Univerzitet u Beogradu - Elektrotehnički fakultet

Multiprocesorki sistemi (13S114MUPS, 13E114MUPS)



Domaći zadatak 1 – OPENMP

Izveštaj o urađenom domaćem zadatku

|  |  |
| --- | --- |
| Predmetni asistent: | Studenti: |
| doc. dr Marko Mišić | Bogdan Bebić 2017/0011  Uroš Krstić 2017/0228 |

Beograd, novembar 2020.

Sadržaj

[Sadržaj 2](#_Toc56515627)

[1. Problem 1 – Računanje broja π (pi) 3](#_Toc56515628)

[1.1. Tekst problema 3](#_Toc56515629)

[1.2. Delovi koje treba paralelizovati 3](#_Toc56515630)

[1.2.1. Diskusija 3](#_Toc56515631)

[1.2.2. Način paralelizacije 3](#_Toc56515632)

[1.3. Rezultati 3](#_Toc56515633)

[1.3.1. Logovi izvršavanja 3](#_Toc56515634)

[1.3.2. Grafici ubrzanja 4](#_Toc56515635)

[1.3.3. Diskusija dobijenih rezultata 4](#_Toc56515636)

[2. Problem 2 – Računanje broja π (pi) 5](#_Toc56515637)

[2.1. Tekst problema 5](#_Toc56515638)

[2.2. Delovi koje treba paralelizovati 5](#_Toc56515639)

[2.2.1. Diskusija 5](#_Toc56515640)

[2.2.2. Način paralelizacije 5](#_Toc56515641)

[2.3. Rezultati 5](#_Toc56515642)

[2.3.1. Logovi izvršavanja 5](#_Toc56515643)

[2.3.2. Grafici ubrzanja 6](#_Toc56515644)

[2.3.3. Diskusija dobijenih rezultata 6](#_Toc56515645)

[3. Problem 3 – Računanje broja π (pi) 7](#_Toc56515646)

[3.1. Tekst problema 7](#_Toc56515647)

[3.2. Delovi koje treba paralelizovati 7](#_Toc56515648)

[3.2.1. Diskusija 7](#_Toc56515649)

[3.2.2. Način paralelizacije 7](#_Toc56515650)

[3.3. Rezultati 7](#_Toc56515651)

[3.3.1. Logovi izvršavanja 7](#_Toc56515652)

[3.3.2. Grafici ubrzanja 8](#_Toc56515653)

[3.3.3. Diskusija dobijenih rezultata 8](#_Toc56515654)

[4. Problem 4 – *Needleman-Wunsch* algoritam 9](#_Toc56515655)

[4.1. Tekst problema 9](#_Toc56515656)

[4.2. Delovi koje treba paralelizovati 9](#_Toc56515657)

[4.2.1. Diskusija 9](#_Toc56515658)

[4.2.2. Način paralelizacije 9](#_Toc56515659)

[4.3. Rezultati 9](#_Toc56515660)

[4.3.1. Logovi izvršavanja 9](#_Toc56515661)

[4.3.2. Grafici ubrzanja 10](#_Toc56515662)

[4.3.3. Diskusija dobijenih rezultata 10](#_Toc56515663)

[5. Problem 5 – Interakcija čvrstih tela (*n-body*) 11](#_Toc56515664)

[5.1. Tekst problema 11](#_Toc56515665)

[5.2. Delovi koje treba paralelizovati 11](#_Toc56515666)

[5.2.1. Diskusija 11](#_Toc56515667)

[5.2.2. Način paralelizacije 11](#_Toc56515668)

[5.3. Rezultati 11](#_Toc56515669)

[5.3.1. Logovi izvršavanja 11](#_Toc56515670)

[5.3.2. Grafici ubrzanja 12](#_Toc56515671)

[5.3.3. Diskusija dobijenih rezultata 12](#_Toc56515672)

1. Problem 1 – Računanje broja π (pi)
   1. Tekst problema

Paralelizovati program koji izračunava vrednost broja PI korišćenjem formule:

.

Tačnost izračunavanja direktno zavisi od broja iteracija, a zbog malog radijusa konvergencije serija konvergira veoma sporo. Program se nalazi u datoteci **piCalc.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. Delovi koje treba paralelizovati
     1. Diskusija

TODO: U okviru ove sekcije će biti opisani uočeni delovi koda koje je moguće paralelizovati. Biće diskutovati delovi nad kojima je izvršena paralelizacija. Ukoliko je neki deo moguće paralelizovati, ali to nije učinjeno, navesti razloge.

* + 1. Način paralelizacije

TODO: Ovde treba opisati način paralelizacije i razloge koji su vodili ka odluci. Ukoliko je isprobano nekoliko načina, ovde ih diskutovati.

* 1. Rezultati

TODO: U okviru ove sekcije su izloženi rezultati paralelizacije problema 1.

