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. Matija Dodović

Studenti:

Nemanja Mehović 2022/3088

Beograd, april 2023.

SADRŽAJ

1. PROBLEM 1 - PRIME

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, N]

1.2. Delovi koje treba paralelizovati

1.2.1. Diskusija

Postoji jedno mesto u programu koje ima smisla paralelizovati, a to je glavna for petlja u funkciji *prime_number*.

1.2.2. Način paralelizacije

Paralelizacija for petlje je urađena ručno korišćenjem id-eva niti za raspodelu rada. Sama raspodela posla je urađena tako da simulira static raspodelu koju nudi OpenMP, korišćenjem schedule direktive sa veličinom chunk-a da bude jednaka jedan.

1.3. Rezultati

U okviru ove sekcije su izloženi rezultati paralelizacije problema 1.

1.3.1. Logovi izvršavanja

```
SEQUENTIAL EXECUTION
22 April 2023 08:16:41 PM

PRIME TEST

  Call PRIME_NUMBER to count the primes from 1 to N.

         N        Pi          Time

         1         0        0.000003
```

```
         2         1        0.000001
         4         2        0.000001
         8         4        0.000001
        16         6        0.000001
        32        11        0.000002
        64        18        0.000005
       128        31        0.000015
       256        54        0.000050
       512        97        0.000178
      1024       172        0.000623
      2048       309        0.002241
      4096       564        0.001678
      8192      1028        0.004981
     16384      1900        0.018390
     32768      3512        0.067879
     65536      6542        0.253619
    131072     12251        0.954604

PRIME_TEST
  Normal end of execution.

22 April 2023 08:16:42 PM
Execution time for sequential 1.304493 s

PARALLEL EXECUTION
22 April 2023 08:16:42 PM

PRIME TEST

  Call PRIME_NUMBER to count the primes from 1 to N.

         N        Pi          Time

         1         0        0.000126
         2         1        0.000002
         4         2        0.000001
         8         4        0.000001
        16         6        0.000001
        32        11        0.000001
```

```
        64         18        0.000002
       128         31        0.000002
       256         54        0.000003
       512         97        0.000010
      1024        172        0.000031
      2048        309        0.000109
      4096        564        0.000383
      8192       1028        0.012727
     16384       1900        0.033007
     32768       3512        0.109096
     65536       6542        0.313004
    131072      12251        1.032983

PRIME_TEST
  Normal end of execution.

22 April 2023 08:16:43 PM
Execution time for parallel 0.339479 s

Test PASSED

SEQUENTIAL EXECUTION
22 April 2023 08:16:43 PM

PRIME TEST

  Call PRIME_NUMBER to count the primes from 1 to N.

         N         Pi          Time

         5          3        0.000001
        50         15        0.000000
       500         95        0.000029
      5000        669        0.001987
     50000       5133        0.152023
    500000      41538       12.409858

PRIME_TEST
  Normal end of execution.
```

```
22 April 2023 08:16:55 PM
Execution time for sequential 12.564042 s

PARALLEL EXECUTION
22 April 2023 08:16:55 PM

PRIME TEST

  Call PRIME_NUMBER to count the primes from 1 to N.

         N        Pi          Time

         5         3        0.000247
        50        15        0.000002
       500        95        0.000010
      5000       669        0.000555
     50000      5133        0.192360
    500000     41538       12.768318

PRIME_TEST
  Normal end of execution.

22 April 2023 08:16:58 PM
Execution time for parallel 3.227136 s

Test PASSED

SEQUENTIAL EXECUTION
22 April 2023 08:16:58 PM

PRIME TEST

  Call PRIME_NUMBER to count the primes from 1 to N.

         N        Pi          Time

         1         0        0.000000
         4         2        0.000000
```

