UNIVERZITET U BEOGRADU - ELEKTROTEHNIČKI FAKULTET

MULTIPROCESORKI SISTEMI (13S114MUPS, 13E114MUPS)

DOMAĆI ZADATAK 4 – CUDA

Izveštaj o urađenom domaćem zadatku

Predmetni asistent:

doc. dr Marko Mišić

Studenti:

Bogdan Bebić 2017/0011

Uroš Krstić 2017/0228

Beograd, decembar 2020.

SADRŽAJ

1. PROBLEM 1 – RAČUNANJE BROJA Π (PI)

1.1. Tekst problema

Paralelizovati program koji izračunava vrednost broja PI korišćenjem formule:

$$\pi = 4 * \sum_{k=1}^{n} \frac{(-1)^{k+1}}{2k-1}.$$

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. Program testirati sa parametrima koji su dati u datoteci **run**.

1.2. Delovi koje treba paralelizovati

1.2.1. Diskusija

Jedino mesto u programu koje se može paralelizovati je glavna for petlja koja radi sumu faktora koji čine vrednost broja pi.

1.2.2. Način paralelizacije

Paralelizacija je implementirana tako što je napravljeno stablo redukcije nad for petljom od interesa. Svaka nit računa vrednosti iteracija i u privatni registar redukuje po 1000 iteracija, pa potom niti kooperativno redukuju svoje međurezultate unutar deljene memorije bloka, a potom nulta nit svakog bloka upiše rezultat u globalnu memoriju – ti međurezultati se redukuju na CPU.

1.3. Rezultati

Paralelizacijom ovog programa na GPU nitima se dobija ubrzanje koje je prikazano na grafiku (Slika 1) – ovo ubrzanje se vidi u svim slučajevima, a najbolje na većem poslu.

1.3.1. Logovi izvršavanja

Ovde su dati logovi izvršavanja za definisane test primere.

```
$ ./z1 1000000
--------------------Sequential execution--------------------
```

```
Before for loop, factor = 0.000000.
After for loop, factor = -1.000000.
With n = 1000000 terms
   Our estimate of pi = 3.14159165358977
   Ref estimate of pi = 3.14159265358979
----------------------Parallel execution----------------------
Before for loop, factor = 0.000000.
After for loop, factor = -1.000000.
With n = 1000000 terms
   Our estimate of pi = 3.14159131050110
   Ref estimate of pi = 3.14159265358979

Sequential elapsed time: 0.003195s
Parallel elapsed time: 0.000626s
Test PASSED

$ ./z1 10000000
----------------------Sequential execution----------------------
Before for loop, factor = 0.000000.
After for loop, factor = -1.000000.
With n = 10000000 terms
   Our estimate of pi = 3.14159255358979
   Ref estimate of pi = 3.14159265358979
----------------------Parallel execution----------------------
Before for loop, factor = 0.000000.
After for loop, factor = -1.000000.
With n = 10000000 terms
   Our estimate of pi = 3.14159221098971
   Ref estimate of pi = 3.14159265358979

Sequential elapsed time: 0.031884s
Parallel elapsed time: 0.000737s
Test PASSED

$ ./z1 100000000
----------------------Sequential execution----------------------
Before for loop, factor = 0.000000.
After for loop, factor = -1.000000.
With n = 100000000 terms
```

