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. Pavle Divović

Studenti:

Lazar Erić 2019/0235

Aleksa Račić 2019/0235

Beograd, decembar 2022.

SADRŽAJ

1. PROBLEM 1 - PRIME

1.1. Tekst problema

Paralelizovati program koji vrši određivanje ukupnog broja prostih brojeva u zadatom opsegu. Prilikom zadavanja izvršne konfiguracije jezgra, koristiti 1D rešetku (grid). Obratiti pažnju na efikasnost paralelizacije i potrebu za redukcijom. Program se nalazi u datoteci prime.c u arhivi koja je priložena uz ovaj dokument. Program testirati sa parametrima koji su dati u datoteci run.

1.2. Delovi koje treba paralelizovati

1.2.1. Diskusija

U okviru ovog zadatka mozemo uociti vise funkcija od kojih je nama od najveceg interesa prime_number. U ostalim funkcijama i main-u ne postoji delova koda koji bi mogli da budu paralelizovani ili smesteni u task-ove.

1.2.2. Način paralelizacije

Iako imamo 2 for petlje, rezultat prve zavisi od druge tako da jedan thread mora da izvrsi unutrasnju petlju kako bi nasao da li je broj prost. Da bi se maksimalno iskoristio uredjaj, broj blokova unutar grida je takav da BlockSize * GridSize bude jednak broju iteracija.

1.3. Rezultati

U okviru ove sekcije su izloženi rezultati paralelizacije problema 1.

1.3.1. Logovi izvršavanja

Ovde su dati logovi izvršavanja za definisane test primere i različit broj niti. Obavezno uključiti u ispis i vremena izvršavanja. Logove pojedinačno uokviriti i obeležiti.

```
19 January 2023 12:37:42 AM

PRIME TEST

  Call PRIME_NUMBER to count the primes from 1 to N.

         N        Pi          Time
```

```
         1          0        0.063463
         2          1        0.000106
         4          2        0.000123
         8          4        0.000104
        16          6        0.000102
        32         11        0.000103
        64         18        0.000107
       128         31        0.000141
       256         54        0.000132
       512         97        0.000164
      1024        172        0.000247
      2048        309        0.000389
      4096        564        0.000698
      8192       1028        0.001347
     16384       1900        0.002614
     32768       3512        0.005837
     65536       6542        0.016828
    131072      12251        0.056249

PRIME_TEST
  Normal end of execution.

19 January 2023 12:37:42 AM
19 January 2023 12:37:42 AM

PRIME TEST

  Call PRIME_NUMBER to count the primes from 1 to N.

         N         Pi          Time

         5          3        0.053804
        50         15        0.000089
       500         95        0.000142
      5000        669        0.000802
     50000       5133        0.010586
    500000      41538        0.668534
```

```
PRIME_TEST
  Normal end of execution.

19 January 2023 12:37:43 AM
19 January 2023 12:37:43 AM

PRIME TEST

  Call PRIME_NUMBER to count the primes from 1 to N.

         N        Pi          Time

         1         0        0.044637
         4         2        0.000082
        16         6        0.000086
        64        18        0.000093
       256        54        0.000108
      1024       172        0.000197
      4096       564        0.000622
     16384      1900        0.002336
     65536      6542        0.015649

PRIME_TEST
  Normal end of execution.

19 January 2023 12:37:43 AM
19 January 2023 12:37:43 AM

PRIME TEST

  Call PRIME_NUMBER to count the primes from 1 to N.

         N        Pi          Time

         1         0        0.000001
         2         1        0.000000
         4         2        0.000000
         8         4        0.000000
        16         6        0.000000
```

