UNIVERZITET U BEOGRADU - ELEKTROTEHNIČKI FAKULTET

MULTIPROCESORKI SISTEMI (13S114MUPS, 13E114MUPS)

DOMAĆI ZADATAK 2 – MPI

Izveštaj o urađenom domaćem zadatku

Predmetni saradnici:

doc. dr Marko Mišić

dipl. ing. Matija Dodović

Studenti:

Nemanja Mehović 2022/3088

Beograd, maj 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. 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

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 koristeći rutina za kolektivnu komunikaciju. Ovo je urađeno tako što master proces prosledi učitane podatke ostalim procesima, nakon čega počinje rad na samom problemu. Svaki proces za sebe određuje koji deo posla će obaviti, koristeći svoj rank u komunikaciji za izbor posla. Nakon samo obrade koristi se mpi funkcija MPI_Reduce za dobijanje samog rezultata.

1.3. Rezultati

U okviru ove sekcije su izloženi rezultati paralelizacije problema 1.

1.3.1. Logovi izvršavanja

```
SEQUENTIAL EXECUTION
08 May 2023 10:41:54 AM

PRIME TEST

  Call PRIME_NUMBER to count the primes from 1 to N.

         N        Pi          Time

         1         0        0.000000
```

```
         2         1        0.000000
         4         2        0.000000
         8         4        0.000001
        16         6        0.000000
        32        11        0.000000
        64        18        0.000000
       128        31        0.000003
       256        54        0.000009
       512        97        0.000033
      1024       172        0.000117
      2048       309        0.000398
      4096       564        0.001456
      8192      1028        0.005310
     16384      1900        0.019596
     32768      3512        0.072483
     65536      6542        0.275159
    131072     12251        1.040566

PRIME_TEST
  Normal end of execution.

08 May 2023 10:41:55 AM
Execution time for sequential 1.415280 s

PARALLEL EXECUTION
08 May 2023 10:41:55 AM

PRIME TEST

  Call PRIME_NUMBER to count the primes from 1 to N.

         N        Pi          Time

         1         0        0.000048
         2         1        0.000003
         4         2        0.000003
         8         4        0.000110
        16         6        0.000004
        32        11        0.000002
```

```
        64         18        0.000003
       128         31        0.000001
       256         54        0.000003
       512         97        0.000008
      1024        172        0.000031
      2048        309        0.000108
      4096        564        0.000389
      8192       1028        0.001343
     16384       1900        0.005043
     32768       3512        0.018822
     65536       6542        0.069432
    131072      12251        0.264521

PRIME_TEST
  Normal end of execution.

08 May 2023 10:41:56 AM
Execution time for parallel 0.359924 s

Test PASSED

SEQUENTIAL EXECUTION
08 May 2023 10:41:56 AM

PRIME TEST

  Call PRIME_NUMBER to count the primes from 1 to N.

         N         Pi          Time

         5          3        0.000000
        50         15        0.000001
       500         95        0.000031
      5000        669        0.002107
     50000       5133        0.158728
    500000      41538       15.627756

PRIME_TEST
  Normal end of execution.
```

```
08 May 2023 10:42:12 AM
Execution time for sequential 15.788736 s

PARALLEL EXECUTION
08 May 2023 10:42:12 AM

PRIME TEST

  Call PRIME_NUMBER to count the primes from 1 to N.

         N        Pi          Time

         5         3        0.000073
        50        15        0.000004
       500        95        0.000003
      5000       669        0.000756
     50000      5133        0.047425
    500000     41538        3.861803

PRIME_TEST
  Normal end of execution.

08 May 2023 10:42:16 AM
Execution time for parallel 3.910140 s

Test PASSED

SEQUENTIAL EXECUTION
08 May 2023 10:42:16 AM

PRIME TEST

  Call PRIME_NUMBER to count the primes from 1 to N.

         N        Pi          Time

         1         0        0.000001
         4         2        0.000000
```