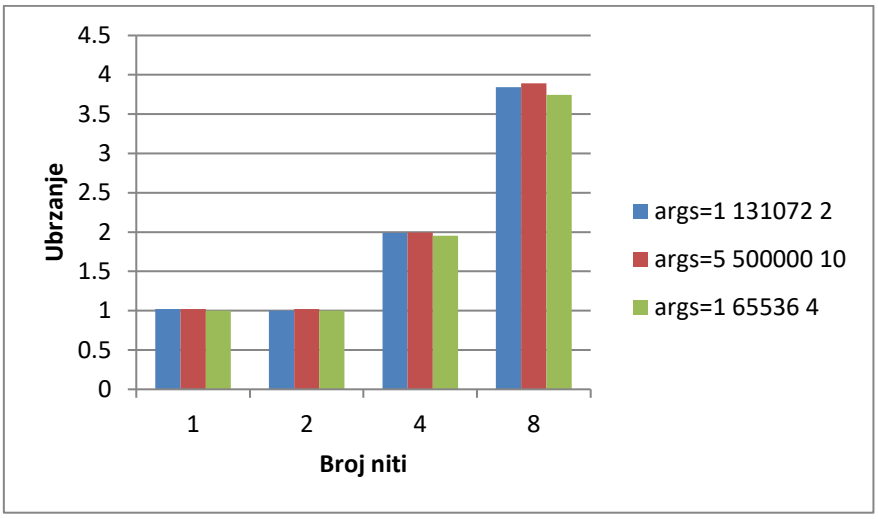
```
        16          6        0.000000
        64         18        0.000001
       256         54        0.000009
      1024        172        0.000105
      4096        564        0.001386
     16384       1900        0.018080
     65536       6542        0.248595

PRIME_TEST
  Normal end of execution.

22 April 2023 08:16:59 PM
Execution time for sequential 0.268212 s

PARALLEL EXECUTION
22 April 2023 08:16:59 PM

PRIME TEST

  Call PRIME_NUMBER to count the primes from 1 to N.

         N         Pi          Time

         1          0        0.000116
         4          2        0.000001
        16          6        0.000001
        64         18        0.000001
       256         54        0.000003
      1024        172        0.000030
      4096        564        0.000382
     16384       1900        0.042718
     65536       6542        0.297237

PRIME_TEST
  Normal end of execution.

22 April 2023 08:16:59 PM
Execution time for parallel 0.071591 s
```

```
Test PASSED
```

Listing 1. Log izvršavanja svih test primera za problem 1

1.3.2. Grafici ubrzanja

U okviru ove sekcije je dat grafik ubrzanja u odnosu na sekvencijalnu implementaciju.



Slika 1. Grafik zavisnosti ubrzanja u odnosu na broj niti i argumenta

1.3.3. Diskusija dobijenih rezultata

Paralelizaciom programa sa većim brojem niti dobili smo željeno ubrzanje, dok izvršavanje programa korišćenjem jedne ili dve niti ima približno isto vreme izvršavanja kao i sekvencijalna verzija problema. Razlog za ovo jeste implementacija same paralelizacije i priroda samog problema. Odlukom da raspodelu posla vršimo statički svakoj niti, korišćenjem id-a, je dovelo do toga da je jednak broj iteracija petlje raspoređen svakoj niti, ali ovo ne znači da je posao koji obavlja svaka nit jednak. Naručito zato što nam se problem glavnim delom bavi proverom da li je određeni broj prost, što za same proste brojeve može oduzeti dosta vremena. Samim tim može doći do situacije da jedna nit obavlja većinu posla što dovodi do vremena izvršavanja sličnoj sekvencijalnoj implementaciji.

2. PROBLEM 2 - PRIME

2.1. Tekst problema

Prethodni program paralelizovati korišćenjem direktiva za podelu posla (worksharing direktive). Program testirati sa parametrima koji su dati u datoteci run. [1, N]

2.2. Delovi koje treba paralelizovati

2.2.1. Diskusija

Problem koji trebamo paralelizovati je isti kao i prethodni, samim tim jedino mesto koje idalje ima smisla paralelizovati jeste glavna for petlja u funkciji *prime_number*.

2.2.2. Način paralelizacije

Paralelizacija for petlje u ovom problemu je urađena korišćenjem workshare(for) direktiva OpenMp-a kao i dinamičkom raspodelom posla koju imamo mogućnost da koristimo zbog schedule direktive. Prilikom same paralelizacije, testirani su različiti načini raspodele posla i dinamička raspodela se pokazala kao najbrža, samim tim smo i nju izabrali.