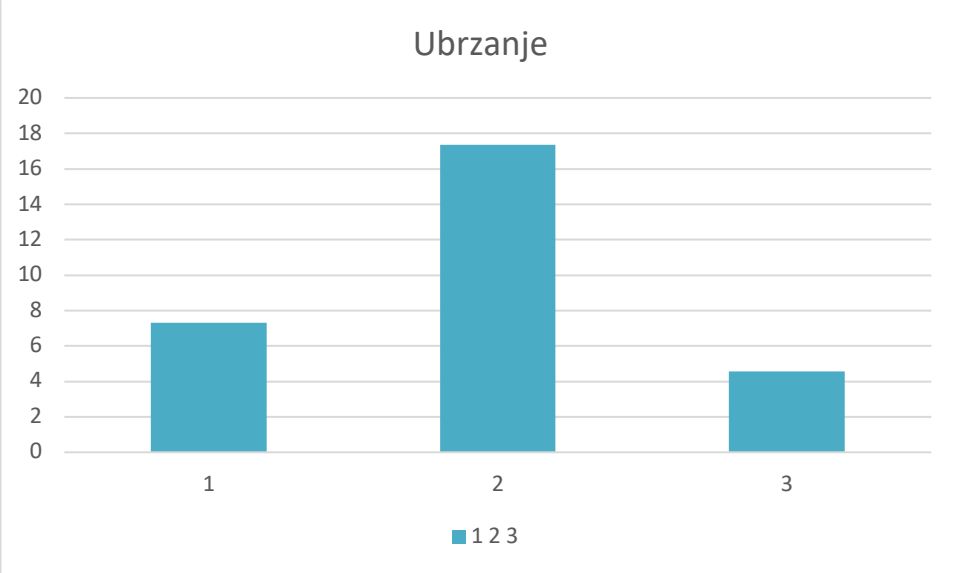
```
        32        11        0.000001
        64        18        0.000001
       128        31        0.000003
       256        54        0.000009
       512        97        0.000032
      1024       172        0.000113
      2048       309        0.000403
      4096       564        0.001480
      8192      1028        0.005144
     16384      1900        0.018555
     32768      3512        0.068269
     65536      6542        0.256072
    131072     12251        0.957547

PRIME_TEST
  Normal end of execution.

19 January 2023 12:37:45 AM
19 January 2023 12:37:45 AM

PRIME TEST

  Call PRIME_NUMBER to count the primes from 1 to N.

         N        Pi          Time

         5         3        0.000000
        50        15        0.000000
       500        95        0.000029
      5000       669        0.002024
     50000      5133        0.152914
    500000     41538       12.550401

PRIME_TEST
  Normal end of execution.

19 January 2023 12:37:57 AM
19 January 2023 12:37:57 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        0.000001
        64        18        0.000001
       256        54        0.000009
      1024       172        0.000109
      4096       564        0.001402
     16384      1900        0.018864
     65536      6542        0.261888

PRIME_TEST
  Normal end of execution.

19 January 2023 12:37:58 AM
```

Listing 1. Paralelno neoptimizovano i optimizovano izvršavanje programa

1.3.2. Grafici ubrzanja

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



Slika 1. Grafik zavisnosti ubrzanja programa programa:
run 1 -./prime 1 131072 2, run 2 - ./prime 5 500000 10, run 3 - ./prime 1 65536 4

1.3.3. Diskusija dobijenih rezultata

Sa grafika mozemo uociti da sto je veci paralelni deo to je ubrzanje vece.

2. PROBLEM 2 - FEYMAN

2.1. Tekst problema

1. 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`**.

2.2. Delovi koje treba paralelizovati

2.2.1. Diskusija

U okviru ovog problema vidimo veci broj funkcija, ali one su sve pomocne koje nam sluze da izvrsimo neke jednostavne matematicke operacije. Zato se one izvrsavaju sekvencijalno, a najveci fokus je na main-u gde ima vise ugnjezndenih for pelji.

2.2.2. Način paralelizacije

Postoje 4 for petlje. Svaka torka vrednosti 3 vanjske petlje predstavlja jednu koordinatu bloka unutar grida. Tok je jedan Blok zaduzen za obradu N trialsa.

