazani su rezultati za raspored *schedule(static, 2)*:

Broj iteracija: 50000

```
OMP_NUM_THREADS=8
[modified_sequential] Elapsed (s): 0.508690
[modifiec_omp] Elapsed (s): 0.139410
Test PASSED
speedup [modifiec_omp]: 3.648877
```

Grupa listing 1. Poređenje rezultata za 50000 iteracija

Broj iteracija: 100000

```
OMP_NUM_THREADS=8
[modified_sequential] Elapsed (s): 1.005575
[modifiec_omp] Elapsed (s): 0.280250
Test PASSED
speedup [modifiec_omp]: 3.588136
```

Grupa listing 2. Poređenje rezultata za 100000 iteracija

Broj iteracija: 1000000

```
OMP_NUM_THREADS=8
[modified_sequential] Elapsed (s): 8.395269
[modifiec_omp] Elapsed (s): 2.378071
Test PASSED
speedup [modifiec_omp]: 3.530285
```

Grupa listing 3. Poređenje rezultata za 1000000 iteracija

Prikazani su rezultati za raspored *schedule(static, 5)*:

Broj iteracija: 50000

```
OMP_NUM_THREADS=8
[modified_sequential] Elapsed (s): 0.508690
[modifiec_omp] Elapsed (s): 0.133098
Test PASSED
speedup [modifiec_omp]: 3.821921
```

Grupa listing 1. Poređenje rezultata za 50000 iteracija

Broj iteracija: 100000

```
OMP_NUM_THREADS=8
[modified_sequential] Elapsed (s): 1.005575
[modifiec_omp] Elapsed (s): 0.270411
Test PASSED
speedup [modifiec_omp]: 3.718691
```

Grupa listing 2. Poređenje rezultata za 100000 iteracija

Broj iteracija: 1000000

```
OMP_NUM_THREADS=8
[modified_sequential] Elapsed (s): 8.395269
[modifiec_omp] Elapsed (s): 2.302063
Test PASSED
speedup [modifiec_omp]: 3.646846
```

Grupa listing 3. Poređenje rezultata za 1000000 iteracija

Prikazani su rezultati za raspored *schedule(static, 10)*:

Broj iteracija: 50000

```
OMP_NUM_THREADS=8
[modified_sequential] Elapsed (s): 0.508690
[modifiec_omp] Elapsed (s): 0.161074
Test PASSED
speedup [modifiec_omp]: 3.158114
```

Grupa listing 1. Poređenje rezultata za 50000 iteracija

Broj iteracija: 100000

```
OMP_NUM_THREADS=8
[modified_sequential] Elapsed (s): 1.005575
[modifiec_omp] Elapsed (s): 0.337470
Test PASSED
speedup [modifiec_omp]: 2.979746
```

Grupa listing 2. Poređenje rezultata za 100000 iteracija

Broj iteracija: 1000000

```
OMP_NUM_THREADS=8
[modified_sequential] Elapsed (s): 8.395269
[modifiec_omp] Elapsed (s): 2.900333
Test PASSED
speedup [modifiec_omp]: 2.894588
```

Grupa listing 3. Poređenje rezultata za 1000000 iteracija

Prikazani su rezultati za raspored *schedule(static, 30)*:

Broj iteracija: 50000

```
OMP_NUM_THREADS=8
[modified_sequential] Elapsed (s): 0.508690
[modifiec_omp] Elapsed (s): 0.360401
Test PASSED
speedup [modifiec_omp]: 1.411456
```

Grupa listing 1. Poređenje rezultata za 50000 iteracija

Broj iteracija: 100000

```
OMP_NUM_THREADS=8
[modified_sequential] Elapsed (s): 1.005575
[modifiec_omp] Elapsed (s): 0.762382
Test PASSED
speedup [modifiec_omp]: 1.318991
```