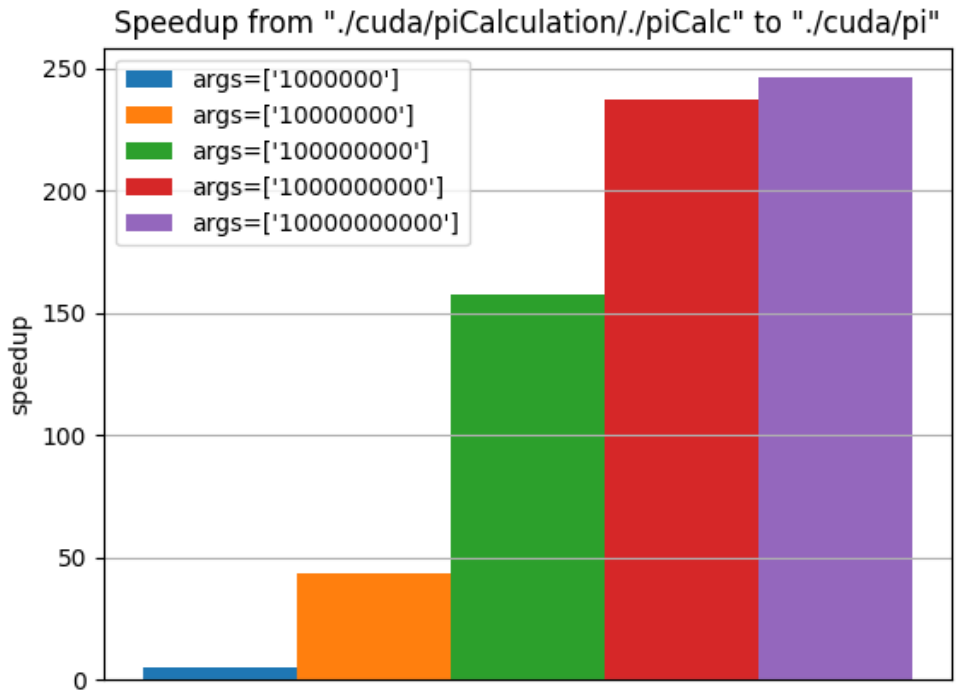
```
   Our estimate of pi = 3.14159264358933
   Ref estimate of pi = 3.14159265358979
----------------------Parallel execution----------------------
Before for loop, factor = 0.000000.
After for loop, factor = -1.000000.
With n = 100000000 terms
   Our estimate of pi = 3.14159234669756
   Ref estimate of pi = 3.14159265358979

Sequential elapsed time: 0.318430s
Parallel elapsed time: 0.002021s
Test PASSED

$ ./z1 1000000000
----------------------Sequential execution----------------------
Before for loop, factor = 0.000000.
After for loop, factor = -1.000000.
With n = 1000000000 terms
   Our estimate of pi = 3.14159265258805
   Ref estimate of pi = 3.14159265358979
----------------------Parallel execution----------------------
Before for loop, factor = 0.000000.
After for loop, factor = -1.000000.
With n = 1000000000 terms
   Our estimate of pi = 3.14159234669756
   Ref estimate of pi = 3.14159265358979

Sequential elapsed time: 3.183109s
Parallel elapsed time: 0.013406s
Test PASSED

$ ./z1 10000000000
----------------------Sequential execution----------------------
Before for loop, factor = 0.000000.
After for loop, factor = -1.000000.
With n = 10000000000 terms
   Our estimate of pi = 3.14159265348835
   Ref estimate of pi = 3.14159265358979
----------------------Parallel execution----------------------
```

```
Before for loop, factor = 0.000000.
After for loop, factor = -1.000000.
With n = 10000000000 terms
   Our estimate of pi = 3.14159234669756
   Ref estimate of pi = 3.14159265358979

Sequential elapsed time: 31.834623s
Parallel elapsed time: 0.129242s
Test PASSED
```

Listing 1. Logovi izvršavanja računanja pi

1.3.2. Grafici ubrzanja

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



Slika 1. Grafik zavisnosti ubrzanja za različite argumente pokretanja

1.3.3. Diskusija dobijenih rezultata

Paralelizacija na GPU nitima je donela značajno ubrzanje. Za posao koji je dovoljno veliki – ovo ubrzanje dostiže skoro 250 puta kada imamo test primer sa najvećim brojem iteracija petlje. Program je embarrassingly parallel i svaka nit dobija određeni chunk iteracija za računanje.

2. PROBLEM 2 – *NEEDLEMAN-WUNSCH* ALGORITAM

2.1. 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 mogućnost korišćenja deljene memorije radi ubrzavanja pristupa podacima. Koristiti 2D organizaciju jezgra, ukoliko je moguće. Program testirati sa parametrima koji su dati u datoteci **run**.