2.3. Rezultati

U okviru ove sekcije su izloženi rezultati paralelizacije problema 2.

2.3.1. Logovi izvršavanja

```
SEQUENTIAL EXECUTION
22 April 2023 08:15:48 PM

PRIME TEST

  Call PRIME_NUMBER to count the primes from 1 to N.

         N        Pi          Time

         1         0        0.000003
         2         1        0.000001
         4         2        0.000001
         8         4        0.000001
        16         6        0.000001
```

```
        32        11        0.000003
        64        18        0.000005
       128        31        0.000014
       256        54        0.000049
       512        97        0.000178
      1024       172        0.000621
      2048       309        0.002232
      4096       564        0.001780
      8192      1028        0.004977
     16384      1900        0.018383
     32768      3512        0.067879
     65536      6542        0.253668
    131072     12251        0.954482

PRIME_TEST
  Normal end of execution.

22 April 2023 08:15:49 PM
Execution time for sequential 1.304488 s

PARALLEL EXECUTION
22 April 2023 08:15:49 PM

PRIME TEST

  Call PRIME_NUMBER to count the primes from 1 to N.

         N        Pi          Time

         1         0        0.000132
         2         1        0.000002
         4         2        0.000001
         8         4        0.000002
        16         6        0.000002
        32        11        0.000002
        64        18        0.000003
       128        31        0.000005
       256        54        0.000007
       512        97        0.000013
```

```
      1024        172        0.000028
      2048        309        0.000072
      4096        564        0.000231
      8192       1028        0.000786
     16384       1900        0.019729
     32768       3512        0.094476
     65536       6542        0.290857
    131072      12251        1.056093

PRIME_TEST
  Normal end of execution.

22 April 2023 08:15:49 PM
Execution time for parallel 0.185690 s

Test PASSED

SEQUENTIAL EXECUTION
22 April 2023 08:15:49 PM

PRIME TEST

  Call PRIME_NUMBER to count the primes from 1 to N.

         N         Pi          Time

         5          3        0.000001
        50         15        0.000001
       500         95        0.000032
      5000        669        0.002007
     50000       5133        0.151877
    500000      41538       12.409987

PRIME_TEST
  Normal end of execution.

22 April 2023 08:16:02 PM
Execution time for sequential 12.564178 s
```

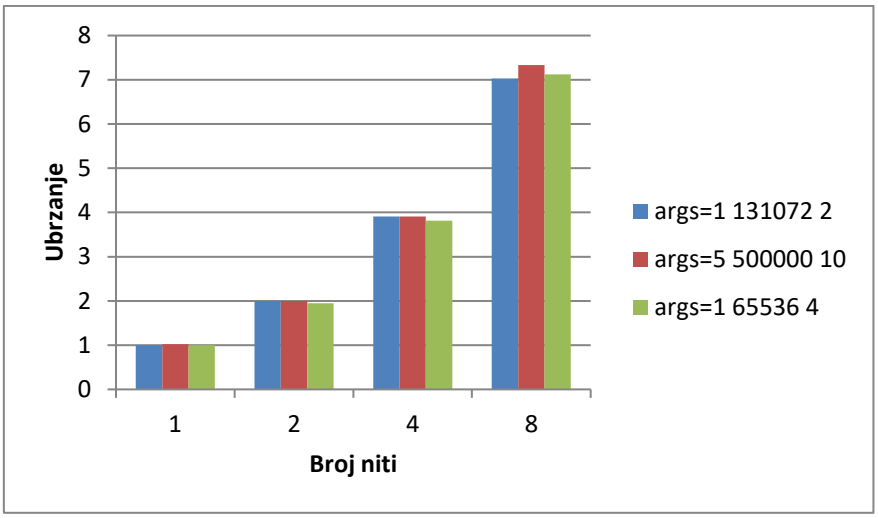
```
PARALLEL EXECUTION
22 April 2023 08:16:02 PM

PRIME TEST

  Call PRIME_NUMBER to count the primes from 1 to N.

         N        Pi          Time

         5         3        0.000177
        50        15        0.000003
       500        95        0.000014
      5000       669        0.000291
     50000      5133        0.160432
    500000     41538       13.536444

PRIME_TEST
  Normal end of execution.

22 April 2023 08:16:04 PM
Execution time for parallel 1.713902 s

Test PASSED

SEQUENTIAL EXECUTION
22 April 2023 08:16:04 PM

PRIME TEST

  Call PRIME_NUMBER to count the primes from 1 to N.

         N        Pi          Time

         1         0        0.000001
         4         2        0.000000
        16         6        0.000001
        64        18        0.000000
       256        54        0.000009
      1024       172        0.000109
```

```
      4096       564        0.001399
     16384      1900        0.018405
     65536      6542        0.253714

PRIME_TEST
  Normal end of execution.

22 April 2023 08:16:04 PM
Execution time for sequential 0.273671 s

PARALLEL EXECUTION
22 April 2023 08:16:04 PM

PRIME TEST

  Call PRIME_NUMBER to count the primes from 1 to N.

         N        Pi          Time

         1         0        0.000170
         4         2        0.000002
        16         6        0.000003
        64        18        0.000003
       256        54        0.000008
      1024       172        0.000026
      4096       564        0.000205
     16384      1900        0.002542
     65536      6542        0.284620

PRIME_TEST
  Normal end of execution.

22 April 2023 08:16:04 PM
Execution time for parallel 0.037675 s

Test PASSED
```