Grupa listing 2. Poređenje rezultata za 100000 iteracija

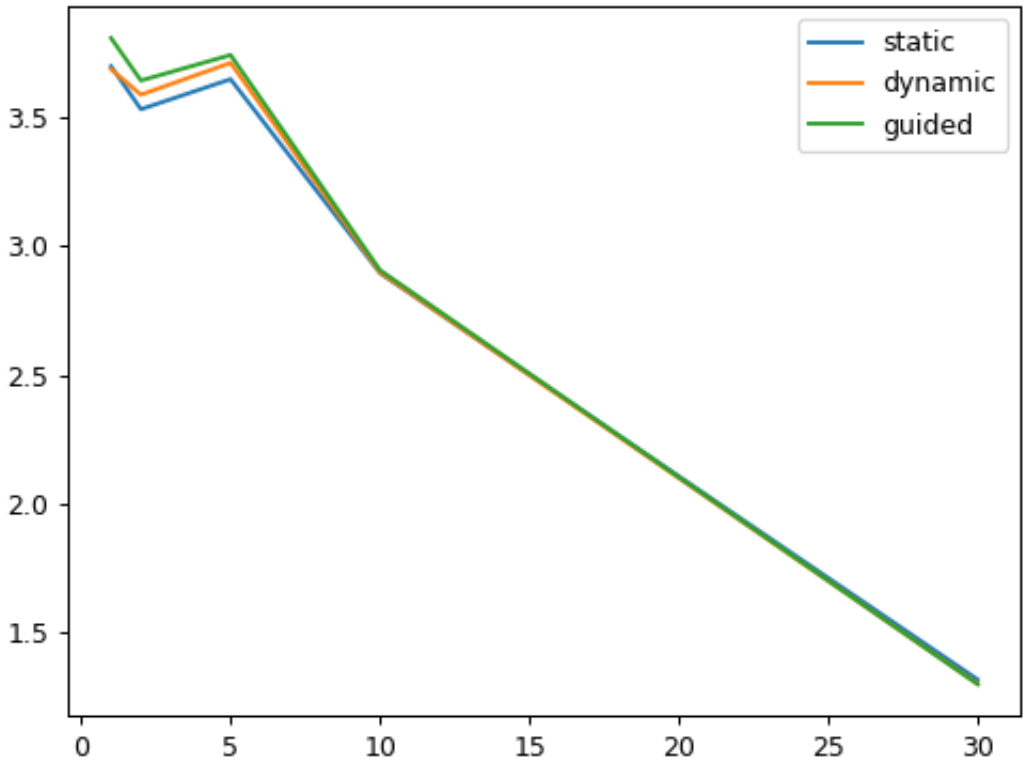
Broj iteracija: 1000000

```
OMP_NUM_THREADS=8
[modified_sequential] Elapsed (s): 8.395269
[modifiec_omp] Elapsed (s): 6.377364
Test PASSED
speedup [modifiec_omp]: 1.316417
```

Grupa listing 3. Poređenje rezultata za 1000000 iteracija

2.3.2. Grafici ubrzanja

U nastavku se nalazi grafik poređenja ubrzanja modifikovane implementacije algoritma sa paralelizovanim modifikovanim algoritmom za 3 različite varijante raspoređivanja za slučaj 1000000 iteracija gde je bilo upotrebljeno 8 niti.



Slika 1. Grafik zavisnosti ubrzanja modifikovanog sekvencijalnog koda sa paralelizovanim modifikovanim sekvencijalnim kodom za 3 moguća načina rasporedjivanja

2.3.3. Diskusija dobijenih rezultata

Dobijeni rezultati su konzistentni sa rezultatima sa ručnom podelom posla (problem 1), bazično sa povećanjem broja niti se povećava i ubrzanje. Tehnike rasporedjivanja igraju ulogu dominantno u izboru *chunksize* argumenta, dok sama tehnika raspoređivanja ima manji uticaj na ubrzanje. Manje vrednosti daju veće ubrzanje za sve 3 tehnike.

3. PROBLEM 3 – GAME OF LIFE

3.1. Tekst problema

Paralelizovati program koji implementira simulaciju ćelijskog automata Game of Life. Simulacija je predstavljena dvodimenzionalnom matricom dimenzija w x h, a svaka ćelija c može uzeti vrednost 1 ukoliko predstavlja živu ćeliju, a 0 ukoliko je mrtva. Za svaku ćeliju se vrši izračunavanje vrednosti n koja predstavlja zbir živih ćelija u susedstvu posmatrane ćelije. Posmatra se osam suseda. Ćelije se rađaju i umiru prema pravilima iz sledeće tabele.

Vrednost C	Vrednost N	Nova vrednost C	Komentar
1	0, 1	0	Usamljena ćelija umire
1	4, 5, 6, 7, 8	0	Ćelija umire usled prenaseljenosti
1	2,3	1	Ćelija živi
0	3	1	Rađa se nova ćelija
0	0, 1, 2, 4, 6, 7, 8	0	Nema promene stanja

Može se smatrati da su ćelije van opsega posmatrane matrice mrtve.

3.2. Delovi koje treba paralelizovati

3.2.1. Diskusija

Program se sastoji od *iter* uzastopnih poziva *evolve* funkcije u kojoj se vrši obrada matrice koja predstavlja stanje sistema. Unutar *evolve* funkcije, najpre se kreira nova matrica na osnovu prethodnog stanja sistema, pri čemu se svako polje računa nezavisno od ostalih, pa ovo predstavlja pogodan deo koda za paralelizaciju. Nakon kreiranja novog stanja, ono se postavlja kao stanje sistema prepisivanjem matrice polje po polje. Ovaj deo koda predstavlja još jednu priliku za paralelizaciju koda. Treći deo koda koji je moguće paralelizovati predstavlja inicijalizacija stanja sistema, u kojoj se svakom polju matrice nasumično dodeljuje jedna od vrednosti 0 ili 1, ali ta paralelizacija nije izvršena iz razloga što režijski troškovi koji nastaju kao posledica paralelizacije premašuju dobitke.

