Univerzitet u Beogradu - Elektrotehnički fakultet

Multiprocesorki sistemi (13S114MUPS, 13E114MUPS)



Domaći zadatak 3 – CUDA

Izveštaj o urađenom domaćem zadatku

|  |  |
| --- | --- |
| Predmetni saradnici: | Studenti: |
| doc. dr Marko Mišić  dipl. ing. Pavle Divović | Andrea Popović 2018/0316  Vukašin Nedeljković 2018/0217 |

Beograd, jun 2022.

Sadržaj

[Sadržaj 2](#_Toc106215586)

[1. Problem 1 - SIMPLEX 3](#_Toc106215587)

[1.1. Tekst problema 3](#_Toc106215588)

[1.2. Delovi koje treba paralelizovati 3](#_Toc106215589)

[1.2.1. Diskusija 3](#_Toc106215590)

[1.2.2. Način paralelizacije 3](#_Toc106215591)

[1.3. Rezultati 4](#_Toc106215592)

[1.3.1. Logovi izvršavanja 4](#_Toc106215593)

[1.3.2. Grafici ubrzanja 5](#_Toc106215594)

[1.3.3. Diskusija dobijenih rezultata 5](#_Toc106215595)

[2. PROBLEM 2 – GAME OF LIFE 6](#_Toc106215596)

[2.1. Tekst problema 6](#_Toc106215597)

[2.2. Delovi koje treba paralelizovati 6](#_Toc106215598)

[2.2.1. Diskusija 6](#_Toc106215599)

[2.2.2. Način paralelizacije 7](#_Toc106215600)

[2.3. Rezultati 7](#_Toc106215601)

[2.3.1. Logovi izvršavanja 7](#_Toc106215602)

[2.3.2. Grafici ubrzanja 8](#_Toc106215603)

[2.3.3. Diskusija dobijenih rezultata 8](#_Toc106215604)

[3. PROBLEM 3 – HOTSPOT 9](#_Toc106215605)

[3.1. Tekst problema 9](#_Toc106215606)

[3.2. Delovi koje treba paralelizovati 9](#_Toc106215607)

[3.2.1. Diskusija 9](#_Toc106215608)

[3.2.2. Način paralelizacije 9](#_Toc106215609)

[3.3. Rezultati 10](#_Toc106215610)

[3.3.1. Logovi izvršavanja 11](#_Toc106215611)

[3.3.2. Grafici ubrzanja 12](#_Toc106215612)

[3.3.3. Diskusija dobijenih rezultata 12](#_Toc106215613)

1. Problem 1 - SIMPLEX
   1. Tekst problema

Paralelizovati program koji računa integral funkcije F na osnovu unutrašnjosti simplex-a (https://en.wikipedia.org/wiki/Simplex) u 20 dimenzija korišćenjem Monte Carlo metode. Program se nalazi u datoteci simplex.c. U izvornom kodu data je matrica eksponenata jednačine i ivica simplexa. Ulazni parametar programa je broj iteracija aproksimacije. Program testirati sa parametrima koji su dati u datoteci run.

* 1. Delovi koje treba paralelizovati
     1. Diskusija

Uočili smo par potencijalnih funkcija za paralelizaciju. Nakon njihove paralelizacije smo zaključili da nam najbolje performanse daje paralelizacija simplex\_unit\_to\_general funkcije.

* + 1. Način paralelizacije

Prvo je alocarana memorija na uredjaju za potrebne nizove. Nakon toga smo pozivali kernel sa dimenzijama numBlocks (predstavlja broj n), m\*m (nismo koristili NUM\_OF\_GPU\_THREADS zato sto je m\*m dosta manje od NUM\_OF\_GPU\_THREADS a nije nam potrebno toliko niti). U krenelu svaka nit računa novu vrednost i na kraju vrsimo redukciju i sabiranje svih elemenata.

* 1. Rezultati

|  |  |
| --- | --- |
| ulaz | Ubrzanje |
| 500000 | = 2.07 |
| 100000 | = 1.89 |
| 1000000 | = 2.00 |
| **Srednje ubrzanje** | **1.99** |

* + 1. Logovi izvršavanja

Simplex 50000

Time elapsed, sequential in ms: 1130.017700

Time elapsed, parallel in ms: 545.342896

Test PASSED

Simplex 100000

Time elapsed, sequential in ms: 2367.957031

Time elapsed, parallel in ms: 1249.830933

Test PASSED

Simplex 1000000

Time elapsed, sequential in ms: 20738.814453

Time elapsed, parallel in ms: 10370.778320

Test PASSED

Listing 1. Sekvencijalna i paralelna izvršavanja

* + 1. Grafici ubrzanja

Slika 1. Grafik ubrzanja za različite ulazne parametre

* + 1. Diskusija dobijenih rezultata

Primećujemo ubrzanja u odnosu na sekvencijalni kod. Ubrzanja nisu velika zato što su u problemu dimenzije matrica dosta male i nije moguće u potpunosti iskoristiti potencijal GPU. Usled numeričke nestabilnosti u radu sa realnim podacima imamo mala odstupanja koja su nešto veća od ACCURACY zbog neslaganja na centralnom i grafičkom procesoru.

