Univerzitet u Beogradu - Elektrotehnički fakultet

Multiprocesorki sistemi (13S114MUPS, 13E114MUPS)



Domaći zadatak 2 –MPI

Izveštaj o urađenom domaćem zadatku

|  |  |
| --- | --- |
| Predmetni saradnici: | Studenti: |
| doc. dr Marko Mišić  dipl. ing. Pavle Divović | Lazar Erić 2019/0235  Aleksa Račić 2019/0235 |

Beograd, decembar 2022.

Sadržaj

[Sadržaj 2](#_Toc123061341)

[1. Problem 1 - PRIME 3](#_Toc123061342)

[1.1. Tekst problema 3](#_Toc123061343)

[1.2. Delovi koje treba paralelizovati 3](#_Toc123061344)

[1.2.1. Diskusija 3](#_Toc123061345)

[1.2.2. Način paralelizacije 3](#_Toc123061346)

[1.3. Rezultati 3](#_Toc123061347)

[1.3.1. Logovi izvršavanja 3](#_Toc123061348)

[1.3.2. Grafici ubrzanja 5](#_Toc123061349)

[1.3.3. Diskusija dobijenih rezultata 6](#_Toc123061350)

[2. Problem 2 - FEYMAN 7](#_Toc123061351)

[2.1. Tekst problema 7](#_Toc123061352)

[2.2. Delovi koje treba paralelizovati 7](#_Toc123061353)

[2.2.1. Diskusija 7](#_Toc123061354)

[2.2.2. Način paralelizacije 7](#_Toc123061355)

[2.3. Rezultati 7](#_Toc123061356)

[2.3.1. Logovi izvršavanja 7](#_Toc123061357)

[2.3.2. Grafici ubrzanja 10](#_Toc123061358)

[2.3.3. Diskusija dobijenih rezultata 11](#_Toc123061359)

[3. Problem 3 - MolDyn 12](#_Toc123061360)

[3.1. Tekst problema 12](#_Toc123061361)

[3.2. Delovi koje treba paralelizovati 12](#_Toc123061362)

[3.2.1. Diskusija 12](#_Toc123061363)

[3.2.2. Način paralelizacije 13](#_Toc123061364)

[3.3. Rezultati 13](#_Toc123061365)

[3.3.1. Logovi izvršavanja 13](#_Toc123061366)

[3.3.2. Grafici ubrzanja 14](#_Toc123061367)

[3.3.3. Diskusija dobijenih rezultata 15](#_Toc123061368)

[4. Problem 4 - MolDyn 16](#_Toc123061369)

[4.1. Tekst problema 16](#_Toc123061370)

[4.2. Delovi koje treba paralelizovati 16](#_Toc123061371)

[4.2.1. Diskusija 16](#_Toc123061372)

[4.2.2. Način paralelizacije 16](#_Toc123061373)

[4.3. Rezultati 16](#_Toc123061374)

[4.3.1. Logovi izvršavanja 16](#_Toc123061375)

[4.3.2. Grafici ubrzanja 17](#_Toc123061376)

[4.3.3. Diskusija dobijenih rezultata 18](#_Toc123061377)

1. Problem 1 - PRIME
   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. Delovi koje treba paralelizovati
     1. Diskusija

U okviru ovog zadatka mozemo uociti fise 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. Način paralelizacije

Kako imamo 2 for petlje u prime\_number funkciji, moramo rucno rasporediti iteracije koje master nit salje ostalima koji rade zapravo svoj deo iteracija u for petljama. Kako je promenljiva total na kraju for-a povecava i samo tu menja, ona ce biti reduction promenljiva.

* 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. Obavezno uključiti u ispis i vremena izvršavanja. Logove pojedinačno uokviriti i obeležiti.