Listing 2. Log izvršavanja svih test primera za problem 2

2.3.2. Grafici ubrzanja

U okviru ove sekcije je dat grafik ubrzanja u odnosu na sekvencijalnu implementaciju.



Slika 2. Grafik zavisnosti ubrzanja u odnosu na broj niti i argumenta

2.3.3. Diskusija dobijenih rezultata

Pošto je ovaj problem isti kao i prvi problem koji smo rešavali sve što smo izajvili za njega važi i za ovaj problem. Razlika u ubrzanju kod ove implementacije paralelizacije i implementacije u prethodnom problemu jeste zbog različitog načina raspodele posla. U ovom rešenju koristilimo dinamičku raspodelu posla. To nam je omogućilo da svaka nit izvršava sličnu količinu posla čak iako svaka nit obrađuje različit broj iteracija petlje, samim tim brzina izvršavanja je približno za broj niti puta brža u odnosu na sekvencijalnu implementaciju.

3. PROBLEM 3 - FEYMAN

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,N]

3.2. Delovi koje treba paralelizovati

3.2.1. Diskusija

Za razliku od prethodna dva problema gde je mogućnost paralelizacije bila samo na jednom mestu kod ovog problema imamo četri mesta u kodu koja je moguće paralelizovati. Te lokacije su ugnježdene for i while petlje u main funkciji. Za paralelizaciju izabrao sam unutrašnju for petlju zato sto odrađuje najveći deo posla i jedina je sa nepoznatim brojem iteracija.

3.2.2. Način paralelizacije

Paralelizacija je implementirana korišćenjem OpenMP for direktive sa dinamičkim raspodelom posla. Prilikom paralelizacije je vođeno posbno računa da se nasumičcan deo algoritma idalje ostvari. Ovo je postignuto tako što je seed za svaku nit bio promenjen, korišćenjem id-a niti i samo se jedan seed čuvao na kraju izvršavanja for petlje za nastavak korišćenja u programu, što je urađeno korišćenjem lastprivate direktive.

3.3. Rezultati

U okviru ove sekcije su izloženi rezultati paralelizacije problema 3.

3.3.1. Logovi izvršavanja

```
SEQUENTIAL EXECUTION
22 April 2023 08:17:49 PM
A = 3.000000
B = 2.000000
C = 1.000000
N = 1000
```

```
H = 0.0010


RMS absolute error in solution = 2.171700e-02
22 April 2023 08:17:52 PM
Execution time for sequential 2.948687 s

PARALLEL EXECUTION
22 April 2023 08:17:52 PM
A = 3.000000
B = 2.000000
C = 1.000000
N = 1000
H = 0.0010


RMS absolute error in solution = 2.163099e-02
22 April 2023 08:17:53 PM
Execution time for parallel 0.438024 s

Test PASSED

SEQUENTIAL EXECUTION
22 April 2023 08:17:53 PM
A = 3.000000
B = 2.000000
C = 1.000000
N = 5000
H = 0.0010


RMS absolute error in solution = 2.127277e-02
22 April 2023 08:18:08 PM
Execution time for sequential 15.080766 s

PARALLEL EXECUTION
22 April 2023 08:18:08 PM
A = 3.000000
B = 2.000000
```

