UniverzitetuBeogradu - Elektrotehničkifakultet

Multiprocesorki sistemi (13S114MUPS, 13E114MUPS)



Domaći zadatak 1 – OPENMP

Izveštaj o urađenom domaćem zadatku

|  |  |
| --- | --- |
| Predmetni saradnici: | Studenti: |
| doc. drMarko Mišić  dipl. ing. Pavle Divović | Ana Radovanović2019/0282  Jovana Jaćimović2019/0593 |

Beograd, novembar 2022.

Sadržaj

[Sadržaj 2](#_Toc119965093)

[1. Problem 1 3](#_Toc119965094)

[1.1. Tekst problema 3](#_Toc119965095)

[1.2. Delovi koje treba paralelizovati 3](#_Toc119965096)

[1.2.1. Diskusija 3](#_Toc119965097)

[1.2.2. Način paralelizacije 3](#_Toc119965098)

[1.3. Rezultati 3](#_Toc119965099)

[1.3.1. Logovi izvršavanja 4](#_Toc119965100)

[1.3.2. Grafici ubrzanja 5](#_Toc119965101)

[1.3.3. Diskusija dobijenih rezultata 7](#_Toc119965102)

[2. Problem 2 8](#_Toc119965103)

[2.1. Tekst problema 8](#_Toc119965104)

[2.2. Delovi koje treba paralelizovati 8](#_Toc119965105)

[2.2.1. Diskusija 8](#_Toc119965106)

[2.2.2. Način paralelizacije 8](#_Toc119965107)

[2.3. Rezultati 8](#_Toc119965108)

[2.3.1. Logovi izvršavanja 9](#_Toc119965109)

[2.3.2. Grafici ubrzanja 9](#_Toc119965110)

[2.3.3. Diskusija dobijenih rezultata 9](#_Toc119965111)

[3. Problem 3 11](#_Toc119965112)

[3.1. Tekst problema 11](#_Toc119965113)

[3.2. Delovi koje treba paralelizovati 11](#_Toc119965114)

[3.2.1. Diskusija 11](#_Toc119965115)

[3.2.2. Način paralelizacije 11](#_Toc119965116)

[3.3. Rezultati 11](#_Toc119965117)

[3.3.1. Logovi izvršavanja 11](#_Toc119965118)

[3.3.2. Grafici ubrzanja 12](#_Toc119965119)

[3.3.3. Diskusija dobijenih rezultata 12](#_Toc119965120)

1. Problem 1

U okviru ovog poglavlja je dat kratak izveštaj u vezi rešenja zadatog problema 1.

* 1. Tekst problema

Paralelizovati program koji vrši određivanje ukupnog broja prostih brojeva u zadatom opsegu. Program se nalazi u datoteci prime.c u arhivi koja je priložena uz ovaj dokument. 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. Delovi koje treba paralelizovati
     1. Diskusija

Paralelizacija se vrši u funkciji prime\_number, koja od računa ukupan broj prostih brojeva u zadatom opsegu. Funkcija sadrži dve for petlje, od kojih spoljašnja predstavlja prolazak kroz brojeve u tom opsegu, dok unutrašnja predstavlja prolazak kroz delioce jednog broja. Paralelizacija je urađena nad spoljašnjom for petljom.

* + 1. Način paralelizacije

Kako je u zadatku naznačeno da nije dozvoljeno korišćenje worksharing direktiva, paralelizacija je u ovom zadatku urađena ručno. Niz je podeljen na onoliko delova koliko ima raspoloživih niti i delovi su približno jednake veličine. Svaka nit dobila je po jedan deo za obrađivanje i kada sve niti završe posao, svoj rezultat beleže tj. sabiraju u zajedničku redukcionu promenljivu koja predstavlja broj prostih brojeva. Problem koji se uočava u ovakvoj paralelizaciji je taj što se niz deli na nekoliko sukcesivnih delova, tako da prva nit obrađuje prvih n elemenata, dok poslednja nit obrađuje poslednjih n elemenata. Kako su prvi elementi niza mali brojevi, druga petlja će imati manje iteracija i samim tim će prva nit dosta ranije završiti posao od poslednje, što dovodi do toga da raspodela poslova nije uniformna. Kako se ručno ne bismo bavili ovim problemom, odlučeno je da se u zadatku broj 2 ovome posveti više pažnje.

* 1. Rezultati

U okviru ove sekcije su izloženi rezultati paralelizacije problema 1.