2.2. Delovi koje treba paralelizovati

2.2.1. Diskusija

Usled zavisnosti po podacima preko granica iteracija for petlji koje rade obradu matrice (for petlje su do-across tipa), nemoguće je paralelizovati spoljašnje petlje obrade, slično tome ni obrade dve trougaone matrice koje čine matricu koja se obrađuje se ne mogu raditi u paraleli. Uočeno je da unutrašnje petlje obrade trouganih delova matrice rade iteriranje po dijagonalama matrice, a da svaki element na dijagonali ima nezavisnu obradu od ostalih. Ove dve unutrašnje petlje su zbog toga paralelizovane. Osim toga, i deo inicijalizacije je paralelizovan kako bi se smanjili režijski troškovi kopiranja sa CPU na GPU. Kako TRACEBACK deo programa radi sekvencijalnu pretragu sa zavisnostima od prošle iteracije, ovaj deo programa mora ostati sekvencijalan zbog korektnosti izvršavanja.

2.2.2. Način paralelizacije

Paralelizovane su unutrašnje petlje obrade trouganih delova matrice tako što svaka nit obrađuj jedan element. Primećeno je da se određenim elementim pristupa više puta u globalnoj memoriji, pa je urađen njihov *prefetching* u deljenu memoriju. Osim ovoga, primećeno je da se kopiranje *reference* niza sa GPU na CPU može raditi nezavisno od obrade trougaonih matrica, pa je ovaj deo posla izdvojen u odvojen *stream* i poslat na kopiranje DMA modulom GPU kartice, čime se paralelizuje kopiranje podataka i obrada.

2.3. Rezultati

Paralelizacijom ovog programa se dobija ubrzanje koje je prikazano na grafiku (Slika 2).

2.3.1. Logovi izvršavanja

Ovde su dati logovi izvršavanja za definisane test primere.

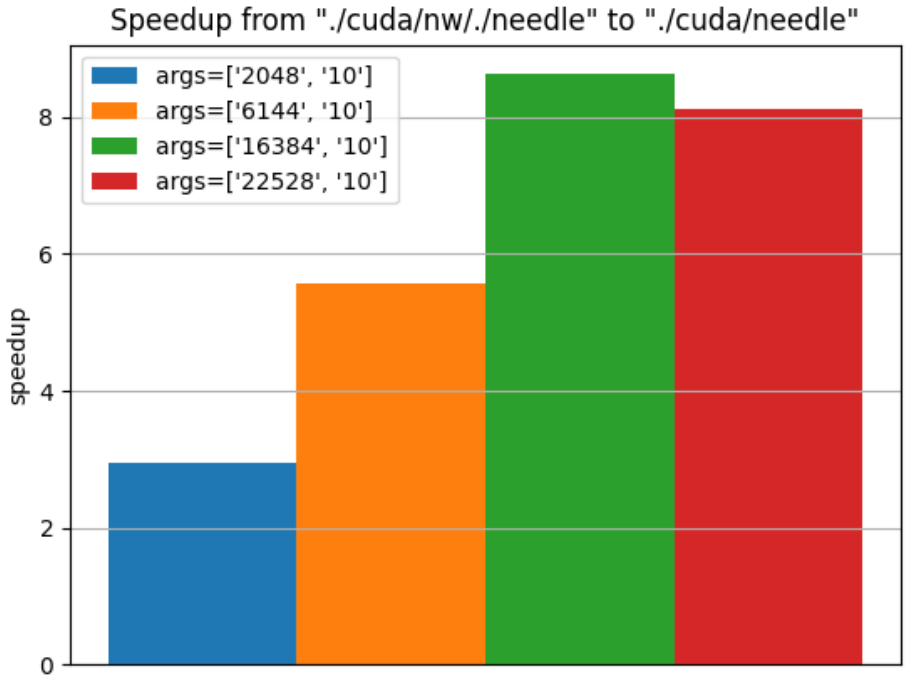
```
$ ./z2 2048 10
----------------------Sequential execution----------------------
Start Needleman-Wunsch
Processing top-left matrix
Processing bottom-right matrix
----------------------Parallel execution----------------------
Start Needleman-Wunsch
Processing top-left matrix
Processing bottom-right matrix

Sequential elapsed time: 0.218328s
Parallel elapsed time: 0.074034s
Test PASSED

$ ./z2 6144 10
----------------------Sequential execution----------------------
Start Needleman-Wunsch
Processing top-left matrix
Processing bottom-right matrix
----------------------Parallel execution----------------------
Start Needleman-Wunsch
Processing top-left matrix
Processing bottom-right matrix

Sequential elapsed time: 1.993039s
Parallel elapsed time: 0.357515s
Test PASSED

$ ./z2 16384 10
----------------------Sequential execution----------------------
Start Needleman-Wunsch
```

```
Processing top-left matrix
Processing bottom-right matrix
----------------------Parallel execution----------------------
Start Needleman-Wunsch
Processing top-left matrix
Processing bottom-right matrix

Sequential elapsed time: 14.725623s
Parallel elapsed time: 1.706326s
Test PASSED

$ ./z2 22528 10
----------------------Sequential execution----------------------
Start Needleman-Wunsch
Processing top-left matrix
Processing bottom-right matrix
----------------------Parallel execution----------------------
Start Needleman-Wunsch
Processing top-left matrix
Processing bottom-right matrix

Sequential elapsed time: 28.038922s
Parallel elapsed time: 3.455601s
Test PASSED
```