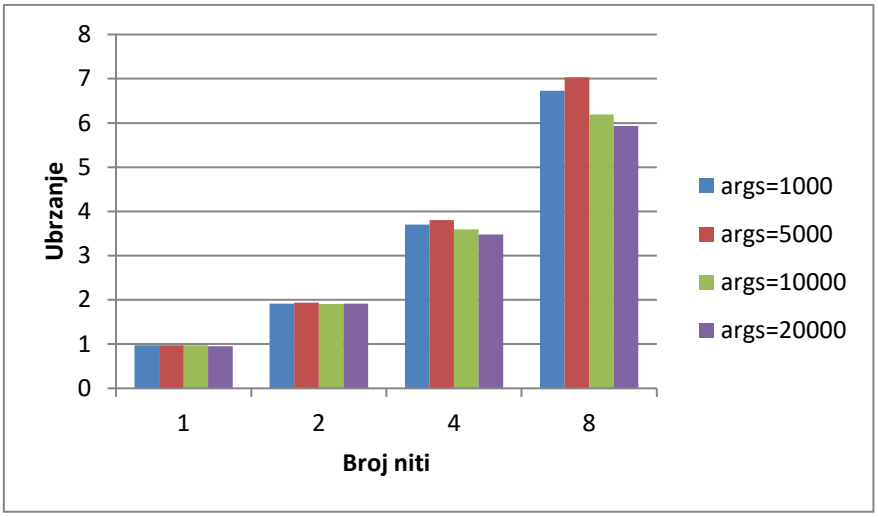
```
C = 1.000000
N = 5000
H = 0.0010


RMS absolute error in solution = 2.137064e-02
22 April 2023 08:18:10 PM
Execution time for parallel 2.143891 s

Test PASSED

SEQUENTIAL EXECUTION
22 April 2023 08:18:10 PM
A = 3.000000
B = 2.000000
C = 1.000000
N = 10000
H = 0.0010


RMS absolute error in solution = 2.109998e-02
22 April 2023 08:18:40 PM
Execution time for sequential 29.547789 s

PARALLEL EXECUTION
22 April 2023 08:18:40 PM
A = 3.000000
B = 2.000000
C = 1.000000
N = 10000
H = 0.0010


RMS absolute error in solution = 2.115149e-02
22 April 2023 08:18:44 PM
Execution time for parallel 4.774195 s

Test PASSED
```

```
SEQUENTIAL EXECUTION
22 April 2023 08:18:44 PM
A = 3.000000
B = 2.000000
C = 1.000000
N = 20000
H = 0.0010


RMS absolute error in solution = 2.102653e-02
22 April 2023 08:19:43 PM
Execution time for sequential 59.039446 s

PARALLEL EXECUTION
22 April 2023 08:19:43 PM
A = 3.000000
B = 2.000000
C = 1.000000
N = 20000
H = 0.0010


RMS absolute error in solution = 2.097130e-02
22 April 2023 08:19:53 PM
Execution time for parallel 9.952296 s

Test PASSED
```

Listing 3. Log izvršavanja svih test primera za problem 3

3.3.2. Grafici ubrzanja

U okviru ove sekcije je dat grafik ubrzanja u odnosu na sekvencijalnu implementaciju.



Slika 3. Grafik zavisnosti ubrzanja u odnosu na broj niti i argumenta

3.3.3. Diskusija dobijenih rezultata

Razultati dobijeni paralelizaciom programa su onakvi kakve smo i očekivali da dobijemo, povećanjem broja niti brzina izvršavanja samog programa se poveća. Takođe možemo primetiti da ubrzanje nije proporcialno u odnosu na količinu niti i da paralelizam najbolje radi sa parametrima koji su manji od deset hiljada. Sam problem nije idealano implementiran za paralelizaciju zbog ručno napravljene funkcije za generisanje nasumičnih brojeva.