* + 1. Logovi izvršavanja

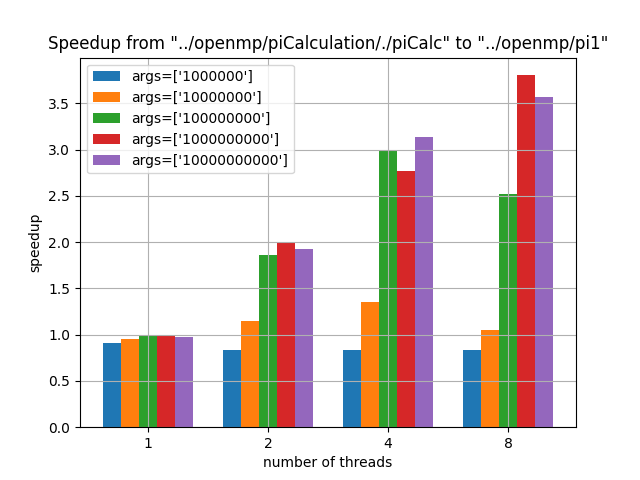
Ovde su dati logovi izvršavanja za definisane test primere i različit broj niti.

// TODO: logovi izvršavajna

Listing 1. Sekvencijalno izvršavanje računanja pi

* + 1. Grafici ubrzanja

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



Slika 1. Grafik zavisnosti ubrzanja u odnosu na broj niti za različite argumente pokretanja

* + 1. Diskusija dobijenih rezultata

TODO: Dobijene rezultate treba izdiskutovati i objasniti sa nekoliko rečenica u odnosu na dobijena ubrzanja (ili usporenja), način paralelizacije, karakteristike problema i sl.

1. Problem 2 – Računanje broja π (pi)
   1. Tekst problema

Paralelizovati program koji izračunava vrednost broja PI korišćenjem formule:

.

Tačnost izračunavanja direktno zavisi od broja iteracija, a zbog malog radijusa konvergencije serija konvergira veoma sporo. Program se nalazi u datoteci **piCalc.c** u arhivi koja je priložena uz ovaj dokument. Prilikom paralelizacije koristiti direktive za podelu posla (*worksharing* direktive). Obratiti pažnju na raspodelu opterećenja po nitima i testirati program za različite načine raspoređivanja posla. Program testirati sa parametrima koji su dati u datoteci **run**. [1, N]

* 1. Delovi koje treba paralelizovati
     1. Diskusija

TODO: U okviru ove sekcije će biti opisani uočeni delovi koda koje je moguće paralelizovati. Biće diskutovati delovi nad kojima je izvršena paralelizacija. Ukoliko je neki deo moguće paralelizovati, ali to nije učinjeno, navesti razloge.

* + 1. Način paralelizacije

TODO: Ovde treba opisati način paralelizacije i razloge koji su vodili ka odluci. Ukoliko je isprobano nekoliko načina, ovde ih diskutovati.

* 1. Rezultati

TODO: U okviru ove sekcije su izloženi rezultati paralelizacije problema 1.

* + 1. Logovi izvršavanja

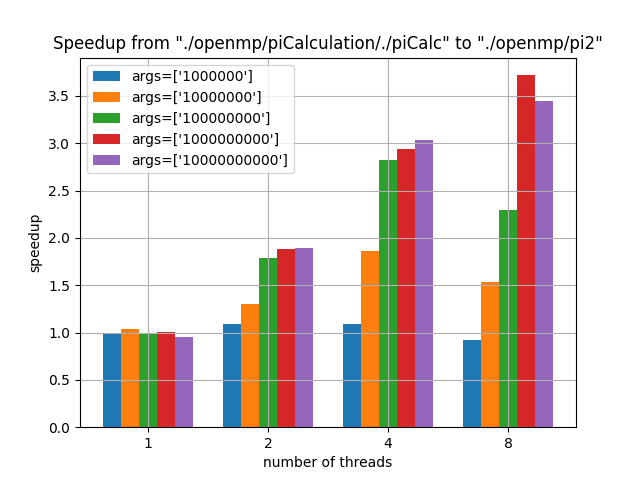
Ovde su dati logovi izvršavanja za definisane test primere i različit broj niti.

// TODO: logovi izvršavajna

Listing 1. Sekvencijalno izvršavanje računanja pi

* + 1. Grafici ubrzanja

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



Slika 1. Grafik zavisnosti ubrzanja u odnosu na broj niti za različite argumente pokretanja

* + 1. Diskusija dobijenih rezultata

TODO: Dobijene rezultate treba izdiskutovati i objasniti sa nekoliko rečenica u odnosu na dobijena ubrzanja (ili usporenja), način paralelizacije, karakteristike problema i sl.