Listing 2. Logovi izvršavanja Needleman-Wunsch algoritma

2.3.2. Grafici ubrzanja

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



Slika 2. Grafik zavisnosti ubrzanja za različite argumente pokretanja

2.3.3. Diskusija dobijenih rezultata

Problem sam po sebi nije preterano paralelan pošto koristi dinamičko programiranje koje unosi dosta zavisnoti po podacima preko granica iteracija svih petlji. Sa znatnim povećanjem broja podataka nad kojima se vrši obrada, povećava se deo nad kojim se mora raditi sekvencijalna obrada jer su svi podaci potrebni u TRACEBACK delu, pa se samim tim povećava i sekvencijalni deo aplikacije. Osim toga, zbog ograničenog broj CUDA jezgara na grafičkoj kartici na kojoj je rađena paralelizacija koji mogu da rade paralelno, značajnim povećanjem se ne dobija mnogo bolji paralelizam obrade podataka.

3. PROBLEM 3 – INTERAKCIJA ČVRSTIH TELA (*N-BODY*)

3.1. 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. Obratiti pažnju na efikasnost paralelizacije, mogućnost upotrebe deljene memorije i potrebu za redukcijom. Verifikaciju paralelizovanog rešenja vršiti nad dobijenim energijama i poslednjem stanju sistema. Način pokretanja programa se nalazi u datoteci **run**.

3.2. Delovi koje treba paralelizovati

3.2.1. Diskusija

Usled zavisnosti po podacima preko granica iteracija for petlje koja radi glavnu obradu (for petlja je do-across tipa), nemoguće je paralelizovati spoljašnju petlju obrade. Unutrašnje petlje se mogu paralelizovati uz vođenje računa o sinhronizaciji pristupa promenljivama. Primećeno je da se funkcije Compute_force, Update_part i Compute_energy mogu paralelizovati uz vođenje računa o redukciji promenljivih, ali se Compute_energy radi samo jednom, pa nije od interesa za paralelizaciju.

3.2.2. Način paralelizacije

Update_part i okružujuća for petlja su trivijalno paralelizovane jer je okružujuća for petlja do-all tipa (nema zavisnosti preko granice iteracija petlje). Compute_force je paralelizovana na način da svaka GPU nit radi računanje sile između dve čestice i da taj rezultat upisuje u globalne promenljive kojima pristupa samo ta nit. Potom se radi odgovarajuća redukcija ovih sila na GPU (ovo je isto trivijalna paralelizacija petlje do-all tipa). Ovaj način pristupa promenljivama dovodi do toga da nema potrebe za sinhroizacijom niti unutar Compute_force jer nema nikakvih *hazard*-a pri pristupima memoriji – sve globalne promenljive su logički privatne.

3.3. Rezultati

Paralelizacijom ovog programa se dobija ubrzanje koje je prikazano na grafiku (Slika 3) – ovo ubrzanje se najbolje vidi u slučaju većeg posla.

3.3.1. Logovi izvršavanja

Ovde su dati logovi izvršavanja za definisane test primere.