4. PROBLEM 4 - FEYMAN

4.1. Tekst problema

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

4.2. Delovi koje treba paralelizovati

4.2.1. Diskusija

Ovaj problem predstavlja nadogradnju trećeg problema samim tim moguće lokacije paralelizacije ostaju iste. Zbog zahteva samog zadatka deo koda koji je paralelizovan je sada promenjen i izabrana je prva for petlja kao početak paralelizacije.

4.2.2. Način paralelizacije

Sam zadatak od nas je tražio da problem paralelizujemo korišćenjem poslova iz OpenMP. Jedan task smo izabrali da bude četvrta ugnježdena for petlja u main programu. Dok same poslove će generisati posebna nit koja počinje sa prvom for petljom. Kao i u prethodnom problemu vođeno je računa da nasumičan deo algoritma idalje ostane nasumičan.

4.3. Rezultati

U okviru ove sekcije su izloženi rezultati paralelizacije problema 4.

4.3.1. Logovi izvršavanja

```
SEQUENTIAL EXECUTION
22 April 2023 08:39:43 PM
A = 3.000000
B = 2.000000
C = 1.000000
N = 1000
H = 0.0010



RMS absolute error in solution = 2.171700e-02
22 April 2023 08:39:46 PM
Execution time for sequential 3.005161 s
```

```
PARALLEL EXECUTION
22 April 2023 08:39:46 PM
A = 3.000000
B = 2.000000
C = 1.000000
N = 1000
H = 0.0010


RMS absolute error in solution = 2.246478e-02
22 April 2023 08:39:47 PM
Execution time for parallel 0.426049 s

Test PASSED

SEQUENTIAL EXECUTION
22 April 2023 08:39:47 PM
A = 3.000000
B = 2.000000
C = 1.000000
N = 5000
H = 0.0010


RMS absolute error in solution = 2.127277e-02
22 April 2023 08:40:02 PM
Execution time for sequential 15.046015 s

PARALLEL EXECUTION
22 April 2023 08:40:02 PM
A = 3.000000
B = 2.000000
C = 1.000000
N = 5000
H = 0.0010


RMS absolute error in solution = 2.116865e-02
```

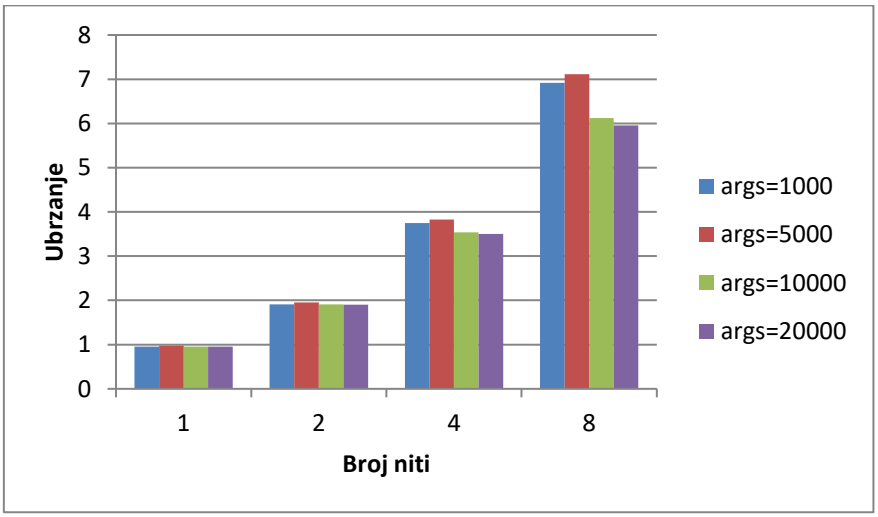
```
22 April 2023 08:40:04 PM
Execution time for parallel 2.119980 s

Test PASSED

SEQUENTIAL EXECUTION
22 April 2023 08:40:04 PM
A = 3.000000
B = 2.000000
C = 1.000000
N = 10000
H = 0.0010


RMS absolute error in solution = 2.109998e-02
22 April 2023 08:40:33 PM
Execution time for sequential 29.452456 s

PARALLEL EXECUTION
22 April 2023 08:40:33 PM
A = 3.000000
B = 2.000000
C = 1.000000
N = 10000
H = 0.0010


RMS absolute error in solution = 2.087167e-02
22 April 2023 08:40:38 PM
Execution time for parallel 4.827917 s

Test PASSED

SEQUENTIAL EXECUTION
22 April 2023 08:40:38 PM
A = 3.000000
B = 2.000000
C = 1.000000
N = 20000
```

```
H = 0.0010


RMS absolute error in solution = 2.102653e-02
22 April 2023 08:41:38 PM
Execution time for sequential 60.143724 s

PARALLEL EXECUTION
22 April 2023 08:41:38 PM
A = 3.000000
B = 2.000000
C = 1.000000
N = 20000
H = 0.0010


RMS absolute error in solution = 2.091587e-02
22 April 2023 08:41:48 PM
Execution time for parallel 9.913756 s

Test PASSED
```