1. Problem 3 – Računanje broja π (pi)
   1. Tekst problema

Paralelizovati program koji izračunava vrednost broja PI korišćenjem formule:

.

Tačnost izračunavanja direktno zavisi od broja iteracija, a zbog malog radijusa konvergencije serija konvergira veoma sporo. Program se nalazi u datoteci **piCalc.c** u arhivi koja je priložena uz ovaj dokument. Prilikom paralelizacije koristiti koncept poslova (*tasks*). Obratiti pažnju na eventualnu potrebu za sinhronizacijom. Rešenje testirati i prilagoditi tako da granularnost poslova bude optimalna. Program testirati sa parametrima koji su dati u datoteci **run**. [1, N]

* 1. Delovi koje treba paralelizovati
     1. Diskusija

TODO: U okviru ove sekcije će biti opisani uočeni delovi koda koje je moguće paralelizovati. Biće diskutovati delovi nad kojima je izvršena paralelizacija. Ukoliko je neki deo moguće paralelizovati, ali to nije učinjeno, navesti razloge.

* + 1. Način paralelizacije

TODO: Ovde treba opisati način paralelizacije i razloge koji su vodili ka odluci. Ukoliko je isprobano nekoliko načina, ovde ih diskutovati.

* 1. Rezultati

TODO: U okviru ove sekcije su izloženi rezultati paralelizacije problema 1.

* + 1. Logovi izvršavanja

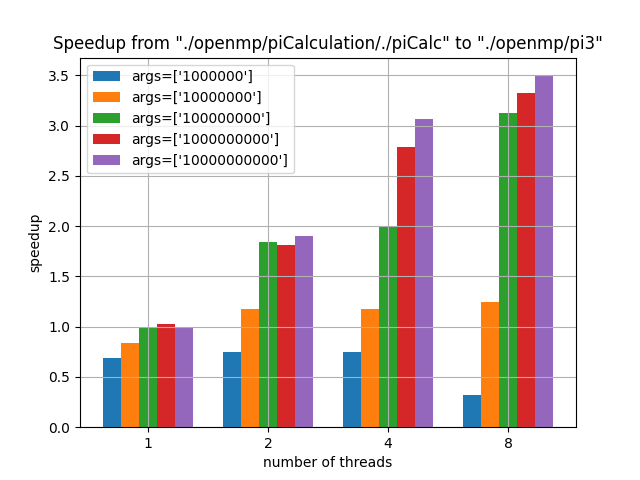
Ovde su dati logovi izvršavanja za definisane test primere i različit broj niti.

// TODO: logovi izvršavajna

Listing 1. Sekvencijalno izvršavanje računanja pi

* + 1. Grafici ubrzanja

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



Slika 1. Grafik zavisnosti ubrzanja u odnosu na broj niti za različite argumente pokretanja

* + 1. Diskusija dobijenih rezultata

TODO: Dobijene rezultate treba izdiskutovati i objasniti sa nekoliko rečenica u odnosu na dobijena ubrzanja (ili usporenja), način paralelizacije, karakteristike problema i sl.

1. Problem 4 – *Needleman-Wunsch* algoritam
   1. Tekst problema

Paralelizovati program koji vrši poravnavanje bioloških sekvenci korišćenjem *Needleman-Wunsch* algoritma. Algoritam predstavlja primenu koncepta dinamičkog programiranja za globalno poravnavanje dve sekvence nukleotida ili aminokiselina (više o algoritmu na adresi: <https://en.wikipedia.org/wiki/Needleman%E2%80%93Wunsch_algorithm>). Program se nalazi u datoteci **needle.c** u arhivi koja je priložena uz ovaj dokument. Obratiti pažnju na raspodelu opterećenja po nitima i testirati program za različite načine raspoređivanja posla. Program testirati sa parametrima koji su dati u datoteci **run**. [1, N]

* 1. Delovi koje treba paralelizovati
     1. Diskusija

TODO: U okviru ove sekcije će biti opisani uočeni delovi koda koje je moguće paralelizovati. Biće diskutovati delovi nad kojima je izvršena paralelizacija. Ukoliko je neki deo moguće paralelizovati, ali to nije učinjeno, navesti razloge.

* + 1. Način paralelizacije

TODO: Ovde treba opisati način paralelizacije i razloge koji su vodili ka odluci. Ukoliko je isprobano nekoliko načina, ovde ih diskutovati.

* 1. Rezultati

TODO: U okviru ove sekcije su izloženi rezultati paralelizacije problema 1.