2.3. Rezultati

U okviru ove sekcije su izloženi rezultati paralelizacije problema 2

2.3.1. Logovi izvršavanja

Ovde su dati logovi izvršavanja za definisane test primere i različit broj niti. Obavezno uključiti u ispis i vremena izvršavanja. Logove pojedinačno uokviriti i obeležiti.

```
--------------------optimized--------------------
19 January 2023 12:34:23 AM
A = 3.000000
B = 2.000000
C = 1.000000
N = 1000
H = 0.0010
```

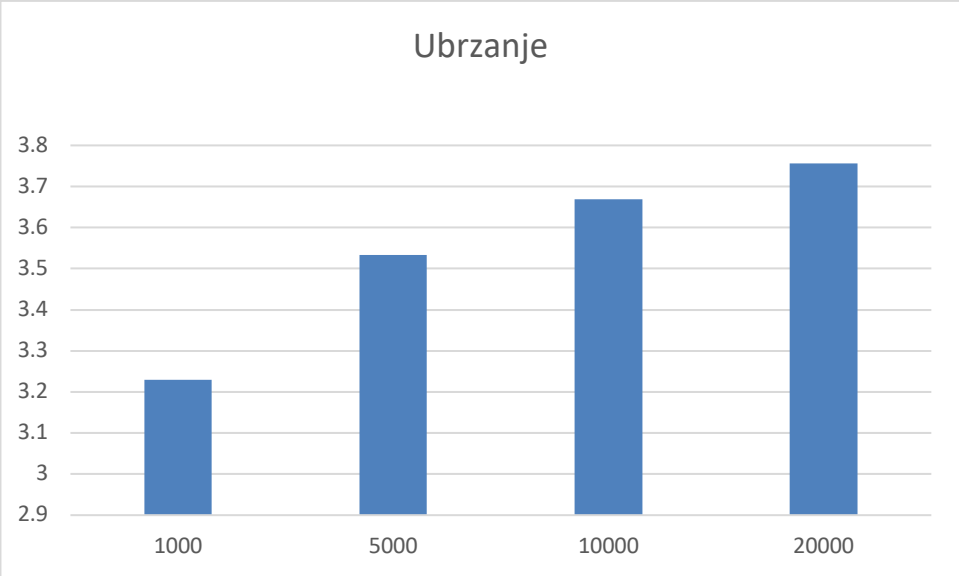
```
RMS absolute error in solution = 2.731885e-02
19 January 2023 12:34:24 AM
19 January 2023 12:34:24 AM
A = 3.000000
B = 2.000000
C = 1.000000
N = 5000
H = 0.0010


RMS absolute error in solution = 2.076798e-02
19 January 2023 12:34:28 AM
19 January 2023 12:34:28 AM
A = 3.000000
B = 2.000000
C = 1.000000
N = 10000
H = 0.0010


RMS absolute error in solution = 2.096349e-02
19 January 2023 12:34:36 AM
19 January 2023 12:34:36 AM
A = 3.000000
B = 2.000000
C = 1.000000
N = 20000
H = 0.0010


RMS absolute error in solution = 2.108210e-02
19 January 2023 12:34:52 AM
19 January 2023 12:34:52 AM
A = 3.000000
B = 2.000000
C = 1.000000
N = 1000
```

```
H = 0.0010


RMS absolute error in solution = 2.171700e-02
19 January 2023 12:34:55 AM
19 January 2023 12:34:55 AM
A = 3.000000
B = 2.000000
C = 1.000000
N = 5000
H = 0.0010


RMS absolute error in solution = 2.127277e-02
19 January 2023 12:35:10 AM
19 January 2023 12:35:10 AM
A = 3.000000
B = 2.000000
C = 1.000000
N = 10000
H = 0.0010


RMS absolute error in solution = 2.109998e-02
19 January 2023 12:35:41 AM
19 January 2023 12:35:41 AM
A = 3.000000
B = 2.000000
C = 1.000000
N = 20000
H = 0.0010


RMS absolute error in solution = 2.102653e-02
19 January 2023 12:36:41 AM
```

Listing 2. Paralelno neoptimizovano i optimizovano izvršavanje programa

2.3.2. Grafici ubrzanja

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



Slika 2. Grafik ubrzanja

2.3.3. Diskusija dobijenih rezultata

Sa grafika mozemo videti da kako se povecava paralelni deo workloada tako se povecava ubrzanje programa. Preko 50 posto izvrsavanja ide na upis

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

U okviru ovog zadatka mozemo uociti fise .c fajlova od kojih je nama od najveceg interesa forces.c i main.c. U ostalim fajlovima se najvise nalazi kod koji ne moze biti paralelizovan.