Listing 4. Log izvršavanja svih test primera za problem 4

4.3.2. Grafici ubrzanja

U okviru ove sekcije je dat grafik ubrzanja u odnosu na sekvencijalnu implementaciju.



Slika 4. Grafik zavisnosti ubrzanja u odnosu na broj niti i argumenta

4.3.3. Diskusija dobijenih rezultata

Možemo videti da uvođenje poslovao kao metod paralelizacije nije dovelo do pogoršanih rezultata u odnosu na regularnu paralelizaciju. Pored toga sve što je rečeno u rezultatima prethodnog problema važi i ovde.

5. PROBLEM 5 - MOLDYN

5.1. Tekst problema

Paralelizovati jednostavan program koji se bavi molekularnom dinamikom. Kod predstavlja simulaciju molekularne dinamike argonovog atoma u ograničenom prozoru (prostoru) sa periodičnim graničnim uslovima. Atomi se inicijalno nalaze raspoređeni u pravilnu mrežu, a zatim se tokom simulacije dešavaju interakcije između njih. U svakom koraku simulacije u glavnoj petlji se dešava sledeće:

- Čestice (atomi) se pomeraju zavisno od njihovih brzina i brzine se parcijalno ažuriraju u pozivu funkcije domove.
- Sile koje se primenjuju na nove pozicije čestica se izračunavaju; takođe, akumuliraju se prosečna kinetička energija (virial) i potencijalna energija u pozivu funkcije forces.
- Sile se skaliraju, završava ažuriranje brzine i izračunavanje kinetičke energije u pozivu funkcije mkekin.
- Prosečna brzina čestice se računa i skaliraju temperature u pozivu funkcije velavg.
- Pune potencijalne i prosečne kinetičke energije (virial) se računaju i ispisuju u funkciji prnout.

Program se nalazi u datoteci direktorijumu MolDyn u arhivi koja je priložena uz ovaj dokument. Program se sastoji od više datoteka, od kojih su od interesa datoteke main.c i forces.c, jer se u njima provodi najviše vremena. Analizirati dati kod i obratiti pažnju na redukcione promenljive unutar datoteke forces.c. Ukoliko je potrebno međusobno isključenje prilikom paralelizacije programa, koristiti kritične sekcije ili atomske operacije. [1, N]

5.2. Delovi koje treba paralelizovati

5.2.1. Diskusija

Problem sadrži prilično veliku količinu lokacija koje je moguće paralelizovati nezavisno jedan od drugog i koje po zahtevima samog zadatka trebamo paralelizovati, ali glavni deo paralelizma se nalazi u funkciji forces koja nas košta najveću količinu vremena prilikom izvršavanja.

5.2.2. Način paralelizacije

Svi oblici paralelizacije su implementirani korišćenjem for direktive iz OpenMp. Prilikom paralelizacije potrebno je bilo obratiti pažnju na globalne promenljive koje se menjaju kao i elemente niza. Problem sa globalnim promenljivama se lako rešio koristeći reduction direktive iz OpenMp, dok kod menjanja elementa niza morali smo da uvedemo vid sinhronizacije korišćenjem atomic operacija.

5.3. Rezultati

U okviru ove sekcije su izloženi rezultati paralelizacije problema 5.

