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 | Uros Jovanovic 2018/0412 |

Beograd, novembar 2021.

Sadržaj

[Sadržaj 2](#_Toc93696169)

[1. Problem 2 – Game of Life 3](#_Toc93696170)

[1.1. Tekst problema 3](#_Toc93696171)

[1.2. Delovi koje treba paralelizovati 3](#_Toc93696172)

[1.2.1. Diskusija 3](#_Toc93696173)

[1.2.2. Način paralelizacije 3](#_Toc93696174)

[1.3. Rezultati 3](#_Toc93696175)

[1.3.1. Logovi izvršavanja 4](#_Toc93696176)

[1.3.2. Grafici ubrzanja 5](#_Toc93696177)

[1.3.3. Diskusija dobijenih rezultata 5](#_Toc93696178)

[2. Problem 1 – SIMPLEX 6](#_Toc93696179)

[2.1. Teskt problema 6](#_Toc93696180)

[2.2. Delovi koje treba paralelizovati 6](#_Toc93696181)

[2.2.1. Diskusija 6](#_Toc93696182)

[2.2.2. Nacin paralelizacije 6](#_Toc93696183)

[2.3. Rezultati 6](#_Toc93696184)

[2.3.1. Logovi izvrsavanja 6](#_Toc93696185)

[2.3.2. Grafici ubrzanja 7](#_Toc93696186)

[2.3.3. Diskusija dobijenih rezultata 7](#_Toc93696187)

[3. Problem – HOTSPOT 8](#_Toc93696188)

[3.1. Tekst problema 8](#_Toc93696189)

[3.2. Delovi koje treba paralelizovati 8](#_Toc93696190)

[3.2.1. Diskusija 8](#_Toc93696191)

[3.2.2. Nacin paralelizacije 8](#_Toc93696192)

[3.3. Rezultati 9](#_Toc93696193)

[3.3.1. Logovi izvrsavanja 9](#_Toc93696194)

[3.3.2. Grafici ubrzanja 10](#_Toc93696195)

[3.3.3. Diskusija dobijenih resenja 11](#_Toc93696196)

1. Problem 2 – Game of Life

U okviru ovog poglavlja je dat kratak izveštaj u vezi rešenja zadatog problema 1. Rešenja svih problema dati u okviru **jednog** dokumenta. U zavisnosti od broja postavljenih problema, dodati nova poglavlja u dokument.

* 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.

* 1. Delovi koje treba paralelizovati
     1. Diskusija

Ono gde moze da se primeni paralelizacija koda je sama funkcija evolve, gde uz pomoc cuda api-ja je moguce izbaciti dva spoljna for-a, a pored toga, oivicavanje matrice sa nulama je moguce izbaciti i dva manja unutrasnja for-a i direktno pristupati celijama matrice. Pored ovoga je moguce izvrsiti i paralelizaciju inicijalizacije matrice.

* + 1. Način paralelizacije

Prvo se izracunaju vrednosti velicine grid-a i velicine samog bloka. Velicina grid-a je kolicnik proizvoda broja vrsta i broja kolona i blockSize-a, a velicina bloka je blockSize. Funkcija evolve se izvrsava na gpu i koordinate matrice se racinaju row = blockIdx.x \* BLOCK\_SIZE + threadIdx.x a col = blockIdx.y \* BLOCK\_SIZE + threadIdx.y, tako da index u matrici racunamo kao index = row \*size + col gde je size w ili h posto su nam dimenzije matrice jednake. I sad svaka nit ce gledati svoje susede u tekucoj matrici(curr\_grid) i racunati broj zivih suseda i na osnovu tog broja ce odluciti da li je ta trenutna celija ziva ili ne, i taj rezultat ce upisati u novu matricu(next\_grid). Kad upise rezultat u novu matricu ta nova matrica se upisuje u matriu na hostu a sledeca matrica koja se proverava je upravo ta nova matrica (next\_grid postaje curr\_grid).

* 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.