* + 1. Logovi izvršavanja

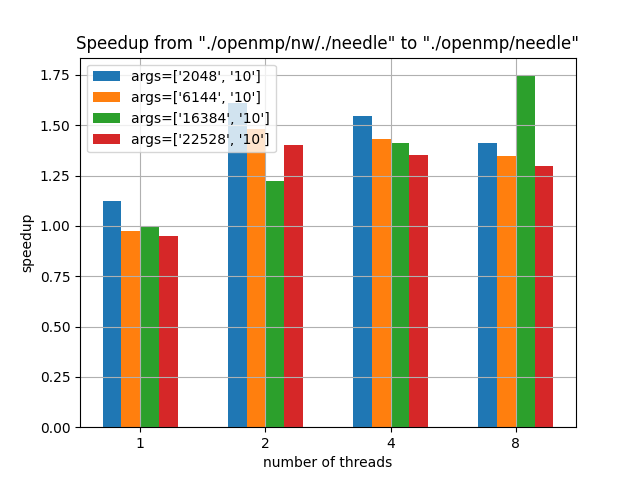
Ovde su dati logovi izvršavanja za definisane test primere i različit broj niti.

// TODO: logovi izvršavajna

Listing 1. Sekvencijalno izvršavanje računanja pi

* + 1. Grafici ubrzanja

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



Slika 1. Grafik zavisnosti ubrzanja u odnosu na broj niti za različite argumente pokretanja

* + 1. Diskusija dobijenih rezultata

TODO: Dobijene rezultate treba izdiskutovati i objasniti sa nekoliko rečenica u odnosu na dobijena ubrzanja (ili usporenja), način paralelizacije, karakteristike problema i sl.

1. Problem 5 – Interakcija čvrstih tela (*n-body*)
   1. Tekst problema

Paralelizovati program koji simulira problem interakcije čvrstih tela u dvodimenzionalnom prostoru (*n-body* problem). Tela interaguju putem gravitacione sile na osnovu sopstvene mase, pozicije u prostoru i trenutne brzine. Program se nalazi u direktorijumu **nbody** u arhivi koja je priložena uz ovaj dokument. Program se sastoji od više datoteka, od kojih je od interesa datoteka **nbody.c**. Analizirati dati kod i obratiti pažnju na način izračunavanja sila i energija. Ukoliko je potrebno međusobno isključenje prilikom paralelizacije programa, koristiti dostupne OpenMP konstrukte. Obratiti pažnju na efikasnost međusobnog isključenja niti i po potrebi ga svesti na što je moguće manju meru uvođenjem pomoćnih struktura podataka. Verifikaciju paralelizovanog rešenja vršiti nad dobijenim energijama i poslednjem stanju sistema. Način pokretanja programa se nalazi u datoteci **run**. [1, N]

* 1. Delovi koje treba paralelizovati
     1. Diskusija

TODO: U okviru ove sekcije će biti opisani uočeni delovi koda koje je moguće paralelizovati. Biće diskutovati delovi nad kojima je izvršena paralelizacija. Ukoliko je neki deo moguće paralelizovati, ali to nije učinjeno, navesti razloge.

* + 1. Način paralelizacije

TODO: Ovde treba opisati način paralelizacije i razloge koji su vodili ka odluci. Ukoliko je isprobano nekoliko načina, ovde ih diskutovati.

* 1. Rezultati

TODO: U okviru ove sekcije su izloženi rezultati paralelizacije problema 1.

* + 1. Logovi izvršavanja

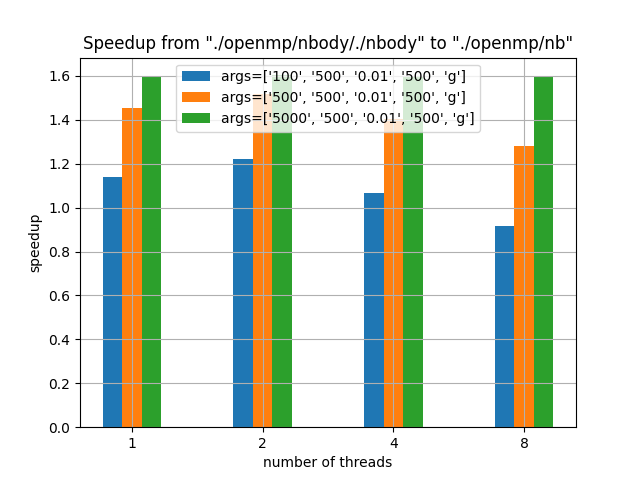
Ovde su dati logovi izvršavanja za definisane test primere i različit broj niti.

// TODO: logovi izvršavajna

Listing 1. Sekvencijalno izvršavanje računanja pi

* + 1. Grafici ubrzanja

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



Slika 1. Grafik zavisnosti ubrzanja u odnosu na broj niti za različite argumente pokretanja

* + 1. Diskusija dobijenih rezultata

TODO: Dobijene rezultate treba izdiskutovati i objasniti sa nekoliko rečenica u odnosu na dobijena ubrzanja (ili usporenja), način paralelizacije, karakteristike problema i sl.