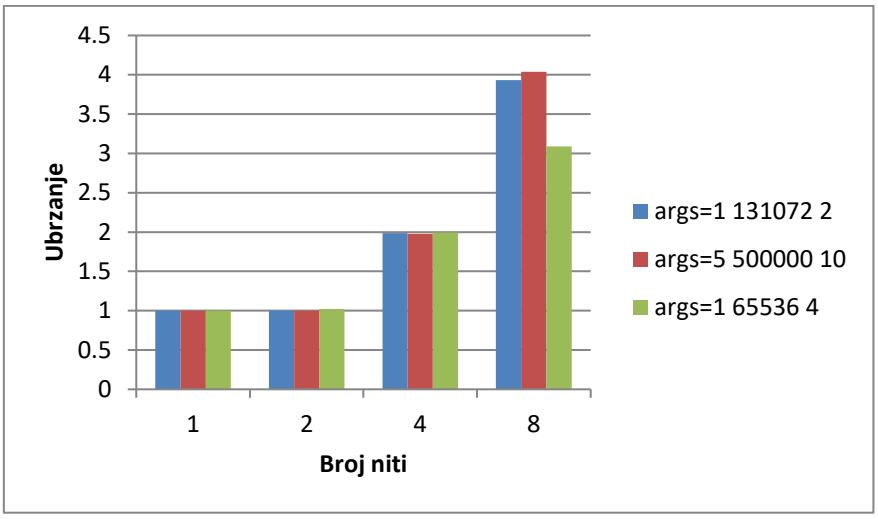
```
        16          6        0.000000
        64         18        0.000002
       256         54        0.000019
      1024        172        0.000227
      4096        564        0.002759
     16384       1900        0.025731
     65536       6542        0.274521

PRIME_TEST
  Normal end of execution.

08 May 2023 10:42:16 AM
Execution time for sequential 0.303482 s

PARALLEL EXECUTION
08 May 2023 10:42:16 AM

PRIME TEST

  Call PRIME_NUMBER to count the primes from 1 to N.

         N         Pi          Time

         1          0        0.000049
         4          2        0.000005
        16          6        0.000002
        64         18        0.000127
       256         54        0.000002
      1024        172        0.000061
      4096        564        0.000868
     16384       1900        0.011754
     65536       6542        0.085355

PRIME_TEST
  Normal end of execution.

08 May 2023 10:42:16 AM
Execution time for parallel 0.098258 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 procesa dobili smo željeno ubrzanje, dok izvršavanje programa korišćenjem jednog ili dva procesa 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, korišćenjem ranka, je dovelo do toga da je jednak broj iteracija petlje raspoređen svakom procesu, ali ovo ne znači da je posao koji obavlja svaki proces 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 jedan proces obavlja većinu posla što dovodi do vremena izvršavanja sličnoj sekvencijalnoj implementaciji.

2. PROBLEM 2 - FEYMAN

2.1. 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]

2.2. Delovi koje treba paralelizovati

2.2.1. 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 što obrađuje najveći deo posla i jedina je sa nepoznatim brojem iteracija.

2.2.2. Način paralelizacije

Paralelizacija je implementirana na sličan način kao i u prethodnom problemu, master proces prosleđuje neophodne podatke ostalim procesima, nakon čega počinje obrada. Prilikom paralelizacije je vođeno posbno računa da se nasumičan deo algoritma idalje ostvari. Ovo je postignuto tako što je seed za svaki proces bio promenjen, korišćenjem ranka procesa.

2.3. Rezultati

U okviru ove sekcije su izloženi rezultati paralelizacije problema 2.

2.3.1. Logovi izvršavanja

```
SEQUENTIAL EXECUTION
08 May 2023 10:44:18 AM
A = 3.000000
B = 2.000000
C = 1.000000
N = 1000
```

```
H = 0.0010


RMS absolute error in solution = 2.171700e-02
08 May 2023 10:44:21 AM
Execution time for sequential 3.299993 s

PARALLEL EXECUTION
08 May 2023 10:44:21 AM
A = 3.000000
B = 2.000000
C = 1.000000
N = 1000
H = 0.0010


RMS absolute error in solution = 2.280089e-02
08 May 2023 10:44:21 AM
Execution time for parallel 0.465143 s

Test PASSED

SEQUENTIAL EXECUTION
08 May 2023 10:44:22 AM
A = 3.000000
B = 2.000000
C = 1.000000
N = 5000
H = 0.0010


RMS absolute error in solution = 2.127277e-02
08 May 2023 10:44:42 AM
Execution time for sequential 20.066761 s