1. PROBLEM 2 – GAME OF LIFE
   1. Tekst problema

Paralelizovati program koji implementira simulaciju ćelijskog automata Game of Life. Simulacija je predstavljena dvodimenzionalnom matricom dimenzija w x h, a svaka ćelija c može uzeti vrednost 1 ukoliko predstavlja živu ćeliju, a 0 ukoliko je mrtva. Za svaku ćeliju se vrši izračunavanje vrednosti n koja predstavlja zbir živih ćelija u susedstvu posmatrane ćelije. Posmatra se osam suseda. Ćelije se rađaju i umiru prema pravilima iz sledeće tabele.

|  |  |  |  |
| --- | --- | --- | --- |
| **Vrednost C** | **Vrednost N** | **Nova vrednost C** | **Komentar** |
| 1 | 0, 1 | 0 | Usamljena ćelija umire |
| 1 | 4, 5, 6, 7, 8 | 0 | Ćelija umire usled prenaseljenosti |
| 1 | 2,3 | 1 | Ćelija živi |
| 0 | 3 | 1 | Rađa se nova ćelija |
| 0 | 0, 1, 2, 4, 6, 7, 8 | 0 | Nema promene stanja |

Može se smatrati da su ćelije van opsega posmatrane matrice mrtve. Kod koji treba paralelizovati se nalazi u datoteci gameoflife.c u arhivi koja je priložena uz ovaj dokument. Program se može prevesti u dve konfiguracije: sa vizuelnim prikazom i bez vizuelnog prikaza, u zavisnosti da li je definisan makro LIFE\_VISUAL. Prevođenje sa vizuelnim prikazom se može izvršiti naredbom make visual. Paralelizovati konfiguraciju bez vizuelnog prikaza, a vreme meriti na nivou cele simulacije i na nivou jednog izvršavanja funkcije evolve. Koristiti 2D organizaciju jezgra, ukoliko je moguće. Program testirati sa parametrima koji su dati u datoteci run

* 1. Delovi koje treba paralelizovati
     1. Diskusija

Paralelizaciju je jedino moguće izvršiti u funkciji evolve. U funkciji postoje po 3 ugneždene for petlje. Poslednju ugneždenu petlju se ne isplati paralelizovati zato što se u njoj jedino izvršava dodela vrednosti, dok druga ugneždena petlja ima mali broj iteracija i nju smo optimizovali korišćenjem tehnike loop unrolling. Zbog toga kompletnu paralelizaciju vršimo nad prvom ugneždenenom petljom.

* + 1. Način paralelizacije

Prvo je alocarana memorija na uredjaju za potrebne matrice. Nakon toga smo pozivali kernel sa dimenzijama numBlocks (predstavlja gornji ceo deo od dimenzije matrice podeljene sa ukupnim brojem niti po bloku), NUM\_OF\_GPU\_THREADS. U krenelu svaka nit računa novu vrednost samo jednog elementa matrice. Na kraju se vrši kopiranje podatak sa uredjaja na host.

* 1. Rezultati

|  |  |
| --- | --- |
| Ulazni parametri | Ubrzanje |
| 30 30 1000 | = 4.899117 |
| 500 500 10 | = 110.557083 |
| 1000 1000 100 | = 283.761780 |
| 1000 1000 1000 | = 281.197937 |
| **Srednje ubrzanje** | **170.10** |

* + 1. Logovi izvršavanja

width=30, height=30, iteration=1000

Time elapsed, sequential in ms: 52.596287

Time elapsed, parallel in ms: 10.735872

Test PASSED

width=500, height=500, iteration=10

Time elapsed, sequential in ms: 139.224091

Time elapsed, parallel in ms: 1.259296

Test PASSED

width=1000, height=1000, iteration=100

Time elapsed, sequential in ms: 5609.303223

Time elapsed, parallel in ms: 19.767649

Test PASSED

width=1000, height=1000, iteration=1000

Time elapsed, sequential in ms: 55344.406250

Time elapsed, parallel in ms: 196.816544

Test PASSED

Listing 2. Sekvencijalna i paralelna izvršavanja

* + 1. Grafici ubrzanja

Slika 2. Grafik ubrzanja za različite ulazne parametre

* + 1. Diskusija dobijenih rezultata

Primećujemo velika ubrzanja u rasponu od 5 puta (za test primer sa malim dimenzijama matrice) do čak 280 puta za veće matrice. Ova ubrzanja su dosta veća u odnosu na OpenMP i MPI biblioteke.