Matrix dimension: 30x30

Iterations: 1000

Time elapsed: 6.172160ms

-------------------------

Matrix dimension: 30x30

Iterations: 1000

Time elapsed: 37ms

Listing 1. Pralelno izvršavanje GAMEOFLIFE

Matrix dimension: 500x500

Iterations: 10

Time elapsed: 1.940704ms

-------------------------

Matrix dimension: 500x500

Iterations: 10

Time elapsed: 72ms

Listing 2. Pralelno izvršavanje GAMEOFLIFE

Matrix dimension: 1000x1000

Iterations: 100

Time elapsed: 65.697090ms

--------------------------

Matrix dimension: 1000x1000

Iterations: 100

Time elapsed: 3041ms

Listing 3. Pralelno izvršavanje GAMEOFLIFE

Matrix dimension: 1000x1000

Iterations: 1000

Time elapsed: 578.350769ms

---------------------------

Matrix dimension: 1000x1000

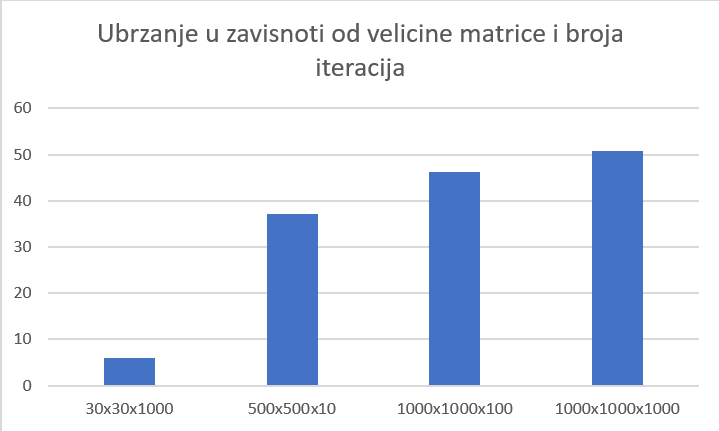
Iterations: 1000

Time elapsed: 29344ms

Listing 4. Pralelno izvršavanje GAMEOFLIFE

* + 1. Grafici ubrzanja

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



Slika 1. Grafik zavisnosti ubrzanja naivnog algoritma od dužine teksta za N = 4 niti (primer)

* + 1. Diskusija dobijenih rezultata

Iz grafika vidimo da se sa povecanjem velicine matrice i broja iteracija znatno povecava i ubrzanje.

1. Problem 1 – SIMPLEX
   1. Teskt 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

Ono gde moze da se izvrsi paralelizacija je u funkcji simplex\_unit\_to\_general gde uz pomoc cuda api-ja mogu da se izbace dve spoljne for petlje.

* + 1. Nacin paralelizacije

Ova funkcija moze da se paralelizuje tako sto se izvrsavanje raspodele na razlicite blokove, u kojima ce razlicite niti racunati simplex za svoju tacku u nizu phy. Svaka nit racuna svoju tacku na osnovu svog id-ja i na osnovu id-ja bloka u kome se nalazi(row = blockIdx.x \* blockDim.x + threadIdx.x i col = blockIdx.y \* blockDim.y + threadIdx.y). Na kraju se tako popunjen niz kopira phy kopira u niz x.

* 1. Rezultati
     1. Logovi izvrsavanja

U okviru ove sekcije su izloženi rezultati paralelizacije problema 1.

Simplex: 50000

Time elapsed: 1062ms

-------------------------

Simplex: 50000

Time elapsed: 1818ms

Listing 1. Pralelno izvršavanje SIMPLEX

Simplex: 100000

Time elapsed: 2188ms

-------------------------

Simplex: 100000

Time elapsed: 3391ms

Listing 2. Pralelno izvršavanje SIMPEX

Simplex: 1000000

Time elapsed: 18412ms

--------------------------

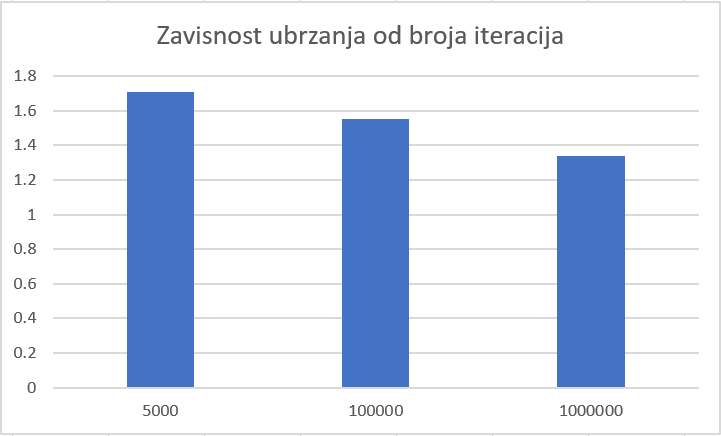
Simplex: 1000000

Time elapsed: 24615ms

Listing 3. Pralelno izvršavanje SIMPLEX

* + 1. Grafici ubrzanja

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



* + 1. Diskusija dobijenih rezultata

Iz dobijenih rezultata vidimo da je paralelizaciom metode simplex\_unit\_to\_general dobijeno ubrzanje za blizu dva puta.