3.2.2. Način paralelizacije

Kako su petlje koje se koriste za obradu matrice savršeno ugneždene pravougaone pretlje za paralelizaciju se koristi direktiva ***#pragma omp parallel for colapse(2)***.

3.3. Rezultati

Analizom rezultata se može zaključiti da paralelizacija koda dovodi do značajnih ubrzanja ukoliko delove koda koji su paralelizovani izvršava više niti. Ukoliko je dostupna samo jedna nit da izvršava kod, režijski troškovi koji nastaju korišćenjem OpenMP API-a imaju za posledicu duže vreme izvršavanja programa.

3.3.1. Logovi izvršavanja

```
Test run variables: w=30 h=30 num_iter=1000 OMP_NUM_THREADS=1
Function execution time(non parallel): 0.000030s
Total simulation execution time(non parallel): 0.033164s
Function execution time(parallel): 0.000030s
Total simulation execution time(parallel): 0.037534s
Function execution time speedup: 1
Total execution time speedup: 0.883572
Test PASSED
```

```
Test run variables: w=30 h=30 num_iter=1000 OMP_NUM_THREADS=2
Function execution time(non parallel): 0.000030s
Total simulation execution time(non parallel): 0.033164s
Function execution time(parallel): 0.000019s
Total simulation execution time(parallel): 0.020830s
Function execution time speedup: 1.578947
Total execution time speedup: 1.592127
Test PASSED
```

```
Test run variables: w=30 h=30 num_iter=1000 OMP_NUM_THREADS=4
Function execution time(non parallel): 0.000030s
Total simulation execution time(non parallel): 0.033164s
Function execution time(parallel): 0.000012s
Total simulation execution time(parallel): 0.014517s
Function execution time speedup: 2.50000
Total execution time speedup: 2.284494
Test PASSED
```

```
Test run variables: w=30 h=30 num_iter=1000 OMP_NUM_THREADS=8
Function execution time(non parallel): 0.000030s
Total simulation execution time(non parallel): 0.033164s
Function execution time(parallel): 0.000010s
Total simulation execution time(parallel): 0.013281s
Function execution time speedup: 3.000000
Total execution time speedup: 2.497101
Test PASSED
```

```
Test run variables: w=500 h=500 num_iter=10 OMP_NUM_THREADS=1
Function execution time(non parallel): 0.006636s
Total simulation execution time(non parallel): 0.070940s
Function execution time(parallel): 0.007696s
Total simulation execution time(parallel): 0.082695s
Function execution time speedup: 0.862266
Total execution time speedup: 0.857851
Test PASSED
```

```
Test run variables: w=500 h=500 num_iter=10 OMP_NUM_THREADS=2
Function execution time(non parallel): 0.006636s
Total simulation execution time(non parallel): 0.070940s
Function execution time(parallel): 0.003886s
Total simulation execution time(parallel): 0.041427s
Function execution time speedup: 1.707669
Total execution time speedup: 1.712410
Test PASSED
```

```
Test run variables: w=500 h=500 num_iter=10 OMP_NUM_THREADS=4
Function execution time(non parallel): 0.006636s
Total simulation execution time(non parallel): 0.070940s
Function execution time(parallel): 0.001961s
Total simulation execution time(parallel): 0.021871s
```

```
Function execution time speedup: 3.338988
Total execution time speedup: 3.243565
Test PASSED
```

```
Test run variables: w=500 h=500 num_iter=10 OMP_NUM_THREADS=8
Function execution time(non parallel): 0.006636s
Total simulation execution time(non parallel): 0.070940s
Function execution time(parallel): 0.001604s
Total simulation execution time(parallel): 0.019168s
Function execution time speedup: 4.137157
Total execution time speedup: 3.700960
Test PASSED
```

```
Test run variables: w=1000 h=1000 num_iter=100 OMP_NUM_THREADS=1
Function execution time(non parallel): 0.027827s
Total simulation execution time(non parallel): 2.793098s
Function execution time(parallel): 0.032063s
Total simulation execution time(parallel): 3.3193359s
Function execution time speedup: 0.867885
Total execution time speedup: 0.874658
Test PASSED
```

```
Test run variables: w=1000 h=1000 num_iter=100 OMP_NUM_THREADS=2
Function execution time(non parallel): 0.027827s
Total simulation execution time(non parallel): 2.793098s
Function execution time(parallel): 0.015879s
Total simulation execution time(parallel): 1.616198s
Function execution time speedup: 1.752440
Total execution time speedup: 1.728190
Test PASSED
```

```
Test run variables: w=1000 h=1000 num_iter=100 OMP_NUM_THREADS=4
Function execution time(non parallel): 0.027827s
Total simulation execution time(non parallel): 2.793098s
Function execution time(parallel): 0.008034s
Total simulation execution time(parallel): 0.814203s
Function execution time speedup: 3.463654
Total execution time speedup: 3.430469
Test PASSED
```

```
Test run variables: w=1000 h=1000 num_iter=100 OMP_NUM_THREADS=8
Function execution time(non parallel): 0.027827s
Total simulation execution time(non parallel): 2.793098s
Function execution time(parallel): 0.006509s
Total simulation execution time(parallel): 0.658751s
Function execution time speedup: 4.275157
Total execution time speedup: 4.239991
Test PASSED
```

```
Test run variables: w=1000 h=1000 num_iter=1000 OMP_NUM_THREADS=1
Function execution time(non parallel): 0.027190s
Total simulation execution time(non parallel): 27.354100s
Function execution time(parallel): 0.031355s
Total simulation execution time(parallel): 31.374130s
Function execution time speedup: 0.867166
Total execution time speedup: 0.871868
Test PASSED
```