3.2.2. Način paralelizacije

Paralelizaciju smo uradili u dva fajla – forces.c i main.c . U fajlu forces.c se odvija najvise izracunavanja. Paralelizacija je izvrsena atomicnim dodavanjem

3.3. Rezultati

U okviru ove sekcije su izloženi rezultati paralelizacije problema 3.

3.3.1. Logovi izvršavanja

Ovde su dati logovi izvršavanja za definisane test primere i različit broj niti. Obavezno uključiti u ispis i vremena izvršavanja. Logove pojedinačno uokviriti i obeležiti.

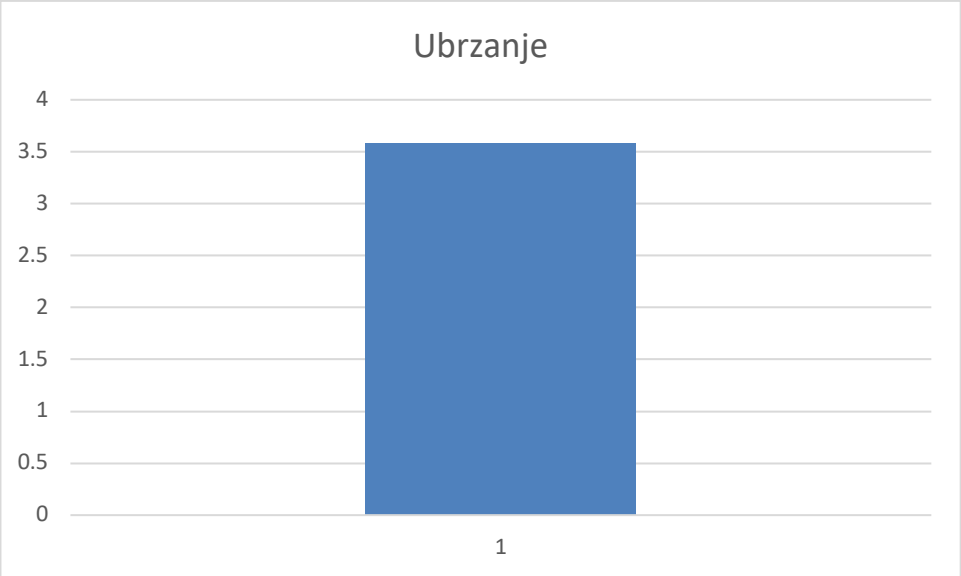
```
Molecular Dynamics Simulation example program
 ----------------------------------------
 number of particles is .............  13500
 side length of the box is .........     25.323179
 cut off is ........................      3.750000
 reduced temperature is ............      0.722000
 basic timestep 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 =  1.644527
rm *.o md
gcc -O3 -fopenmp  -c main.c
gcc -O3 -fopenmp  -c dfill.c
gcc -O3 -fopenmp  -c domove.c
gcc -O3 -fopenmp  -c dscal.c
gcc -O3 -fopenmp  -c fcc.c
gcc -O3 -fopenmp  -c forces.c
```

```
gcc -O3 -fopenmp  -c mkekin.c
gcc -O3 -fopenmp  -c mxwell.c
gcc -O3 -fopenmp  -c prnout.c
gcc -O3 -fopenmp  -c velavg.c
gcc -O3 -fopenmp  -o md main.o dfill.o domove.o dscal.o fcc.o forces.o mkekin.o
mxwell.o prnout.o velavg.o -lm
 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 =  5.713348
```

Listing 3.