1. Problem – 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

Paralelizacija moze da se izvrsi tako sto se uz pomoc cuda api-ja obezbedi da se svaka iteracije glavne for petlje u funkciji compute\_tran\_temp izvrsava na gpu procesorima, odnosno treba paralelizovati metodu single\_iteration.

* + 1. Nacin paralelizacije

Funkcija single\_iteration je laka za paralelizaciju obzirom da se sastoji samo od dva ugnjezdena fora koja mogu da se zamene sa dva intigera koja predstavljaju poziciju u matrici koju odredjena nit racuna. Velicina grida se poklapa sa velicinom matrice result podeljena sa velicinom bloka, dok nam je jedna velicina bloka 16. Sad na osnovu id bloka i id niti unutar bloka mozemo da nadjemo poziciju matrice koju nit izvrsava. Nakon izvrsavanja svake iteracije vrsi se swap rezult matrice i temp matrice. Na izlasku for-a u metodi compute\_tran\_temp vrsi se prepisivanje rezultata sa gpu-a na odgovarajuce nizove.

* 1. Rezultati
     1. Logovi izvrsavanja

U okviru ove sekcije su izloženi rezultati paralelizacije problema 1.

Paralelno izvrsavanje:

Input: 32 32 8192

Time elapsed: 72.119682ms

-------------------------

Sekvencijalno izvrsavanje:

Input: 32 32 8192

Time elapsed: 20ms

Listing 1. Pralelno izvršavanje HOTSPOT

Paralelno izvrsavanje:

Input: 256 256 8192

Time elapsed: 368.864960ms

-------------------------

Sekvencijalno izvrsavanje:

Input: 256 256 8192

Time elapsed: 1330ms

Listing 2. Pralelno izvršavanje HOTSPOT

Paralelno izvrsavanje:

Input: 1024 1024 4096

Time elapsed: 2326.470215ms

--------------------------

Sekvencijalno izvsavanje:

Input: 1024 1024 4096

Time elapsed: 10773ms

Listing 3. Pralelno izvršavanje HOTSPOT

Paralelno izvrsavanje:

Input: 1024 1024 8192

Time elapsed: 4632.452637ms

--------------------------

Sekvencijalno izvrsavane

Input: 1024 1024 4096

Time elapsed: 21921ms

Listing 4. Pralelno izvršavanje HOTSPOT

Paralelno izvrsavanje:

Input: 1024 1024 16384

Time elapsed: 9216.209961ms

--------------------------

Sekvencijalno izvrsavanje:

Input: 1024 1024 16384

Time elapsed: 42580ms

Listing 5. Pralelno izvršavanje HOTSPOT

Paralelno izvrsavanje:

Input: 1024 1024 32768

Time elapsed: 18373.375000ms

--------------------------

Sekvencijalno izvrsavanje:

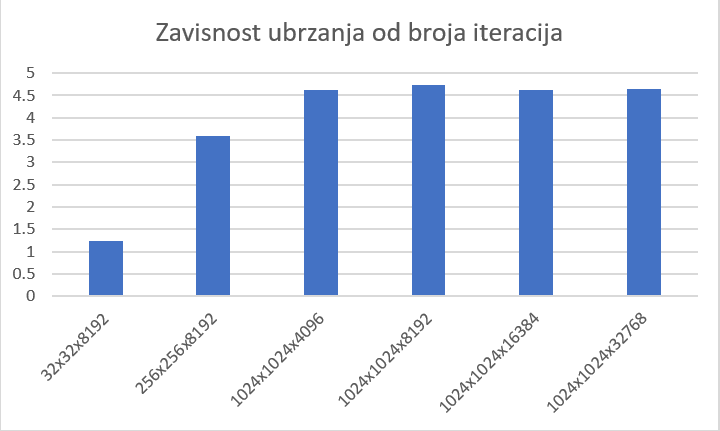
Input: 1024 1024 32768

Time elapsed: 85204ms

Listing 6. Pralelno izvršavanje HOTSPOT

* + 1. Grafici ubrzanja

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



* + 1. Diskusija dobijenih resenja

Iz dobijenih resenja vidimo da smo za male vrednosti matrice i broja iteracija dobili duze izvrsavanje paralelne implementacije u odnosu na sekvencijalnu. A sa porastom broja iteracija povecava se i ubrzanje, koje posle varira oko 5 puta.