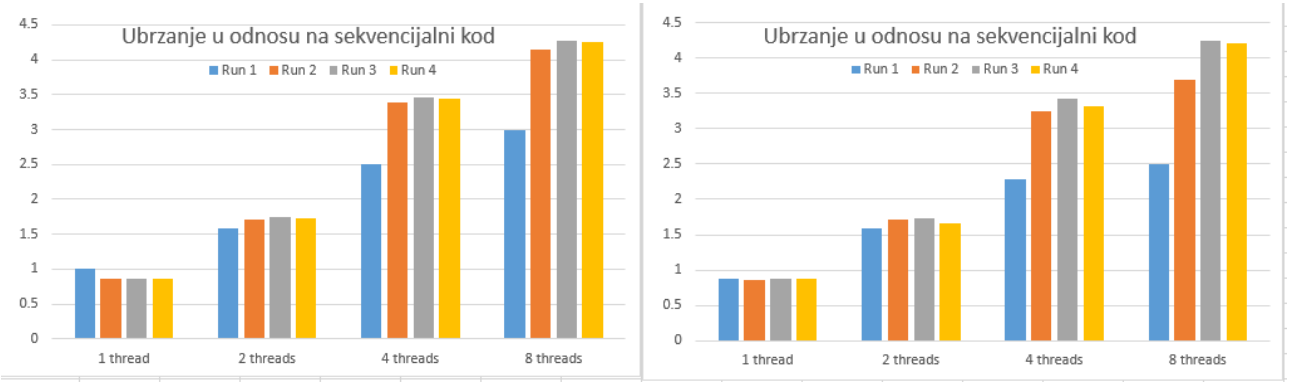
```
Test run variables: w=1000 h=1000 num_iter=1000 OMP_NUM_THREADS=2
Function execution time(non parallel): 0.027190s
Total simulation execution time(non parallel): 27.354100s
Function execution time(parallel): 0.015677s
Total simulation execution time(parallel): 16.463590s
Function execution time speedup: 1.734388
```

```
Total execution time speedup: 1.661490
Test PASSED
```

```
Test run variables: w=1000 h=1000 num_iter=1000 OMP_NUM_THREADS=4
Function execution time(non parallel): 0.027190s
Total simulation execution time(non parallel): 27.354100s
Function execution time(parallel): 0.007900s
Total simulation execution time(parallel): 8.263759s
Function execution time speedup: 3.441772
Total execution time speedup: 3.310128
Test PASSED
```

```
Test run variables: w=1000 h=1000 num_iter=1000 OMP_NUM_THREADS=8
Function execution time(non parallel): 0.027190s
Total simulation execution time(non parallel): 27.354100s
Function execution time(parallel): 0.006385s
Total simulation execution time(parallel): 6.489204s
Function execution time speedup: 4.258418
Total execution time speedup: 4.215325
Test PASSED
```

3.3.2. *Grafici ubrzanja*



Grafici ubrzanja izvršavanja funkcije *evolve* (levo) i izvršavanja kompletne simulacije (desno).

3.3.3. *Diskusija dobijenih rezultata*

Kao što je i očekivano, paralelizacija dovodi do poboljšanja performansi, kao i povećanje broja raspoloživih niti koje daje dodatno smanjuju vreme potrebno za izvršanvanje programa. Može se uočiti da su ubrzanja totalnog vremena izvršavanja niža od ubrzanja izvrčavanja pojedinačne funkcije kao posledica toga što u svakom programu koji je paralelizovan postoje delovi koda koji moraju da se izvršavanju sekvencijalno. Korišćenje različitih rasporeda *schedule* direktive nije dovelo do značajnih razlika u rezultatima.

4. PROBLEM 4 – GAME OF LIFE

4.1. Tekst problema

Rešiti prethodni problem korišćenjem koncepta poslova (tasks). Obratiti pažnju na eventualnu potrebu za sinhronizacijom. Rešenje testirati i prilagoditi tako da granularnost poslova bude optimalna. Program testirati sa parametrima koji su dati u datoteci run. [1, N]

4.2. Delovi koje treba paralelizovati

4.2.1. Diskusija

Identična kao u zadatku 3.