* + 1. Logovi izvršavanja

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

SEKVENCIJALNO IZVRŠAVANJE

time ./prime 1 131072 2

real 1.325s

time ./prime 5 500000 10

real 12.614s

time ./prime 1 65536 4

real 0.279s

Listing 1. Sekvencijalno izvršavanje problema 1

NUM\_THREADS=8

time ./dz1z1.exe 1 131072 2

real 0.259s

time ./ dz1z1.exe 5 500000 10

real 2.114s

time ./ dz1z1.exe 1 65536 4

real 0.06s

Listing 2. Izvršavanje problema 1 za N=8 niti

NUM\_THREADS=4

time ./dz1z1.exe 1 131072 2

real 0.563s

time ./ dz1z1.exe 5 500000 10

real 5.417s

time ./ dz1z1.exe 1 65536 4

real 0.122s

Listing 3. Izvršavanje problema 1 za N=4 niti

NUM\_THREADS=2

time ./dz1z1.exe 1 131072 2

real 0.946s

time ./ dz1z1.exe 5 500000 10

real 9.265s

time ./ dz1z1.exe 1 65536 4

real 0.206s

Listing 4. Izvršavanje problema 1za N=2 niti

NUM\_THREADS=1

time ./dz1z1.exe 1 131072 2

real 1.308s

time ./ dz1z1.exe 5 500000 10

real 12.360s

time ./ dz1z1.exe 1 65536 4

real 0.276s

Listing 5. Izvršavanje problema 1 za N=1 niti

* + 1. Grafici ubrzanja

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

Slika 1. Grafik zavisnosti ubrzanja izvršavanja problema 1 od ulaznih parametara za N = 8 niti

Slika2. Grafik zavisnosti ubrzanja izvršavanja problema 1 od ulaznih parametara za N = 4 niti

Slika3. Grafik zavisnosti ubrzanja izvršavanja problema 1 od ulaznih parametara za N = 2 niti

Slika4. Grafik zavisnosti ubrzanja izvršavanja problema 1 od ulaznih parametara za 8, 4 i 2 niti

* + 1. Diskusija dobijenih rezultata

Kao što možemo videti iz priloženih logova i grafika, a i kao sto je očekivano, ubrzanje raste srazmerno sa porastom broja niti. Najoptimalnije performanse dobijaju sa kada je broj niti = 8 i tada je ubrzanje 4.5 – 6, u zavisnosti od veličine podataka nad kojim se algoritam izvršava. Naravno, što je više podataka, to je ubrzanje veće, te se na test primeru broj 2 dobijaju najbolja ubrzanja. Sa smanjenjem broja niti opada i ubrzanje, pa je ubrzanje koje dobijemo koristeći 4 niti nešto manje, dok je ubrzanje koristeći 2 niti još manje, kao što se može jasno videti u priloženim graficima. Kako Open MP nudi mogućnost da se for petlje paralelizuju korišćenjem worksharing direktiva, rešenje koje je dobijeno ručnom paralelizacijom u ovom zadatku nije najoptimalnije, što se može videti odmah u narednom zadatku.

1. Problem 2

U okviru ovog poglavlja je dat kratak izveštaj u vezi rešenja zadatog problema 2.

* 1. Tekst problema

Paralelizovati program koji vrši određivanje ukupnog broja prostih brojeva u zadatom opsegu. Program se nalazi u datoteci prime.c u arhivi koja je priložena uz ovaj dokument. Obratiti pažnju na ispravno deklarisanje svih promenljivih prilikom paralelizacije. Program paralelizovati korišćenjem direktiva za podelu posla (worksharing direktive). Program testirati sa parametrima koji su dati u datoteci run.

* 1. Delovi koje treba paralelizovati
     1. Diskusija

Paralelizacija se vrši u funkciji prime\_number, koja od računa ukupan broj prostih brojeva u zadatom opsegu. Funkcija sadrži dve for petlje, od kojih spoljašnja predstavlja prolazak kroz brojeve u tom opsegu, dok unutrašnja predstavlja prolazak kroz delioce jednog broja. Paralelizacija je urađena nad spoljašnjom for petljom.

* + 1. Način paralelizacije

U ovom zadatku, izvršena je paralelizacija worksharing direktivo za for petlju. Kao što je spomenuto u prethodnom zadatku, ovde će biti priče o raspodeli podataka po nitima. Ukoliko ne primenimo ni jednu klauzulu za raspored, podaci će se podeliti statički, odnosno niti će raditi sa sukcesivnim elementima niza. Naznačeno je da ovo predstvalja manji problem, jer podaci nisu uniformno raspoređeni, te će poslednja nit imati najviše posla. Ovde se kao najoptimalnije rešenje nameće kaluzula guided, koja na početku nitima daje više odbiraka, dok se kasnije taj broj smanjuje do naznačenog. Ovo i jeste najpogodnija optimizacija za ovaj problem. Pored ovog pristupa, isprobano je i korišćenje dynamic i static klauzule, gde se korišćenjem static klauzule vreme izvršavanja poveća za nekih 20% - 25%, dok klauzula dynamic daje dosta slične rezultate kao guided.