1. PROBLEM 3 – HOTSPOT
   1. Tekst problema

Paralelizovati program koji rešava problem promene temperature na čipu procesora u dvodimenzionalnom prostoru kroz vreme, ako su poznati početna temperatura i granični uslovi. Simulacija rešava seriju diferencijalnih jednačina nad pravilnom mrežom tačaka kojom se aproksimira površina procesora. Svaka tačka u mreži predstavlja prosečnu temperaturu za odgovarajuću površinu na čipu. Mreža tačaka je predstavljena odgovarajućom matricom koja opisuje trenutne temperature. Program se nalazi u direktorijumu hotspot u arhivi koja je priložena uz ovaj dokument. Program se sastoji od više datoteka, od kojih je od interesa datoteka hotspot.c. Verifikaciju paralelizovanog rešenja vršiti nad dobijenim temperaturama u poslednjem stanju sistema. Koristiti 2D organizaciju jezgra, ukoliko je moguće. Način pokretanja programa se nalazi u datoteci run. Kao pomoćno sredstvo, data je i python skripta koja izlaznu datoteku formatira u heatmap sliku u PNG formatu.

* 1. Delovi koje treba paralelizovati
     1. Diskusija

Diskusija Paralelizacija je izvršena nad nad funkcijom single\_iteration koju smo podelili na tri funkcije. I svaka od funkcija je obrađivala jedan od tri slučaja (ugao, ivica, sredina).

* + 1. Način paralelizacije

Prvo je alocarana memorija na uredjaju za potrebne matrice. Kako smo posao podelili na tri funkcije za svaku od funkcija smo odredili koliko blokova niti je potrebno i pozvali smo potrebne kernele. U kernelu svaka nit računa novu vrednost samo jednog elementa matrice. Na kraju se vrši kopiranje podatak sa uredjaja na host.

* 1. Rezultati

|  |  |
| --- | --- |
| Ulaz | Ubrzanja |
| 32 32 8192 1 ./data/temp32\_32 ./data/power32\_32 ./output/out32\_32 | = 0.540607 |
| 256 256 8192 1 ./data/temp256\_256 ./data/power256\_256 ./output/out256\_256 | = 16.429686 |
| 1024 1024 4096 1 ./data/temp1024\_1024 ./data/power1024\_1024 ./output/out1024\_1024\_4096 | = 70.337936 |
| 1024 1024 8192 1 ./data/temp1024\_1024 ./data/power1024\_1024 ./output/out1024\_1024\_8192 | = 75.171761 |
| 1024 1024 16384 1 ./data/temp1024\_1024 ./data/power1024\_1024 ./output/out1024\_1024\_16384 | = 77.442940 |
| 1024 1024 32768 1 ./data/temp1024\_1024 ./data/power1024\_1024 ./output/out1024\_1024\_32768 | = 77.692184 |
| **Srednje ubrzanje** | **52.94** |

* + 1. Logovi izvršavanja

grid\_rows=32, grid\_cols=32, sim\_time=8192, tempfile=./data/temp32\_32, powerfile=./data/power32\_32

Time elapsed, sequential in ms: 92.896797

Time elapsed, parallel in ms: 171.837860

Test PASSED

grid\_rows=256, grid\_cols=256, sim\_time=8192, tempfile=./data/temp256\_256, powerfile=./data/power256\_256

Time elapsed, sequential in ms: 5790.399414

Time elapsed, parallel in ms: 352.435181

Test PASSED

grid\_rows=1024, grid\_cols=1024, sim\_time=4096, tempfile=./data/temp1024\_1024, powerfile=./data/power1024\_1024

Time elapsed, sequential in ms: 46810.578125

Time elapsed, parallel in ms: 665.509705

Test PASSED

grid\_rows=1024, grid\_cols=1024, sim\_time=8192, tempfile=./data/temp1024\_1024, powerfile=./data/power1024\_1024

Time elapsed, sequential in ms: 97473.789062

Time elapsed, parallel in ms: 1296.680908

Test PASSED

grid\_rows=1024, grid\_cols=1024, sim\_time=16384, tempfile=./data/temp1024\_1024, powerfile=./data/power1024\_1024

Time elapsed, sequential in ms: 196322.156250

Time elapsed, parallel in ms: 2535.055664

Test PASSED

grid\_rows=1024, grid\_cols=1024, sim\_time=32768, tempfile=./data/temp1024\_1024, powerfile=./data/power1024\_1024

Time elapsed, sequential in ms: 392788.468750

Time elapsed, parallel in ms: 5055.701172

Test PASSED

Listing 3. Sekvencijalna i paralelna izvršavanja

* + 1. Grafici ubrzanja

Slika 3. Grafik ubrzanja za različite ulazne parametre

* + 1. Diskusija dobijenih rezultata

Primećujemo velika ubrzanja za sve test primere osim prvog gde je dimenzija čipa dosta mala. Za ostale ulazne parametre ubrzanje je primetno i u rasponu su od 16 do 78. Ova ubrzanja su dosta veća u odnosu na OpenMP i MPI biblioteke. Usled numeričke nestabilnosti u radu sa realnim podacima imamo mala odstupanja koja su nešto veća od ACCURACY zbog neslaganja na centralnom i grafičkom procesoru.