4.2.2. Način paralelizacije

Paralelni region koji se formira unutar *evolve* funkcije sadrži *single* region, tj. kreira se jedna nit koja će da prolazi kroz matricu i kreirati po jedan task za svaku vrstu te matrice. Ti taskovi će se dodeljivati nitima na izvršavanje. Nakon *single* regiona potrebno je uraditi jedan *omp taskwait* kako bi sledeći deo funkcije, prepisivanje nove matrice u stanje sistema, bilo odrađeno u istom paralelnom regionu, na isti način kao i glavna obrada matrice, jedna nit kreira taskove koji obradjuju po jednu vrstu. Kako su obe petlje koje se paralelizuju unutar istog *parallel* regiona potreban nam je *taskwait* da bi osigurali kompletnu obradu matrice pre početka prepisivanja, a razlog za taj način realizacije paralelizacije je taj da se smanje režijski troškovi koji nastaju prilikom kreiranja niti.

4.3. Rezultati

U okviru ove sekcije su izloženi rezultati paralelizacije problema 1.

4.3.1. Logovi izvršavanja

```
Test run variables: w=30 h=30 num_iter=1000 OMP_NUM_THREADS=1
Function execution time(non parallel): 0.000030s
Total simulation execution time(non parallel): 0.034775s
Function execution time(parallel): 0.000035s
Total simulation execution time(parallel): 0.041611s
Function execution time speedup: 0.857143
Total execution time speedup: 0.835717
Test PASSED
```

```
Test run variables: w=30 h=30 num_iter=1000 OMP_NUM_THREADS=2
Function execution time(non parallel): 0.000030s
Total simulation execution time(non parallel): 0.034775s
Function execution time(parallel): 0.000033s
Total simulation execution time(parallel): 0.040218s
Function execution time speedup: 0.909091
Total execution time speedup: 0.864663
Test PASSED
```

```
Test run variables: w=30 h=30 num_iter=1000 OMP_NUM_THREADS=4
Function execution time(non parallel): 0.000030s
Total simulation execution time(non parallel): 0.034775s
Function execution time(parallel): 0.000034s
Total simulation execution time(parallel): 0.040548s
Function execution time speedup: 0.882353
Total execution time speedup: 0.857626
Test PASSED
```

```
Test run variables: w=30 h=30 num_iter=1000 OMP_NUM_THREADS=8
Function execution time(non parallel): 0.000030s
Total simulation execution time(non parallel): 0.034775s
Function execution time(parallel): 0.000055s
Total simulation execution time(parallel): 0.059584s
Function execution time speedup: 0.545455
Total execution time speedup: 0.5836300
Test PASSED
```

```
Test run variables: w=500 h=500 num_iter=10 OMP_NUM_THREADS=1
Function execution time(non parallel): 0.006938s
Total simulation execution time(non parallel): 0.070940s
Function execution time(parallel): 0.006739s
Total simulation execution time(parallel): 0.07283s
Function execution time speedup: 1.029530
Total execution time speedup: 0.976019
Test PASSED
```

```
Test run variables: w=500 h=500 num_iter=10 OMP_NUM_THREADS=2
Function execution time(non parallel): 0.006938s
Total simulation execution time(non parallel): 0.070940s
Function execution time(parallel): 0.003554s
Total simulation execution time(parallel): 0.039423s
Function execution time speedup: 1.952167
Total execution time speedup: 1.799457
Test PASSED
```

```
Test run variables: w=500 h=500 num_iter=10 OMP_NUM_THREADS=4
Function execution time(non parallel): 0.006938s
Total simulation execution time(non parallel): 0.070940s
Function execution time(parallel): 0.002002s
Total simulation execution time(parallel): 0.022549s
Function execution time speedup: 3.465534
Total execution time speedup: 3.14038
Test PASSED
```

```
Test run variables: w=500 h=500 num_iter=10 OMP_NUM_THREADS=8
Function execution time(non parallel): 0.006938s
Total simulation execution time(non parallel): 0.070940s
Function execution time(parallel): 0.001956s
Total simulation execution time(parallel): 0.022068s
Function execution time speedup: 3.547035
```

```
Total execution time speedup: 3.214609
Test PASSED
```

```
Test run variables: w=1000 h=1000 num_iter=100 OMP_NUM_THREADS=1
Function execution time(non parallel): 0.027633s
Total simulation execution time(non parallel): 2.973603s
Function execution time(parallel): 0.029545s
Total simulation execution time(parallel): 2.841090s
Function execution time speedup: 0.935285
Total execution time speedup: 1.046642
Test PASSED
```

```
Test run variables: w=1000 h=1000 num_iter=100 OMP_NUM_THREADS=2
Function execution time(non parallel): 0.027633s
Total simulation execution time(non parallel): 2.973603s
Function execution time(parallel): 0.014281s
Total simulation execution time(parallel): 1.438663s
Function execution time speedup: 1.934949
Total execution time speedup: 2.066921
Test PASSED
```

```
Test run variables: w=1000 h=1000 num_iter=100 OMP_NUM_THREADS=4
Function execution time(non parallel): 0.027633s
Total simulation execution time(non parallel): 2.973603s
Function execution time(parallel): 0.007613s
Total simulation execution time(parallel): 0.770889s
Function execution time speedup: 3.629712
Total execution time speedup: 3.857369
Test PASSED
```

```
Test run variables: w=1000 h=1000 num_iter=100 OMP_NUM_THREADS=8
Function execution time(non parallel): 0.027633s
Total simulation execution time(non parallel): 2.973603s
Function execution time(parallel): 0.006998s
Total simulation execution time(parallel): 0.710133s
Function execution time speedup: 3.948700
Total execution time speedup: 4.187389
Test PASSED
```

```
Test run variables: w=1000 h=1000 num_iter=1000 OMP_NUM_THREADS=1
Function execution time(non parallel): 0.027036s
Total simulation execution time(non parallel): 27.35239s
Function execution time(parallel): 0.027934s
Total simulation execution time(parallel): 28.001500s
Function execution time speedup: 0.967853
Total execution time speedup: 0.976819
Test PASSED
```

