Multiprocesorski sistemi

Domaći zadatak 4
CUDA – osnove
(10 poena)

Uvod

Cilj zadatka je da studente obuči da samostalno razvijaju osnovne CUDA programe.

Podešavanje okruženja

Detaljna uputstva za instaliranje, podešavanje i prvo izvršavanje CUDA programa se mogu naći na adresi http://developer.nvidia.com/nvidia-gpu-computing-documentation ili na sajtu predmeta pod nazivom CUDA Getting Started Guide (Windows) ili CUDA Getting Started Guide (Linux) u zavisnosti koji operativni sistem se koristi. Po tom uputstvu podesiti okruženje za razvoj i kontrolisano izvršavanje (engl. debugging) CUDA programa na lokalnom računaru. Alternativno, koristiti CUDA (**`nvcc`**) na računaru **`rtidev5.etf.rs`**. Prevodilac se nalazi u direktorijumu: **`/usr/local/cuda/bin/`**.

Zadaci

Svi programi treba da koriste GPU za bilo koju obradu. Smatrati da je broj GPU niti na nivou jednog bloka niti određen konstantom **`NUM_OF_GPU_THREADS`**, čija je vrednost za sve zadatke 1024. Obezbediti da niti koje u nekom koraku nemaju posla na korektan način stignu do kraja tela CUDA jezgra.

Kod zadataka gde je to zahtevano, korisnik zadaje samo dimenzije nizova/matrica, a sve potrebne ulazne podatke generisati u operativnoj memoriji uz pomoć generatora slučajnih brojeva iz biblioteke jezika C, a zatim prebaciti u GPU memoriju. Generisani brojevi treba da budu odgovarajućeg tipa u opsegu od **`-MAX`** do **`+MAX`**, gde **`MAX`** ima vrednost 1024. Za sve zadatke je potrebno napisati ili iskoristiti zadatu sekvencijalnu (CPU) implementaciju odgovarajućeg problema koja će biti korišćena kao referentna (*gold*) implementacija prilikom testiranja programa.

Svaki program treba da:

- Generiše ili koristi već obezbeđene ulazne test primere.
- Kopira test primere u GPU memoriju i rezultat iz GPU memorije.
- Izvrši CUDA jezgro nad zadatim test primerom.
- Izvrši sekvencijalnu implementaciju nad zadatim test primerom.
- Ispiše vreme izvršavanja CUDA i sekvencijalne implementacije problema.
- Uporedi rezultat CUDA i sekvencijalne implementacije problema.
- Ispiše "**`Test PASSED`**" ili "**`Test FAILED`**" u zavisnosti da li se rezultat izvršavanja CUDA implementacije podudara sa rezultatom izvršavanja sekvencijalne implementacije.

Kod zadataka koji koriste realne tipove (**`float`**, **`double`**) tolerisati maksimalno odsupanje od **`±ACCURACY`** prilikom poređenja rezultata CPU i GPU implementacije. Smatrati da konstanta **`ACCURACY`** ima vrednost 0.01. **Prilikom rešavanja zadataka voditi računa da se postigne maksimalni mogući paralelizam.** Dozvoljeno je ograničeno preuređivanje dostupnih sekvencijalnih implementacija prilikom paralelizacije. **Ukoliko u nekom zadatku nešto nije dovoljno precizno definisano, student treba da uvede razumnu pretpostavku i da nastavi da izgrađuje svoje rešenje na temeljima uvedene pretpostavke.**

Dostupne sekvencijalne implementacije se nalaze u arhivi **`MPS_DZ4_CUDA.zip`** ili **`MPS_DZ4_CUDA.tar.bz2`** koje se mogu preuzeti na adresi http://mups.etf.rs/dz/2016-2017/. Na **`rtidev5.etf.rs`** računaru arhiva se može dohvatiti i raspakovati sledećim komandama:

Dohvatanje: **`wget http://mups.etf.rs/dz/2016-2017/MPS_DZ4_CUDA.tar.bz2`**

Raspakivanje: **`tar xjvf MPS_DZ4_CUDA.tar.bz2`**

1. Paralelizovati program koji vrši izračunavanje određenog integrala korišćenjem kvadratnog pravila, pravila trapeza i Simpsonovog pravila u dvostrukoj preciznosti. Prilikom zadavanja izvršne konfiguracije jezgra, koristiti 1D rešetku (*grid*). Program se nalazi u datoteci **`quad2.c`** u arhivi koja je priložena uz ovaj dokument. Program testirati sa parametrima koji su dati u datoteci **`run`**.
2. Paralelizovati program koji rešava problem prostiranja toplote u čvrstom telu (ploči) pravougaonog oblika, ako su poznati početni uslovi. Simulacija rešava seriju diferencijalnih jednačina nad pravilnom mrežom tačaka kojom se aproksimira površina tela. Svaka tačka u mreži predstavlja prosečnu temperaturu za odgovarajuću površinu na telu. Mreža tačaka je predstavljena odgovarajućim matricama. Program se iterativno izvršava sve dok se ne zadovolji kriterijum konvergencije. Program se nalazi u datoteci **`heated_plate.c`** u arhivi koja je priložena uz ovaj dokument. Program testirati sa parametrima koji su dati u datoteci **`run`**. Primeri izlaza za zadate ulaze su dati u direktorijumu **`output.`**
3. Paralelizovati program koji vrši statističku analizu prostorne distribucije posmatranih astronomskih tela korišćenjem tzv. *two point angular correlation function* (TPACF). Algoritam proračunava uglovne distance između svih parova tačaka (tela) sa ulaza i generiše histogram posmatranih distanci. Program se nalazi u direktorijumu **`tpacf`** u arhivi koja je priložena uz ovaj dokument. Program se sastoji od više datoteka, od kojih su od interesa datoteke **`main.c`** i **`model_compute_cpu.c`**. Paralelizovati funkciju **`doCompute`** iz **`model_compute_cpu.c`** datoteke. Dozvoljeno je korišćenje atomičnih operacija za rešavanje zadatka, ukoliko su potrebne. Ulazni test primeri se nalaze u direktorijumu **`data`**, a način pokretanja programa u datoteci **`run`**.