* 1. Rezultati

U okviru ove sekcije su izloženi rezultati paralelizacije problema 1.

* + 1. Logovi izvršavanja

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

SEKVENCIJALNO IZVRŠAVANJE

time ./prime 1 131072 2

real 1.325s

time ./prime 5 500000 10

real 12.614s

time ./prime 1 65536 4

real 0.279s

Listing 6. Sekvencijalno izvršavanje problema 2

NUM\_THREADS=8

time ./dz1z2.exe 1 131072 2

real 0.185s

time ./dz1z2.exe 5 500000 10

real 1.722s

time ./dz1z2.exe 1 65536 4

real 0.044s

Listing 7. Izvršavanje problema 2 za N=8 niti

NUM\_THREADS=4

time ./dz1z2.exe 1 131072 2

real 0.340s

time ./dz1z2.exe 5 500000 10

real 3.914s

time ./dz1z2.exe 1 65536 4

real 0.077s

Listing 8. Izvršavanje problema 2 za N=4 niti

NUM\_THREADS=2

time ./dz1z2.exe 1 131072 2

real 0.653s

time ./dz1z2.exe 5 500000 10

real 6.321s

time ./dz1z2.exe 1 65536 4

real 0.141s

Listing 9. Izvršavanje problema 2 za N=2 niti

NUM\_THREADS=1

time ./dz1z2.exe 1 131072 2

real 1.285s

time ./dz1z2.exe 5 500000 10

real 12.616s

time ./dz1z2.exe 1 65536 4

real 0.279s

Listing 10. Izvršavanje problema 2 za N=1 niti

* + 1. Grafici ubrzanja

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

Slika5. Grafik zavisnosti ubrzanja izvršavanja problema 2 od ulaznih parametara za N = 8 niti

Slika6. Grafik zavisnosti ubrzanja izvršavanja problema 2 od ulaznih parametara za N = 4 niti

Slika7. Grafik zavisnosti ubrzanja izvršavanja problema 2 od ulaznih parametara za N = 2 niti

Slika8. Grafik zavisnosti ubrzanja izvršavanja problema 2 od ulaznih parametara za 8, 4 i 2 niti

* + 1. Diskusija dobijenih rezultata

Kao što je već spomenuto u prethodnom zadatku, korišćenjem worksharing direktiva koje pruža Open MP, performanse postaju najoptimalnije. U prethodnom zadatku videli smo da se performanse korišćenjem 8 niti povećaju za približno 6 puta, odnosno vreme izvršavanja se za toliko smanji, dok se u ovom zadatku, parelelizacijom istog problema vreme izvršavanja smanjilo za preko 7 puta. Ovim pokazujemo da je korišćenje Open MP direktiva još više ubrzan rad algoritma. Na graficima su prikazani i rezultati ukoliko se koriste 2 ili 4 niti, gde se može uočiti da ubrzanje znatno pada, mada iako se broj niti smanjuje duplo, ubrzanja opadaju nešto manje. Ova pojava objašnjava se time što paralelizacija sama sa sobom donosi određene overhead-ove i samim tim rezultati ne mogu biti na onom nivou na kom bi to teorijski bilo izvodljivo.

1. Problem 3

U okviru ovog poglavlja je dat kratak izveštaj u vezi rešenja zadatog problema 3.

* 1. Tekst problema

Paralelizovati program koji vrši izračunavanje 3D Poasonove jednačine korišćenjem Feyman-Kac algoritma. Algoritam stohastički računa rešenje parcijalne diferencijalne jednačine krenuvši N puta iz različitih tačaka domena. Tačke se kreću po nasumičnim putanjama i prilikom izlaska iz granica domena kretanje se zaustavlja računajući dužinu puta do izlaska. Proces se ponavlja za svih N tačaka i konačno aproksimira rešenje jednačine. Program se nalazi u datoteci feyman.c u arhivi koja je priložena uz ovaj dokument. Program testirati sa parametrima koji su dati u datoteci run.

* 1. Delovi koje treba paralelizovati
     1. Diskusija

Paralelizacija se vrši u main-u, u kome se tačke generišu i računa dužina puta do izlaska. Kako je problem dosta složen, funkcija sadrži 4 for petelje. Odlučeno je da se paralelizacija radi nad prvom for petljom.