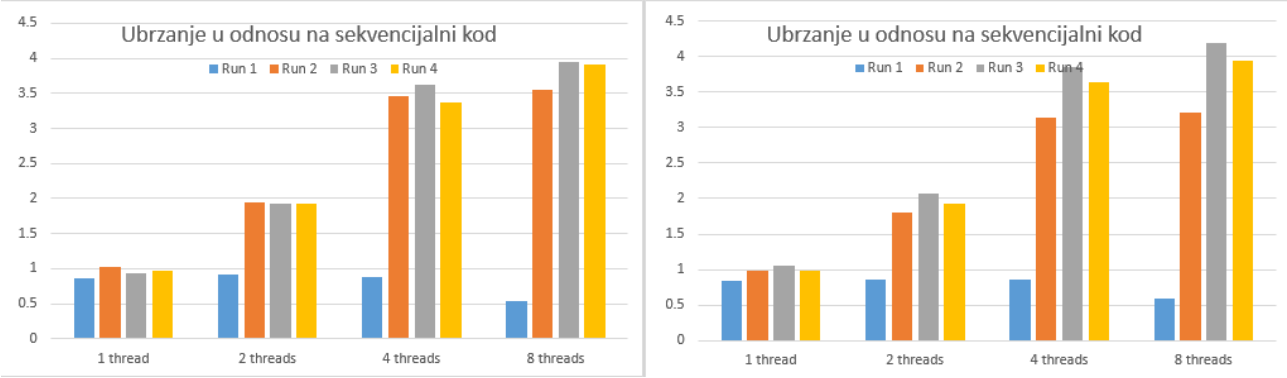
```
Test run variables: w=1000 h=1000 num_iter=1000 OMP_NUM_THREADS=2
Function execution time(non parallel): 0.027036s
Total simulation execution time(non parallel): 27.35239s
Function execution time(parallel): 0.013967s
Total simulation execution time(parallel): 14.193920s
Function execution time speedup: 1.935706
Total execution time speedup: 1.927049
Test PASSED
```

```
Test run variables: w=1000 h=1000 num_iter=1000 OMP_NUM_THREADS=4
Function execution time(non parallel): 0.027036s
Total simulation execution time(non parallel): 27.35239s
Function execution time(parallel): 0.008038s
Total simulation execution time(parallel): 7.515491s
```

```
Function execution time speedup: 3.363523
Total execution time speedup: 3.639468
Test PASSED
```

```
Test run variables: w=1000 h=1000 num_iter=1000 OMP_NUM_THREADS=8
Function execution time(non parallel): 0.027036s
Total simulation execution time(non parallel): 27.35239s
Function execution time(parallel): 0.006926s
Total simulation execution time(parallel): 6.949921s
Function execution time speedup: 3.904679
Total execution time speedup: 3.935640
Test PASSED
```

4.3.2. Grafici ubrzanja



Grafici ubrzanja izvršavanja funkcije *evolve* (levo) i izvršavanja kompletne simulacije (desno).

4.3.3. Diskusija dobijenih rezultata

Kao i u prethodnom zadatku paralelizacija generalno dovodi do ubrzanja (više detalja kasnije u tekstu), pri čemu se može uočiti šablon blago boljih ubrzanja pri manjem broju niti, dok pri većem broju niti ubrzanja su blago niža u odnosu na prethodni problem, tj. različit način paralelizacije. Jedan izuzetak u ovom problemu predstavlja Run 1 sa promenljivama w=30, h=30, num_iter=1000. Za ovaj skup parametara, ne samo da se ne dobija očekivano ubrzanje izvršavanja programa, već se program izvršava sporije nego sekvencijalna implementacija, a povećanjem broja niti se sve više ispoljava uočeni efekat.