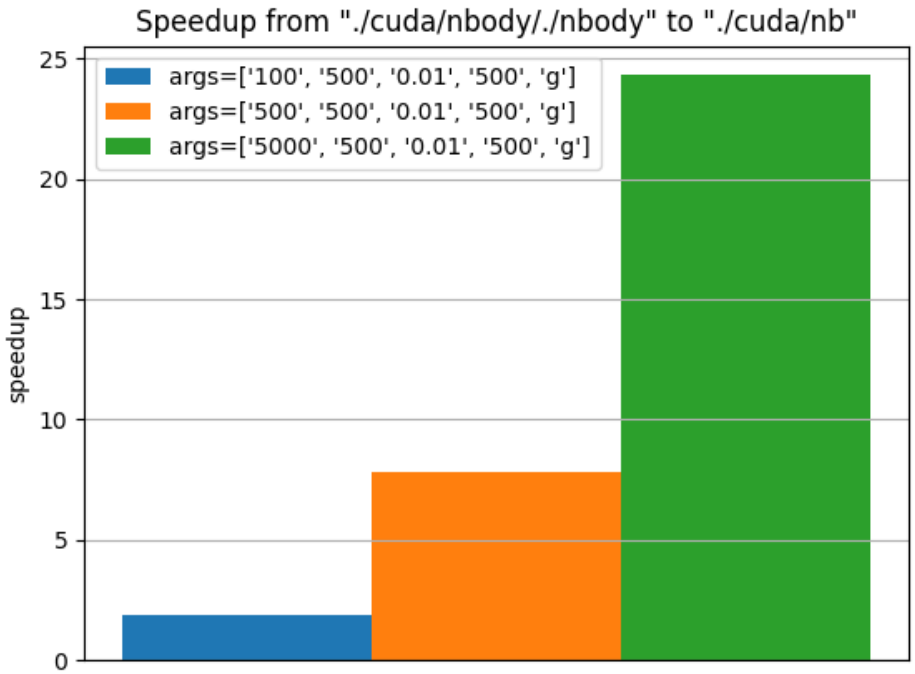
```
$ ./z3 100 500 0.01 500 g
----------------------Sequential execution----------------------
   PE = -6.985593e+36, KE = 2.250000e+35, Total Energy = -6.760593e+36
   PE = -7.035612e+36, KE = 1.304554e+36, Total Energy = -5.731058e+36
Elapsed time = 6.552005e-02 seconds
----------------------Parallel execution----------------------
   PE = -6.985593e+36, KE = 2.250000e+35, Total Energy = -6.760593e+36
   PE = -7.035612e+36, KE = 1.304554e+36, Total Energy = -5.731058e+36
Elapsed time = 3.521395e-02 seconds

Sequential elapsed time: 0.065531s
Parallel elapsed time: 0.035248s
Test PASSED

$ ./z3 500 500 0.01 500 g
----------------------Sequential execution----------------------
   PE = -4.831939e+37, KE = 1.125000e+36, Total Energy = -4.719439e+37
   PE = -4.754056e+37, KE = 1.360414e+36, Total Energy = -4.618014e+37
Elapsed time = 1.621206e+00 seconds
----------------------Parallel execution----------------------
   PE = -4.831939e+37, KE = 1.125000e+36, Total Energy = -4.719439e+37
   PE = -4.754056e+37, KE = 1.360414e+36, Total Energy = -4.618014e+37
Elapsed time = 2.064412e-01 seconds

Sequential elapsed time: 1.621213s
Parallel elapsed time: 0.206455s
Test PASSED

$ ./z3 5000 500 0.01 500 g
----------------------Sequential execution----------------------
```

```
   PE = -6.751832e+38, KE = 1.125000e+37, Total Energy = -6.639332e+38
   PE = -6.649074e+38, KE = 2.481116e+36, Total Energy = -6.624263e+38
Elapsed time = 1.625907e+02 seconds
----------------------Parallel execution----------------------
   PE = -6.751832e+38, KE = 1.125000e+37, Total Energy = -6.639332e+38
   PE = -6.649074e+38, KE = 2.481116e+36, Total Energy = -6.624263e+38
Elapsed time = 6.687269e+00 seconds

Sequential elapsed time: 162.589359s
Parallel elapsed time: 6.687247s
Test PASSED
```

Listing 3. Logovi izvršavanja n-body problema

3.3.2. Grafici ubrzanja

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



Slika 3. Grafik zavisnosti ubrzanja za različite argumente pokretanja

3.3.3. Diskusija dobijenih rezultata

Primećeno je ubrzanje za svaki test primer, a ubrzanje se najbolje vidi povećanjem broja podataka nad kojima se vrši obrada, jer se time CUDA jezgra više iskorišćavaju.