3.3.2. Grafici ubrzanja

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



Slika 3. Grafik zavisnosti ubrzanja programa za N = 1000 kada rade 1,2,3 i 4 niti

3.3.3. Diskusija dobijenih rezultata

Sa grafika vidimo da je ubrzanje oko 3.5 puta

4. PROBLEM 4 - MOLDYN

4.1. Tekst problema

4.2. Delovi koje treba paralelizovati

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]

4.2.1. Diskusija

U okviru ovog zadatka mozemo uociti fise .c fajlova od kojih je nama od najveceg interesa forces.c i main.c. U ostalim fajlovima se najvise nalazi kod koji ne moze biti apralelizovan.

4.2.2. Način paralelizacije

Paralelizaciju smo uradili u dva fajla – forces.c i main.c . U fajlu forces.c se odvija najvise izracunavanja. Izvrsili smo paralelizaciju for direktivom i redukciju epot i vir promenjivih uz pomoc broadcasta kada svi procesi salju masteru koji na kraju to redukuje. S tim stoe je sada razlika da master sam ne ucestvuje u obradi vec deli poslove worker-ima, pa sje tako kada imamo N = 2, tj. Jednog worker-a i master-a sporije neko sekvencijalno.

4.3. Rezultati

U okviru ove sekcije su izloženi rezultati paralelizacije problema 5.

4.3.1. Logovi izvršavanja

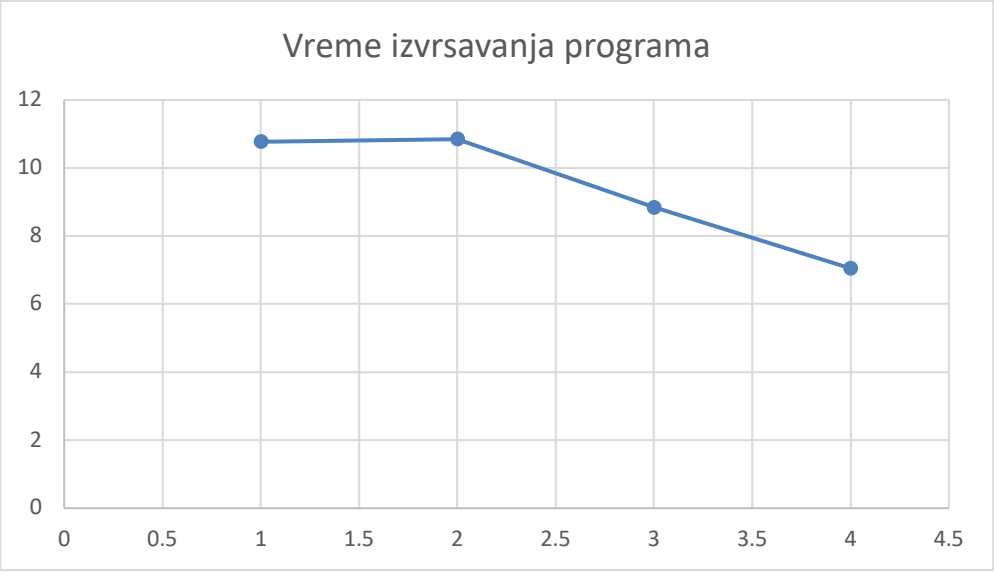
Ovde su dati logovi izvršavanja za definisane test primere i različit broj niti. Obavezno uključiti u ispis i vremena izvršavanja. Logove pojedinačno uokviriti i obeležiti.

```
Thread_Num   Time
1      10.778501
2      10.845185
3      8.841843
4      7.049351
```

Listing 4.

4.3.2. Grafici ubrzanja

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



Slika 4. Grafik zavisnosti ubrzanja programa za N = 1,2,3 i 4 niti

4.3.3. Diskusija dobijenih rezultata

Sa datih grafika možemo primeniti da je ubrzanje sve uocljivije kako se povecava broj niti i kako se povecava.