* + 1. Način paralelizacije

U ovom zadatku, izvršena je paralelizacija worksharing direktivom za for petlju. Najbolje performanse dobijene su korišćenjem klauzula (dynamic,1), (static,1), kao i ne stavljanjem nikakve klauzule što predstavlja static blokovsku raspodelu, međutim tradeoff ovog pristupa je preciznost izračunavanja greške. Naime, što algoritam duže računa, to je rezultat približniji rezultatu sekvencijalnog koda. Kako je u tekstu zadatka naznačeno da se odstupanja veličine [-0.01,0.01] zanemaruju, odlučeno je da se pribegne rešenju koje daje što manje vreme izvršavanja. U kodu je ostavljena static blokovska raspodela, kao krajnje rešenje.

* 1. Rezultati

U okviru ove sekcije su izloženi rezultati paralelizacije problema 1.

* + 1. Logovi izvršavanja

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

SEKVENCIJALNO IZVRŠAVANJE

time ./feyman 1000

RMS absolute error in solution = 2.171700e-02

real 3.030s

time ./ feyman 5000

RMS absolute error in solution = 2.127277e-02

real 15.257s

time ./ feyman 10000

RMS absolute error in solution = 2.109998e-02

real 30.220s

time ./ feyman 20000

RMS absolute error in solution = 2.102653e-02

real 1m0.488s

Listing 11. Sekvencijalno izvršavanje problema 3

NUM\_THREADS=8

time ./dz1z3.exe 1000

RMS absolute error in solution = 2.205990e-02

real 0.430s

time ./dz1z3.exe 5000

RMS absolute error in solution = 2.130991e-02

real 2.036s

time ./dz1z3.exe 10000

RMS absolute error in solution = 2.104680e-02

real 3.990s

time ./dz1z3.exe 20000

RMS absolute error in solution = 2.111877e-02

real 9.316s

Listing 12. Izvršavanje problema 3 za N=8 niti

NUM\_THREADS=4

time ./dz1z3.exe 1000

RMS absolute error in solution = 2.249594e-02

real 1.292s

time ./dz1z3.exe 5000

RMS absolute error in solution = 2.102691e-02

real 6.418s

time ./dz1z3.exe 10000

RMS absolute error in solution = 2.105578e-02

real 12.838s

time ./dz1z3.exe 20000

RMS absolute error in solution = 2.106925e-02

real 25.692s

Listing 13. Izvršavanje problema 3 za N=4 niti

NUM\_THREADS=2

time ./dz1z3.exe 1000

RMS absolute error in solution = 2.248462e-02

real 2.417s

time ./dz1z3.exe 5000

RMS absolute error in solution = 2.100388e-02

real 7.769s

time ./dz1z3.exe 10000

RMS absolute error in solution = 2.097929e-02

real 16.264s

time ./dz1z3.exe 20000

RMS absolute error in solution = 2.102237e-02

real 30.675s

Listing 14. Izvršavanje problema 3 za N=2 niti

NUM\_THREADS=1

time ./dz1z3.exe 1000

RMS absolute error in solution = 2.171700e-02

real 3.037s

time ./dz1z3.exe 5000

RMS absolute error in solution = 2.127277e-02

real 15.497s

time ./dz1z3.exe 10000

RMS absolute error in solution = 2.109998e-02

real 30.468s

time ./dz1z3.exe 20000

RMS absolute error in solution = 2.102653e-02

real 1m2.249s

Listing 15. Izvršavanje problema 3 za N=1 niti

* + 1. Grafici ubrzanja

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

Slika9. Grafik zavisnosti ubrzanja izvršavanja problema 3 od ulaznog parametara za N = 8 niti

Slika 10. Grafik zavisnosti ubrzanja izvršavanja problema 3 od ulaznog parametara za N = 4 niti

Slika11. Grafik zavisnosti ubrzanja izvršavanja problema 3 od ulaznog parametara za N = 2 niti

Slika12. Grafik zavisnosti ubrzanja izvršavanja problema 3 od ulaznog parametara za 8, 4 i 2 niti

* + 1. Diskusija dobijenih rezultata

Kada posmatramo dobijena rešenja, vidimo da ubrzanje raste za preko 7 puta ukoliko na raspolaganju imamo 8 niti. Taj broj je nešto manji ukoliko je broj niti 4 i 2, kao što se može videti iz priloženih grafika. Kao što se može primetiti, rezultat greške menja u zavisnoсti od svih parametara, tek postoje mala odstupanja u grešci u sekvencijalnom i u paralelizovanim kodovima. Ovo odstupanje je u opsegu [-0,01, 0,01], što se može smatrati relativno malim odstupanjem i neznačajnim za ovaj problem. I u ovom zadatku, kao i u prethodnom pokazalo se da Open MP directive postižu više nego zadovoljavajuće rezultate.