5. PROBLEM 5 - HOTSPOT

5.1. Tekst problema

Paralelizovati program koji rešava problem promene temperature na čipu procesora u dvodimenzionalnom prostoru kroz vreme, ako su poznati početna temperatura i granični uslovi. Simulacija rešava seriju diferencijalnih jednačina nad pravilnom mrežom tačaka kojom se aproksimira površina procesora. Svaka tačka u mreži predstavlja prosečnu temperaturu za odgovarajuću površinu na čipu. Mreža tačaka je predstavljena odgovarajućom matricom koja opisuje trenutne temperature. Analizirati dati kod i obratiti pažnju na način izračunavanja temperatura. Verifikaciju paralelizovanog rešenja vršiti nad dobijenim temperaturama u poslednjem stanju sistema. Način pokretanja programa se nalazi u datoteci run. [1, N].

5.2. Delovi koje treba paralelizovati

5.2.1. Diskusija

Glavni deo programa predstavlja petlja koja se sastoji od *num_iterations* iteracija u kojima se izvršava *single_iteration* funkcije koja vrši obradu nad marticom veličine *row*col* pri čemu se svako polje matrice računa nezavisno. Vrednost jednog izvršavanja funkcije se koristi u narednoj iteraciji.

5.2.2. Način paralelizacije

Glavnu petlju nije moguće paralelizovati jer se vrednost jedne iteracije koristi u narednoj, ali je moguće paralelizovati obradu unutar jednog poziva funkcije. Paralelizaciju je moguće uraditi na dva načina, korišćenjem *#pragma omp parallel for colapse(2)* direktive ili kreiranjem paralelnog regiona i pravljenja taskova. Korišćenje taskova zahteva malu prepravku koda tako da se pri pozivu funkcije *single_iteration* zadaje kao parametar i broj početnog i krajnjeg reda matrice koji se obrađuje. Na taj način je moguće u paraleli zadati izvršavanje više poziva funkcije koji će obrađivati deo matrice umesto jednog poziva koji obrađuje celu matricu.

5.3. Rezultati

Analizom rezultata se pokazuje da korišćenje taskova daje bolje performanse, ali ono zahteva promenu koda, kako bi se niti kreirale van poziva funkcije *single_iteration,* jer u suprotnom ne bi bilo poboljšanja u odnosu na *#pragma omp parallel for colapse(2)* direktivu, kod koje se pri svakom pozivu kreiraju nove niti.

5.3.1. Logovi izvršavanja

```
Test run variables: row=32 col=32 num_iterations=8192
Total simulation execution time(sequential):  0.03029s
Time (parallel; 1 thread; for): 0.037638s
Speedup: 0.801772
Time (parallel; 1 thread; task): 0.029017s
Speedup: 1.043871
Time (parallel; 2 thread; for): 0.025498s
Speedup: 1.187936
Time (parallel; 2 thread; task): 0.01903s
Speedup: 1.591697
Time (parallel; 4 thread; for): 0.018109s
Speedup: 1.672649
Time (parallel; 4 thread; task): 0.016848s
Speedup: 1.797840
Time (parallel; 8 thread; for): 0.020598s
Speedup: 1.470531
Time (parallel; 8 thread; task): 0.015051s
Speedup: 2.012491
Test PASSED
```

```
Test run variables: row=256 col=256 num_iterations=8192
Total simulation execution time(sequential):  1.304012s
Time (parallel; 1 thread; for): 1.822592s
Speedup: 0.715471
Time (parallel; 1 thread; task): 1.26827s
Speedup: 1.028182
Time (parallel; 2 thread; for): 1.005028s
Speedup: 1.297488
Time (parallel; 2 thread; task): 0.706593s
Speedup: 1.845492
Time (parallel; 4 thread; for): 0.465275s
Speedup: 2.802669
Time (parallel; 4 thread; task): 0.325797s
```

```
Speedup: 4.002529
Time (parallel; 8 thread; for): 0.591472s
Speedup: 2.204689
Time (parallel; 8 thread; task): 0.341207s
Speedup: 3.821762
Test PASSED
```

```
Test run variables: row=1024 col=1024 num_iterations=4096
Total simulation execution time(sequential):  14.044399s
Time (parallel; 1 thread; for): 15.927118s
Speedup: 0.881792
Time (parallel; 1 thread; task): 10.979757s
Speedup: 1.279117
Time (parallel; 2 thread; for): 7.138147s
Speedup: 1.967513
Time (parallel; 2 thread; task): 5.305784s
Speedup: 2.646998
Time (parallel; 4 thread; for): 3.676159s
Speedup: 3.820400
Time (parallel; 4 thread; task): 2.948303s
Speedup: 4.763553
Time (parallel; 8 thread; for): 4.892868s
Speedup: 2.870382
Time (parallel; 8 thread; task): 2.643538s
Speedup: 5.312728
Test PASSED
```