PARALLEL EXECUTION
08 May 2023 10:44:42 AM
A = 3.000000
B = 2.000000
```

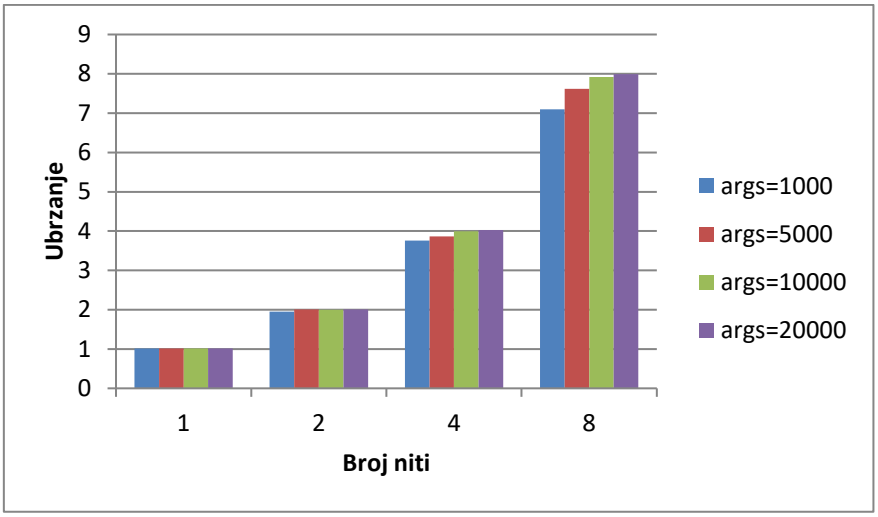
```
C = 1.000000
N = 5000
H = 0.0010


RMS absolute error in solution = 2.109291e-02
08 May 2023 10:44:44 AM
Execution time for parallel 2.635296 s

Test PASSED

SEQUENTIAL EXECUTION
08 May 2023 10:44:45 AM
A = 3.000000
B = 2.000000
C = 1.000000
N = 10000
H = 0.0010


RMS absolute error in solution = 2.109998e-02
08 May 2023 10:45:26 AM
Execution time for sequential 41.194665 s

PARALLEL EXECUTION
08 May 2023 10:45:26 AM
A = 3.000000
B = 2.000000
C = 1.000000
N = 10000
H = 0.0010


RMS absolute error in solution = 2.084098e-02
08 May 2023 10:45:31 AM
Execution time for parallel 5.201354 s

Test PASSED
```

```
SEQUENTIAL EXECUTION
08 May 2023 10:45:31 AM
A = 3.000000
B = 2.000000
C = 1.000000
N = 20000
H = 0.0010


RMS absolute error in solution = 2.102653e-02
08 May 2023 10:46:54 AM
Execution time for sequential 82.668168 s

PARALLEL EXECUTION
08 May 2023 10:46:54 AM
A = 3.000000
B = 2.000000
C = 1.000000
N = 20000
H = 0.0010


RMS absolute error in solution = 2.089977e-02
08 May 2023 10:47:04 AM
Execution time for parallel 10.348239 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

Razultati dobijeni paralelizaciom programa su onakvi kakve smo i očekivali da dobijemo, povećanjem broja procesa brzina izvršavanja samog programa se povećava. Sam problem nije idealano implementiran za paralelizaciju zbog ručno napravljene funkcije za generisanje nasumičnih brojeva.

3. PROBLEM 3 - MOLDYN

3.1. 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]

3.2. Delovi koje treba paralelizovati

3.2.1. Diskusija

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

3.2.2. Način paralelizacije

Paralelizacija je urađena koristeći rutina za kolektivnu komunikaciju. Ovo je urađeno tako što master proces prosledi učitane podatke ostalim procesima, nakon čega počinje rad na samom problemu. Svaki proces za sebe određuje koji deo posla će obaviti, koristeći svoj rank u komunikaciji za izbor posla. Nakon samo obrade koristi se mpi funkcija MPI_Reduce za dobijanje samog rezultata.

3.3. Rezultati

U okviru ove sekcije su izloženi rezultati paralelizacije problema 3.