5.3.1. Logovi izvršavanja

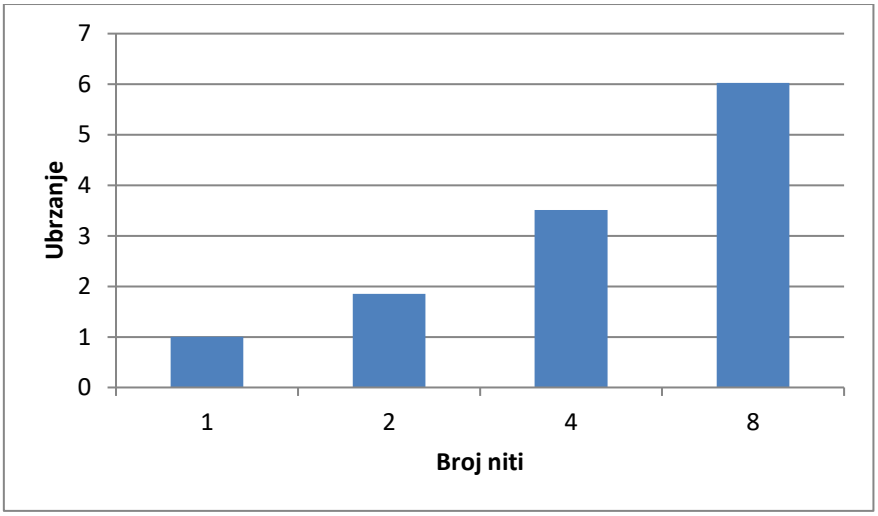
```
SEQUENTIAL EXECUTION
 Molecular Dynamics Simulation example program
 ---------------------------------------------
 number of particles is .............  13500
 side length of the box is .........     25.323179
 cut off is ........................      3.750000
 reduced temperature is .............      0.722000
 basic timestep is ..................      0.064000
 temperature scale interval .........     10
 stop scaling at move ...............     20
 print interval .....................      5
 total no. of steps .................     20

    i       ke         pe             e          temp      pres      vel      rp
  -----  ----------  ----------  ----------  --------  --------  --------  ----
      5  12619.1758 -91985.3542 -79366.1784    0.6232   -5.2880    0.1821  39.7
     10  14619.4170 -86181.5919 -71562.1749    0.7220   -2.8265    0.1336  14.1
     15  11405.1707 -82966.3254 -71561.1547    0.5633   -1.5094    0.1714  33.6
     20  10825.0423 -82385.8646 -71560.8222    0.5346   -1.2219    0.1679  32.2
Time =  5.773351
PARALLEL EXECUTION
 Molecular Dynamics Simulation example program
 ---------------------------------------------
 number of particles is .............  13500
 side length of the box is .........     25.323179
 cut off is ........................      3.750000
```

```
 reduced temperature is ...........      0.722000
 basic timestep is ................      0.064000
 temperature scale interval ........     10
 stop scaling at move ..............     20
 print interval ....................      5
 total no. of steps ................     20

    i        ke          pe             e          temp      pres      vel       rp
 -----  ----------  ----------  ----------  --------  --------  --------  ----
     5  12619.1758 -91985.3542 -79366.1784    0.6232   -5.2880    0.1821  39.7
    10  14619.4170 -86181.5919 -71562.1749    0.7220   -2.8265    0.1336  14.1
    15  11405.1707 -82966.3254 -71561.1547    0.5633   -1.5094    0.1714  33.6
    20  10825.0423 -82385.8646 -71560.8222    0.5346   -1.2219    0.1679  32.2
Time =  0.957988
Test PASSED
```

Listing 5. Log izvršavanja problema 5

5.3.2. Grafici ubrzanja

U okviru ove sekcije je dat grafik ubrzanja u odnosu na sekvencijalnu implementaciju.



Slika 5. Grafik zavisnosti ubrzanja u odnosu na broj niti

5.3.3. Diskusija dobijenih rezultata

Možemo videti da se povećanjem broja niti povećava i brzina izvršavanja samog programa, aku nije proporcionalno boju niti. Razlog za ovo je uvođenje atomic operacija koje su nam neophodne za tačnost programa ali usporavaju sam program zbog sinhronizacije. Takođe sama sinhronizacija će više usporavati program što više niti imam što se može i videti na grafu Slika 5.