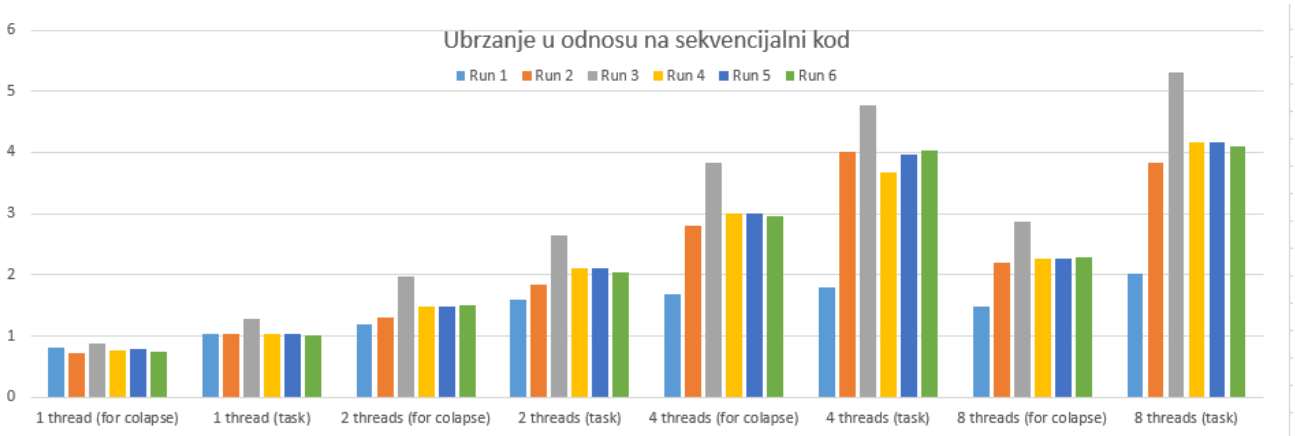
```
 Test run variables: row=1024 col=1024 num_iterations=8192
Total simulation execution time(sequential):  21.954688s
Time (parallel; 1 thread; for): 28.512548s
Speedup: 0.769873
Time (parallel; 1 thread; task): 21.268475s
Speedup: 1.032093
Time (parallel; 2 thread; for): 14.926427s
```

```
Speedup: 1.470616
Time (parallel; 2 thread; task): 10.442765s
Speedup: 2.102034
Time (parallel; 4 thread; for): 7.326150s
Speedup: 2.996259
Time (parallel; 4 thread; task): 5.983003s
Speedup: 3.668901
Time (parallel; 8 thread; for): 9.738173s
Speedup: 2.254124
Time (parallel; 8 thread; task): 5.256901s
Speedup: 4.175663
Test PASSED
```

```
Test run variables: row=1024 col=1024 num_iterations=16384
Total simulation execution time(sequential):  43.954688s
Time (parallel; 1 thread; for): 56.234499s
Speedup: 0.781632
Time (parallel; 1 thread; task): 42.286887s
Speedup: 1.039440
Time (parallel; 2 thread; for): 29.829287s
Speedup: 1.473541
Time (parallel; 2 thread; task): 20.862693s
Speedup: 2.106856
Time (parallel; 4 thread; for): 14.649824s
Speedup: 3.000356
Time (parallel; 4 thread; task): 11.110411s
Speedup: 3.956171
Time (parallel; 8 thread; for): 19.442866s
Speedup: 2.260710
Time (parallel; 8 thread; task): 10.563800s
Speedup: 4.160878
Test PASSED
```

```
Test run variables: row=1024 col=1024 num_iterations=32768
Total simulation execution time(sequential):  86.571069s
Time (parallel; 1 thread; for): 116.461971s
Speedup: 0.743342
Time (parallel; 1 thread; task): 86.088170s
Speedup: 1.005609
Time (parallel; 2 thread; for): 57.783456s
Speedup: 1.498198
Time (parallel; 2 thread; task): 42.487947s
Speedup: 2.037544
Time (parallel; 4 thread; for): 29.295556s
Speedup: 2.955092
Time (parallel; 4 thread; task): 21.480544s
Speedup: 4.030208
Time (parallel; 8 thread; for): 37.900712s
Speedup: 2.284154
Time (parallel; 8 thread; task): 21.087956s
Speedup: 4.105238
Test PASSED
```

5.3.2. Grafici ubrzanja



Grafik ubrzanja vremena izvršavanja programa u zavisnosti od broja niti i načina paralelizacije.

5.3.3. Diskusija dobijenih rezultata

Iz priloženih rezultata se može uočiti da se paralelizacijom postiže ubrzanje programa, kao i to da izbacivanjem režijskih troškova koji nastaju kreiranjem novih niti pri svakom pozivu funkcije *single_iteration* postiže dodatno ubrzanje. Iz rezultata koda koji je paralelizovan pomoću for direktive se može uočiti to da performanse rastu kako se broj niti povećava do 4, ali onda opadaju kada se broj niti poveća na 8. To je posledica činjenice da se pri svakom pozivu funkcije *single_iteration* ove niti iznova kreiraju, pa se može primetiti da dodavanje više niti ne garantuje poboljšanje performansi, već da je potrebno balansirati dobitke koji nastaju paralelizacijom koda i gubitke usled režijskih troškova kreiranja novih niti kako bi smo dobili maksimalne dobitke za specifičan problem.