Univerzitet u Beogradu – Elektrotehnički fakultet

Multiprocesorki sistemi (13S114MUPS, 13E114MUPS)



Domaći zadatak 4 – CUDA

Izveštaj o urađenom domaćem zadatku

|  |  |
| --- | --- |
| Predmetni saradnici: | Studenti: |
| doc. dr Marko Mišić  dipl. ing. Pavle Divović | Ana Radovanović 2019/0282  Jovana Jaćimović 2019/0593 |

Beograd, januar 2023.

Sadržaj

[Sadržaj 2](#_Toc125016339)

[1. Problem 1 3](#_Toc125016340)

[1.1. Tekst problema 3](#_Toc125016341)

[1.2. Delovi koje treba paralelizovati 3](#_Toc125016342)

[1.2.1. Diskusija 3](#_Toc125016343)

[1.2.2. Način paralelizacije 3](#_Toc125016344)

[1.3. Rezultati 3](#_Toc125016345)

[1.3.1. Logovi izvršavanja 3](#_Toc125016346)

[1.3.2. Grafici ubrzanja 4](#_Toc125016347)

[1.3.3. Diskusija dobijenih rezultata 5](#_Toc125016348)

[2. Problem 2 6](#_Toc125016349)

[2.1. Tekst problema 6](#_Toc125016350)

[2.2. Delovi koje treba paralelizovati 6](#_Toc125016352)

[2.2.1. Diskusija 6](#_Toc125016353)

[2.2.2. Način paralelizacije 6](#_Toc125016354)

[2.3. Rezultati 6](#_Toc125016355)

[2.3.1. Logovi izvršavanja 6](#_Toc125016356)

[2.3.2. Grafici ubrzanja 7](#_Toc125016357)

[2.3.3. Diskusija dobijenih rezultata 8](#_Toc125016358)

1. Problem 1

U okviru ovog poglavlja je dat kratak izveštaj u vezi rešenja zadatog problema 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. Delovi koje treba paralelizovati
     1. Diskusija

Paralelizacija se vrši u funkciji prime\_number, koja od računa ukupan broj prostih brojeva u zadatom opsegu. Funkcija sadrži dve for petlje, od kojih spoljašnja predstavlja prolazak kroz brojeve u tom opsegu, dok unutrašnja predstavlja prolazak kroz delioce jednog broja. Paralelizacija je urađena nad spoljašnjom for petljom.

* + 1. Način paralelizacije

U funkciji test pozvana je funkcija kernela prime\_number. Alocirani su podaci za potrebe kernela, izvršeni potrebni transferi i zatim je pozvana funkcija kernela. Kako je u zadatku naznačeno da grid bude 1D, grid se i sastoji iz jednog bloka, koji sadrži 1024 niti. Ova funkcija paralelizuje spoljnu for petlju, tako da svaka nit radi jednu iteraciju spoljašnje petlje, u zavisnosti od svog id-a. Privatna promenljiva total\_sum smešta se u registar i nad njom se radi sabiranje, a ona se zatim upisuje u redukcioni niz total, nad kojim se kasnije radi redukcija na CPU. Nakon završetka kernela, kao što je već naznačeno radi se redukcija nad nizom i tako redukovana promenljiva predstavlja rešenje.

* 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.

SEKVENCIJALNO IZVRŠAVANJE

./prime 1 131072 2

Time elapsed: 1.302937

./prime 5 500000 10

Time elapsed: 12.574481

./prime 1 65536 4

Time elapsed: 0.274239

Listing 1. Sekvencijalno izvršavanje problema 1

./dz4z1.exe 1 131072 2

Time elapsed: 1.74756

./dz4z1.exe 5 5000000 10

Time elapsed: 19.70387

./dz4z1.exe 1 65536 4

Time elapsed: 0.36204

Listing 2. Izvršavanje problema 1 nakon paralelizacije koda

* + 1. Grafici ubrzanja

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

Slika 1. Grafik zavisnosti ubrzanja izvršavanja problema 1 od ulaznih parametara

* + 1. Diskusija dobijenih rezultata

Kao što se vidi sa grafika, optimizacija sekvencijalnog algoritma nije uspela, odnosno dobijena vremena su duža od izvršavanja sekvencijalnog koda. Ovo može biti posledica velikih overhead-ova.

1. Problem 2

U okviru ovog poglavlja je dat kratak izveštaj u vezi rešenja zadatog problema 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. Delovi koje treba paralelizovati
     1. Diskusija

Paralelizacija se vrši u main-u, u kome se tačke generišu i računa dužina puta do izlaska. Kako je problem dosta složen, funkcija sadrži 4 for petelje. Odlučeno je da se paralelizacija radi nad spoljašnje tri for petlje.

* + 1. Način paralelizacije

U main funkciji izvršena je alokacija podataka za potrebe kernela i izvršeni su potrebni transferi. Korišćen je grid dimenzije 2D i sastoji iz dva bloka, od kojih oba sadrže 1024 niti, mada u prvom bloku sve niti rade dok u drugom bloku aktivne su niti sa id-em manjem od 32. Zatim je pozvana funkcija kernela feyman. Ova funkcija, koju izračunava svaka nit, računa grešku nad jednom tačkom, tako da su sve tri petlje maltene paralelizovane u jednu funkciju. Na kraju funkcije kernela radi se kopija povratnih vrednosti iz GPU-a u CPU. Zatim se radi redukcija nad redukcionim nizovima i obračunava konačno rešenje.

* 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.

SEKVENCIJALNO IZVRŠAVANJE

time ./feyman 1000

RMS absolute error in solution = 2.171700e-02

real 21.243

time ./feyman 5000

RMS absolute error in solution = 2.127277e-02

real 105.367

time ./feyman 10000

RMS absolute error in solution = 2.109998e-02

real 211.274

time ./feyman 20000

RMS absolute error in solution = 2.102653e-02

real 422.919

Listing 3. Sekvencijalno izvršavanje problema 2

./dz4z2.exe 1000

RMS absolute error in solution = 2.889379e-02

Time elapsed: 210.28564

./dz4z2.exe 5000

RMS absolute error in solution = 1.962858e-02

Time elapsed: 1026.1004

./dz4z2.exe 10000

RMS absolute error in solution = 1.971610e-02

Time elapsed: 2103.46819

./dz4z2.exe 20000

RMS absolute error in solution = 1.980972e-02

Time elapsed: 4196.1238

Listing 4. Izvršavanje problema 2 nakon paralelizacije koda

* + 1. Grafici ubrzanja

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

Slika 2. Grafik zavisnosti ubrzanja izvršavanja problema 2 od ulaznog parametara

* + 1. Diskusija dobijenih rezultata

Kao što se vidi sa grafika, optimizacija sekvencijalnog algoritma nije uspela, odnosno dobijena vremena su duža od izvršavanja sekvencijalnog koda. Ovo može biti posledica velikih overhead-ova.