--------------------unoptimized--------------------

N Pi Time

1 0 0.000125

2 1 0.000001

4 2 0.000001

8 4 0.000001

16 6 0.000001

32 11 0.000001

64 18 0.000001

128 31 0.000001

256 54 0.000003

512 97 0.000008

1024 172 0.000026

2048 309 0.000092

4096 564 0.000332

8192 1028 0.001218

16384 1900 0.004714

32768 3512 0.016289

65536 6542 0.05972

131072 12251 0.227838

5 3 0.000119

50 15 0.000001

500 95 0.000008

5000 669 0.000511

50000 5133 0.039446

500000 41538 3.176562

1 0 0.000104

4 2 0.000001

16 6 0.000001

64 18 0.000001

256 54 0.000003

1024 172 0.000026

4096 564 0.00033

16384 1900 0.004619

65536 6542 0.059847

TOTAL TIME : 3.591951

--------------------optimized--------------------

Size MaxWallTime MinWallTime AvgWallTime

1 1.311489 1.311489 1.311489

1 12.636851 12.636851 12.636851

1 0.276818 0.276818 0.276818

2 0.671683 0.663852 0.667768

2 6.437754 6.420637 6.429196

2 0.140734 0.139886 0.14031

3 0.669759 0.666336 0.668615

3 6.699568 6.682187 6.693769

3 0.140915 0.140383 0.140735

4 0.35569 0.346772 0.350689

4 3.488552 3.460801 3.470624

4 0.073767 0.072387 0.073201

Size MaxCpuTime MinCpuTime AvgCpuTime Max-Avg Avg-Min

1 1.311462 1.311462 1.311462 0 0

1 12.636303 12.636303 12.636303 0 0

1 0.276707 0.276707 0.276707 0 0

2 0.671627 0.663755 0.667691 0.003915 0.003916

2 6.437492 6.420613 6.429053 0.008558 0.008559

2 0.14065 0.139841 0.140246 0.000424 0.000424

3 0.669694 0.666252 0.668472 0.001144 0.002279

3 6.699526 6.681907 6.693575 0.005799 0.011582

3 0.140805 0.140377 0.140661 0.00018 0.000352

4 0.355681 0.34677 0.35067 0.005001 0.003917

4 3.488227 3.460696 3.470485 0.017928 0.009823

4 0.073671 0.072314 0.073127 0.000566 0.000814

Listing 1. Paralelno neoptimizovano i optimizovano izvršavanje programa

* + 1. Grafici ubrzanja

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

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

run 1 - mpirun -np $t ./prime 1 131072 2

run 2 - mpirun -np $t ./prime 5 500000 10

run 3 - mpirun -np $t ./prime 1 65536 4

* + 1. Diskusija dobijenih rezultata

Sa datih grafika možemo primeniti da je ubrzanje sve uocljivije kako se povecava broj niti i kako se povecava. Takodje, vidi se ogromna razlika izmedju optimizovanog i neoptimizovanog koda.

1. Problem 2 - FEYMAN
   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]

* 1. Delovi koje treba paralelizovati
     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.

* + 1. Način paralelizacije

S obzirom da broj iteracija prve tri for petlje nije preveliki, mi se koncentrisemo na paralelizaciju najuugnjezdenije for petlje koju rade i master i ostali procesi sa svojim chunk\_size-om.

* 1. Rezultati

U okviru ove sekcije su izloženi rezultati paralelizacije problema 2

* + 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--------------------

Num\_Threads N RMS MaxTime MinTime AvgTime

1 1000 0.021717 3.635621 3.635621 3.635621

1 5000 0.021273 19.510467 19.510467 19.510467

1 10000 0.0211 37.529069 37.529069 37.529069

1 20000 0.021027 65.397179 65.397179 65.397179

2 1000 0.022108 1.545373 1.542642 1.544008

2 5000 0.021083 7.66284 7.651001 7.65692

2 10000 0.020968 17.362141 17.346726 17.354434

2 20000 0.021022 38.345505 38.317076 38.331291

3 1000 0.021974 1.345273 1.342823 1.344196

3 5000 0.021201 6.709455 6.699402 6.703298

3 10000 0.02105 13.582462 13.563605 13.575155

3 20000 0.020961 29.316467 29.279916 29.304087

4 1000 0.021577 1.048798 1.047524 1.048242

4 5000 0.020977 5.203013 5.195298 5.198469

4 10000 0.020979 10.532916 10.517366 10.524769

4 20000 0.020875 21.087035 21.030404 21.062726

Listing 2. Paralelno neoptimizovano i optimizovano izvršavanje programa

* + 1. Grafici ubrzanja

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

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

* + 1. Diskusija dobijenih rezultata

Sa datih grafika možemo primeniti da je ubrzanje uočljivo i da je sve veće kako je i broj niti veći. Takodje se vidi i ogromna razlika u vremenu izvrsavanja neoptimizovanog i optimizovanog programa.

1. Problem 3 - MolDyn
   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]

* 1. Delovi koje treba paralelizovati
     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.

* + 1. 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.

* 1. Rezultati

U okviru ove sekcije su izloženi rezultati paralelizacije problema 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 10.837424

2 8.206704

3 6.572324

4 5.513658

Listing 3.

* + 1. 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

* + 1. Diskusija dobijenih rezultata

Sa datih grafika možemo primeniti da je ubrzanje uočljivo i da je sve veće kako je i broj niti veći. Takodje, jasno se vidi povecanje vremena izvrsavanja kada se povecava broj N.

1. Problem 4 - MolDyn
   1. Tekst problema
   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]

* + 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.

* + 1. 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.

* 1. Rezultati

U okviru ove sekcije su izloženi rezultati paralelizacije problema 5.

* + 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.

* + 1. 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

* + 1. Diskusija dobijenih rezultata

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