3.3.1. Logovi izvršavanja

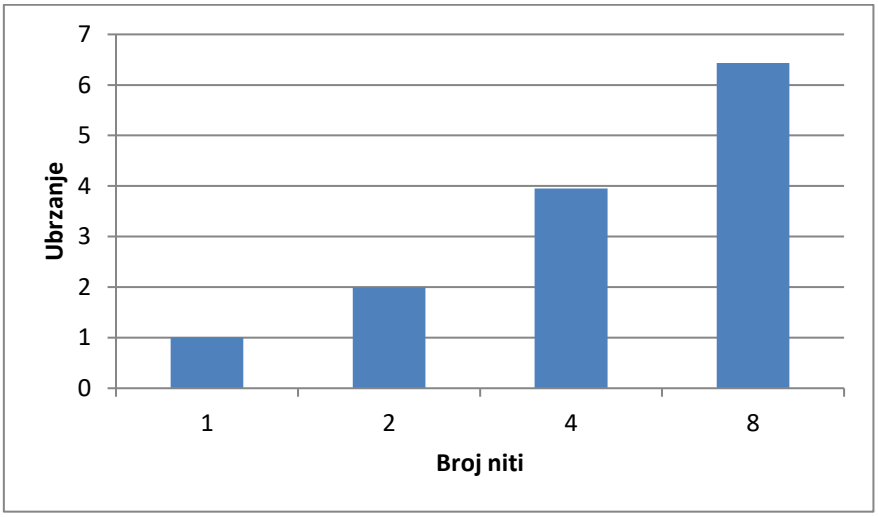
```
SEQUENTIAL EXECUTION
 Molecular Dynamics Simulation example program
 ---------------------------------------------
 number of particles is .............  13500
 side length of the box is .........     25.323179
 cut off is ........................      3.750000
 reduced temperature is .............      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 =  6.721519
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 =  1.044698
Test PASSED
```

Listing 3. Log izvršavanja problema 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

3.3.3. Diskusija dobijenih rezultata

Možemo videti da se povećanjem broja procesa povećava i brzina izvršavanja samog programa, ali možemo videti da ubrzanje nije proporcionalno količini procesa i da nakon četri procesa ubrzanje se uvećava za manju količinu.

4. PROBLEM 4 - MOLDYN

4.1. 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. Način pokretanja programa se nalazi u datoteci run. [1, N]

4.2. Delovi koje treba paralelizovati

4.2.1. Diskusija

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

4.2.2. Način paralelizacije

Paralelizacija je urađena po zahtevima zadatka korišćenjem manager – worker modela. Sam manager radnicima prosleđuje neophodne podatke za rad nakon čega im i pošalje jednu jedinicu posla da obrade. Za jedinicu posla izabrana je jedna iteracija petlje. Radnici kada završe sa poslom prosleđuju rezultate masteru i čekaju na dalje instrukcije.

4.3. Rezultati

U okviru ove sekcije su izloženi rezultati paralelizacije problema 4.

4.3.1. Logovi izvršavanja

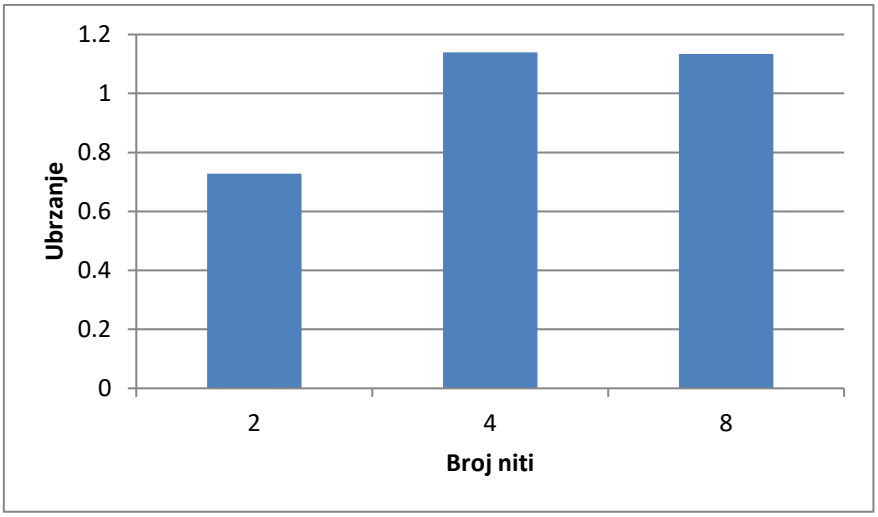
```
SEQUENTIAL EXECUTION
 Molecular Dynamics Simulation example program
 ---------------------------------------------
 number of particles is .............  13500
 side length of the box is .........     25.323179
 cut off is ........................      3.750000
```

```
 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 =  14.005826
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 =  12.351927
Test PASSED
```

Listing 4. Log izvršavanja problema 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

4.3.3. Diskusija dobijenih rezultata

Možemo videti na grafu Slika 4 da korišćenjem ovog modela ne dolazi do ubrzanja samog programa. Razlog za ovo jeste konstantna komunikacija između menadžera i radnika koja usporava rad programa. Ovo se dešava zato što u jednom trenutku menadžer može da komunicira samo sa jednim radnikom i u tom periodu komunikacije ako ostali radnici čekaju da komuniciraju isto sa menadžerom ne dolazi do bilo kakve obrade posla.